Knowledge representation and exploitation for interactive and cognitive robots

Guillaume Sarthou

▶ **To cite this version:**

Guillaume Sarthou. Knowledge representation and exploitation for interactive and cognitive robots. Robotics [cs.RO]. UPS Toulouse - Université Toulouse 3 Paul Sabatier, 2021. English. NNT : . tel-03404043v1

HAL Id: tel-03404043

https://laas.hal.science/tel-03404043v1

Submitted on 26 Oct 2021 (v1), last revised 10 Mar 2022 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *20/09/2021* **par :**

Guillaume SARTHOU

Knowledge representation and exploitation for interactive and cognitive robots

---

JURY

Michael BEETZ	Professeur	Rapporteur
Paulo MENEZES	Professeur Associé	Rapporteur
Rachid ALAMI	Directeur de Recherche	Directeur de Thèse
Aurélie CLODIC	Ingénieure de Recherche	Directrice de Thèse
Simon LACROIX	Directeur de Recherche	Président du Jury
Kerstin FISCHER	Professeure Associée	Membre du Jury

École doctorale et spécialité :
MITT : Informatique et Télécommunications
Unité de Recherche :
LAAS-CNRS
Directeur(s) de Thèse :
Rachid ALAMI et *Aurélie CLODIC*
Rapporteurs :
Michael BEETZ et *Paulo MENEZES*

Acknowledgments

En démarrant cette thèse, je pensais qu'une thèse était un travail personnel et que personnel rimait avec solitaire. Ces trois ans (et même plus en soi) au sein de cette équipe ont su me démontrer mon erreur. Je ne serais jamais arrivé à réaliser le travail que ce manuscrit retranscrit sans toutes ces personnes qui m'ont accompagné durant ces trois années.

En tout premier lieu, je souhaite remercier ma compagne Margaux pour m'avoir supporté (et c'est peu dire) et soutenu depuis certes de nombreuses années, mais avant tout durant cette thèse. Ces matins où j'étais déjà parti, ces soirs où je n'étais pas encore rentré, ces week-ends où j'étais absent, ces vacances quasi-inexistantes, tu as su les accepter, les comprendre, et ne jamais me les reprocher. Merci de m'avoir toujours remotivé, d'avoir écouté mes complaintes et de m'avoir poussé à toujours aller plus loin même si cela voulait dire se voir moins.

Une thèse démarre avant tout avec un directeur de thèse et dans mon cas deux. Je tiens d'abord à remercier Aurélie Clodic pour avoir cru en mon travail avant même le début de cette thèse, et cela, jusqu'au bout. Ton soutien sans failles a été une véritable force pour moi. Si je voulais résumer ce soutien, je pourrais citer qu'une phrase de toi: *"Je n'ai jamais vu d'article comme ça et je ne sais pas si ça se fait mais si tu penses que c'est ce qu'il faut faire alors vas-y"*[1].

Je remercie également Rachid Alami, directeur de cette thèse, pour l'ensemble des idées qui ont nourri cette thèse, toujours précises, affinées et enrichies, et donnant un véritable cap à suivre et évitant ainsi de se perdre dans des contributions trop vagues. Merci pour ta disponibilité, tes relectures toujours attentives et anticipées et ton sens de la phrase pour trouver les bons titres.

Mes remerciements vont également à Micheal Beetz et Paulo Menezes pour avoir accepté d'être rapporteurs de cette thèse, ainsi qu'aux autres membres du jury, Kerstin Fischer et Simon Lacroix.

Je me dois bien évidemment de remercier une seconde fois Simon, qui en plus de sa magnifique prestation de président de jury et de ses questions probabilistes a également été un camarade de pause toujours de bon conseil, grâce à qui j'ai pu mieux comprendre le monde de la recherche.

En restant sur la thématique des pauses probabilistes, je souhaite remercier Arthur pour son regard critique et bienveillant sur mon travail.

Outre les collègues quelque peu formels disons, une thèse est également accompagnée de compagnons d'infortune parmi lesquels j'ai su trouver mes mousquetaires, pour affronter ensemble le cardinal de Richelieu et ainsi créer une super démo. Sans vous, ces années n'auraient sûrement pas été aussi agréables et cette thèse n'aurait pas été aussi riche. Je pourrais, j'en suis sûr, noircir quelques pages pour vous remercier mais promis je vais faire court.

Guilhem, qui eût cru que tu l'aies fait, peut-être pas toi, mais nous oui. Malgré ton pessimisme constant, tu as été une véritable source d'inspiration pour moi,

[1] Nominated for the best paper.

avec une vision de la robotique et de la HRI toujours juste, réfléchi, critique et réaliste. Toujours prêt à tout remettre en question, j'ai pu mieux construire ma vision, toujours prêt à débattre, j'ai pu étayer mes arguments, toujours prêt à tout recommencer à zéro, j'ai trouvé un camarade de code. Merci pour toutes ces longues discussions, ces débats, ces engueulades et ces bières.

Amandine, j'espère qu'après ces quelques années je te fais enfin moins peur. Véritable juke-box ambulant, tu as su transformer de longues soirées d'intégration laborieuses en moments agréables où la bonne humeur était le maître-mot. Camarade de confinement non respecté, merci pour ces longues discussions, ces midis de jeux et ces gâteaux.. Ho ces gâteaux... Merci également pour tous ces développements sans lesquels nos travaux seraient restés théoriques et pour tes exigences nous ayant poussés à aller toujours plus loin.

Kathleen, petite folle dont je n'oublierai pas la rencontre quelque peu incongrue pour quelqu'un devant nous aider à savoir comment interagir avec un humain, j'espère qu'un robot ne fera jamais comme toi. Un grand caractère dans une si petite personne. Merci pour toutes ces pauses de complaintes, cette éthique sans failles qu'on a eu tant de mal à respecter. Merci également pour ton partage de connaissance passionné qui a donné une autre envergure a cette thèse. J'en suis sûr, ce n'est que le début d'une collaboration.

Merci ensuite à tous ceux que j'ai obligés à faire tant de pauses café, matin, midi et soir. Je me doute à quel point cela a dû être dur pour vous. Yannick, fidèle ingénieur ayant un sens du mouvement tout personnel. Amélie, dont le rire illuminait le bâtiment tel un feu d'artifice. Jules, programmeur talentueux à qui le rouge allait si bien. Rafa, roi d'Espagne ne craignant pas la coupe. Phani, danseuse étoile nous ayant fait tourner la tête. Philippe, dont l'âge n'a d'égale que l'humour. Jérémy, athlète friand de punch ing ball. Ilinka, loup garou par défaut. Alexandre, peintre attentif. Léa, pile sur pâtes. David, la chaleur dégagée par son réseau de neurones a eu raison de ce qu'il se trouvait dessus. Dario et Margot, camarades de week-ends. Merci aussi à Alejandro, Christophe, Ellon, Anthony, Antoine, Jerôme, Martin, Tanguy, Victor, Jean-Hugues, Élise, Vivien, Gianluca, Andréa, Pierre, Florian un et deux, Idriss, Paul et tous ceux que j'ai pu croiser durant ces années.

Merci également à tous les permanents du bâtiment H toujours prêts à aider et à discuter de tout et de science. Christelle, encadrante de la première heure. Matthieu, dompteur de robot. Anthony, plus efficace que marraine la bonne fée avec une baguette (de mocap). Felix, actionnaire principal du CAF40. Merci également à Daniel, Nic, Juan, Corrine et Patrick.

Je souhaite également remercier tous les enseignants que j'ai pu avoir durant mon parcours scolaire et qui, alors que je voulais faire des études courtes, ont su me donner le goût du travail. Pour n'en citer que certains, merci à Damien, Gaelle, Bruno, Bruno, Alexandre, Pierre-Emmanuel, Claude, Didier, Christophe et tous les autres.

Merci à l'ensemble de l'équipe du fablab. Vous avez été ma petite bulle de grand n'importe quoi durant ces années, toujours partants pour une bière, une raclette,

un barbecue, éclater des composants, faire des arcs électriques, imprimer des choses utiles, casser des machines, faire un ordinateur dans une boîte à pizza et j'en passe. Alors merci pour votre passion, vos idées et vos rires.

D'un point de vue plus personnel, je souhaiterais vivement remercier Julien, toujours présent depuis bien des années. Tu es celui que mes collègues connaissent sous la casquette d'ingénieur de recherche personnel. Merci pour toutes tes relectures de code, tes conseils avisés et m'avoir poussé à aller toujours plus loin, à ne pas me reposer sur mes acquis. Je pense que bien des fois, tu as dû te questionner sur l'utilité de mes logiciels, mais qu'importe, tu voulais les améliorer. Merci également pour ton amitié et tous ces projets durant toutes ces années.

Merci enfin à ma famille et tout particulièrement à ma mère et mes sœurs. Je sais que vous n'avez pas toujours compris ce que je faisais et l'importance que ça avait pour moi, mais vous m'avez toujours soutenu et vous avez su accepter mes absences sûrement trop nombreuses. Alors certes, il semblerait que je ne sois pas encore au bout du chemin, mais soyez sûres de la place que vous avez dans mon cœur.

Contents

1 Introduction **1**
- 1.1 A prototypical scenario . . . 1
- 1.2 Interacting with a robot: What can we expect ? . . . 4
- 1.3 Knowledge organization . . . 7
- 1.4 Contributions . . . 10
  - 1.4.1 Knowledge management . . . 10
  - 1.4.2 Knowledge exploitation . . . 11
  - 1.4.3 Cognitive architectures . . . 11
- 1.5 A reader's guide . . . 12

2 Ontologenius: A semantic memory for HRI **15**
- 2.1 Design and features . . . 16
  - 2.1.1 Why an ontology? . . . 16
  - 2.1.2 Desired features . . . 19
  - 2.1.3 Ontology formalism . . . 21
- 2.2 Architecture . . . 28
  - 2.2.1 Permanent versus temporary data structure . . . 28
  - 2.2.2 Reasoning to enrich the knowledge . . . 31
  - 2.2.3 Concepts name in natural language . . . 32
- 2.3 Managing partners' estimated knowledge . . . 32
  - 2.3.1 Ontologenius multi-instances principle . . . 32
  - 2.3.2 Catching knowledge at a given moment . . . 33
  - 2.3.3 Exploring several possible mental states at once . . . 34
- 2.4 Using Ontologenius in robotic applications . . . 35
  - 2.4.1 Updating the knowledge base . . . 35
  - 2.4.2 Retrieving knowledge . . . 36
  - 2.4.3 Interfacing with Ontologenius . . . 38
- 2.5 Computational performance evaluation . . . 40
  - 2.5.1 Comparing with Knowrob . . . 40
  - 2.5.2 Comparing with ORO . . . 44
  - 2.5.3 Additional tests . . . 47

3 Elaborating a route for a human partner based on semantic knowledge **51**
- 3.1 Introduction . . . 52
- 3.2 Related work . . . 53
- 3.3 The Semantic Spatial Representation . . . 55
  - 3.3.1 The SSR classes . . . 56
  - 3.3.2 The SSR properties . . . 57
  - 3.3.3 An example of description . . . 59

3.4 Finding routes: A two-level search 60
3.4.1 The region-level: Trim down the search 61
3.4.2 The place-level: Refine the search 63
3.5 Generating an explanation in natural language 66
3.5.1 Reconstructing the paths 66
3.5.2 The robot putting itself in your shoes 68
3.5.3 A pattern-based generation 69
3.6 Experiment in the mockup and the real environment 70

4 Ontology-based Referring Expression Generation 73
4.1 Introduction 74
4.2 Related work 76
4.3 Define the REG problem 80
4.3.1 The knowledge representation 80
4.3.2 Contextualization and restriction for situated REG 82
4.3.3 Expected solution: structure and validity criteria 84
4.4 Uniform Cost Search REG 86
4.4.1 Formalisation as a search problem 86
4.4.2 Algorithm choice 88
4.4.3 Algorithm presentation 88
4.4.4 Replanning and explaining failures 91
4.5 Results 92
4.5.1 Solution analysis: The pen in the cup 92
4.5.2 Scaling up: The three-room apartment 93
4.5.3 Comparisons with other state-of-the-art algorithms 95
4.6 Integration on a robotic system 97

5 Estimating communication feasibility and cost at task planning 101
5.1 Introduction 102
5.2 Related work 104
5.3 The involved components 106
5.3.1 The Hierarchical Task Planner 107
5.3.2 The Semantic Knowledge Base 108
5.4 Integrating planners 108
5.4.1 The representation of the communication action 110
5.4.2 Maintaining the right knowledge base, at the right time . . . 111
5.4.3 Reducing the number of updates 112
5.5 Results 113
5.5.1 Prevent execution dead-end 113
5.5.2 Reduce the overall communication complexity 115
5.5.3 Compare with other communication means 115
5.6 Integration in a robotic system 116

6 Extending the REG with knowledge about past activities 119
6.1 Introduction . . . 120
6.2 Related work . . . 122
6.2.1 Interaction based Referring expression . . . 122
6.2.2 HTN-based tasks representation in ontology . . . 123
6.3 Structuring and gathering the knowledge . . . 124
6.3.1 The three knowledge representations . . . 124
6.3.2 The knowledge gathering scheme . . . 126
6.3.3 Building the ontology . . . 128
6.4 REG algorithm modifications . . . 132
6.5 Results . . . 134
6.5.1 One execution trace for five referring expressions . . . 134
6.5.2 Impact of the extension on the performances . . . 138
6.6 Discussion . . . 139

7 Beyond binary relations in the REG 141
7.1 Introduction . . . 142
7.2 Related work . . . 143
7.3 Through the use of compound relations . . . 145
7.3.1 Defining a compound relation . . . 146
7.3.2 A lightweight representation of the verbal link . . . 147
7.3.3 A strategy to explore compound relations . . . 148
7.4 REG with compound relations . . . 152
7.4.1 Exploring the compound relations . . . 152
7.4.2 Determining a referring expression validity . . . 154
7.4.3 From tree to radix tree . . . 155
7.5 Results . . . 155
7.5.1 The actor playing James Bond . . . 155
7.5.2 The description of past activities as compound relations . . . 157
7.5.3 Assessing compound relations impact on performance . . . 159

8 A robot in the mall: The MuMMER project 161
8.1 Introduction . . . 162
8.2 Related work . . . 162
8.3 Learning from exploratory studies . . . 163
8.4 The deliberative architecture . . . 164
8.4.1 Environment representation . . . 165
8.4.2 Perceiving the partner . . . 168
8.4.3 Managing the robot resources . . . 168
8.4.4 Describing the route to follow . . . 170
8.4.5 Planning a shared visual perspective . . . 170
8.4.6 Navigating close to human . . . 173
8.4.7 Executing and controlling the task . . . 173
8.5 Embody architecture in a physical robot . . . 174

8.5.1 Pepper in Ideapark . . . 174
8.5.2 Pepper "in the wild" . . . 175

9 The Director Task: Assessing cognitive architectures 177
9.1 Introduction . . . 178
9.2 From psychology to Human-Robot Interaction . . . 180
9.2.1 The original task . . . 180
9.2.2 The Director Task setup . . . 182
9.2.3 The adapted task . . . 184
9.2.4 Additional rules for the first implementation . . . 184
9.2.5 Additional abilities . . . 186
9.3 The cognitive architecture . . . 187
9.3.1 Storing and reasoning on symbolic statements . . . 188
9.3.2 Assessing the world: from geometry to symbols . . . 188
9.3.3 Planning with symbolic facts . . . 191
9.3.4 Managing the interaction . . . 191
9.3.5 Speaking and understanding . . . 192
9.4 Experiments . . . 194
9.4.1 PR2 as the director . . . 195
9.4.2 PR2 as the receiver . . . 196
9.5 Open challenges for the community . . . 198
9.5.1 Challenges to take up . . . 199
9.5.2 User studies to perform . . . 201

Conclusion: represent, store, explore, communicate 203

A Route description supplementary material 209
A.1 Routes verbalization patterns . . . 209
A.2 Routes search solutions . . . 212

B REG suplementary material 213
B.1 Referring Expression Generation solutions . . . 213
B.2 Compare REG with other communication means . . . 215
B.3 Referring Expression Generation comparisons . . . 216

C Résumé en Français 219

Bibliography 245

Acronyms

CE Compound Entity. 146, 147, 148, 152, 153, 154, 157

CR Compound Relation. 143, 145, 146, 147, 148, 149, 151, 152, 154, 155, 157, 158, 159, 160, 205

CT Compound Tree. 152, 153, 154, 155

GBA Graph-Based Algorithm. 77, 96, 97

HATP Hierarchical Agent-based Task Planner. 107, 108, 110, 111, 116, 121, 125, 126, 128, 131, 191, 204, 205, 215

HET Hierarchical Execution Trace. 122, 123, 124, 128, 205

HRI Human Robot Interaction. 7, 10, 11, 12, 13, 15, 19, 32, 40, 44, 52, 71, 72, 102, 106, 120, 123, 126, 138, 141, 160, 178, 182, 184, 185, 203, 204, 205, 206, 265

HTN Hierarchical Task Network. 107, 109, 110, 119, 121, 122, 123, 124, 125, 127, 128, 129, 130, 131, 134, 139

IA Incremental Algorithm. 77, 78, 207

KB Knowledge Base. 10, 11, 12, 17, 18, 19, 20, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 40, 41, 42, 43, 44, 47, 48, 49, 101, 106, 107, 127, 134, 159, 160, 161, 188, 203, 204, 207

MuMMER MultiModal Mall Entertainment Robot. 11, 51, 72, 161, 162, 170, 205

RE Referring Expression. 11, 13, 76, 77, 78, 79, 80, 83, 84, 85, 86, 87, 88, 90, 91, 92, 93, 95, 96, 97, 99, 102, 103, 106, 110, 113, 115, 119, 121, 122, 128, 133, 134, 135, 136, 137, 138, 147, 148, 152, 153, 154, 155, 158, 159, 160, 205, 207

REG Referring Expression Generation. 11, 12, 13, 73, 75, 76, 77, 78, 79, 80, 82, 83, 84, 86, 87, 88, 89, 90, 92, 95, 96, 98, 100, 101, 102, 105, 106, 107, 108, 109, 110, 111, 112, 114, 117, 119, 120, 121, 122, 124, 127, 128, 132, 134, 139, 143, 146, 147, 148, 149, 152, 153, 154, 155, 178, 191, 192, 193, 198, 204, 205, 206, 207, 213, 215, 216, 265

SSR Semantic Spatial Representation. 11, 53, 55, 56, 58, 59, 60, 67, 70, 72, 167, 175, 203, 204

UCS Uniform-Cost Seach. 88, 89, 97, 100, 119, 132, 133, 134, 153, 213

Chapter 1

Introduction

Contents

1.1 A prototypical scenario 1
1.2 Interacting with a robot: What can we expect ? 4
1.3 Knowledge organization 7
1.4 Contributions 10
1.4.1 Knowledge management 10
1.4.2 Knowledge exploitation 11
1.4.3 Cognitive architectures 11
1.5 A reader's guide 12

In order to introduce this thesis and to provide a global picture of its content, we first narrate a short story about a robot in a store. This story will then be used to identify some of the mains abilities a robot needs in order to interact with humans with a focus on the knowledge it needs. While this thesis is from a roboticist point of view, working on interaction naturally leads to the study of cognitive psychology. From this field, we want to present an overview of the knowledge organisation models that have had an impact on the field, at the point to be now used for robotic research.

1.1 A prototypical scenario

A company selling cameras have recently invested in a robot to support its employees in its stores. Their goal with these robots is to help the employees during their daily tasks in the stores. The robots can prepare orders, put items on the shelves, cash customers, or advise them.

Max is one of these robots. It is a PR2 equipped with a head, two arms, and a mobile base with wheels. It is 9 a.m. and the robotic company having developed Max, power it on for the first time in the store. The robot starts navigating in the store, looking at all the products on the shelves. Liam, the human employee is counting the cash register when a customer enters the store. This customer is Tony. He looks to some cameras displayed on the shelves, going from one shelf to another.

Max, seeing Tony looking at all the cameras and Liam occupied to count the cash register, decides to go to see whether the customer needs help.

Max - "Hi, I am Max. Have you found a camera that interests you or do you need advice?"
Tony - "Hi, I don't really know anything about cameras. I am planning to go on a trip in a month and I would like a camera to take animal pictures during the trip."

For an amateur, Max chooses to present to Tony some automatic models. Moreover, since it is for a trip, it advises Tony to prefer a small camera. Looking at the prices, the customer explains that he did not want to spend more than 500 euros. Max thus selects three options for him:

Tony - "You have this one at 350, the small black one here in front of you at 475, and on the other shelf there the small brown one near to the screen costs 230."

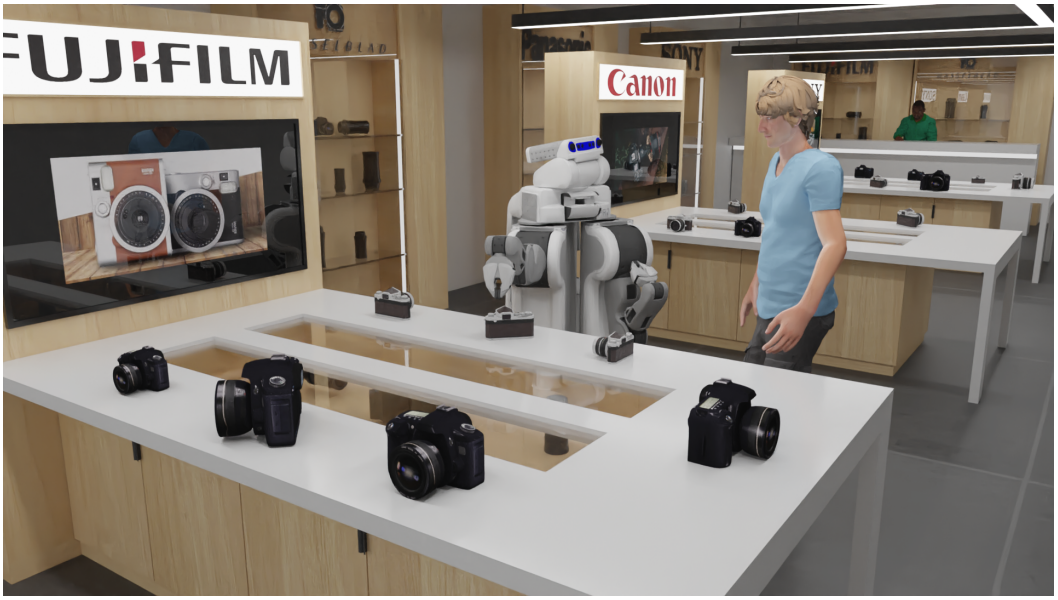

Figure 1.1: A PR2 robot, as an employee of a camera store, advises a customer.

Explaining that, Max points to the cameras. They continue to discuss when Tony's phone rings. He has to go to join his wife, in another store in the mall. Not knowing where the other store is located, he asks:

Tony - "I am sorry I have to go. I will come back in the week. I have to go to a store selling video games but I do not remember the name."
Max - "There is only one store selling video games in this mall, it is Game-ania."
Tony - "Do you know how to reach it from there ?"
Max - "For sure"

Max moves next to the entrance, followed by Tony. It raises one arm pointing to the aisle and said:

Max - "Go down this aisle then turn left straight after the salad bar. After that Game-ania will be on your right when we walk."

Tony leaves and comes back the morning after. Max recognizes him and moves towards him. It asks Tony if he has easily found the video game shop then recalls the camera identified the day before.

Max - "Yesterday we stop on two Fujifilm cameras and a Canon for your trip."

They continue to discuss and finally Tony selects the camera at 350 euros. Following Max's advice, he takes a memory card and a second battery to have enough storage and power during his trip. In addition, he takes a zoom lens to take pictures of animals at long range. Unfortunately, the desired camera is not available at the moment. The last one in the store is the demonstration one. Max proposes to Tony to order the camera. The client agrees and leaves.

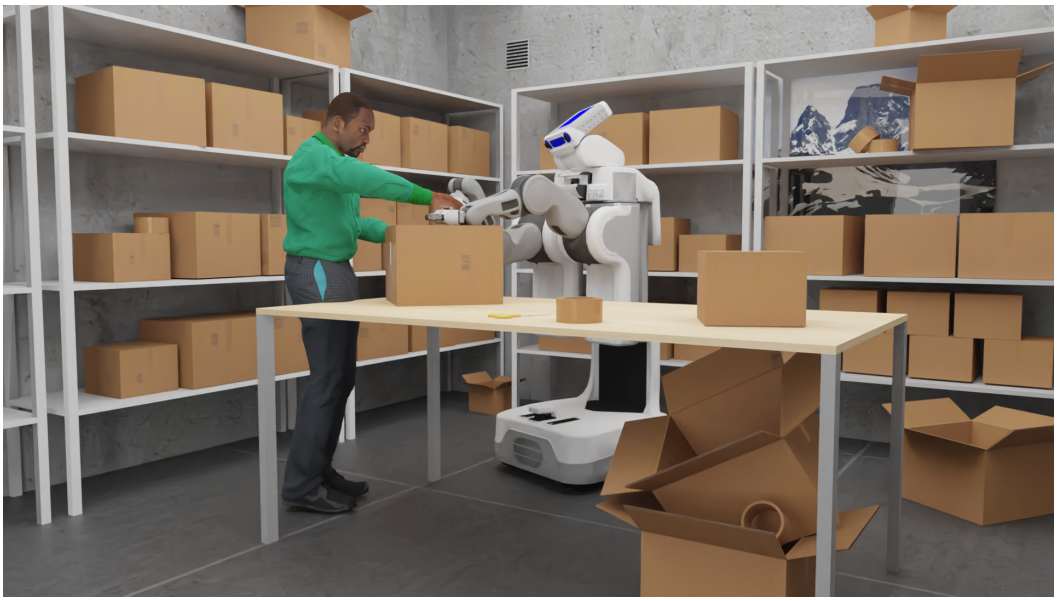

Figure 1.2: A PR2 robot and a human employee collaborating to close a box.

A few days later, before opening, several boxes are delivered to the store. Liam, the human employee, and Max have to open them, fill stocks and prepare Tony's order. This is the first time since Max arrival they have to do it. They both go to the backroom in order to do this task together. There are two boxes. Liam starts opening one and so Max starts opening the other. Max informs Liam about the camera which has been ordered and is planned to be in the delivery.

Max - "Tell me if there is a X-T100, a customer orders one."

Liam finds the camera. The robot explains to him that it also needs an SD card of 32Go and a lens XC 15-45mm. It informs the human that the SD cards are too small for it to grasp them. It thus proposes to Liam to take some of the cameras to put on the shelves and to bring back the card and the lens at the same time. During this time, Max gets a box, puts the camera and the battery inside. When Liam comes back, he puts the two other items in the same box. Then, Max maintains the box close while Liam tapes it. When finished, they both take the last items to put on the shelves and go to the main room.

The afternoon of the same day, while Max is cashing in a customer, Tony enters the store. Seeing him, Max requests Liam:

Max - "Can you bring the order we prepared this morning? The client is the one just entering, with the blue T-shirt."

Liam goes to the backroom and brings back the correct box. Max charges the customer who finally gets his camera on time.

1.2 Interacting with a robot: What can we expect ?

Even if the scenario of the robot in a store is not intended to be entirely implemented, it allows us to identify what is needed in terms of knowledge representation, for a robot interacting with humans. First of all, we can draw a rough partition of the needed knowledge:

1. **common-sense knowledge**. It is the knowledge not necessarily related to the current situation but required to understand it. In the scenario, this is not because the robot is in a camera store that it knows the concept of a camera, this concept is more general. Thanks to this common ground it is able to understand the concept of video games, trips, boxes, or animals in our scenario.

2. **knowledge of the environment**, grounded in the space. The robot does not only need to know how it can move in its environment, it has to identify the elements composing it, link them to the common ground, and refine them. When Max is powered on for the first time, it goes around the environment to analyse it. Seeing an object, Max identifies it as being a camera. With additional cues, it can refine its knowledge through the model of this camera and its characteristics. This object is on a support, this is a shelf, dedicated to a specific brand, and so on.

3. **knowledge of the activities**, grounded in the space and the time. More than how to perform a given task, we are here interested in how it has been achieved. By whom? With whom? Where? When? On which entities? In

our scenario, Liam has brought back an SD card from the main room, during this time, Max has prepared a box in the backroom, and these two tasks have been made to prepare, together, Tony's command.

Through this rough partition, we can identify three types of knowledge. The general ones, the knowledge related to space, and those related to time. Where the first can be seen as partially static, all have to be dynamic. The robot should be **able to gather knowledge**, update it, and create links in between.

At this stage, one can ask: where is the interaction in it? Even if this knowledge is mandatory for a robot interacting with a human, it holds also for a robot acting alone in an environment.

Speaking about interaction, the first use of the knowledge we can think about is communication, meaning sharing information. Consequently, an important property of the robot's knowledge is that a part of it has to be **narrative-enabled**. The robot should be able to communicate its knowledge. As humans, we can naturally think about verbal communication, for example when it explains the characteristics of the cameras. It can however also be through gestures, like pointing or other dietic gestures. When the robot says "turn right" while explaining a route to follow, the language can be accompanied by a hand gesture, turning it on the right. Moreover, for a robot equipped with a screen, we could also imagine communication through texts or images. We claim that only a part of its knowledge has to be narrative-enabled since another part can be dedicated to its internal functioning, from a pure engineering point of view.

When we qualify a piece of knowledge to be narrative-enabled, we can restrict ourselves to communication from the robot and to the human. However, if we consider the robot able to express knowledge, we can assume it to be able to understand it. In this way, in our conception of narrative-enabled knowledge, the robot should be able to formulate sentences and understand humans speech, based on its knowledge. Moreover, to understand its partner the robot not simply has to match the expressed concepts to its knowledge. Since some communications are underspecified, like when the client requests a camera for a trip, the robot can make use of **inference** to better understand the human. In our scenario, the client explains he is not an expert so the robot advises for an automatic camera. Since he goes on a trip, more storage and battery could be necessary. Since he wants to take pictures of animals, a zoom lens could be useful. All these pieces of information are not explicit to the communication, but thanks to inference on the knowledge can be understood by the robot. Finally, to fully understand its partner, the robot has to consider the **context** of the entire interaction. When Tony said he wants to acquire a camera at 350 euros, it seems trivial that it is the one at this price among the three previously identified.

Moving toward the knowledge about activities, we have to speak about **plans**. Keeping it simple for the moment, we can consider a plan as a succession of actions allowing an agent to achieve a task. In this way, a cooking recipe is a kind of plan allowing to achieve the task to prepare a dish. Through the narrative-enabled

characteristic of knowledge, we thus want a robot to be able to express a plan. In our scenario, Max asked Liam to bring back an SD card and a lens, it thus expresses a part of a plan. However, it could also have explained to Liam to go in the main room, to move toward the second shelf, to open the rightmost door, to take the SD card in a blue package, and to come back in the backroom. This explanation expresses the same thing but goes deeper into the details, taking a lower level of **abstraction**. Here we see that the robot has to be able to express a plan at various levels of abstraction. For example, if the robot says "Let us prepare the order" it expresses the global task.

Such an explanation of a plan to be carried out highlights a number of key elements regarding the robot's need for knowledge. Here we introduce two of them:

- If the robot can express the same plan with different granularity, how does it choose the right one? To answer this question, we have to introduce the self-and-other distinction. To know which information to share, a robot has to **estimate its partner knowledge**, that it is the common-sense knowledge as well as the knowledge about the environment and the activities. It is on the basis of this estimation that the robot can know how to communicate. In our scenario, the robot interacts with the human employee to prepare the order. Since Liam was in the store before the robot, it could estimate that the employee knows where the SD cards are. Communicating this low-level information would thus be useless and inefficient for the task. The human employee would be a new one, the robot would have interacted in a different way. This estimation of the others knowledge can be found somewhere else in the scenario. When Max refers to a specific camera to the client, it describes it in relation to its attributes and location. To the difference, to refer to the same entities to Liam, Max uses the precise model name of the camera. It thus estimates that the client does not know the model's name but that the employee does.

- To allow the robot to communicate at the right level of abstraction, estimating the other knowledge allows the detection of **belief divergence**. It appears when the robot estimates that the other does not have a piece of knowledge or still considers a piece of knowledge that no more holds. Detecting such a divergence can be used to prevent errors for example. In our scenario, if the robot has taken the last zoom lens on a given shelf and estimates that the employee is not aware of this information, it can detect a belief divergence. Consequently, rather than just asking for a zoom lens, it can inform the other that there is no more lens at this place. The employee will not have to search at the original place and will be able to directly go to another place where there is still some zoom lens.

Continuing to explore the knowledge implies by the realisation of a plan, we can move toward the elaboration of the plan, the planning of the task. By achieving the task together, the human and the robot are performing a **joint-task**, meaning

having a joint-goal and collaborating to achieve it. To collaborate the robot elaborates a **plan considering the human**. It can thus propose to him to achieve a part of the task. To know how to dispatch the actions, the robot needs knowledge about **human abilities** as well as its own abilities. In the scenario, the SD card is assigned to the human because the robot is aware that it can not grasp it. If the object to bring back was graspable by it, maybe it would have done it by considering it as uncomfortable for the human to walk too much. Underlying, this ability to plan for himself and others also requires a projection into future situations and thus a **representation of possible worlds state**, like what is done in task planning.

Finally, the human is not an agent like the robot, he can not be controlled. Even if the robot plans by considering him, the human can act freely. The robot thus has to **monitor, interpret, and ground human actions**. From there and with regard to the plan and thus the joint-task, it can react and adapt. In our scenario, when Liam starts opening one of the two boxes, the robot adapts and takes the other one, even if it planned a different course of action. In this situation inverting does not raise any issue, they can thus continue.

In this section, we have identified some of the key elements need and use about knowledge for Human Robot Interaction (HRI). Even if we were not able to tackle all of them in this thesis, they give the context in which the presented contributions have been thought and they aim to be integrated. In this section, we have often use the term "knowledge". However under this general term can be found a number of concepts related to memory and how, as humans, we are able to remember things. Before moving to the contributions of this thesis, we propose in the next section an overview of some model from the cognitive psychology, allowing to better understand how we represent our knowledge.

1.3 Knowledge organization: Drawing inspiration from cognitive psychology

Even if our goal in robotics is not to create a copy of the human, either in terms of body shape or cognition, drawing inspiration from it is nevertheless important. In the same way that roboticists take inspiration from the human body to create robots able to act in a world created by and for the human, the field of cognitive robotics takes inspiration from the human cognition to create robots able to interact with humans. While we do not aim to imitate human cognition, we think that a robot must be endowed with some similar capabilities if we want them to interact with us efficiently and in an acceptable way.

Regarding the knowledge representation, the first experimental study has been realized in 1885 by Ebbinghaus [Ebbinghaus 1885]. Since then, the definition of memory as the capacity to encode, store, and retrieve knowledge [Roediger 1996] has been widely accepted. At the same time, the word "memory" has become a generic term suggesting a unique system. However, the human memory can rather be seen as several sub-systems that differ from their storage duration, storage capacity,

and the level of consciousness necessary for information retrieval. In the rest of the section, we present some memory models, presented in a non-chronological way, and focussing on what is called long-term memory. During all this section, it is important to keep in mind that we only present models aiming to understand human cognition with a focus on knowledge management. No formal truth is stated here given that there is no consensus in the field. The presented models and terms will allow us to better understand the existing robotic cognitive architectures and give inspiration for the design and structuring of the components developed in this thesis.

The primary division of the memory is done concerning the storage duration and capacity. From there has been defined the **short-term memory** (STM) and the **long-term memory** (LTM) [Atkinson 1966]. Short-term memory is characterized by its small capacity and its capability to retrieve information that has just been seen. It is often stated that it is near to twenty seconds and seven items (or chunks)[Miller 1956]. Long-term memory, on the opposite, refers to any situation where we use information that has not just been seen. It is often stated that it has an infinite capacity and duration. It is this latter that allows the acquisition of new knowledge and the retrieval of information acquired a long time ago.

Over the years, what was called short-term memory has become the **working memory** (WM) [Baddeley 1986]. This change has been made to add a notion of knowledge manipulation. Instead of focusing on the only temporal aspect, it reflects its functional aspect. It is thus a system that retains information for the time necessary for its use by other cognitive functions. The working memory and the long-term memories are two independent but related systems. For information to be stored in long-term memory, we think that it has to pass by the working memory.

For the structure of the long-term memory, a first dichotomy is proposed by Graf and Schacter [Graf 1985] with the **implicit** and **explicit** memory. It reflects the way knowledge can be retrieved. Knowledge from explicit memory can be retrieved consciously and voluntarily. On the opposite, knowledge from implicit memory is retrieved in situations in which our behaviors are influenced by an experience. This is the case when one pours water into a glass. We do not need to retrieve explicitly how to perform this task. It is our experience that influences how we do it.

Another dichotomy has been proposed by Squire and Cohen [Squire 1982] with the **procedural** and **declarative** memory. Procedural memory is the system allowing us to retain knowledge about our cognitive, motor, or perceptual skills. It is the memory of the know-how. A specificity of this memory is that it is difficult to verbalize the knowledge it stores and we use them unconsciously. The declarative memory stores representations of facts, events, general knowledge, and memories of past events. This knowledge can be retrieved consciously and we can speak about them. Taking the example of the code of your credit card. Initially, this knowledge is stored in the declarative memory. When you need it, you can easily remember it and say it. The more you use it and it slowly becomes an automatism. It became hard to remember it consciously while you use it every day and if you need to say

it you need to type it on a virtual keyboard to remember it. It has slowly moved into your procedural memory. We see that both dichotomies are almost equivalent. The declarative memory of Squire is related to the implicit memory of Graf, and the procedural one is related to the implicit memory.

Going deeper into the declarative memory, Tulving has proposed a dichotomy between **episodic** and **semantic** memory [Tulving 1995]. The episodic memory retains knowledge related to past experienced events, which are specific to our individual experience, and localized both in space and time. The semantic memory is more general and retains knowledge that we accumulate all along with our life concerning our environment. It can be seen as an encyclopedic memory independent of the acquisition context. While the episodic memory is related to "remembering", the semantic one is related to "knowing". Taking as an example your visit to Toulouse [1], this visit is encoded in the episodic memory while the knowledge that Toulouse is a city in France is encoded in the semantic one.

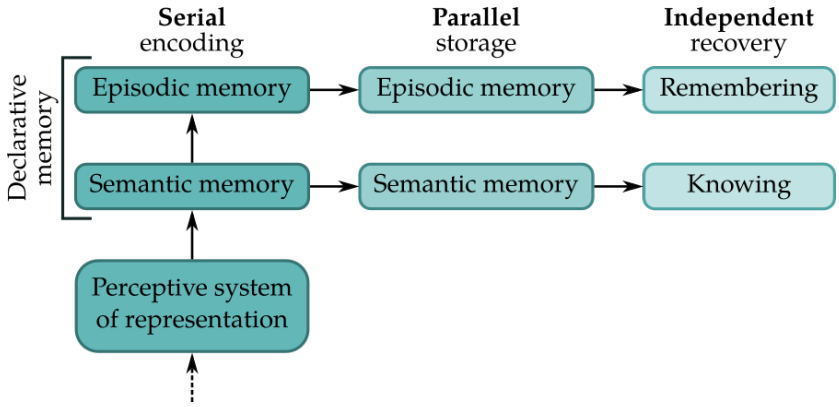

Figure 1.3: Relations between episodic memory and semantic memory according to the SPI (Serial, Parallel, and Independent) model of Tulving (1995).

With the previous example, we saw that even if these two sub-systems of the declarative memory are independent, they are therefore in interaction with each other as we can apply a semantic treatment on the episodic memory. To better understand their relation, Tulving has proposed the Serial, Parallel, and Independent model (SPI). As illustrated in Figure 1.3, the encoding of information is serial (S) passing first by the semantic memory, then in the episodic one. The storage in both memories is performed in parallel (P). Finally, the information recovery is independent between the two memories. As we explain at the beginning of this section, the presented theory can not be proven and must therefore be taken as [Tulving 1995] "an explicit starting point for a more systematic pursuit of what is clearly the next problem that needs to be tackled".

For this thesis, and robotic in general, these hypotheses about the knowledge organisation for humans allows a better understanding of the kind of knowledge we

[1] If you never visit it, you can take another city you have visited but you should plan to visit it anyway.

have to manage. We have identified three types: semantic, episodic, and procedural. Wanting to insert a new piece of knowledge, we can thus identify the types it is of and thus where to store it.

1.4 Contributions

This section summarizes the main contributions of this thesis. They are organised around three complementary topics: knowledge management, knowledge exploitation, and cognitive architectures. All have been thought in the context of Human Robot Interaction, meaning what a robot needs to take the human into account and interact with him.

1.4.1 Knowledge management

The starting point of this thesis is the need to store and maintain both the robot's and humans' estimated knowledge. We focused our efforts on semantic knowledge, using an ontology to represent it. The core contribution of this thesis is thus a software component to manage ontology instances. Each instance represents an agent knowledge, with the robot one being its "ground truth"[2] and its partners' ones being an estimation of their knowledge[3]. The ontology is used to represent the common-sense knowledge, the knowledge about the environment, knowledge about activities, as well as knowledge for the proper functioning of the robot.

The resulting software is called Ontologenius. It is a lightweight and open-source software developed in C++ and working as a server within a robotic architecture. Any component of the architecture can thus access the knowledge it maintains through the ROS middleware, ensuring uniformity of the knowledge among the entire architecture. It comes with extensive documentation, debugging tools, and an API in C++ and python.

Ontologenius supports dynamic updates of the Knowledge Bases it maintains and keeps them consistent at any instant thanks to reasoners. At the date, five main reasoners are available in the form of plugins, each dedicated to a specific axiom. Others can thus be added depending on the need. They can be activated or deactivated at runtime and propose different reasoning managements. We can for example choose to run then upon the query, at an update or periodically. Ontologenius is able to make the difference between a stated fact and a deduced one.

At the difference of other ontology management software, Ontologenius comes with more than 60 low-level inbuilt parametrizable queries, allowing a fast and precise exploration of the knowledge. These queries have been designed in such a way to be called from search algorithms to create higher-level cognitive processes. In addition, it proposes a SPARQL interface.

Ontologenius has also been designed to be used for task planning applications by representing several possible world states in a single instance, switching from

[2]This is the knowledge about what has been directly perceived by the robot.

[3]Either estimated to be perceived by the human or explicitly provided by the programmer.

one to another.

It was made from scratch and thought like a sandbox, based on ontology standards but sometimes allowing to deviate from them to better explore the possibilities and never be limited.

During this thesis, a software to manage episodic knowledge has also been developed and linked with Ontologenius. It is called Mementar. Due to its early stage, it will not be presented in-depth but used in the final contribution of this thesis.

1.4.2 Knowledge exploitation

On the basis of the semantic knowledge management software Ontologenius, several knowledge exploitation contributions have been developed. We can group them all into the topic of spatial referring.

The first tackles the route description task. Describing an indoor environment using an ontology, we have proposed two complementary algorithms able to find several routes leading to a target place. To represent an indoor environment with an ontology, we have proposed a piece of ontology to describe the topology, called the Semantic Spatial Representation. It allows the description of the static elements of the environment, perceptible by humans, as well as elements necessary for the verbal description of a route. We have then developed an algorithm to verbalise routes, respecting good practices.

The second tackles the Referring Expression Generation task. The goal is to search for the knowledge to communicate to a hearer allowing him to identify the target entity without ambiguity, in a given context. The resulting algorithm has been shown as being the most efficient to date, even if the input KB is not dedicated to the task. This contribution has been an important source of inspiration and has lead to three other related contributions. We have linked this algorithm to a symbolic task planner to evaluate communication feasibility and cost during the planning process. After this link with a planner, we tried to exploit the shared Human-Robot experience as a new kind of knowledge usable to generate Referring Expression. The latter contribution leads us to a proposal of task representation in an ontology. To move toward a more generic method to generate Referring Expression, we finally start a preliminary work aiming to support n-ary relations.

1.4.3 Cognitive architectures

The last contributions of this thesis are two cognitive architectures embodied on a Pepper and a PR2 robotic platforms. The first architecture has been developed in the context of the MultiModal Mall Entertainment Robot project. It makes use of the route description to guide customers in a mall. The second has been developed later at LAAS-RIS to experiment recent developments of the team. This architecture makes use of the contributions around the Referring Expression Generation. It has been tested in a task inspired by cognitive psychology and adapted to the Human Robot Interaction, called the Director Task.

Both architectures use Ontologenius to manage the robot and humans semantic KBs. Since both architectures were developed two years apart, we will see in this thesis how the semantic Knowledge Base has become a central element of our cognitive architecture for Human Robot Interaction.

1.5 A reader's guide

Thesis organisation

This thesis is partly based on published or submitted work. Some of the key contributions are more detailed than the paper version, while work involving several students is rather summarized to provide an overview of an integrated contribution. This thesis is organised in an almost progressive way. Such an organisation aims at following step by step the way in which each contribution has been thought about regarding the previous ones. For example, from Chapter 5 to Chapter 7 an architecture is built proposing each time the integration of new components and new abilities.

This thesis is organized into three parts: knowledge management, knowledge exploitation through referring applications, and finally the integration into robotic architectures. The knowledge management is presented in Chapter 2 with the software Ontologenius. This software is then the core of all the other contributions. The knowledge exploitation has been studied with referring tasks. Chapter 3 proposes a knowledge representation and algorithm to describe routes for guide robots. Chapters 4 to 7 are all around the Referring Expression Generation problem. This thesis ends with two robotic architectures embodied respectively into a Pepper and a PR2 robotic platform. The first architecture has been developed approximatively from 2017 to 2019 in the context of a European project. The second architecture is an internal project of the team, giving more flexibility to explore new designs and new organisation of its components. The latter has been developed from 2020 to 2021. While the first architecture uses the route description knowledge exploitation, the second use the contributions on the Referring Expression Generation. Having these two architectures grouped at the end of this thesis allows us to compare them and see how knowledge management has taken a primary place in a robotic architecture for Human Robot Interaction applications.

All along this thesis, special attention has been paid to the performance of each contribution. Each time our aim is to integrate the presented contribution into a wider system and work in an incremental manner. To not spoil the final Human-Robot interaction and be able to create more high-level cognitive capabilities, this attention is essential to us. At the end of most of the chapter, a performance analysis is thus provided.

To get to the point

For the readers having less time and wanting to go to the point of this thesis, we can propose a shorter reading route. For sure, this selection is subjective, we often prefer our latter work, which seems to us more mature, having a better background on the topics. We thus suggest to read to following:

- Chapter 2: Ontologenius, the ontology management software, the core of this thesis,
- Chapter 4: The Referring Expression Generation algorithm,
- Introductions of chapters 5 and 6: Some challenges around the Referring Expression Generation,
- Chapter 7: A preliminary work to create more generic Referring Expressions,
- Section 9.3: Our most advanced cognitive architecture for Human Robot Interaction applications.

List of Publications

Published

- Sarthou, G., Alami, R., & Clodic, A. (2019, June). Semantic Spatial Representation: a unique representation of an environment based on an ontology for robotic applications. In Combined Workshop on Spatial Language Understanding (SpLU) and Grounded Communication for Robotics (RoboNLP).
- Sarthou, G., Clodic, A., & Alami, R. (2019, October). Ontologenius: A long-term semantic memory for robotic agents. In 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN) (pp. 1-8). IEEE.
- Heikkilä, P., Lammi, H., Niemelä, M., Belhassein, K., Sarthou, G., Tammela, A., Clodic, A., & Alami, R. (2019, November). Should a robot guide like a human? A qualitative four-phase study of a shopping mall robot. In International Conference on Social Robotics (pp. 548-557). Springer, Cham.
- Buisan, G.*, Sarthou, G.*, Bit-Monnot, A., Clodic, A., & Alami, R. (2020, August). Efficient, situated and ontology based referring expression generation for human-robot collaboration. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)* (pp. 349-356). IEEE.
- Buisan, G., Sarthou, G., & Alami, R. (2020, November). Human aware task planning using verbal communication feasibility and costs. In *International Conference on Social Robotics* (pp. 554-565). Springer, Cham.

- Sarthou, G., Mayima, A., Buisan, G., Belhassein, K., & Clodic, A. The Director Task: a Psychology-Inspired Task to Assess Cognitive and Interactive Robot Architectures. To be published in *2021 30th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*.

- Sarthou, G., Buisan, G., A., Clodic, A., & Alami, R. Extending Referring Expression Generation through shared knowledge about past Human-Robot collaborative activity. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*

Submitted

- Mayima, A., Sarthou, G., Buisan, G., Singamaneni, P., Sallami, Y., Belhassein, K., Waldhart, J., Clodic, A., & Alami, R. Direction-giving considered as a Human-Robot Joint Action. Submitted to *User Modeling and User-Adapted Interaction (UMUAI) Journal*.

Chapter 2

Ontologenius: A semantic memory for HRI

Contents

2.1 Design and features 16
2.1.1 Why an ontology? 16
2.1.2 Desired features 19
2.1.3 Ontology formalism 21
2.2 Architecture 28
2.2.1 Permanent versus temporary data structure 28
2.2.2 Reasoning to enrich the knowledge 31
2.2.3 Concepts name in natural language 32
2.3 Managing partners' estimated knowledge 32
2.3.1 Ontologenius multi-instances principle 32
2.3.2 Catching knowledge at a given moment 33
2.3.3 Exploring several possible mental states at once 34
2.4 Using Ontologenius in robotic applications 35
2.4.1 Updating the knowledge base 35
2.4.2 Retrieving knowledge 36
2.4.3 Interfacing with Ontologenius 38
2.5 Computational performance evaluation 40
2.5.1 Comparing with Knowrob 40
2.5.2 Comparing with ORO 44
2.5.3 Additional tests 47

In this chapter, we present the software Ontologenius. It is a lightweight and open-source software to store an ontology, perform reasoning on it, update it, and query it. Ontologenius has been developed especially for Human Robot Interaction applications. To do so, it allows to manage several ontology instances in parallel and puts a focus on the concepts' names in natural language. It is aimed to be used as a server, shared by an entire robotic architecture, providing consistent and always up to date knowledge to every component of the architecture. Consequently, it is thread-safe and can be queried by several components at a time while being constantly updated.

A previous version of this contribution has been presented and published at the RO-MAN 2019 conference [Sarthou 2019b]. The current chapter presents new features and a more mature contribution.

2.1 Design and features

In this section, we first explain our choice to use ontology as a means of representing knowledge, both from the point of view of its expressiveness and its growing use in robotics. Then, we present the desired features and the level of expressiveness we have selected for this software. Finally, we introduce a formalism of the kind of ontology we will use all along this thesis.

2.1.1 Why an ontology?

In cognitive psychology, semantic memory refers to the encyclopedic knowledge of words associated with their meanings. After several studies about participants response times to questions, some authors have proposed a model of this semantic memory as being a semantic network [Collins 1969, Collins 1970]. With this model, they put the hypothesis that knowledge is organized in a hierarchical way, respecting a principle of inclusion among classes. For example, a class representing the concept of cat would inherit from an upper class, representing the concept of animal. In addition, instances of these classes would be linked to others through properties, and in the same way, as for the classes, a notion of hierarchy over the properties would exist. Such a structure of knowledge in humans would allow a cognitive economy as well as efficient storage of this knowledge. Even if they were not the first to formalize the principle of a semantic network, Collins and Quillian have provided prominent work and computer implementations.

As reported in [Prasad 2020], such a semantic network, also called semantic graph or knowledge graph, is today frequently used as knowledge representation in robotic applications to represent among others:

- the categories of entities at different levels of abstraction [Bálint-Benczédi 2018], e.g. a handle is a physical object
- the characteristics of entities [Tenorth 2017], e.g. a fridge has a handle
- the function or purpose of entities [Paulius 2019], e.g. the fridge handle allows to open the fridge
- the location of an entity with respect to another one (i.e. spatial relations) [Singh 2020], e.g. the milk bottle is in the fridge

It is on the basis of this knowledge that we can then create programs to provide cognitive capabilities to robots. Here is an important point, this knowledge should support the components of the architecture but is not necessarily part of it. It is thus

called the knowledge-enabled robot programming paradigm [Beetz 2012], where the knowledge is separated from the program. It allows to have several subsystems of a robotic architecture, each providing a specific cognitive capability but all sharing the same knowledge. The choice to share common knowledge leads to important challenges requiring first to agree on the knowledge content and second to provide it in a machine-understandable way. Considering the use of a semantic network as knowledge representation, the use of ontology is intended to respond to these challenges.

The first definition of an ontology has been proposed by Guber in [Guber 1993] as: *"an ontology is an explicit specification of a conceptualization"*. Later, Borst [Borst 1999], has introduced two new concepts defining an ontology as a *"formal specification of a shared conceptualization"*. The new concepts are the notions of "formal" and "shared". Merging these two definitions, Studer et al. [Studer 1998] reached the definition: *"An ontology is a formal, explicit specification of a shared conceptualization"*. With this last definition, we start to see that an ontology can be a powerful tool to create a Knowledge Base (KB) common to an entire robotic architecture and used by subsystems providing specific cognitive capabilities.

Even if all the previous definitions trend to refine what is an ontology, they all rely on the common notion of conceptualization for which no formal definition is provided. A former definition, presented in [Genesereth 1987], was:

> *"A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly."*

However, as explained by Guarino in [Guarino 2009], such a definition of a conceptualization does not correspond to our intuition either our need for an ontology. Indeed, we do not want the conceptualization to depend on the current situation. It should rather be what is common to any situation, allowing to represent them in a uniform way. As explained in [Guarino 1995], when the world is modified, the conceptualization should stay the same. We are thus interested in the meaning of the concepts used to describe a world state. However, the final definition of a conceptualization provided by Guarino goes with difficulty in this direction despite its initial goal. For the simplicity of the current purpose, the conceptualization should be the meaning of any concept allowing to represent a given situation or world state. Consequently, an ontology could be defined as:

> *a formal and explicit specification of the shared meaning of any concept allowing to represent a given situation.*

In other words, an ontology aims at constraining the interpretation of a used vocabulary. It is thus a logical theory. However, looking at the literature, we notice

that the definition of an ontology is still today blurry. Indeed, even if it represents the meaning of the usable concepts, we want to apply it to the KB representing a given world state, even if it changes over time. Where a semantic network would represent this instantiation, because the meaning of its edges could be defined by the use of an ontology, in this thesis, we will use the term ontology to refer to **a semantic network owning a restriction on the interpretation of the used vocabulary**. As the goal of this thesis is not to model these restriction rules but rather to use them, it should not lead to any confusion.

Considering the study of ontology in robotics, and more generally in computer science, we can distinguish three kinds of contribution:

- ontology creation,
- ontology storage and inference,
- framework using ontology

Ontology creation focuses on the definition of vocabulary and rules. The created ontologies are often divided into four categories: upper-level, reference, domain, and application. The upper-level ontologies define widely applicable concepts, transversal to several disciplines. The more used are DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [Masolo 2003], Cyc [Lenat 1989], or SUMO (Suggested Upper Merged Ontology) [Niles 2001]. Reference ontologies are based on upper-ontology and describe the vocabulary of a discipline, like engineering or medicine. For example, [Schlenoff 2015] covers robotics and automation, describing the robot from an engineering point of view, with its sensors, its processes, and its pose. Domain ontologies aim at refining a discipline, focussing on a more restricted domain in it, like the SOMA ontology [Beßler 2020] focussing on activities representation. Finally, applications ontologies extend domain ontologies for precise applications. The ontology design uses languages like RDF (Resources Description Framework), FOL (First Order Logic), or OWL (Web Ontology Language). Each language comes with characteristics like formality or computability.

Using these ontologies, a number of frameworks have been developed to support high-level reasoning processes in robotic applications. Among them, we find ontology used for decision making, situation assessment, planning, or belief maintenance. A complete review of these reasoning capabilities and the corresponding frameworks is available in [Olivares-Alarcos 2019]. Among the reviewed frameworks, the emblematic ones are KnowRob [Tenorth 2013], ROSETTA [Stenmark 2013], CARESSES [Bruno 2017], RoboBrain [Saxena 2014], or ORO [Lemaignan 2010].

The last contribution type, being the one of the current chapter, is the ontology storage and inference to form a Knowledge Base on which the framework can rely to perform high-level reasoning. KnowRob uses Prolog [Wielemaker 2003] with a library to manage RDF triples. ROSETTA uses a Sesame triple store [Broekstra 2002]. ORO uses the Jena triple store in addition to the Pellet [Sirin 2007] reasoner. Finally, RoboBrain uses a custom graph database. We

can see that no standard solution has emerged as it mostly depends on the application needs in terms of query capability, expressiveness support, or technical support and compatibility. As other tools we can find in C++ owlcpp [Levin 2011] based on Raptor and Fact++ [Tsarkov 2006], or in Python RDFlib, OWLReady2, or AllegroGraph.

To sum up, a Knowledge Base in the form of an ontology, can be viewed as a semantic graph relying on a formal and explicit specification of the shared meaning of concepts used to represent a given situation. In order to be used in a framework to perform high-level reasoning, ontology has to be stored to expose the knowledge it contains to other components. No standard storage solution has emerged so far as it mostly depends on the application needs. Consequently, in the next part of this section, we discuss our needs with a focus on Human Robot Interaction applications and taking inspiration from the existing solutions, we define the features we want for our storage system.

2.1.2 Desired features

Work as a server: The robotic architecture we target is composed of several modules, able to communicate. On top of the architecture, we consider a supervision system, being a kind of "puppet-master" which calls each component when needed. A common issue with such an architecture is that each component owns a part of the general knowledge which can lead to inconsistencies. Consequently, the Knowledge Base we want should be a server, accessible by query and update by any component. It thus provides uniform knowledge among the architecture. To create a server, we could use any existing tool and attach a communication layer to it. This is for example the case of ORO, which works as a server, based on the Jena triple store and providing a Telnet interface to send the queries and updates.

Support multiple instances: Since we target HRI application, we need to be able to represent an estimation of the human partners' knowledge. Where some could use a single Knowledge Base to represent the robot knowledge as well as the humans' knowledge, we want our software to support the "self-other distinction". It is presented in [Pacherie 2012] as the fact that "for shared representations (...) to foster coordination rather than create confusion, it is important the agents be able to keep apart representations of their own and other's actions and intentions". To the best of our knowledge, ORO is the only software proposing this ability. However, it uses a simple way to implement it by running several instances of the same triple store, each attached to an agent. As we will see in the sequel, in robotic application such a solution is not sufficient. For task planning usage, we could need to estimate a future state of a KB. To implement this feature, we need to be able to capture the state of a KB at a given instance then be able to modify it without impacting the original KB, which is continually updated by perception. At the date of this thesis, no ontology-based tool provides this possibility.

Query at the semantic level: As explained in [Broekstra 2002], to query RDF graphs, three levels are possible. Considering an ontology written in XML, the syntactic level would query the XML structure. Consequently, it would query a tree rather than a graph. A query would be "give me the resources nested in a *Description* element having the attribute *about* with the value X". It is thus dependent on the language and requires to known the used keywords. The structural level, as available in [Lassila 1998], provides an abstraction to the language. It allows query of the kind "give me all the elements of type A" regardless of how it is represented in the language. However, the structural level only looks at the explicit triples. If B is a subtype of A and the element x is described as being of type B, requesting for the elements of type A, x will not be returned. The last query level is the semantic level which is not limited to explicit knowledge. At this level, requesting for the elements of type A, x would be returned as x is of type B and B is a specification of A. This means that we do not consider a simple triple store trying to match patterns. To support this query level, two solutions can be used: computing the closure of the graph (adding A as a type of x and storing it), or inferring it at query. While the first solution removes a part of the semantic by flattening the hierarchy[1], the second can be time-consuming for the queries.

Specific queries: We aim to use our software with solver algorithms. This means that we want to provide fine access to the stored knowledge at high frequency. At the difference of Prolog having its own search algorithm, we would prefer lower-level queries such as: "which are the direct types of X?", "which are the inverse properties of Y?", "which properties link C and D?", or "is E in the domain of application of F?", and that at the semantic level. Nevertheless, for fast queries, we should be careful with the number of inferences needed at query time.

Reasoning at update: To avoid too many inferences at query time, we want some inferences to be applied at update. While computing the entire closure would flatten the KB, some relations can still be computed at updates like the ones coming from inverse relations, or chain axioms.

Thread safe: Since the software should work as a sever, we want it to support multiple queries in parallel but to be safe on update.

Expressiveness level: Considering the Web Ontology Language (OWL), there exist different levels of expressiveness depending on the needs. The most expressive one is OWL-full, however, it is said to be not computational, meaning that the inferences it allows cannot be resolved at query time. Then, OWL-DL (Description Logic) supports property and class hierarchy, enumeration value restriction, inverse properties, or cardinality restriction. Finally, OWL-lite is the less expressive one, not supporting the cardinality restriction and the restriction on value. Even if

[1] We lose the fact that x is of type A because it is of type B.

OWL-DL would be suitable, in a first version OWL-lite could be sufficient. It is, however, the minimal expressiveness to support as a whole.

Custom reasoning: For research purposes, we do not want to be limited to First-Order Logic reasoning. We want to be able to integrate application-dependent reasoning processes having direct access to the internal structure. Such a feature could be provided by the support of plugins for example. The advantage of plugins is that anyone can add reasoning capability without modifying the core of the software or owning a custom version.

In the light of our desired features, of the number of available tools no more maintained or without documentation, and our need to be able to implement new capacities according to our research needs evolving over time, we choose to develop new software from scratch. The resulting software is Ontologenius. Such a choice is risky because it will be difficult to reach the level of much more mature software. On the other hand, this allowed us during this thesis to have control over its internal functioning and to be able to make it evolve easily during the different projects in which we used it.

Ontologenius is part of the continuity of the ORO software but by offering more advanced multi-instance management and a query system more suited to integration into algorithms. So far we have used the terms of class, properties, inverse properties, etc. Before starting the presentation of Ontologenius implemented features, we first propose a formalisation of what is an ontology and an explicit definition of the terms. This formalism will then be used all along this document and give an overview of the expressiveness supported by Ontologenius at the date.

In the rest of this thesis, we will often use the expression "to query the ontology". Through this term we always consider a query of a semantic graph built with a formal and shared vocabulary based on OWL, using Ontologenius.

2.1.3 Ontology formalism

Even if we saw that the use of ontology is today a common way to represent semantic knowledge, we will recall in this subsection the contents of an ontology. For each element composing it, we will draw a formalization based on Description Logic (DL), then give examples using the Turtle syntax. The pieces of ontologies used in the examples of this subsection are voluntarily simplified. The introduced notations will be the ones used in the rest of this document and the graphical representations, both in terms of color and shape, will be used as often as possible.

On the base of the definition of a Description Logic ontology presented in [Fokoue 2006] and [Krötzsch 2013], we define a semantic knowledge base K_S represented as an ontology by $K_S = \langle \mathbb{A}, \mathbb{T}, \mathbb{R} \rangle$. $\mathbb{A}$, $\mathbb{T}$, and $\mathbb{R}$ are respectively called the ABox, TBox, and RBox of the ontology.

In the Description Logic knowledge base it is common to only consider a TBox, containing extensional knowledge (general knowledge about a domain), and an

ABox containing intensional knowledge (instantiated knowledge) [Baader 2003]. The first describes the terminology while the latter contains assertions. The definition we choose with the RBox extracts a part of the TBox to put it into the RBox, making it clearer.

2.1.3.1 The ontology TBox: Terminological Knowledge

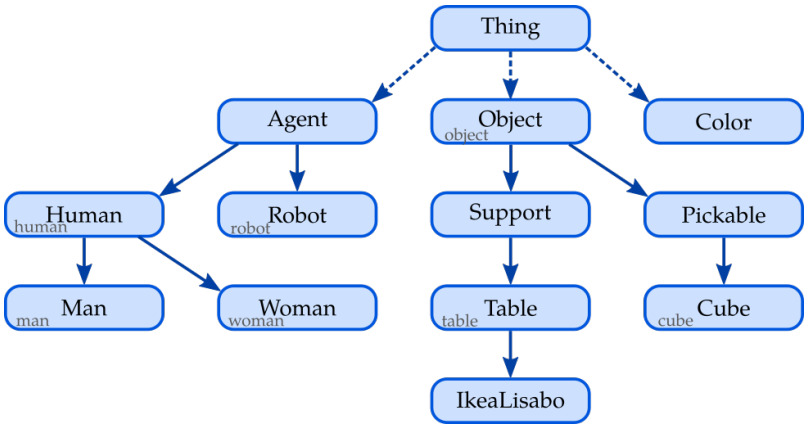

Figure 2.1: Representation of an ontology class hierarchy graph to illustrate the composition of a TBox. Taking the class Human, the arrow linking "Human" to "Man" has to be read as *"A man is a kind of Human"*. The texts at the bottom left of the class are the classes' labels in natural language, when they exist.

The TBox $\mathbb{T}$ contains assertions about the **classes** (types) of the ontology. It is defined by $\mathbb{T} = \langle T, H \rangle$. It can be seen as a directed acyclic graph as presented in Figure 2.1. T is the set of all the classes of the ontology. In our example, $T = \{Thing,\ Agent,\ Object, ...,\ IkeaLisabo\}$. Considering the TBox as a graph, H stores its directed edges. They represent the inheritance links between the classes (i.e. the subsumption assertions). We often use the special property "isA" to refer to these links (e.g. *(Human, isA, Agent)*). In the OWL language, they are described with the property rdfs:subClassOf, as illustrated in Listing 2.1.

```
:Human rdf:type owl:Class ;
       rdfs:subClassOf :Agent .

:Man   rdf:type owl:Class ;
       rdfs:subClassOf :Human .
```

Listing 2.1: Description of ontology classes in the OWL language using the Turle syntax.

2.1.3.2 The ontology RBox: Relations between Roles

The RBox $\mathbb{R}$ contains axioms about the **properties** (roles). It is at least defined by $\mathbb{R} = \langle P, Incl, Inv, Dom, Ran \rangle$. P is the set of properties, and $Incl$ stores the directed edges of the finite directed acyclic graph representing the inheritance links

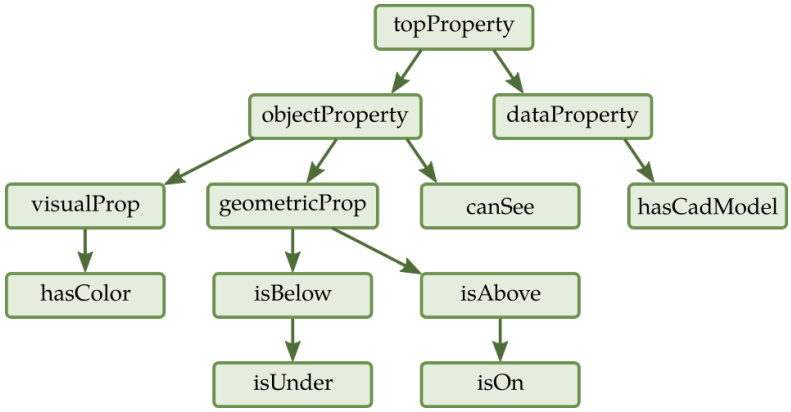

Figure 2.2: Representation of an ontology property hierarchy graph to illustrate the composition of an RBox. Taking the property isBelow, the bottom arrow has to be read as: *"The property isUnder is a specification of the property isBelow"*.

between the properties. Such a graph is represented in Figure 2.2. These inheritance links aim at specifying properties. In our example, the property IsOn is a specification (more detailed) of the property isAbove in the way that an object being on another is an object that is above the latter and being in contact with. It is described with the property `rdfs:subPropertyOf` in the OWL language.

$Inv = \{(p, p^{-1}) \in P^2\}$ is the set representing the properties inverses (*e.g.* $(isOn, isUnder) \in Inv$). Describing the inverse of a property is useful first to reduce description work since if some describe a relation involving a property for which an inverse is defined, the inverse relation is also described in an underlying way. Moreover, for an algorithm exploring an ontology, knowing that a relation uses a property having an inverse can allow to reduce the algorithm complexity by not considering the inverse relation into the exploration.

Finally, Dom and Ran are two sets representing respectively the properties domains and ranges. Their are define by $Dom = \{(p, t)\}$ and $Ran = \{(p, t)\}$ with $p \in P$ a property and $t \in T$ a class. The domain of a property informs on the type of resources that may use the property, thus the type of the subject of a triplet. The range of a property informs on the valid values applied to the property, thus the type of the object of a triplet. For the property isOn, we would therefore have $(isOn,\ Object) \in Dom$ and $(isOn,\ Support) \in Ran$. In this way, we state that the property IsOn can be used to describe that an object is on top of an object being support. Domains and ranges can be used in two ways. It can be to check the consistency of an ontology by checking if the way the properties have been used corresponds to their definition. It can also be used to reason on the ontology and extract new knowledge from a given situation. If, for example, an entity is said to be on top of another that is not described as being a support, we could deduce that this second entity is a support.

The formalization above considers only a general kind of property while the OWL language makes the distinction between two main categories. The **object**

properties, linking two entities, and **data properties**, linking an entity to a value (representing entities' attributes). While both are slightly different, we will only keep a general definition of a property for our formalization. This is only to simplify the future algorithm explanations. An example of the description of an object property and a data property from the Figure 2.2 are illustrated in Listing 2.2 using the OWL language.

```
:isOn  rdf:type owl:ObjectProperty ;
       rdfs:subPropertyOf :isAbove ;
       owl:inverseOf :isUnder ;
       rdfs:domain :Object ;
       rdfs:range :Support .

:hasCadModel rdf:type owl:DatatypeProperty ;
             rdfs:domain :Object .
```

Listing 2.2: Description of ontology properties in the OWL language using the Turle syntax.

2.1.3.3 The ontology ABox: Asserting facts

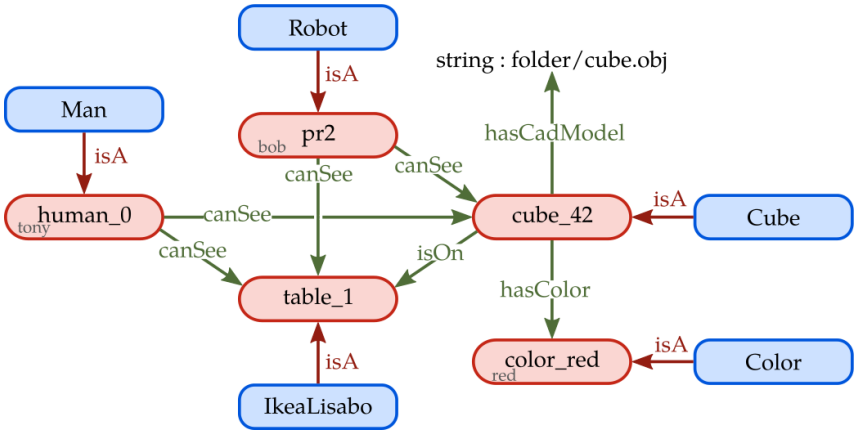

Figure 2.3: Representation of an ontology instances graph to illustrate the composition of an ABox. Red boxes are individuals of the ontology. Green arrows are properties coming from the RBox and applied to individuals. Red arrows represent a direct inheritance link between an individual and a class coming from the TBox. The texts at the bottom left of the individuals are the individuals' labels in natural language, when exist.

The ABox $\mathbb{A}$ contains assertions about the **entities** (individuals) of the ontology. When we refer about entities, we no more speak about general concepts but rather of instantiated concept, being either a physical or conceptual entity. The ABox is defined by $\mathbb{A} = \langle A, C_0, R \rangle$. A is the set of all the entities represented in the ontology. C_0 the set of direct types of A such as $C_0 = \{(a, t)\}$ with $a \in A$ an individual and $t \in T$ a class. In the graphical representation of an ABox in Figure 2.3, the red blocks are the ABox entities ($A = \{human_0,\ pr2,\ ...,\ table_1\}$) and the red

arrows with the label "isA" are the entities direct types ($(cube_42, Cube) \in C_0$). R is finally the set of **relations** between entities. Such relations are in the form of triplets (s, p, o) where s is the subject, p the property and o the object. The set of relations is thus defined by $R = \{(s, p, o) | (s, o) \in A^2, p \in P\}$. These relations are represented by the green arrows between the entities in Figure 2.3. We can note in this figure the presence of the use of a data property "hasCadModel". This property does not link two entities, which goes against the previous definition. Regarding our formalization and to keep it tractable, we can however keep it as it is, and view the string value as an entity having for direct type a concept "String". An example of the description of an entity from Figure 2.3 is illustrated in Listing 2.3 using the OWL language.

```
:cube_42  rdf:type      :Cube ;
          :hasColor     :color_red ;
          :hasCadModel  ``folder/cube.obj''^^string ;
          :isOn         :table_1 .
```

Listing 2.3: Description of an ontology individual in the OWL language using the Turle syntax.

We just saw that in the ABox, C_0 contains the direct types of entities. We also saw that the classes can inherit from one another in the TBox, thanks to the classes inheritance directed edges stored in H. This means that the individuals of the ABox have inherited types. Taking the entity cube_42 of Figure 2.3, its direct type is the class Cube ($(cube_42,\ Cube) \in C_0$). Regarding the TBox represented in Figure 2.1, a Cube is a kind of Pickable ($(Cube,\ Pickable) \in H$), itself being a kind of Object ($(Pickable,\ Object) \in H$). We can thus say that the entity cube_42 is a Cube, a Pickable, and an Object. To represent it, we use C to denote the set of direct and inherited types. We thus have $\{(cube_42,\ Cube), (cube_42,\ Pickable), (cube_42,\ Object)\} \subset C$.

2.1.3.4 Extending the ontology TBox

With the use of the relation set R of the ABox, we saw that we can apply properties to individuals to link them together and form relations in the form of triplets. However, one could want to apply properties to classes to describe general links between classes. While properties domains and ranges already give such relations this can be not enough. Taking an object property hasMother, we can assign to it the class Human for domain and Woman for range. With such description, we state that a human CAN have a mother, that is a woman. However, we do not describe that even if we do not know who it is, a human has a mother who is a woman. For this particular example, we could use cardinality constraint but we will not go that far. Taking now the data property hasCadModel of Figure 2.2, we have applied it to a specific entity in the example of Figure 2.3. But what about a Table Lisabo (IkeaLisabo in Figure 2.1)? Any table of this model will have the same CAD model and we do not want to put this relation to every entity of this type. Here domains and range are not sufficient to represent it. To do so, we will use

annotation properties applied to classes. Annotation properties are usually used to document ontologies and not to describe general relations on classes. We take thus some liberty regarding the OWL standard for convenience. However, we will try to use it in very particular cases where no other simple solution can be applied. Relations to classes using annotation properties are thus added to the definition of a TBox $\mathbb{T} = \langle T, H, E \rangle$, where E is the set of relations between classes in the form of triplets.

2.1.3.5 Advanced use of properties

In this sub-section, we present a formalism of an ontology in the form of $K_S = \langle \mathbb{A}, \mathbb{T}, \mathbb{R} \rangle$. All the knowledge stored in K_S is sufficient to build exploration algorithms on top of it. However, to reason on ontology, additional descriptions are necessary in the form of properties characteristics. We do not add them to the knowledge base formalism but enumerate them below:

- **Symmetric property**: If the relation (x, p, y) holds in R with p being a symmetric property, the relation (y, p, p) is also part of R.
  e.g. *hasSpouse*

- **Asymmetric property**: If the relation (x, p, y) holds in R with p being an asymmetric property, the relation (y, p, p) cannot be part of R.
  e.g. *hasChild*

- **Reflexive property**: A reflexive property can be used to link an individual to itself.
  e.g. *hasRelative*

- **Irreflexive property**: An irreflexive property cannot be used to link an individual to itself.
  e.g. *hasParent*

- **Functional property**: Every individual can be linked by a functional property to at most one other individual. By this way, if $(x, p, y), (x, p, z) \subset R$, then $y = z$.
  e.g. *hasFather*

- **Inverse functional property**: Every individual can hold at most one inverse functional property. By this way, if $(x, p, y), (z, p, y) \subset R$, then $x = z$.
  e.g. *hasHusband*

- **Transitive property**: A transitive property describes a link between two individuals x and z whenever it exists a link between x and y, and y with z with this property. If $(x, p, y), (y, p, z) \subset R$ with p a transitive property, then $(x, p, z) \in R$.
  e.g. *hasAncestor*

- **Property chain axiom**: While the transitive property characteristic decsribes a link between several individuals with the same property, the chain axiom does the same with distinct properties. Given the chain $p_1 \bullet p_2 \Rightarrow p_3$, if $(x, p_1, y), (y, p_2, z) \subset R$, then $(x, p_3, z) \in R$.
  e.g. $hasParent \bullet hasParent \Rightarrow hasGrandparent$

- **Disjunction**: Given two disjoint elements (classes or properties), a third element cannot inherit of two disjoint elements.
  e.g. $Man \sqcup Woman$

2.1.3.6 Labeling functions

We saw in the previous chapter that the semantic knowledge base is part of what we assimilate to be the declarative memory. The particularity of such memory is the ability to verbalize the knowledge it stores. In this way, we introduce a labeling function $\mathcal{L}$ for any element of the ontology. This labeling function is specified for the individuals ($\mathcal{L}_a$), the classes ($\mathcal{L}_t$), and the properties ($\mathcal{L}_p$). Considering the individuals labeling function $\mathcal{L}_a : A \rightarrow Lbl$ with Lbl a set of communicable names encoded as UTF8 string in our implementation. The same holds for the other two labeling functions.

2.1.3.7 Ontology for Human-Robot Interaction

Since we are working in the field of Human-Robot Interaction, it is mandatory for the robotic agent to be able to represent its own knowledge but also to represent an estimation of its human partners' knowledge. Such features will be explained later in this thesis and we only introduce the related notation for the moment. We define the robot's own symbolic knowledge base $K_S^R = \langle \mathbb{A}^R, \mathbb{T}^R, \mathbb{R}^R \rangle$. Then, for each human agent H_i the robot knows, we consider the agent's semantic knowledge $K_S^{H_i} = \langle \mathbb{A}^{H_i}, \mathbb{T}^{H_i}, \mathbb{R}^{H_i} \rangle$. The robot's global knowledge thus encompasses both its own semantic representation of the environment as well as an **estimation** of the other agent's knowledge.

To go further, when the robot meets a new human partner, it can estimate him to have a minimal set of knowledge. It is what is called the common ground. We define it as $K_S^{CG} = \langle \mathbb{A}^{CG}, \mathbb{T}^{CG}, \mathbb{R}^{CG} \rangle$. Since it represent the minimal knowledge we estimate an agent to have, we can state that $K_S^{CG} \subset K_S^R$ and $K_S^{CG} \subset K_S^{H_i}$. The common ground represents the fact that, for example, anybody knows the color blue, the concept of a table, or the property isAbove.

In the rest of this document, we will simply use the notation K_S in cases where the used knowledge base does not matter (i.e. either the robot one or an estimated one can be used).

2.1.3.8 Ontology formalism recap

The ontology definition used all along this thesis is summarized in Table 2.1.

Table 2.1: The list of symbols used to define a semantic knowledge base as an ontology

$\mathbb{A}$ **ABox entities/indiv**	$\mathbb{T}$ **TBox classes/concepts**
A: set of entities	T: set of classes
C_0: entities' direct types	H: classes inheritance links
R: relations between entities	E: relations between classes
$\mathcal{L}_a$: individuals labeling function	$\mathcal{L}_t$: classes labeling function

$\mathbb{R}$ **RBox roles/properties**	
P: set of properties	
$Incl$: properties inheritance links	Inv: properties inverses
Dom: properties domains sets	Ran: properties ranges sets
$\mathcal{L}_p$: properties labeling function	

2.2 Architecture

Now we have presented what an ontology is and the desired features for a software managing a Knowledge Base (KB) based on an ontology, we introduce the resulting software, Ontologenius. In this section, we start with the software architecture able to manage a single ontology instance, meaning the KB of a single agent. The goal of this section is not to go deep in the technical implementation but rather to give cues to understand how the KB is managed.

The architecture of Ontologenius is divided into three major modules as shown in Figure 2.4: the knowledge storage (permanent and temporary data structures), the KB exploration (transient data) and the reasoning processes (plugins). Ontologenius has been developed in C++14 and is based on ROS for the communication layer and the plugins management. It is compatible with ROS Kinetic (2016), Melodic (2018), and Noetic (2020).

In this section, we first explain how ontology files are loaded and converted into a semantic graph. Then, we present the available reasoners and their management. Finally, we explain how the system deals with natural language names.

2.2.1 Permanent versus temporary data structure

To store the Knowledge Base as an ontology, we first use OWL files using the XML/RDF syntax. The files can be read from a local folder on the computer or retrieved from the Internet. Ontology files often include others, as an upper ontology. However, Ontologenius is not able for now to solve these dependencies. All the required files thus have to be listed. These files are what we consider to be a permanent data structure. Nevertheless, our goal is not just to load a static Knowledge Base. This latter aims to be updated all along an interaction and if the robot learns new concepts, we want it to be able to memorize them for the next interaction. In the same way, if the robot interacts with a given agent and estimates that he/she knows some concepts, it needs to memorize them for the next

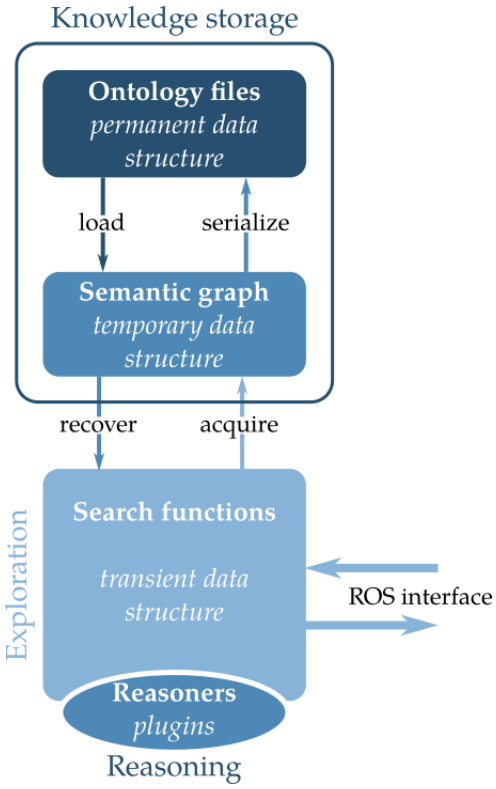

Figure 2.4: The software architecture of Ontologenius to manage a single Knowledge Base in the form of an ontology.

interaction with this specific agent. Consequently, we consider two kinds of files. On one hand, we have what we call the input files, being a default Knowledge Base (i.e. the common ground). On the other hand, we have the internal file, being the Knowledge Base of a given agent. Their management is illustrated in Figure 2.5. If no internal file exists for the managed agent, the input files are loaded (a). This will be the case for the first time we start the robot if the Knowledge Base is the robot's one, or the first time the robot interacts with a new human if the Knowledge Base is an estimation of an agent Knowledge Base. The input files thus represent the minimal knowledge we estimate an agent to have. Once we power off the robot or when an agent leaves an interaction, all the maintained Knowledge Base is serialized into a single OWL file, the internal file. This file is thus a backup of an agent's Knowledge Base. The next time the robot is powered on, or the next time it interacts with a known agent, the internal file is load and the input files are no more considered (b). The robot can thus refine its estimation of the others knowledge as well as its own knowledge over interactions.

When ontology files are loaded, they are converted into a semantic graph, at the difference of triple store. Elements of the ontology (individuals, classes, object properties, and data properties) are represented as nodes. We thus have four types of nodes. The nodes have several sets of pointers toward other nodes, each having

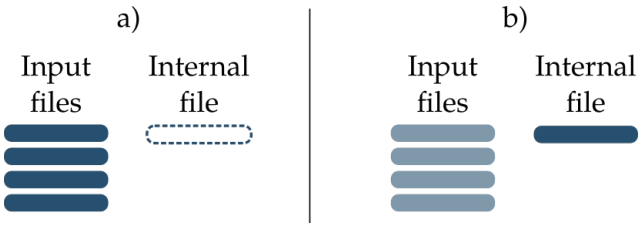

Figure 2.5: a) In the case where no internal file already exists, the input files are considered as the default Knowledge Base of the agent b) In the case where an internal file exists for a given agent, this means that the robot is already interacting with this agent. In the latter case, the input files are not taken into account and the previously estimated Knowledge Base of the agent is loaded. If the current Knowledge Base represents that the robot's one, the robot simply restarts with its previous knowledge base.

a semantic meaning. Such a representation is useful for efficient search algorithms like query resolution. For example, wanting to know all classes from which an individual inherits, from the node of this individual, we just have to iterate over all the nodes in the set of inheritance and perform recursion on each of them, iterating once again on the set of inheritance.

To find the entry point of the graph, we have four containers, one for each element type, as illustrated in Figure 2.6. These containers provide efficient retrieval of an element from its identifier thanks to the use of maps to store the nodes. In addition, the containers provide a readers-writer lock mechanism, allowing concurrent access for ready-only operation while ensuring exclusive access for a write operation. Thanks to this synchronization primitive, Ontologenius is thread-safe.

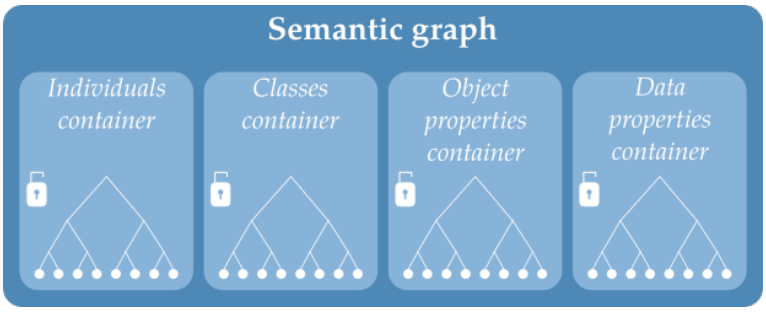

Figure 2.6: The semantic graph has four containers, one for each type existing in an ontology (i.e individuals, classes, object properties, and data properties). The containers own the nodes representing the elements of the ontology and store them in maps where the keys are the identifiers of the elements and the values are the nodes themselves. Each container also provides a lock mechanism allowing to prevent concurrency issues in multi-thread usage.

2.2.2 Reasoning to enrich the knowledge

Reasoning on a Knowledge Base is a primal need to deduce new knowledge from the described one. At the difference of tableau algorithm [Zuo 2006], we do not want to create a dedicated model to do so. Consequently, we made the choice to develop our own reasoners, able to run on the Ontologenius internal representation.

The reasoners are implemented on the basis of plugins for an easy extension[2]. The proposed plugin interface allows the execution of reasoning processes either before the resolution of a query, after an update of the KB, or periodically. The reasoners are not necessarily limited to first-order logic. Users can implement any process depending on their needs. To date, we have four usual reasoners to solve the symmetric properties, the inverse properties, to reason on domain and range, and to solve chained axioms. We have also implemented a reasoner dedicated to the elaboration of complementary labels and another to create relations on classes, using annotation properties, being a generalization of the existing relation. For example, if several entities own relation toward the same CAD model file, the reasoner will generate a relation toward this CAD model to an upper common class to all these entities.

A YAML configuration file can be provided to choose the reasoners to use and to tune each reasoner depending on the application. Any available reasoner could also be activated and deactivated at run time. However, no order among the reasoners can be defined. This means that each reasoner has to be independent of the others.

To avoid time loss during reasoning, two mechanisms are available. For the reasoners needing to run once on each element, they can mark the elements on which they have run. By checking these marks they can know if an element has already been analysed or not. The second can be compared to a to-do list. Any element owns an update flag. When an element is modified by an external process, like a perception process, the flag of the element is raised. Reasoners then run only on the elements having this flag raised. If they find a new relation to insert in the KB, they can raise the flags of the impacted elements. The other reasoners will thus check again for these newly updated elements. A reasoning pass ends when no more update has been performed by any of the reasoners. While a reasoning pass is in progress, no additional external modifications of the KB are applied to avoid conflicts.

Finally, to facilitate the deletion of inferred relations, we attach a reference to any relations that have allowed those inferences. In this way, by removing a relation, we are able to find side relations that also need to be removed. For example, consider the chain axiom $isIn \bullet isOn \Rightarrow isOn$. If A isIn B and B isOn C, the reasoners will generate the relation A isOn C. The two former relations have thus allowed the inference of the latter. If one of them is removed, the third also has

[2]Even if dedicated symbolic planners, like the ones presented in the rest of this document, could be implemented using the Ontologenius plugin system, in this section, we only consider as reasoners algorithms aiming to deduce new pieces of knowledge from the existing ones and storing them in the ontology.

to be removed. If A is no more in B, it is consequently no more on C. In this way, the KB can always be kept up to date without performing reasoning from scratch at each update, neither by creating a side model dedicated to the reasoning. However, it is less efficient than classic reasoning methods which can use dedicated models.

2.2.3 Concepts name in natural language

Having labels attached to elements is quite usual. Like other systems, Ontologenius attaches to any element a map linking a country-language code to a set of labels. The slight difference of Ontologenius is that it supports a special kind of label, called muted labels. They are labels that are only used to retrieve an element from it but that can not be retrieved from an element. For HRI application, it allows the robot to understand some words but to not use them when it speaks. For example, for the restaurant Mac Donalds, we want the robot to understand "Mickey D's" in the US, "Mekkes" in Germany, or "McDo" in France. However, we prefer the robot to use the true name.

2.3 Managing partners' estimated knowledge

A major feature of Ontologenius is the ability to manage several ontology instances, to represent both the robot's knowledge and the estimation of its partners' knowledge. In this section, we first present how the multiple instances are managed by Ontologenius. We then present two advanced uses: the deep copy of an instance and the representation of multiple knowledge states in a single instance.

2.3.1 Ontologenius multi-instances principle

What we refer to when we use the term instance is an instantiation of the previously presented architecture. It thus represents a single Knowledge Base being either the one of the robot or an estimation of the Knowledge Base of one of its partners. To manage multiple instances, we use the architecture represented in Figure 2.7.

The architecture of each instance stays the same as previously, this means that each instance has its own ROS instance and can be queried and updated independently. However, to void to manually run several instances and not have any conflict on the ROS topic and service names, we add an instance manager. This manager owns all the instances and can create new ones as well as deleting existing ones. In addition, it attributes an identifier to each instance. It is often the identifier of the agent it represents. With this identifier, each instance creates dedicated and non-overlapping ROS services and topics. The manager itself owns a ROS interface to perform the deletions and creations.

When a new instance is created, it runs in a dedicated thread. Consequently, having multiple instances does not slow down Ontologenius. The manager keeps a reference to the instance and can have access to the knowledge of each of them without relying on ROS. It allows the manager to propose queries involving several

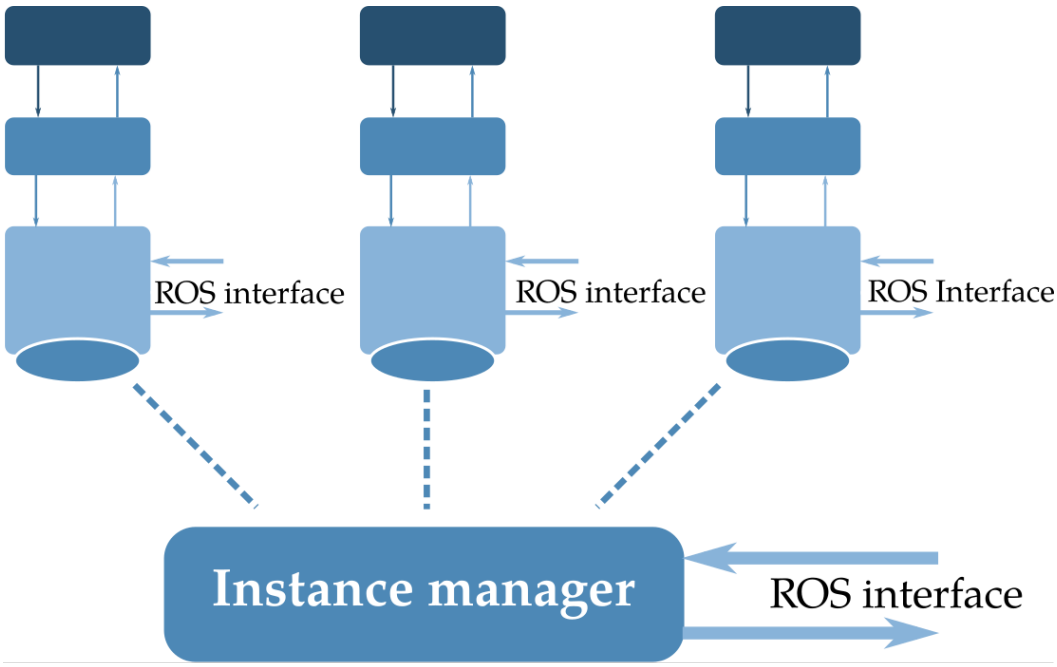

Figure 2.7: The software architecture of Ontologenius to manage multiple Knowledge Bases in the form of an ontology. An instance manager makes the link between each instance (see Figure 2.4) and has a direct access to each semantic graph to propose queries.

instances. For example, the only existing one for the moment computes the difference of knowledge between two instances, about a given entity. With such direct access to the semantic graphs, it allows easy and efficient belief divergence checking.

Finally, because each instance has an identifier, when one is killed, it stores the maintained knowledge in an OWL file referenced with this identifier. Consequently, when an instance is started, it can automatically load the previous state of the KB, using the instance identifier.

2.3.2 Catching knowledge at a given moment

The instances representing the robot's KB and the estimation of its partners' KBs aim to be continuously updated by perception processes. However, some processes, like task planning, need a stable version of a KB. In addition, some processes like (once again) task planning may need to modify a KB to represent hypothetical future states of the KB. Nevertheless, if a process modifies the same KB that is updated by the perception, it would neither represent the future state, neither the current states.

To avoid such a situation, Ontologenius provides a mechanism to copy an instance. It performs a "deep copy", keeping all the inferred knowledge, the inference

traces, or the reasoners' configurations. The resulting instance becomes completely independent from the previous one and can be modified without altering it.

The only difference is that an instance resulting from a copy will not be saved into an OWL file when it will be deleted. It is a draft, to test possible states.

2.3.3 Exploring several possible mental states at once

Having a dedicated instance to represent possible futures is useful, however planning processes could need to maintain multiple states at a time. A deep copy is far from being a free operation as presented in the result section of this chapter. It takes time, especially for large KBs. Moreover, we cannot imagine to maintain too many instances at the time[3].

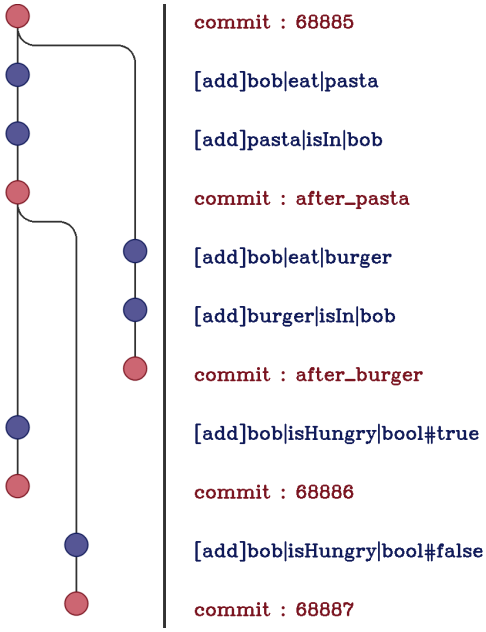

Figure 2.8: An example of a versioning tree drawn with Ontologenius. In red are the commits and in blue the changes. The temporal order of the operations is respected from top to bottom.

To face this issue, we propose with Ontoogenius a kind of versioning system. First of all, this system is only available on instances resulting from a copy, as the others do not aim to represent multiple states. Taking the term of the git versioning system, we perform some modifications on a KB and all the changes are tracked. When a user considers that all the changes represent a version, he/she performs a "commit". For example, in use for task planning, a commit would correspond to a plan step. This commit has an identifier, and a parent commits. Wanting to go back to a previous version of the KB, we can checkout this version thanks to

[3]From our unlucky tests, the process died after the creation of 82 instances with small knowledge bases.

the identifier. The versioning system thus applies the modifications in the inverse order to create the previous state. From there we can thus have multiple versions of the same KB but it only represents a linear evolution. Most of the planning systems work with branching, allowing from one state to imagine multiple future states. Where git would use branches also, with ontologenius, we do not and we can simply perform a new commit on an old one. It automatically creates a new branch without explicit management of it. We can thus easily checkout a commit of another branch. However, once a branch is created, it will never be possible to merge it with another one. Consequently, it creates a tree.

From the commit identifier, the users are free to set it or to let Ontologenius setting it.

Finally, for debug purposes, Ontologenius can generate an XML file representing all the versioning, with the commits and the changes. An executable allows then to generate an image of it to understand what an algorithm has performed. An example of such an image is presented in Figure 2.8.

2.4 Using Ontologenius in robotic applications

Now we explain Ontologenius architecture and its management of instances, in the current section, we present how to use it once it is launched. We explain how knowledge is inserted at run-time and how knowledge can be retrieved. Finally, we present tools to facilitate its use with API and GUI.

2.4.1 Updating the knowledge base

In order to represent the state of an evolving environment, updating the KB is a key feature for software managing KB. First of all, the update process is asynchronous in Ontologenius. Updates are filled in an internal buffer which is periodically analysed. For a usual instance, updates are looked up at 20hz. Such a frequency is sufficient for updates coming from perception processes. For the instances coming from a copy, the update process runs at a higher frequency of 1000hz, meaning that a published update will start to be processed at most 1ms after its publication. The need for a higher frequency for instances coming from copy is designed not to hamper planners using it and having to explore a large number of states[4]. The goal of such instances is to make a caption of the state of a KB at a given instant and then to freely modify it to run an external process like task planning. A delay of 50ms would not be acceptable for this kind of process. Once the update process is triggered by a non-empty buffer, it runs until the buffer is empty. In this way, even if a lot of updates are published at once, no additional delay is introduced.

Since the updates are asynchronous but that some components may need to be sure that all updates have been applied before continuing, Ontologenius raises a

[4]We could have chosen to provide a dedicated method to request the execution of the reasoners but we preferred to keep the same mechanism among the instances, whether they come from a copy or not.

signal once it has no more update to process. Before rasing this signal Ontologenius also applied all the needing reasoning so that the KB is consistent and populates with all the inferred knowledge. It allows a synchronisation for components needing it (like task planner), to start a new process on a stable KB.

The content of a single update is composed of an operator, either an addition or removal, and a triplet. In the current version, the triplet can be relations to individuals using object or data properties, relations to classes using annotation properties, inheritance links, inverse axioms on object properties, or labels to any elements. At least the subject or the object of the triplet must be already known by the system to apply the update. ALL the other elements are created with a deduction of their types (i.e. individual, class, property).

If the content of an update is a removal, all the inferences made on the basis of the original triplet are removed too. In addition, for any update, the related triplet is republished on another topic with a timestamp. For the inferred triplet, they are published by Ontologenius on a dedicated topic with a reference to the triplets having been used to perform the inference or to remove it. It allows, if needed, temporal storage of all the updates of the world with a temporal consistency about the inferred knowledge. This means that even if the knowledge has been inferred later than the original knowledge update, it is possible to temporally align them.

2.4.2 Retrieving knowledge

To retrieve knowledge from Ontologenius, two ways are available. The main query system consists in a set of low-level and parametric queries for precise and efficient retrieval. The second system is based on a more classic RDF query language that is SPARQL.

2.4.2.1 Low-level queries

The principal query system proposed by Ontologenius is a set of low-level queries organized around the four main types used in the ontology: the individuals/entities, the classes/concepts, the object properties, and the data properties. For each of these types, dedicated queries are available.

First, all four types have 10 common queries related to the exploration of the upper hierarchy and the names in natural language. We will not review them one by one but rather take two of them to give an indication of their composition and functioning.

```
string getName(string uri,
               bool take_id = true);
```

The getName query allows a user to retrieve one of the labels of a given element. If the element has multiple labels, one is selected randomly. This query is impacted by the general language configuration. If Ontologenius is configured to work in English, only English labels will be retrieved. The language configuration can be

changed at run-time. This query has an optional parameter take_id. If it is set to true, if the queried element does not have labels, its id can be used as a default label, regardless of the language setting.

```
vector<string> getUp(string uri,
                     int depth = -1,
                     string selector = "");
```

The getUp query allows a user to explore the upper hierarchy of a given element. By setting only the mandatory parameter, it returns all the elements for which the given element inherits. If A inherits from B and B from C, interrogating the hierarchy of A will give B and C. With the depth parameter, the user can set the depth exploration. Setting it to 1, the query will only return the direct hierarchy. The last optional parameter, the selector, can be used to filter the results to be returned. Among the returned elements, Ontologenius will only keep the ones inheriting from the selector. Wanting to know if entity A is of type B, one can query a getUp on A with B as the selector. If the result is empty, A does not inherit B.

To query individuals, Ontologenius provides 14 additional queries. We can query for the properties applied to a given entity, the entities it is linked with, the properties it is the range of, etc. Most of them support the optional parameters of depth and selector.

For the classes, we have 13 additional queries. We can explore the relations using annotation properties, get the classes inheriting from a given one, get the properties for which it is the domain of, etc.

For the object properties, we have 5 additional queries. We can know their domain and range, their inverse, their disjunction, or their hierarchy. For the data properties, we have the same excepted the inverse.

2.4.2.2 SPARQL-like queries

The low-level queries provide a fine and efficient exploration of the KB. Encouraging the user to select the right query depending on its need, Ontologenius does not have to perform analysis on the query itself to find how to resolve it. However, such queries can be difficult to use in a first time and are not adapted to every usage. Consequently, with Ontologenius we also provide a SPARQL-like query interface.

SPARQL, for SPARQL Protocol and RDF Query Language, is a semantic query language for databases. A SPARQL query looks like:

```
SELECT ?name
       ?email
WHERE
  {
    ?person  is    Person .
    ?person  name  ?name .
    ?person  mbox  ?email
  }
```

Through the SELECT keyword, we first specify the variables we want to get the possible binding. Here we ask for pairs of names and emails. With the WHERE block, we explain the conditions to respect in order to fill the variable. It is the query pattern. In the provided example, we define a variable (always prefixed with a question mark) person which should be of type Person. Then, for all the possible values of the variable person, we retrieve their name and email. With the SELECT keyword, we can specify a sub-set of the involved variable to be returned.

Three other keywords can be used instead of SELECT. It exists CONSTRUCT, ASK, and DESCRIBE. In addition, we can apply filters or aggregation rules. SPARQL is a really rich language but we have chosen to take the minimum as it is not the main entry point of Ontologenius.

Consequently, Ontologenius only implements the SELECT query variation with the pattern in the WHERE, as a list of triplets. Nevertheless, we have added the keyword DISTINCT, optional after the SELECT, which ensures to only retrieve distinct results.

In the rest of the document, and for better readability, we only use the patterns of the SPARQL queries when we present some.

2.4.3 Interfacing with Ontologenius

We provide to the users several tools to work with Ontologenius in the easiest way as possible. Here we present the Application Programming Interface (API) and the debugging Graphical User Interface (GUI).

2.4.3.1 The Application Programming Interfaces

To provide an abstraction of the ROS middleware, and thus of the ROS messages, we propose two APIs to use Ontologenius. One is in C++[5] and the other in Python[6]. Both are constructed in the same way and use approximately the same methods names to easily go from one to the other depending on the needs. They are composed of 13 classes for a total of more than 90 distinct methods. Such a number of methods allows a fine and precise use for the instance management, the reasoners' management, the knowledge insertion, or the ontology query. In addition, the methods support all the exploration options in an intuitive way.

[5] https://sarthou.github.io/ontologenius/cpp_API/CppAPI.html
[6] https://sarthou.github.io/ontologenius/python_API/PythonAPI.html

The API also takes advantage of advanced ROS use for the services. It provides a direct TCP connection with a recovery mechanism. For intensive query use in a limited time, the first query will take more time to establish a direct connexion and all the following will benefit from it. It thus provides efficient and safe communication without additional complexity for the users.

Finally, Ontologenius comes with five tutorials[7] covering its more basic usage to the more complex ones including the use of the versioning mechanism.

All the APIs description and the tutorials are available online on a website dedicated to Ontologenius: `https://sarthou.github.io/ontologenius`.

2.4.3.2 Debbuging tool

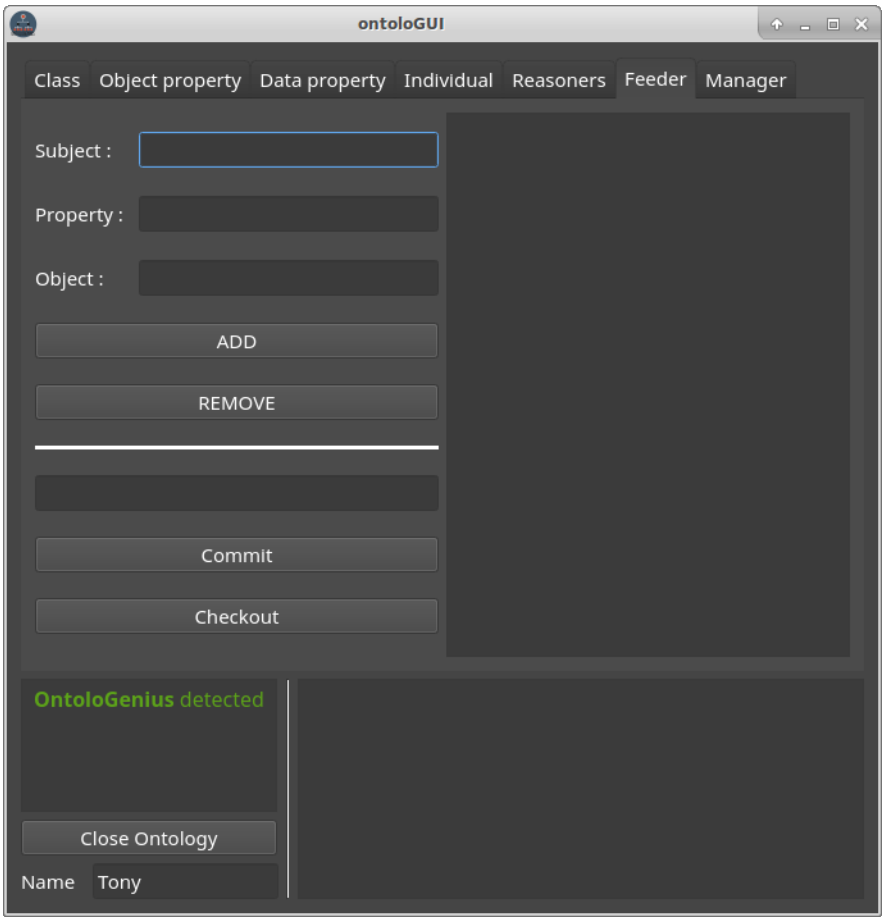

Figure 2.9: A view of the Ontologenius Graphical User Interface. With the displayed panel a user can insert or remove triplets from a given instance. On the left-hand bottom, we can see that the current ontology instance is the one of the human Tony.

[7] `https://sarthou.github.io/ontologenius/cpp_Tutorials/Tutorials.html` and `https://sarthou.github.io/ontologenius/python_Tutorials/Tutorials.html`

To help new users to take in hand Ontologenius and to help more experienced users to debug their applications using Ontologenius, we provide a Graphical User Interface (GUI). It has been developed using Qt. It is linked to ROS and provides almost all the methods available with the API like the instance management, the reasoners' management, the knowledge insertion, and the ontology query. In addition, for all the exploration queries, when hovering a button the interface provides a brief explanation of the hovered method and its equivalent in the ROS command lines.

In this chapter, we have seen that Ontologenius is based on a lot of low-level queries for a precise and efficient exploration of an ontology. This GUI is thus often used during the development of an application using Ontologenius to choose the right method to use by allowing the developer to directly test the kind of results he/she can expect with a given query. In addition, at run-time, developers can easily explore the KB to understand the origin of potential bugs in their application.

2.5 Computational performance evaluation

In this section, we evaluate the Ontologenius performance and scalability through comparison with two systems and using their own tests. First, we compare with the KnowRob system with a focus on the required CPU time to insert new knowledge and required memory to store this knowledge. The second system is ORO. With this latter, we measure query resolution time. Finally, we present some additional measurements like concept recovery time and deep-copy time.

2.5.1 Comparing with Knowrob

We start these comparisons with the KnowRob system presented in [Tenorth 2013]. This system is composed of several modules able to perform dedicated reasoning like temporal reasoning, CAD model segmentation, or object perception. All these modules are integrated around the logical programming language SWI Prolog [Wielemaker 2012]. OWL ontologies are loaded using the SWI Prolog's Semantic Web library [Wielemaker 2003] which provides an efficient and scalable way to manipulate RDF structures in Prolog. Internally, the triplet structure of the ontologies is represented as Prolog predicates. Unlike Ontologenius, the Prolog system and thus KnowRob does not aim to be used as a server but rather as a monolithic and highly integrated system. To provide a fair comparison, in the part, we use Ontologenius without the ROS communication layer. Consequently, a single process runs both Ontologenius and the test application.

[Tenorth 2017] proposes a detailed presentation of the internal representation of KnowRob and presents some performance and scalability analysis. The following comparisons have to be considered carefully. KnowRob is a wider and more mature system than Ontologenius. It proposes more advanced capability in terms of reasoning and integration. Ontologenius is more focused on HRI applications as presented earlier in this section. In this way, some simplifications that have been

done to fit at our best the KnowRob tests may impact the results. The final reason why the following results should be taken with caution is that due to the complexity of the KnowRob system, we did not perform their test on our end. The results for the KnowRob system comes from their paper [Tenorth 2017]. Consequently, in this section, we do not aim to show that Ontologenius is "better" than KnowRob. We rather want to show that Ontologenius is not out of scope in terms of performance and scalability regarding a well-established software.

For the tests we have replicated, they use the description of visual perception entities. Such an entity represents the occurrence of the perception of a given object, having a given pose at a given instant. An example of one of these entities is illustrated in Listing 2.4. The entity *cup_i* is the perceived object. The entity $VisualPerception_i$ is the perception occurrence. It uses an object property to make a link with the perceived object and two data properties to represent the object's pose and the perception time stamp. The first simplification we had to make is the representation of the matrix of position. In Ontologenius, the data types are only represented in a serialized way as no internal manipulation of these types are made. Thanks to the use of Prolog, KnowRob can manipulate such a matrix and perform operations on it. Even if it is not said if they have inverse properties or not, we add one for the *objectActedOn* property in order to know in which perception instance an object has been perceived.

```
:cup_i  rdf:type   :Cup ;

:VisualPerception_i  rdf:type          :VisualPerception ;
                     :objectActedOn    :cup_i ;
                     :eventOccursAt    [[1,0,0,2.56], ... ,[0,0,0,1]]^^RotMat ;
                     :startTime        6572^^Time .
```

Listing 2.4: Description of a visual perception entity created in a comparable way as in the KnowRob system. The description is provided in the OWL language using the Turtle syntax.

The first test consists of creating and inserting N visual perception entities in the KB and measuring the required CPU time to insert one of them. This means that we measure the average CPU time over N insertions. For each visual perception entity, we thus have to create two entities (the visual perception entity and the perceived object), two inheritance links, two raw data, and three relations (two based on data property and one based on object property). The results are shown in Figure 2.10. KnowRob has a constant insertion time around 1.3ms. With Ontologenius we do not have constant time. The first decreasing part can be explained by the asynchronous Ontologenius mechanism. It looks for updates at 20hz. Consequently, we can have a delay between the moment we publish updates and the moment they are processed. The effect of the mechanism disappears with the amount of data to process since once the update mechanism started, it does not stop while data have to be processed. The general trend is thus an increase in the required CPU time. This increase can come from the fact that Ontologenius performs reasoning, like the creation of the inverse relations, at update where KnowRob resolves it at query.

In addition, Ontologenius also performs consistency checks at updates. However, until 1,000,000 insertions and thus 2,000,000 of entities, 2,000,000 raw data, and 5,000,000 triples (2 for inheritances and 3 for relation per perception entity), the required CPU time is under 1ms.

Considering the same insertions as previously, the second test consists of measuring the required memory. The results are shown in Figure 2.11. For a high number of individuals, Ontologenius required a bit less memory. It can be explained by the fact that it has a simplified matrix representation. In addition, no information had been provided about the initial content of the KB. Both systems, therefore, require memory in the same order of magnitude.

The last test with KnowRob is about the required CPU time to perform queries. They took the same KB as made previously, with N visual perception entities. The goal here is to select randomly one of the perceived cups, then retrieve its pose. In Prolog, the query is:

?- owl_individual_of(Obj, kr:'Cup'), current_object_pose(Obj, Pose).

Using the low-level queries of Ontologenius, we have made a function doing the equivalent. It requests for all the cups, selects one randomly, retrieves the visual perception entity it is linked to, and fetches the pose. The required CPU time is reported in Figure 2.12.

Concerning KnowRob, the query time is almost constant, jumping between one and two milliseconds. It has to be noted that in Prolog, the time measurement resolution is about 1 millisecond. Nevertheless, due to the jump, we can assume that

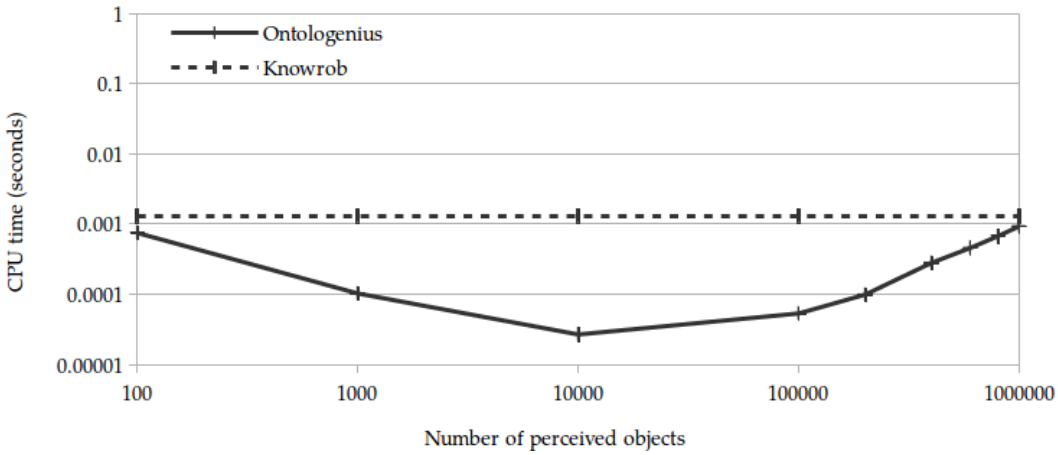

Figure 2.10: Comparison of the CPU time required to create a large number of object perceptions in the knowledge base. Each inserted object corresponds to a visual perception entity linked to a cup, a pose matrix and a timestamp (see Listing 2.4. Ontologenius is used without the ROS communication layer to provide comparable usage and thus results.

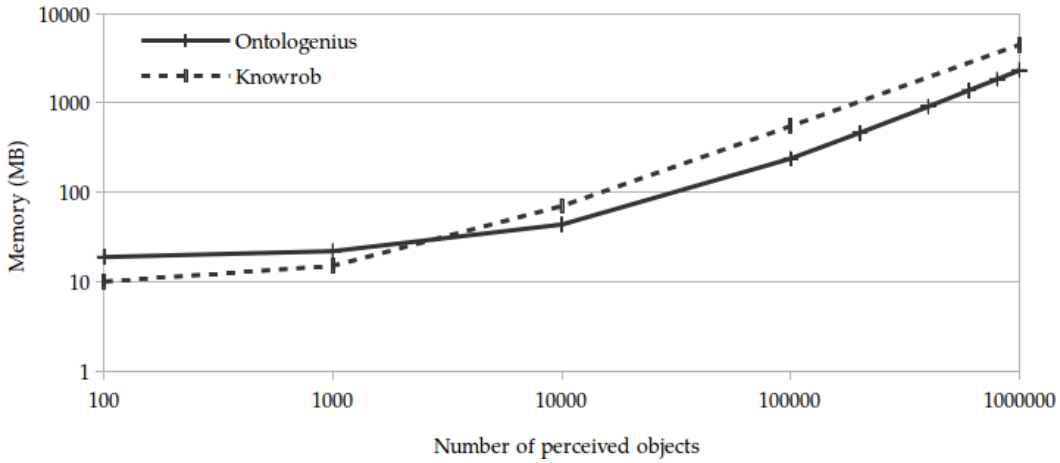

Figure 2.11: Comparison of memory required to create a large number of object perceptions in the knowledge base. Each inserted object corresponds to a visual perception entity linked to a cup, a pose matrix and a timestamp.

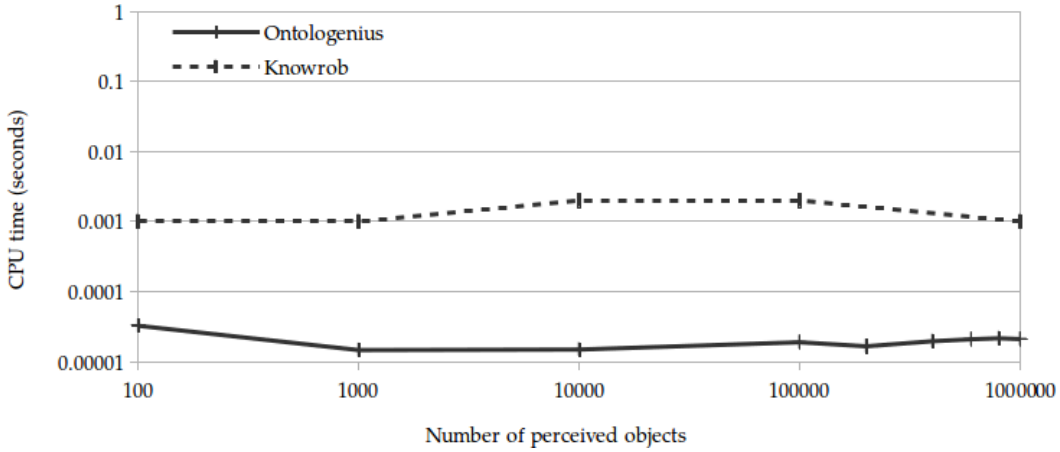

Figure 2.12: Comparison of the required CPU time for querying the pose matrix of a randomly selected object after that N perception entities have been created. Ontologenius is used without the ROS communication layer to provide comparable usage and thus results.

the real value is not far from the millisecond. For Ontologenius, the query time is also almost constant, with values around 0.02ms. This significant difference, with an average factor of 75, can be due to the more precise queries provided by Ontologenius and to the fact that Ontologenius does not have to solve inference at query time. For the more precise queries, submitting a query to Prolog, KnowRob has to perform a kind of search among the KB to make the submitted predicates true. At the difference, Ontologenius requires the programmer to refine and decompose the high-

level query, allowing the execution to be more efficient. For the inference, most of the relations are inferred at the update, like the ones coming from inverse properties. At query, Ontologenius only has to go through the existing relations and only reasons about the classes and properties inclusion axioms. For example, if we had a query for all the objects, no direct link would have been created between the cups instances and the object class, this part would thus have been solved at query thanks to exploration.

In light of the presented results, even if both software have not the exact same goal, way of use, and maturity, we can at least conclude that Ontologenius is not out of scope with acceptable performance and scalability. Since Ontologenius provides fewer functionalities or at least different ones, the results are encouraging.

2.5.2 Comparing with ORO

For the second comparison, we select the software ORO [Lemaignan 2010]. At the difference of KnowRob which does not address the same HRI applications, ORO does. It works as a central server, usable by all the components of architecture and is able to manage several instances at a time. In this way, it has been designed for HRI applications. We can however note three major differences from a technical point of view. It is based on the Jena framework for the RDF triplets storage and uses Pellet [Sirin 2007] for the reasoning part. Pellet supports OWL-DL expressiveness level to perform reasoning where with Ontologenius we are still at the OWL-lite expressiveness. Regarding the way to query the ontology, ORO uses the Jena SPARQL interface. It also provides some built-in high-level queries made for specific applications. Finally, to communicate with the server, ORO uses a TCP connection with a Telnet protocol.

In [Lemaignan 2010] they propose three test queries to assess the software performance. We thus reproduce these tests and we add a second dimension being the scalability. Consequently, the three test queries have been performed on KB of different sizes. Before presenting the results, let us see the content of the test ontology. This ontology does not aim to be semantically correct. We first have three object properties: *isAt*, *isOn*, and *isUnder*. The property *isUnder* is described as being the inverse of *isOn* and *isOn* as being a sub-property of *isAt*. Moreover, the property *isOn* has for domain the class *animal*[8]. No additional information about this ontology is needed for the presented results.

Before performing any query, N triplets are inserted in the KB. These triplets are of the form:

individual_i isOn apple

In this triplet, *apple* is an entity already existing in the ontology and which have as type the class *Plant*. Since the property *isOn* has for domain the class *animal*, all the *individual_i* should be inferred as inheriting of this latter class. Moreover,

[8]We said that the ontology does not have any real meaning.

because of the inverse property, the inverse relation *(apple isUnder individual_i)* should exist.

The first test query concerns inheritance. The goal is to retrieve all the individuals inheriting the class *animal*. Regarding the inserted relations, if N relations have been inserted, the query should return the N individuals consequently created, meaning the N *individual_i*. In SPARQL the query would be:

?i rdf:type animal

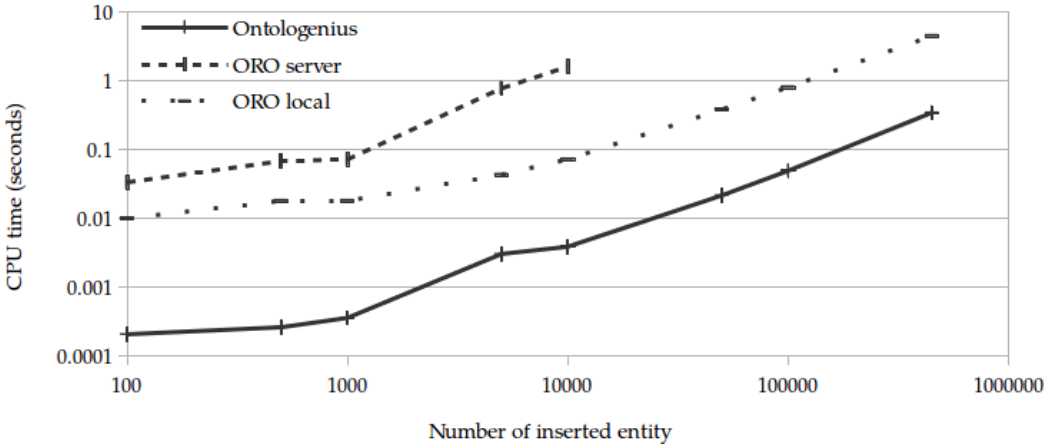

Figure 2.13: Comparison of the required CPU time for querying all the entities being animals, after that N entities have been created. Ontologenius and ORO are used with their own communication layer. To assess the impact of the communication layer on ORO performance, we also provide measures without the ORO communication layer (ORO local).

For the test, ORO has been queried through the SPARQL query where Ontologenius has been queried through low-level queries. To provide results representing the way the software should be used, both have been queried through their respective API. This means that results include the communication times. However, to not fall in a comparison of the communication layers but rather of the software, we have also tested to query ORO without the communication layer. The three plots (one for Ontologenius with the communication layer and two for ORO one with the communication layer and one without) are presented in Figure 2.13.

First of all, we can note that with the communication layer we have not achieved to go over 10,000 entities with ORO. Rather than a limitation of the software itself, here it is a limitation of the test which requires the retrieval of too many entities. Nevertheless, with ORO without the communication layer, we succeed to go to 450,000 entities. We can see that Ontologenius performs far better than ORO, even when ORO is tested without its communication layer. This means that even if the communication time impacts the results, it does not spoil them. In server usage, Ontologenius is more efficient of a factor around 255 on average. In this

test Ontologenius takes advantage of its reasoning process applied at the update, allowing a more direct search.

For the two next queries, we only present the results in server usage as we just saw that this additional time is not inordinate. The second query uses the inverse properties. It aims at retrieving all the entities being under the *apple*. As for the previous query, we thus expect to retrieve the N newly created entities. The corresponding SPARQL query is:

?i isUnder apple

The third query is more complex and uses a conjunction. The query aims at retrieving all the entities having a relation involving the property *isAt* toward an entity being of type *Plant*. Since all the newly created entities have a relation using the property *isOn*, which is a sub-property of *isAt*, toward the entity *apple*, which inherits of the *Plant* class, the N individual are expected to be retrieve. The corresponding SPARQL query is:

?i isAt ?p, ?p rdf:type Plant

The results of these two queries are presented for both software in Figures 2.14 and 2.15. They are almost the same as those of the first test query. Ontologenius performs on average better of a factor around 250 and we previously saw that the ORO communication layer has a limited impact which cannot explain such difference.

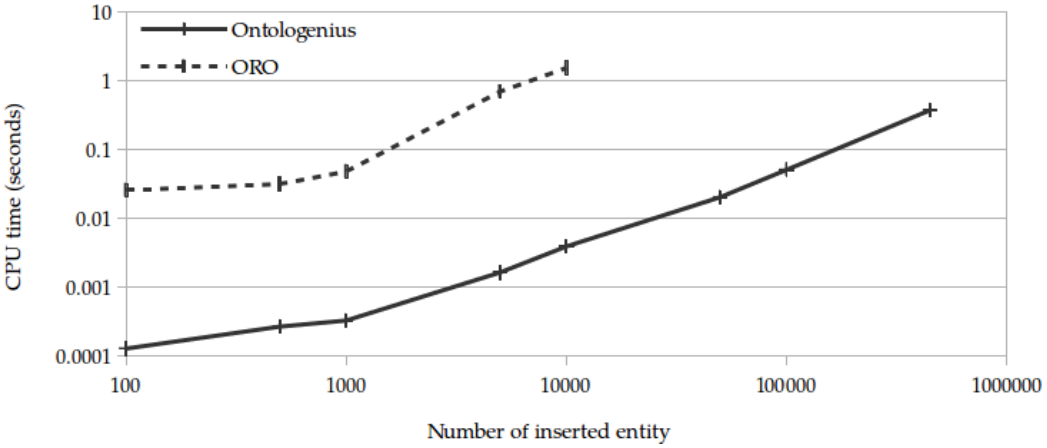

Figure 2.14: Comparison of the required CPU time for querying all the entities having a relation of the kind "isUnder" toward a given entity. Since the invert relation has been inserted for all the created entities, the software has to solve the inverse relation to answer the query. Ontologenius and ORO are used with their own communication layer.

Through the comparison with ORO, considering its expected usage, we can fairly conclude that Ontologenius seems to be more adapted for performance requirements. Nevertheless, the test itself could be discussed. The fact that the

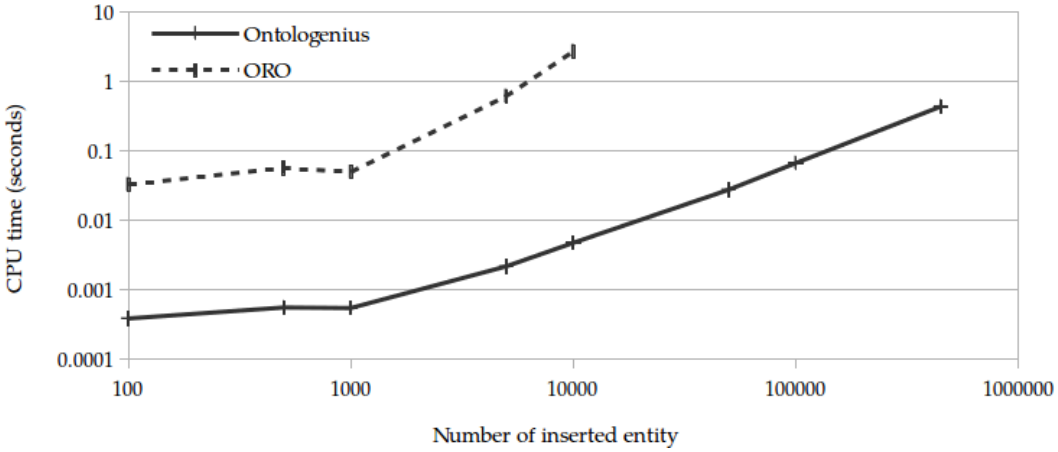

Figure 2.15: Comparison of the required CPU time for querying all the entities having a relation of the kind "isAt" toward an entity of type "Plant". Ontologenius and ORO are used with their own communication layer.

software has to retrieve such an amount of entities does not represent real use-cases. Moreover, in the light of such equivalent results from a query to another, we can question what is really measured here. Looking backwards to the tests with KnowRob, the query time of Ontologenius was quite constant where here we have a constant increase. This test is however complementary with the ones of KnowRob, showing Ontologenius ability to retrieve a large number of entities. In addition, even with few entities (100), the performance gap is already significant, confirming the usability of Ontologenius even if it is not based on any established libraries like Jena or Pellet.

2.5.3 Additional tests

With the previous tests, we have performed comparisons based on tests proposed by other systems. We now present some additional tests to assess the performance of Ontologenius. Among the number of features, we have selected two of them.

We first propose to compare the required CPU time to retrieve an entity by its name in natural language and its identifier. For this test, we have taken 466,508 English words. These words have a length going from 1 letter to 45 letters with a means of 9.42 words. For each step of the test, N words have been randomly selected and inserted in the ontology as individuals. In addition, to each entity, we have defined a name in natural language, which is the same as the identifier. The N inserted entities have then been retrieved with their name in natural language and their identifier. The results are presented in Figure 2.16. The retrieve of an entity through the use of its identifier is constant with a required CPU time of around 0.046ms. At the difference, retrieving an entity through the use of its name in natural language is highly impacted by the size of the KB. We reach a CPU

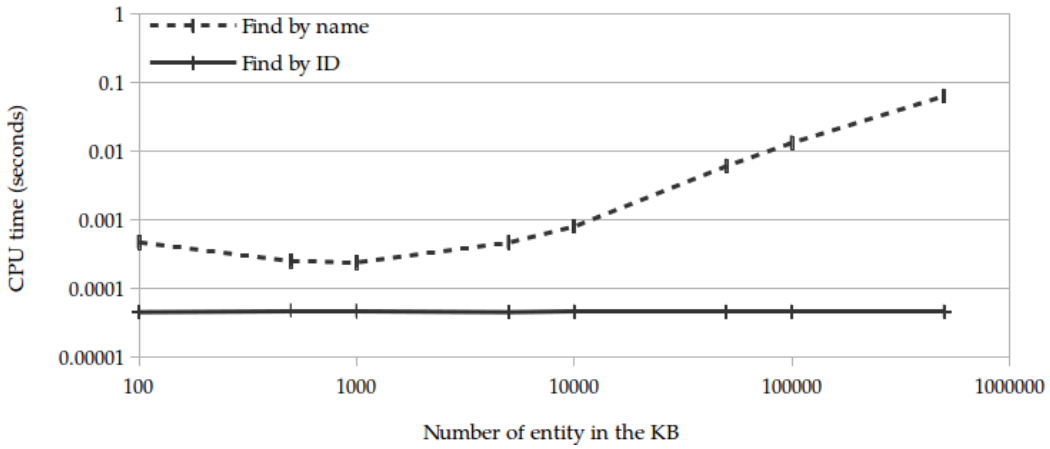

Figure 2.16: Comparison of the required CPU time to retrieve an entity by its name in natural language and its identifier. Ontologenius is used with its communication layer for both cases.

time of 0.62ms for a KB with 450,000 entity. This result highlights the fact that the name in natural language should be used, with Ontologenius, as an interface with the human partner and that all the algorithms should rather use the identifier. Even if this result could seem evident, it still shows the constant CPU time with the use of identifier, and that, even with large KB.

The second feature of Ontologenius we choose to evaluate is the instance copy. As explained earlier in this chapter, with Ontologenius we are able to make a deep copy of an ontology instance, resulting in a new and independent instance. Moreover, we have presented a kind of versioning mechanism allowing to represent multiple knowledge states in a single instance. We explained that the expected

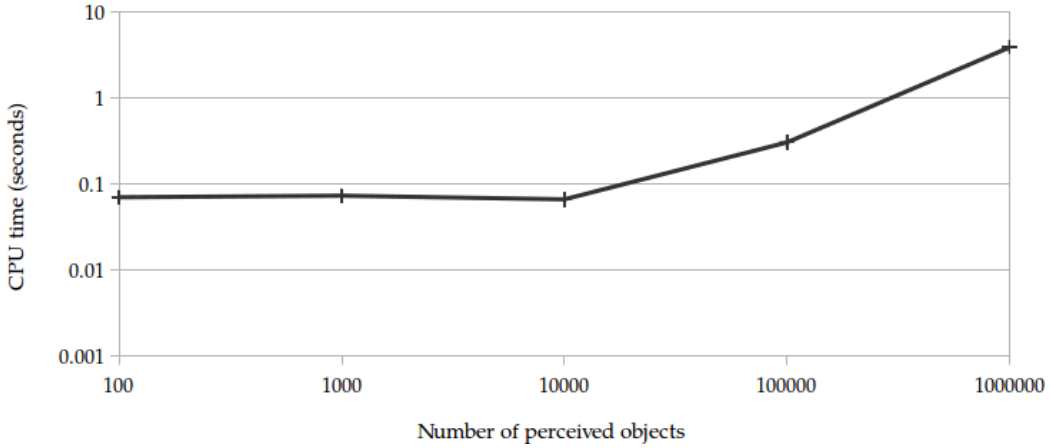

Figure 2.17: Evaluation of the required CPU time to perform a deep-copy depending on the number of perceived entity, represented in the Knowledge Base. Ontologenius is used with its communication layer.

usage is to perform a single copy to create an independent instance, not altered by other components, then to use the versioning mechanism on the newly created instance. The presented test aims a measuring the required CPU time to perform such a copy, depending on the ontology size.

For this test, we have taken again, the ontology content proposed by the KnowRob tests. It consists of inserting visual perception structures. We thus add N structures resulting in the creation of 2*N entities, 2*N raw data, 2*N inheritance links, and 3*N relations. Once the N structures were inserted, we performed an instance copy and measured the required CPU time. The results for N going from 100 to 1,000,000 are represented in Figure 2.17. As expected, the required CPU time grows with the KB size. Starting around 100ms for the smallest KB, it grows to 4s for the largest one. Even if the results seem acceptable regarding the job to perform, we can however conclude that such an operation should not be performed too frequently. This reinforces the idea of doing a single copy then using the versioning mechanism if we want to represent many states of the same KB.

Chapter 3

Elaborating a route for a human partner based on semantic knowledge

Contents

3.1 Introduction 52
3.2 Related work 53
3.3 The Semantic Spatial Representation 55
3.3.1 The SSR classes 56
3.3.2 The SSR properties 57
3.3.3 An example of description 59
3.4 Finding routes: A two-level search 60
3.4.1 The region-level: Trim down the search 61
3.4.2 The place-level: Refine the search 63
3.5 Generating an explanation in natural language 66
3.5.1 Reconstructing the paths 66
3.5.2 The robot putting itself in your shoes 68
3.5.3 A pattern-based generation 69
3.6 Experiment in the mockup and the real environment 70

In this chapter, we propose a representation of indoor environments in a semantic way by building a minimal but sufficient ontology, oriented around a route description task. This representation is then used to plan a route that a human has to follow to reach a goal destination. Finally, from the computed route a second process allows the robot to verbalize the route, seen as a procedure to be followed, allowing the guided human to mentally navigate along the route.

The contribution presented in this chapter is excerpted from our work, published in the proceedings of the Spatial Language Understanding (SpLU) 2019 workshop [Sarthou 2019a]. In this manuscript, the contribution is more detailed and discussed. This work is part of the MuMMER project, aiming at developing a robot guide in a mall. At the end of this document (Chapter 8), a chapter is dedicated to the presentation of the project and the integration of the current contribution in a robotic system.

3.1 Introduction

We all have already been requested, or have ourselves asked for, for a route in a city, in a shopping center, or more simply in a house. When providing such information to a person, we perform what is commonly called a guidance task. Even if it can seem trivial for us, developing a robot able to perform it can be challenging. In this chapter, we choose to focus on the sub-task consisting at generating the explanation sentence. This sub-task is called the route description. To perform it, we first need a set of knowledge about the environment in which the guided person will walk, such as the paths, the intersections of the paths, or the elements alongside them. Then, we need a set of "good practices" to provide a route easy enough to follow and to remember.

In the Human Robot Interaction (HRI) field, robots guides have been studied intensively and deployed into shopping centers [Okuno 2009], museums [Burgard 1999, Clodic 2006, Siegwart 2003], or airport [Triebel 2016]. From a knowledge representation point of view, we can notice the use of metrical representations [Thrun 2007] or topological representations [Morales Saiki 2011] to represent the environment in which the robot acts. Since we focus on the route description task, we consider that the robot does not accompany the human to his final destination but rather explains how to reach it. Consequently, the metrical representation will not be considered as being mainly used for navigation purpose [Thrun 2007]. To perform more specifically a route description, topological knowledge is not sufficient. In addition to the topology of the environment, the robot needs to know the types of the elements composing the environment and their names in natural language. Some contributions have thus tried to mix metric or topologic representations with semantic ones to hold this additional knowledge [Satake 2015b, Chrastil 2014, Zender 2008]. However, mixing them can create a lack of uniformity among the overall knowledge representation. In this way, creating a unique representation allowing a robot to compute routes and expressing them could ensure uniformity among the knowledge.

Even equipped with a consistent representation of its environment, the robot has to find a route not for itself but for the guided human. A robot accompanying the human only has to determine a path, adapted to its capacities and interpretable only by itself. Providing a route to a human, the route has to be adapted to the human capabilities and knowledge. For example, in an outdoor environment, we will not give the same route for a car driver or a cyclist. In the context of a mall, we will not give a route with stairs to a mobility-impaired person or to someone with a shopping cart. Once computed, the robot has to explain the route. Where interactive maps only have to highlight a path, here, the robot has to generate a sentence that the human will be able to memorize. For sure the robot will not instruct a human with a sentence like "walk 30 meters them turn -90 degrees". This would not be adapted. The use of orientation and reference to elements of the environment will be needed through a sentence like "walk until the florist then turn left". We thus want the robot to generate plans understandable and executable by

the human.

The first contribution of this chapter is a **unified representation** of an indoor environment using an ontology, to include both topological and semantic knowledge. Then, on the basis of this representation, we propose a first algorithm to **find a suitable route** to be explained to a human and a second algorithm to **verbalize a route** in an appropriate way.

First, we review the literature on route description. Then, we introduce our unified semantic representation under the name of Semantic Spatial Representation (SSR). We then present the algorithm used to compute the route and in a second time the algorithm to verbalize the previously computed route. We end this chapter with experimental results on both mockup and real environments.

3.2 Related work: the route description task

In the literature, a route description task is defined as being a particular kind of spatial description. First, from a cognitivist point of view, Denis in [Denis 1997] has identified three main cognitive operations used to generate such a spatial discourse: 1) the activation of an internal representation of the environment, 2) the planning of a route in this representation, 3) and finally the formulation of the procedure to follow. From a computer science point of view, Cassell [Cassell 2007] considers the second operation as the fact of finding a set of route segments, each connecting two important points, and the third operation as chronologically explaining the route segments. In the same way, Mallot in [Mallot 2009], sees the second operation as the fact of selecting a sequence of places leading to the objective, and the third as managing declarative knowledge to choose the right action to be explained at each point of the sequence. While both second operations are equivalent, the third about the formulation of the procedure are rather complementary.

The route description task has been extensively studied in linguistic through verbal and textual communication to understand how humans communicate spatial knowledge. The goal of such studies has been to identify the invariants but also the good practices ensuring the success of the task. Through five experiments in both urban and indoor environments, Allen [Allen 2000] has identified three basic practices seen as being important for communicating knowledge about routes. They can be summarized as follows: a) respect the spatiotemporal order, b) concentrate on the information about the points of choice and c) use landmarks that the listener can easily identify.

This latter practice about the use of landmarks, also called reference marks, has been identified by Tversky [Tversky 1999] as a critical piece of information for the success of a route description. [Tversky 1998] finds that in addition to information about actions, reorientation, and direction, 91% of the guidance instructions contains the use of landmarks. These results confirm the ones of [Denis 1997]. Montello [Montello 1993] tries to identify when the use of landmarks appears in a description. Defining the *Vista* space as being the area within sight and the *Environ-*

mental as being the rest of the environment reachable through locomotion, he finds that guides usually use landmarks when the target places are no longer in the *Vista* space but in the *Environmental* one. Moreover, with regard to [Tversky 1999], the use of landmarks appears during an explanation of a direction changing. In addition, their choice is based on salient features over a route description [Nothegger 2004].

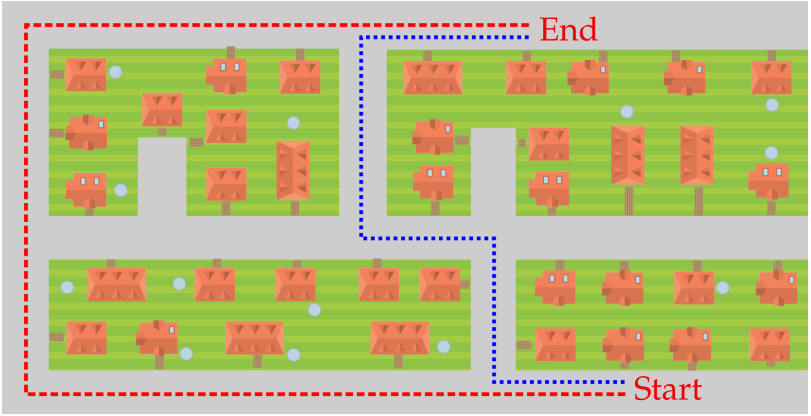

Figure 3.1: Comparison of two routes in terms of complexity and length. Even if the blue route (. . .) is the shortest, many directions changing are required. Each of them is a risk for the guided person to make a mistake and be lost again. The red route (- - -), although being a bit longer, is easier to explain and to remember, and has few directions changing.

Even if the use of landmarks helps at understanding direction changing by anchoring in the environment the action to be performed, there is still a risk for the guided person to make a mistake, taking the wrong path. While the length of the route is an important criterion in the choice of a route, its complexity has also to be taken into account when we need to explain it. Morales [Morales 2015] argued that reducing the route complexity, in terms of the number of stages composing it, should be preferred to its length. This feature reduces the risk of mistakes concerning the choice to make along the route and also has an impact on its understanding and memorization. This criteria of minimal explanation can be compared to the Grice's Maxim of quality [Grice 1975]. In the example of Figure 3.1, some should prefer to explain the red route rather than the blue one, even if it is longer, since it is easier to remember.

Finally, to explain the same route, Taylor [Taylor 1992] has noticed that a speaker can use two kinds of perspectives. First, the *survey* perspective tend to adopt a bird's eye view point of the environment, meaning a top view of it like looking at a map. With this perspective, the speaker refers to the different landmarks of the route with respect to one another. They are thus referred to using terms including north-south-east-west. This perspective is opposite to the *route* perspective. With such a perspective, the speaker mentally navigates along the route, making an imaginary tour of the environment. As a result, he refers to the landmarks with respect to the future guided person position along the route. The landmarks are

thus referred to using terms like left, right, front, or back. [Taylor 1996] notices that the survey perspective is generally used for open environments whereas the route perspective is generally used in environments with already identified paths. For indoor environments, the route perspective should thus be preferred to facilitate route understanding and memorization.

3.3 The Semantic Spatial Representation

Satake [Satake 2015a] has developed a topological graph for the route search process. To understand the human request about the destination node in the graph, they build an ontology, providing a semantic description of them. The description informs about the places types (e.g. cloth shop, restaurant, ...), names and nicknames, and the sold items. However, as explained in [Morales 2015], such a map with annotated elements does not provide a suitable base to generate route explanations using a route perspective as it misses directional information. Nevertheless, we can note that the use of an ontology is suitable to describe the meaning of the elements of the environment. In the same way, one can also describe other elements than shops, like stairs, elevators, entrances, or escalators. To unify the representations, we could thus, represent the topology in an ontology. Since both are graphs, this representation should be feasible and provide more information about the elements along the paths. As explained in the previous chapters of this thesis, ontologies are widely used for knowledge representation, to capture the meaning of the elements of an environment but also to encode knowledge dedicated to the robot's internal process. In this way, they provide a good interface between the robot's processes and its human partner in terms of interaction.

To semantically represent an environment, Kuipers in [Kuipers 2000] introduces the Spatial Semantic Hierarchy (SSH). He defines a 'topological level' composed of three main elements being the places, paths, and regions, as well as relations between them. To create our semantic description of an indoor environment, we thus choose to take it as a basis. However, this representation does not use an ontology, we thus present how we extend it to represent the topology in an ontology. The resulting representation and its underline pattern are what we will call the Semantic Spatial Representation (SSR).

Since this contribution is focused on a pattern to describe environments and algorithms using it to compute routes, the presented representations are made by hand. However, many contributions trend to automatically generate a topological graph from geometric measurements (e.g. Region Adjacency Graphs [Kuipers 2004], Cell and Portal Graphs [Lefebvre 2003], hierarchical models [Lorenz 2006], or from natural language [Hemachandra 2014]). Even if we do not have used such techniques for the moment, our contribution could benefit from these works, solving the complexity of creating such a representation by hand.

In the following, we first present the used classes and properties before ending with an example of a description.

3.3.1 The SSR classes

Kuipers has defined three distinct concepts being the **region**, the **path**, and the **place**. To represent the topology, we have refined the concepts of place and path. Here below, we define these concepts and their refinement. The resulting class hierarchy is representing with the TBox of Figure 3.2.

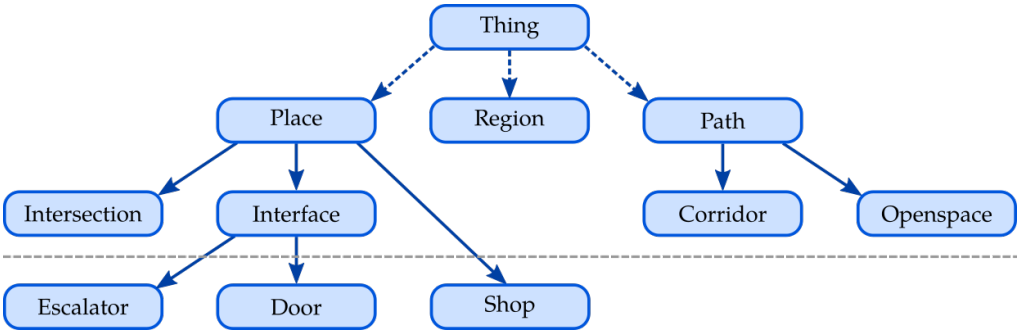

Figure 3.2: Representation of the TBox (classes hierarchy) of the Semantic Spatial Representation used to describe the topology of an indoor environment. While the top part is inherent to the SSR, the bottom one extends the latter to provide more granularity.

Region: It represents a two-dimensional element, drawing an area. An area is a subset of an overall environment. A description of an environment must include at least one region being the entire environment itself. However, defining multiple regions aims at creating a finer representation. For example, for multi-storey buildings, we will at least represent each floor by a distinct region. Regions can be described as being nested if needed.

Path: It is a one-dimensional element, like a line. We can materialize it as an element along which it is possible to move by following it. Even if it is not directly semantically represented, a path must have a direction in order to describe other elements in relation to it. A path can be refined into:

- **Corridor:** It represents a kind of path having a distinct beginning and end. In this way, a corridor cannot be a loop. The direction of the corridor can be chosen arbitrarily. However, its direction defines the position of its beginning and end. Consequently, it also defines the right and left of the corridor. The direction does not mean that the corridor can be used in a unique way but will be used to describe the elements along with it.
- **Open space:** It is a kind of path which does not have any defined beginning or end. It thus forms a loop. It can also be viewed as a "potato-shaped" describing the outline of open space. It materializes the possibility of turning the gaze around the room and the fact of not having to go through a defined

path to reach one of its points. In a building, a hall would typically be described with such a path.

Place: It represents a point of zero dimension[1]. It can either represent a physical or symbolic element. In the context of a mall description, a shop would inherit the place class as a point representing its door can be sufficient to describe it. To represent the topology, we refine it into:

- **Path intersection:** It represents the connection between two and only two paths. It is thus a waypoint to go from one path to another. In the case of a crossing between three paths, three intersections have to be described.
- **Interface:** It represents the connection between two and only two regions. It is thus a waypoint to move from one region to another. It can be physical, like a door or a staircase, or symbolic like a passage.

To better catch the difference between paths and places, it can be related to the differences between the types of rooms made by [Andresen 2016]. Some rooms in a building such as a house have the main use to circulate from a room to another, they are corridors. Other rooms have an explicit use that is not the traffic, they are places even if you can move in.

3.3.2 The SSR properties

The relations introduced by Kuipers aim at representing the connections of places and paths, the order of the places along a path, and the elements in the regions. The main relations are the following:

$on(place, path)$	$place$ is on $path$
$order(path, place1, place2, dir)$	the order on $path$ from $place1$ to $place2$ is dir
$right_of(path, dir, region)$	$path$, facing direction dir, has $region$on its right
$left_of(path, dir, region)$	$path$, facing direction dir, has $region$on its left
$in(place, region)$	$place$ is in $region$

Through these relations, we can first see a slight difference in our use of the main elements. The regions are used to group a set of places along a path, avoiding in this way to describe each individual place as being at the right/left of a path. That way, our use of the concept of region will thus require more description for each place but has the advantage to give a finer granularity of huge environments representation with a first level of abstraction. With it, we could imagine a short prior description like “it is on the first floor”.

[1]For sure it has a 3D location, but here we are speaking of its use in navigation from one place to another.

A major limitation of these relations is that they cannot be used in an ontology as they are not all in the form of triplets. The issue comes from the need for a direction to determine the order and the side positions. The properties we introduce aim at fixing this issue and give a more precise description. For example, with the representation of Kuipers, we cannot have a place at the edge of a path. All the properties presented here can be extended with their inverse (e.g. $isIn$ and $hasIn$) for a more expressive model and thus easier handling. The resulting property hierarchy is represented with the partial RBox of Figure 3.3

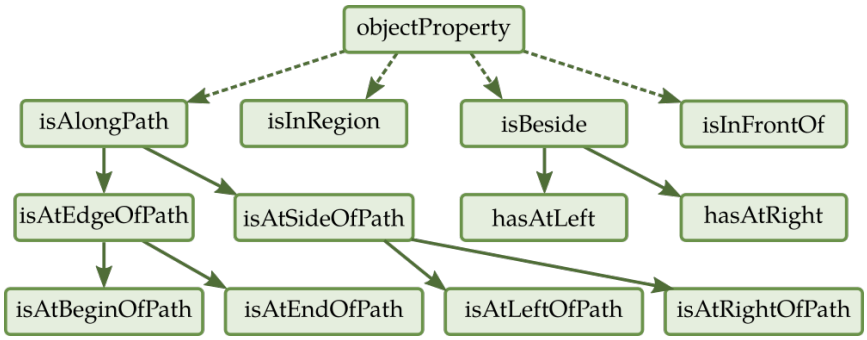

Figure 3.3: Representation fo the RBox (properties hierarchy) of the Semantic Spatial Representation used to describe the topology of an indoor environment.

isInRegion: $isInRegion(path/place,\ region)$ describes the fact that a *path* or a *place* is in a *region*.

isAlongPath: $isAlongPath(place,\ path)$ describes the fact that a *place* is along *path*. Depending if the path is a corridor or an openspace, a finer description can be needed:

- $isAlong(place,\ corridor)$: Since a corridor is a line, the places along the path can first be described as being either on a side of the corridor with the property $isAtSideOfPath$ or at the edge with the property $isAtEdgeOfPath$. Thanks to direction of the corridor, these properties can be refined with $isAtBeginOfPath$ and $isAtEndOfPath$ for the places at an edge, and $isAtRightOfPath$ and $isAtLeftOfPath$ for the places along a side of the corridor. Even if the direction of the corridor is not directly represented in the ontology, since the places have been positioned in relation to the direction, we can retrieve it.

- $isAlong(place,\ openspace)$: For open spaces, since they do not have a beginning or an end, places are only defined as being along with an open space.

isBeside: $isBeside(place1, place2)$ describes the fact that $place1$ is beside $place2$. To really represent their order, we use the properties $hasAtLeft$ and $hasAtRight$.

The choice of these properties is made by positioning themselves at the place and facing the path the place is along.

isInfrontOf: $isInfrontOf(place1, place2)$ describes the fact that $place1$ is in front of $place2$. This property is not mandatory for all the places but provides a finer description. The more it is used, the more the verbalization of the itinerary will be easy. We will see in the sentence generation section that it is however important to always define a place in front of an intersection. This information will be used to determine if the guided human will have to go left or right in some cases. If there is no described place in front of an intersection, we can use a *emptyPlace* class that would inherit the *place* class.

To simplify the description few axioms are made. The property $isInfrontOf$ is defined as reflexive. The properties $hasAtRight$ and $hasAtLeft$ are defined as inverse from one another. Finally the chain axiom $isAlong \bullet isIn \rightarrow isIn$ is used to not have to set the $isIn$ property for each place. It can be deduced thanks to the path they are along.

3.3.3 An example of description

To illustrate the use of our SSR, we will describe the shop pl_5 (with the red floor) represented in Figure 3.4. For the description of this place, we will use the previously introduced classes and properties.

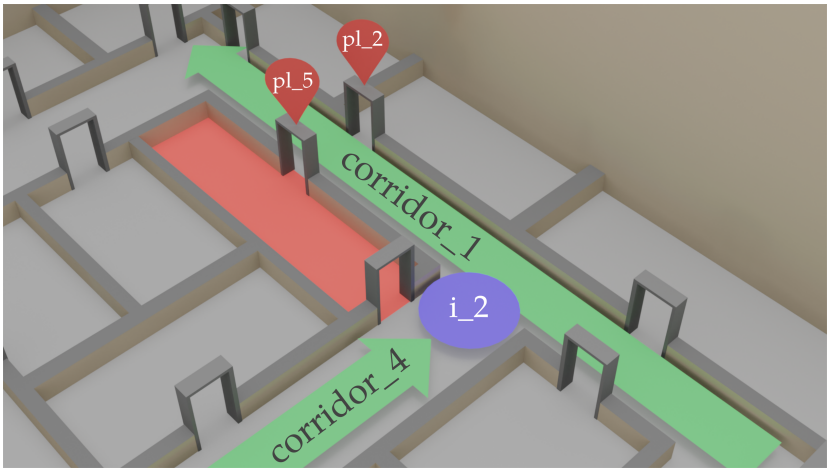

Figure 3.4: An example of environment with two corridors, an intersection, and two shops represented as places. All the elements are in the same region. The arrows represent the corridors directions.

To describe our environment, we start with the description of its paths. The corresponding ontology is available in Listing 3.1. In the example, we have two corridors. We assume that both are in a common region named $region_1$. The paths do not need further description. To pass from one to the other, we create an

intersection, here denoted *i*_2. To describe its position along with the two paths we have to take a look at the paths' directions. For *corridor*_4, the intersection is at its end edge. For *corridor*_1, the intersection is at its left.

```
:corridor_1  rdf:type      :Corridor ;
             :isInRegion   :region_1 .

:corridor_4  rdf:type      :Corridor ;
             :isInRegion   :region_1 .

:i_2  rdf:type         :Intersection ;
      :isAtEndOfPath   :corridor_4;
      :isAtLeftOfPath  :corridor_1.
```

Listing 3.1: Description of the two corridors and their common intersection in the OWL language using the Turle syntax.

Once the paths and intersections are described, we can describe the other places of the environment. For this example, we only describe the shop *pl*_5. We would do the same process for all the others. The place *pl*_5 is first a shop, which is a particular kind of place. Regarding the corridors, it is at the left of *corridor*_1. However, it also has an entrance overlooking *corridor*_4. We can thus describe it as being along two paths. Depending on the starting point of the description, we will be able to instruct for one of its two entrances. For *corridor*_4, *pl*_5 is at its left. Now that the place is positioned along paths, we have to describe its position according to its adjacent places. Facing *corridor*_1, the intersection *i*_2 is at the right of *pl*_5 and the shop *pl*_2 is in front of it. Facing *corridor*_4, the intersection would be at its left. However, *pl*_5 is on a side of the corridor while *i*_2 is at one of its edges. Consequently, we do not have to describe any order between them. The resulting ontology part is provided in Listing 3.2.

```
:pl_5  rdf:type          :Shop ;
       :isAtLeftOfPath   :corridor_4 ;
       :isAtLeftOfPath   :corridor_1 ;
       :hasAtRight       :i_2 ;
       :isInFrontOf      :pl_2.
```

Listing 3.2: Description of the shop pl_5 using the SSR in the OWL language using the Turle syntax.

3.4 Finding routes: A two-level search

At this point, we have created and built a representation of the environment using the Semantic Spatial Representation (SSR). Using this representation, we now want to compute routes from one place to another. We already saw that even if the length of a route can be a criterion, the complexity of the description is a more important criterion when we have to describe it. Moreover, depending on the guided person, some routes could be prefered to others, to avoid stairs, to favor elevators, or to reduce the use of potentially crowded paths.

In this section, the goal is to provide multiple routes so that we can then choose the best route based on personal preferences, during the interaction. In order to reduce the complexity of this exhaustive search, especially for large scale environments, we propose to work at two levels:

- **First** the region-level. It considers only regions and interfaces such as doors, stairs, or elevators. In such a way, we will not consider paths in regions not leading to the goal destination. If we are on the ground floor and want to go to a place also on the ground floor, exploring other floors would be useless.
- **Then** the place-level. It provides complete routes computation including paths and intersections within regions.

3.4.1 The region-level: Trim down the search

In large-scale environments such as multi-storey buildings, routes computation can lead to combinatorial explosion. Exploration at the region-level can decrease this effect, giving a first high-level exploration. To study this first level, we take the example of Figure 3.5. It represents a building composed of five regions. Each is linked to others through the use of interfaces, being doors, stairs, or escalators. In this environment, the goal is to guide a human from the flag "start" to the flag "end". Quickly analysing this figure, we can see that the human will have to pass by regions 1, 2, and 3. To pass from *region_1* to *region_2* the only interface is

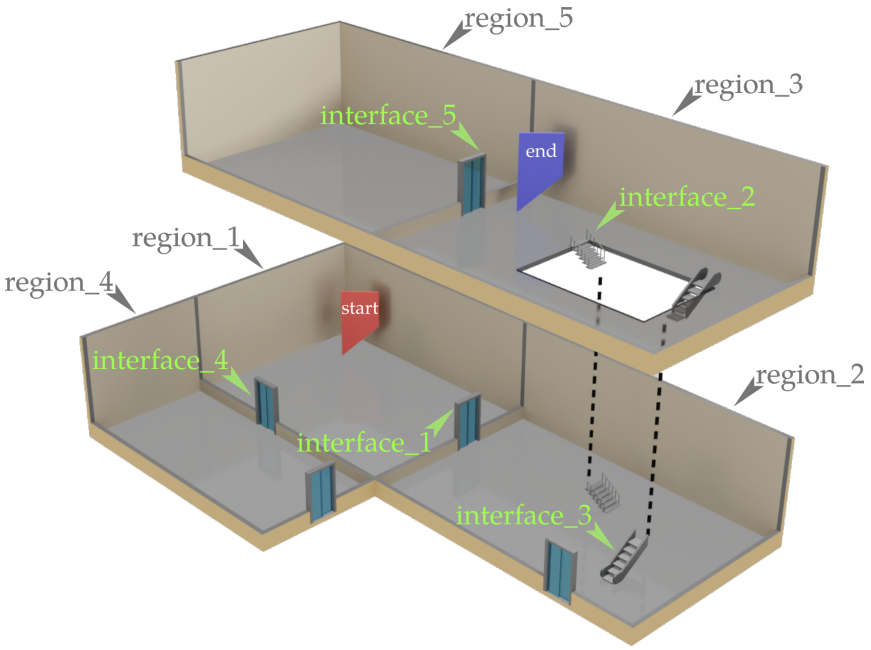

Figure 3.5: Representation of an environment at the region-level. Regions are linked through interfaces. We know that the starting point of the search is in *region_1* and the goal place is in *region_3*.

interface_1. To pass from *region_2* to *region_3*, two interfaces can be used, either *interface_2* or *interface_3*. The exploration of the two other regions is useless.

For this first level of exploration, we only use the regions and the interfaces as they link the regions. Since the places are along paths and the paths are in regions, we know in which region is each place. Using the property *isInRegion*, we get the starting region and the goal region respectively from the start place and the end place. In our example, we get that the human has to go from *region_1* to *region_3*. These two regions define the initial state and goal state of a search algorithm. Considering the regions as the nodes of a graph and the interfaces as its edges, we can thus apply a graph search algorithm to find routes at the region-level.

Algorithm 1 Exhaustive Graph Search algorithm for region exploration

```
 1: function EXHAUSTIVE_REGION_SEARCH(problem)
 2:   state ← (problem.initial_region, _)
 3:   frontier ← a FIFO queue of state
 4:   frontier ← INSERT(state, frontier)
 5:   solutions ← an empty set
 6:
 7:   loop
 8:     if EMPTY(frontier) then
 9:       return solutions
10:
11:     state ← POP(frontier)
12:     if state.current_region is problem.goal_region then
13:       solutions ← INSERT(state, solutions)
14:     else
15:       for all interface of state.current_region.hasInRegion s.t. ∉ state do
16:         for all region s.t. interface.isInRegion do
17:           if region ∉ state then
18:             frontier ← INSERT(state∪ (region, interface), frontier)
```

Since we want to provide several routes, we use the exhaustive graph search algorithm represented in Algorithm 1. Such an exhaustive search allows us to take into account all possible interfaces. In this algorithm, we consider a state as being a list of pairs composed of a region and the intersection used to come to the region. The initial state (line 2) thus takes the initial region and no interface. When we explore a state (line 11) we first test if its current region (the last one in the list) is the goal region (line 12). If it is, rather than returning the state, we simply add it to a solution set (line 13). If the current region is not the goal one, we retrieve from the ontology all the interfaces connected to the current region thanks to the property *hasInRegion* which is the inverse of *isInRegion*. For each of these interfaces not already used in the state (line 15), we get the regions it is connected to (line 16). With each of these regions not already used in the state (line 17), we finally create a new state which is added to the frontier (line 18). The algorithm only stops when no more state has to be explored (line 8). By including the knowledge base

exploration directly inside the search algorithms, it is not necessary to extract a topological graph with nodes and edges. It is carried out within the search algorithm without preprocessing.

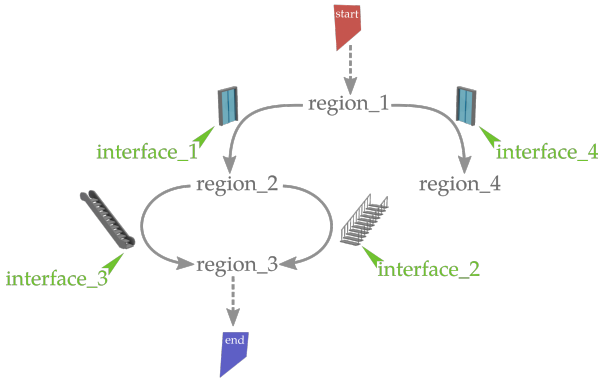

Figure 3.6: An illustartion of the exploration process at the region-level for the example of Figure 3.5.

The exploration performed by this algorithm on our example is illustrated in Figure 3.6. The *region_4* has been detected as being a deadend and the *region_5* has not been explored at all. The two found solutions are the routes:

$$region_1 - interface_1 - region_2 - interface_2 - region_3$$
$$region_1 - interface_1 - region_2 - interface_3 - region_3$$

This type of result makes it possible to quickly eliminate unnecessary regions to be analysed next and thus reduces the complexity for a more detailed search at a second time. This technique is similar to what is done for GNSS road navigation systems where the main roads are studied upstream of secondary roads with pyramidal (or hierarchical) route structure [Bovy 2012].

3.4.2 The place-level: Refine the search

Place-level search is based on the Region-level search results with the aggregation of start and end places. The format changes from:

$$region - place - region - ... - region$$
to
$$\mathbf{place} - region - place - region - ... - region - \mathbf{place}$$

Place-level search works from one place to another through a single region. We have therefore divided the previous solutions to meet this constraint. This step aims to reduce complexity again. Indeed, several routes can pass through the same region with the same places of departure and arrival. The inner route can thus

be calculated once and for all. In our example, the division gives five sub-routes instead of six:

$$start - region_1 - interface_1$$
$$interface_1 - region_2 - interface_2$$
$$interface_2 - region_3 - end$$
$$interface_1 - region_2 - interface_3$$
$$interface_3 - region_3 - end$$

The place-level algorithm aims to replace each region in the sub-routes with a succession of paths and intersections. Focusing on *region_1*, illustrated in Figure 3.7, we need to find a route from the start place to *intersection_1*. Using the property *isAlongPath*, we get from the ontology that the starting place is along *corridor_1* and the interface 1 (*i_1*) is along *corridor_5*. Solving it by hand, a possible solution would be to take *corridor_1*, then pass by the intersection 1 (*i_1*) to reach *corridor_5*.

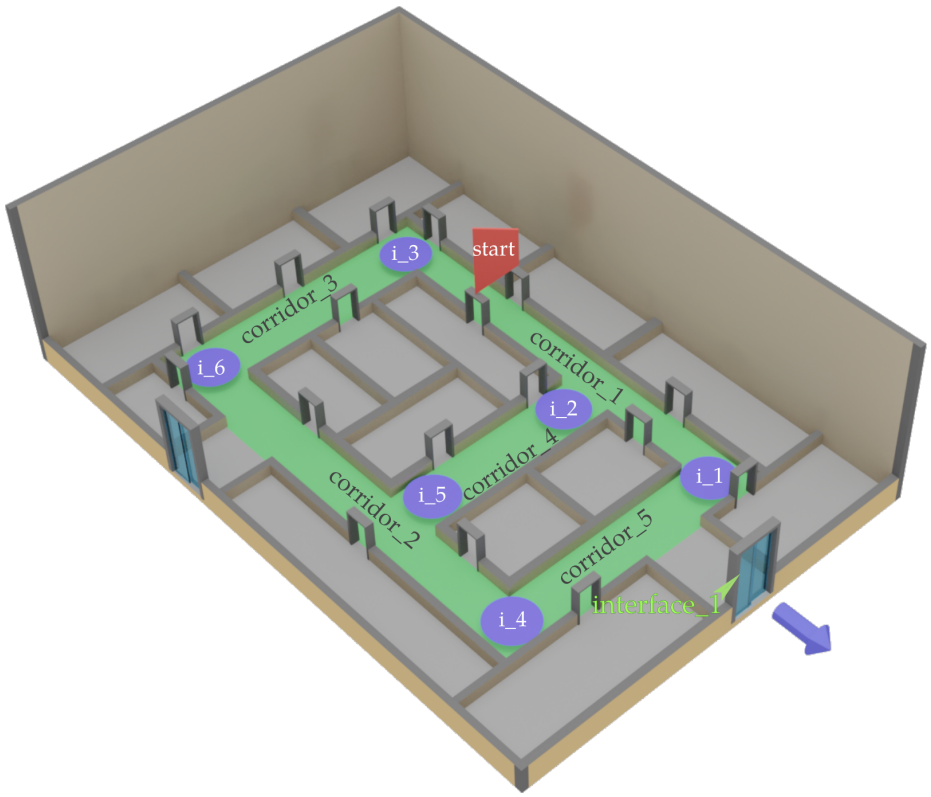

Figure 3.7: Representation of *region_1* at the place-level. A region is composed of paths (here corridors only) connected through intersections. We know that the starting point of the search is along *corridor_1* and the local goal place is in *corridor_5*.

Whereas at the region-level an exhaustive search was acceptable due to the limited number of regions and their limited connectivity, such a search cannot be applied to the place-level. In our example of *region_1* we could find four routes, without loops, to go from the start place to *interface_1*. In a more realistic environment, it could be of the order of several tens. We thus choose to only get the less complex route, at the place-level, for each sub-route. Since the complexity of the route is only impacted by the number of items composing the route, a not cost-based algorithm can be used. Moreover, considering the places of the region as nodes and the paths as edges, we thus have a graph. To find the optimal route in a graph without costs associated with the edges or nodes, we choose a breadth-first algorithm. Its pseudocode is provided in Algorithm 2.

Algorithm 2 Adapted breadth-first search algorithm for paths exploration. This version does not return the first valid solution but all the solutions having the same minimum length.

```
1:  function EXHAUSTIVE_PATH_SEARCH(problem)
2:      frontier ← a FIFO queue of state
3:      explored ← a set of explored path
4:      solutions ← an empty set
5:      min_length ← inf
6:
7:      for all path ∈ problem.initial_paths do
8:          state ← (path, _)
9:          explored ← INSERT(path, explored)
10:         if path ∉ problem.goal_paths then
11:             return state
12:         else
13:             frontier ← INSERT(state, frontier)
14:
15:     loop
16:         if EMPTY(frontier) then
17:             return solutions
18:
19:         state ← POP(frontier)
20:         for all intersection s.t. state.current_path.hasAlong ∧ ∉ state do
21:             for all path s.t. intersection.isAlong ∧ ∉ state do
22:                 if path ∈ problem.goal_paths then
23:                     state ← state∪ (path, intersection)
24:                     if state.size ≤ min_length then
25:                         solutions ← INSERT(state), solutions)
26:                         min_length ← state.size
27:                 else if path ∉ explored then
28:                     frontier ← INSERT(state∪ (path, intersection), frontier)
29:                     explored ← INSERT(path, explored)
```

This algorithm has been adapted to better fit our problem. First of all, a place can be along several paths. In the case of our starting place, in addition to be along *corridor_1*, we have described it as being also along *corridor_4*. Performing a search for each combination of departure path and destination path would be time-consuming. Where a breadth-first algorithm commonly takes one initial state and one goal state, we have adapted it to assume several initial and goal states. This modification is visible from line 7 to line 12 of our algorithm. For each initial path, we create an initial state and test if it is one of the goal paths or not. We also put them all in the explored set to avoid to search route passing by them since we are already along with them. The second adaptation comes from the number of solutions. A breadth-first search only returns the shortest solution. However, if several solutions exist with the same length, only the first found is selected. Since we want to propose several routes, we do not exit the algorithm once a solution is found but rather bound it from there with the variable *min_length*. At line 23, any state having a size bigger than the minimum length is forsaken. Once all the routes under exploration reach this limit the frontier becomes empty and the algorithm stops.

This algorithm is applied to each sub-route found at the previous stage. The overall routes are then recomposed with the results found at the place-level. The final routes are in the form of:

$$start\ place - path - place - ... - place - path - goal\ place$$

3.5 Generating an explanation in natural language

This section describes the third cognitive operation of [Denis 1997] to generate a spatial discourse: the formulation of the procedure. Similarly to [Cassell 2007], we define a route description as a set of route segments, each connecting two important points. Taking the format of the previously found routes, the segments correspond to each path of the route and the two places linked to. The segments have then to be verbalized in a chronological way. To generate the full explanation, we start with the determination of the actions to be performed and the computation of the positions of the landmarks along the route, taking the route perspective. Then, with this information, we present how we elaborate an explanation in natural language.

3.5.1 Reconstructing the paths

In the same way as [Tversky 1999], we consider that to enable their explanation, each segment of a route corresponds to a triplet: orientation, action, and landmark. A route description can thus be realized through the repetition of three steps: designating a landmark, reorienting the listener, and continuing the progression by prescribing an action.

The issue with the environment representation used to compute the route is that it is not directly usable to generate the formulation of the procedure. With the current representation, the orientation and action are too complex to extract, given that they depend on the direction by which the guided human arrives. Nevertheless, thanks to the additional information provided by the Semantic Spatial Representation, it is possible to interpret the representation in relation to the estimated future position of the human. This interpretation is what we call an internal representation. It is an interpretation of the SSR based on the current task. To generate the triplets of the segments, we create such an internal representation for each segment of the route to explain. The segments are thus represented and analysed independently from one another.

For the corridors, we start by defining four sets. These sets are used to represent the places at the left of a path, the places at its right, the places at its beginning, and the places at its end. Given the corridor c_i such that $(c_i, Corridor) \in C$, the set of places Pi_{right} at its right is:

$$Pi_{right} = \{pl_i \mid (pl_i,\ isAtRightOfPath,\ c_i) \in R\}$$

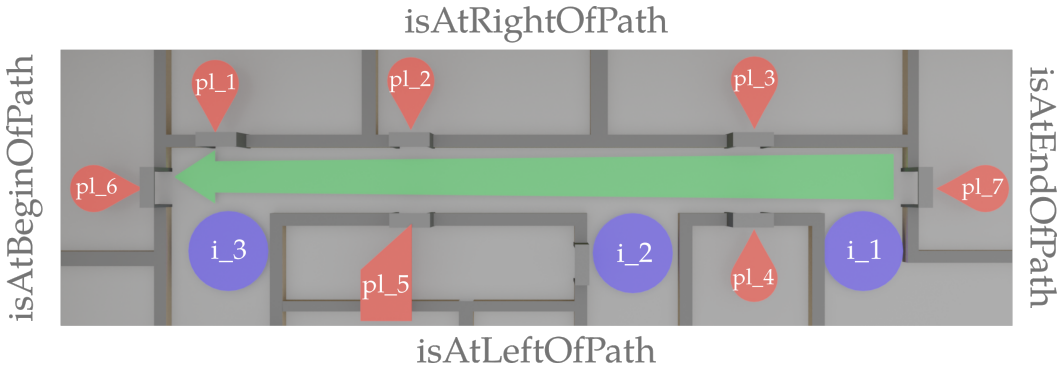

Figure 3.8: Internal representation of a *corridor_1* extracted from the semantic representation. Thanks to the specialisations of the *isAlongPath* property (i.e. *isAtLeftOfPath*, ...) we know on which side of the path each place is. With the specialisations of the *isBeside* property, we can reconstruct the order of the places for each side.

The three other sets are built in the way using relations including the properties $isAtLeftOfPath$, $isAtBeginOfPath$, and $isAtEndOfPath$. We then consider these sets as being partialy ordered sets (posets) through the binary relation $<$. For two elements a and b of a set Pi, we said that $a < b$ if $(a,\ hasAtRight,\ b) \in R$. Thanks to these four partially ordered sets[2] we can reconstruct the arrangement of

[2]The sets are not totally ordered because we do not consider the transitivity of the relations $hasAtRight$ and $hasAtLeft$ directly in the ontology. Consequently not all the elements can be compared.

a corridor like in Figure 3.8 for *corridor_1* of our running example and of Figure 3.9 for an automaticaly generated visualisation of the reconstruction of a corridor of a real mall (used in Chapter 8). To match the positions of places on opposite sides, we use the relations involving the property $isInFrontOf$.

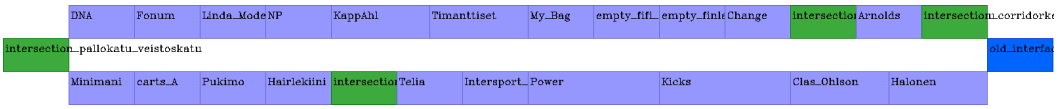

Figure 3.9: An example of an automaticaly generated visualisation of a reconstruction based on the semantic knowledges. The places in green are intersections, the place in blue is an interface, and the others are shops.

For the open spaces, we only generate one partially ordered set representing all the locations along with it. We still use the property $isInFrontOf$ to improve the placement of the places.

3.5.2 The robot putting itself in your shoes

Once we have an internal representation of each segment, we can determine the procedure that the user must perform. Cassell in [Cassell 2007] mentions that an action, a reorientation, a progression, or a positioning must be carried out at the end of each segment. The end of one segment being the beginning of the next, we choose to determine the actions at the beginning of each segment (which corresponds more precisely to our internal representation). It allows to work on one path at a time. This rule is formalized in [Mallot 2009] as:

$$\text{“}choosing\ action\ A_i\ at\ place\ P_j\ will\ lead\ to\ place\ P''_k\text{”}$$

This determination of actions can be made thanks to our internal representation. It is represented in red on the top of Figure 3.10 assuming the human to come from the place pl_5 (P_j) and P_k as being one of the other places of the corridor. For example, if the local place to reach is pl_1 the action will be *“turn left”*. If the local place to reach is pl_2 the action will be *“go in front of you”*. If the place is on the same side, the ordering of the set is sufficient to determine the action. If the place is on an edge, the action depends on the side from which the humans come and the edge to reach. If the place to reach is on the other side of the path, we have to use the property $isInFrontOf$ to determine the action to perform.

The information in blue on the two sides of Figure 3.10 gives the orientation of the sub-goal place P_k taking into account the previous reorientation. With this orientation information, we can inform the guided human about the location of the next step once the previous action has been performed. Wanting to reach intersection i_1, we can give the explanation *“take the corridor at your right”*.

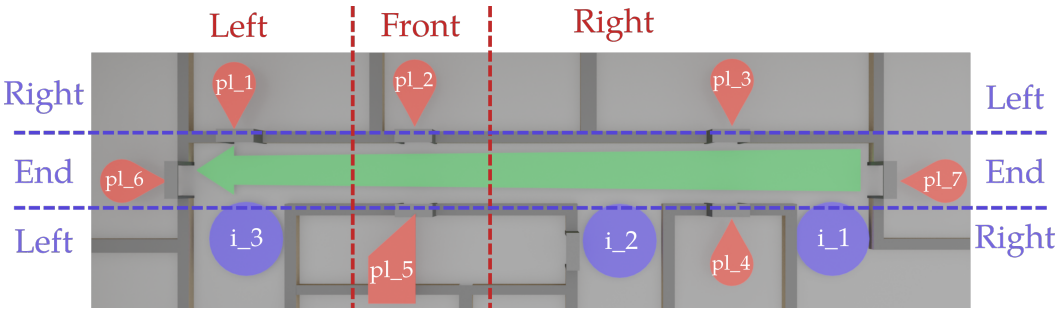

Figure 3.10: In red (on top) is represented the resolution of the action to be performed (i.e. going in front, turning left, or turning right) from the starting place *pl_5*. In blue (on the sides) is represented the resolution of the position of the place to reach depending on the performed action. Starting from place *pl_5*, to reach the intersection *i_1*, the guided human will have to turn right (red) then the place will be on his right (blue). Starting from place *pl_5*, to reach the place *pl_3*, the guided human will have to turn left (red) then the place will be on his right (blue).

The full sentence would thus be *turn right then take the corridor on your right*"[3]. Consequently, taking into account the orientation of the guided person after an action, we provide directions in the route perspective. The guided human can thus perform an imaginary tour of the environment [Cassell 2007].

Working segment by segment in the order given by the route search algorithm, we necessarily generate the explanations with a spatiotemporal ordering. This criterion is an important point in Allen's best practice in communicating route knowledge [Allen 2000].

The latest critical information is the landmarks. With our internal representation, we provide all the landmarks (corresponding to places along the paths) around which action must be taken. In the previous example, the sentence may be confusing because there will have two corridors on the right. We are able to refer to place *pl_4* to ground the action. With this new information, a sentence without ambiguity would be *"turn right then take the corridor at your right straight after pl_4"*. With this last sentence, we can no longer go to *i_2*.

3.5.3 A pattern-based generation

In order to generate the sentences in natural language, we first analyzed explanations used by human guides in a human-human study [Belhassein 2017]. From this study, we have identified four types of explanation components: those corresponding to the beginning of the route, the end of the route, the progress in it, and the particular case of routes with only one step. Each type has sub-types of explanation components. For example, for the sentences expressing progress in the route, we can mention two sub-types: sentences expressing a redirection action and

[3]We can note that this sentence can also lead to *i_2*. We will fix it after.

the ones expressing the fact of continuing forward. Thus, each explanation belongs to a sub-type according to the kind of information it expresses. This classification allows us to have multiple ways to express the same information, varying the robot turn of phrase and vocabulary when talking to a person. To represent similar sentences and to be able to generate sentences with variations, we have grouped sentences with close lexical structures. Each sentence is then represented with its variations through the use of patterns. An example of pattern is: [“you will”], [“see”, “find”],[“it”, “/X”], [“on”], [“your”, “the”], [“/D”],[“side”,“when you walk”, “”]. When using a sentence, the variations are randomly chosen with a uniform distribution. We can notice in the previous example the use of place-holders such as X or D. The place-holder X corresponds to the name of an element of the environment and D to a direction (right or left). We also implemented the use of a Y variable to refer to another element of the environment (a landmark) and DY to refer to its location. The pattern *“You will see X at the DY of Y”* becomes for example *“You will see Burger King at the right of Espresso House”* after instantiation. If a sentence requires a variable that the algorithm is not able to extract from our internal representation, then we select another sentence with the same meaning or another variation of the sentence that does not require the previously used variable.

In our English verbalization, we have 238 unique patterns to express the end of a route description, 64 to express its beginning, 20 to express the continuation, and 66 for the special case of a route with a single step. All the patterns are available in appendix A.1.

For the example of Figure 3.7, a possible verbalization would be:

“Go through this corridor, turn right straight after pl_4, then you will see the door on your left.”

For the same route, the system would be able to generate another sentence like:

“Walk across this corridor and turn right straight after pl_4. After that, the door is on the left there, straight after pl_7.”

3.6 Experiment in the mockup and the real environment

The SSR was first used to describe a mockup mall to test the related algorithms. This representation has then been tested in a real mall to study its applicability in a larger environment. Table 3.1 indicates the number of elements described in both environments. The number of places does not correspond to the sum of the shops, interfaces and intersections since much more elements have been described, such as ATMs, restrooms or carts location. These two environments are detailed in Chapter 8.

To assess the usability of such a description and its related algorithms, we propose to measure the resolution time of several places on the real mall representation.

	Mockup mall	Real mall
Places	83	251
Shops	19	140
Interfaces	11	18
Path intersections	10	50
Paths	11	41
Regions	5	3

Table 3.1: Number of elements described in the mockup and real environment.

Over the 140 shops of the mall, we have selected 26 of them in such a way that they all are along a single and distinct path. Computing routes for all the places along the same path would give the same results. Figure 3.11 reports the resolution time for each target shop in function of the maximum route length. The resolution times include the ROS communication time to query the ontology. It is thus a real use case. The first element to note in these results is the maximum resolution time of 40.79 ms for a maximal route length of 5. This place is a bit particular as being in another region than the one of departure and that only one interface allows to reach this place. It can be seen as an unadapted description due to a special case where an additional region would be required. Nevertheless, resolution times under 45ms are still acceptable for HRI applications.

Speaking about the division of the environment into regions, we have here circled all the measures related to places into the same region. In blue are the places being in the same region as the departure place. In green ones are places in a small region composed of 4 paths and three interfaces. In red are the places on the first floor of the building. Eleven interfaces have been identified to pass from the departure region to the first floor. The results over these three regions draw three distinct and non-overlapping clusters. Results in the same region as the departure one give resolutions under 5ms with a maximum route length of four. Results over the small region give resolutions time between 12.5ms and 15ms with routes going from 4 to 5 paths. Results over the last region create an important gap in terms of resolution time with a minimum around 28ms.

These results could be discussed at length. On one hand, we could say that the division into regions has a positive impact on the resolution time by restricting the search space. For the places in the same region as the departure one, the paths in the other regions do not impact the search. Such an explanation would be in adequation with the presented graph. On the other hand, the “high” resolution times for the region in red can be attributed to this same region division. Because we want to propose several routes, and at least one passing by each interface, we artificially increase the search space with the need of at least 22 sub-routes computation for places in a region connected with 11 interfaces (one sub-route to go from the departure place to the interface and one sub-route to go from the interface to the destination place). For the smallest region (in green) 6 sub-routes

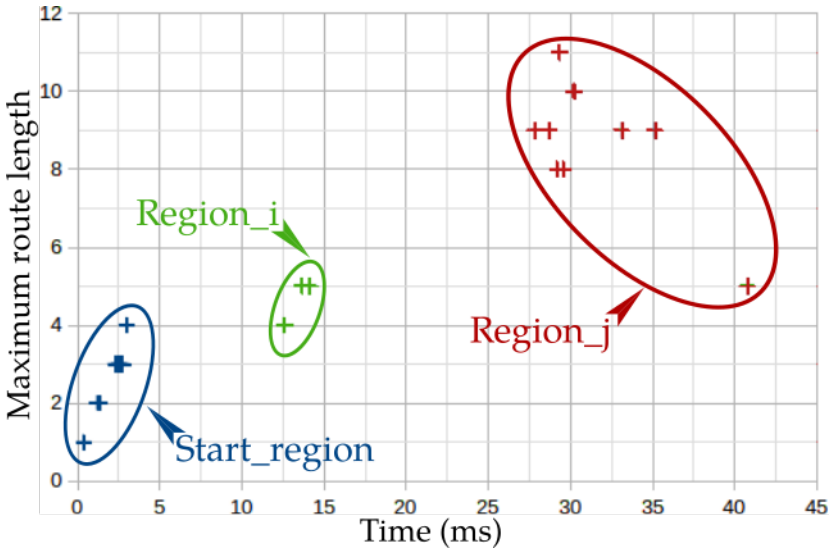

Figure 3.11: The resolution times of 26 places of the real mall related to the maximum route length found for each of them. The places are each on a unique and distinct path. We identify three clusters. The blue one on the left corresponds to places in the same region as the departure place. The green cluster on the center corresponds to places in a small region with few paths, different from the region of the departure place. The red cluster on the right corresponds to places in a third region with more paths in it and more interfaces to reach it.

computations are needed because of 3 interfaces.

To conclude the analysis of these results, we can say that regarding the need of searching several routes passing by different interfaces, the proposed algorithm is fully suitable with resolution time adapted to HRI applications. However, the number of interfaces linking two regions has a non-negligible impact on these resolution times. A careful description of such an environment is thus important and attention must be put on the division into regions.

Results about the integration of the SSR and its related algorithms in a robotic architecture will be provided at the end of this document over a chapter dedicated to the MuMMER project (Chapter 8).

Chapter 4

Ontology-based Referring Expression Generation

Contents

4.1 Introduction 74
4.2 Related work 76
4.3 Define the REG problem 80
4.3.1 The knowledge representation 80
4.3.2 Contextualization and restriction for situated REG 82
4.3.3 Expected solution: structure and validity criteria 84
4.4 Uniform Cost Search REG 86
4.4.1 Formalisation as a search problem 86
4.4.2 Algorithm choice 88
4.4.3 Algorithm presentation 88
4.4.4 Replanning and explaining failures 91
4.5 Results 92
4.5.1 Solution analysis: The pen in the cup 92
4.5.2 Scaling up: The three-room apartment 93
4.5.3 Comparisons with other state-of-the-art algorithms 95
4.6 Integration on a robotic system 97

In this chapter, we present a method that allows a robot to generate the "optimal" set of assertions that are necessary to produce an unambiguous reference to an entity of an environment. We propose a novel approach to the Referring Expression Generation (REG) problem and its integration into an interactive robotic system. The method is domain-independent and based on an ontology as a knowledge base.

The contribution presented in this chapter is excerpted from our work, published in the proceedings of the RO-MAN 2020 conference [Buisan 2020b]. In this manuscript, the contribution is more detailed and discussed. The presented work has been achieved in collaboration with Guilhem Buisan, with an equal contribution. Several algorithms have been developed by both of us, giving, as a result, the one presented in this chapter, merging the best of our trials. My focus was mainly on how to fully take advantage of the ontology as a knowledge base and on algorithmic optimisations to make our method the most efficient with respect to the current literature.

4.1 Introduction

Referring to an entity is one of the most common tasks that we perform every day. "Can you bring me my mug? It is the black one next to the sink". "I don't remember the name of the man with the red shirt and the glasses". "I lost my keys, they are on a keychain with a unicorn-shaped plush toy". Such kind of communication, accurate and efficient, is a key aspect for the success of a collaborative task. Nevertheless, in complex environments with a wide variety of objects, places, or people, referring to a specific entity can become a real challenge for a robot. It has to take into account the context of the upper task, the diversity of facts that can be extracted from the situation and which depend on available perception modalities, and the available common ground between the robot and its partner.

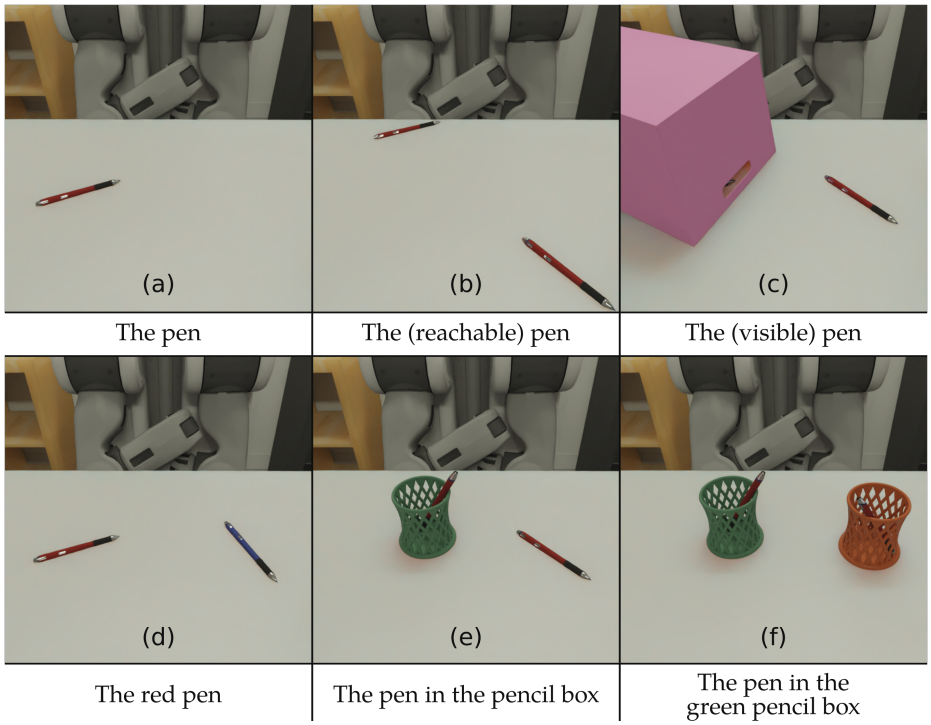

Figure 4.1: Six situations viewed from the hearer perspective, with the robot placed on the other side of the table. Referring to the same pen involves different mechanisms to solve the ambiguity, in each situation. The sentence below each situation is a possible referring expression to refer to said pen unambiguously.

Consider the situation where you are around a table with your collaborative robot and the robot needs a pen. The simple statement *"Give me the pen"* can result in several situations of various complexities. In the case where there is only one pen (Fig.4.1a), referring to it is obvious. Now consider two pens on the table. If one is reachable by you (the human) and the other is not (Fig.4.1b), reachability can be used to refer to the pen. If both pens are not reachable but one is visible to

you and the other is not (Fig.4.1c where a pen is hidden under the box), visibility through perspective-taking could be used to determine that the other pen does not lead to ambiguity. Now both pens are visible and reachable, but one is blue and the other is red (Fig.4.1d). The addition of the color of the pen solves the ambiguity. If both pens have the same color but one is in a pencil box (Fig.4.1e), the relation to the pencil box resolves the ambiguity. If unfortunately, both are in a pencil box but one is green and the other orange (Fig.4.1f), the relation to the pencil box and the color of the latter resolves the ambiguity. We could continue like this for a long time, considering that one is at your left and one at your right, that there is no pen on the table but there is one on a shelf and so on.

Until now, we considered that the robot knows the concepts of pen and pencil box as well as their names in natural language to speak about them. However, imagine yourself travelling in another country and having to speak english[1], you can sometimes miss some words and thus use a more generic one instead. However, by doing so some new ambiguity can be raised because this generic word also refers to other entities. It is the same for our robot if it has to speak French and does not know the translation of the pencil box concept. It will have to use a more generic term, such as "container". However, this more generic term could also refer to other entities, like boxes.

In addition to the concepts names, the robot must pay attention to the relationships it uses. The exact weight or length of an object wouldn't be useful as a human cannot easily evaluate them. On the contrary, the color of the object seems to be a suitable property to use, unless the robot partner is color blind. This means that the robot has to use relations that it estimates to be known and easily observable by its partner. Such can be done considering the theory of mind and performing the Referring Expression Generation on the human knowledge base estimated by the robot.

This task to refer to a precise entity among others is commonly called the **Referring Expression Generation (REG)** task. It is often decomposed into two sub-tasks: the **content determination** and the **linguistic realisation** [Krahmer 2012]. The content determination aims at determining the relations to be used to refer to an entity while the linguistic realisation aims at choosing the words to be used to communicate the content. The contribution of this chapter is focused on content determination but we can not consider these two sub-tasks as entirely independent. As explained earlier, if the robot does not know some concept names in natural language, the linguistic realisation will fail in the case the content determination select them. Another possibility for the linguistic realisation would be to choose a more general word but it would not correspond to the determined content. We could imagine a dedicated knowledge base for the REG with only concepts usable in natural language, but such a solution is not suitable if we want a unique knowledge base for the entire robotic system. Moreover, it could be hard to maintain this dedicated knowledge base during the interaction, in addition to

[1] I apologize to the native English speakers who will not take full advantage of this example.

others since it would be a knowledge duplication.

The main contribution presented in this chapter is an **ontology-based and domain-independent algorithm for the generation of referring expressions**. It uses a customizable cost function estimating the cognitive load required for a human to interpret the RE in order to produce **the "optimal" set of verbalizable assertions** which allows to refer unambiguously to a given entity.

First, we review the literature concerning the REG problem and discuss the issues we aim to tackle. Then, we refine the definition of the problem to manage it at a search problem in a second part. We then compare our algorithm with two states of the art algorithms to assess its solutions and its performance. We end this chapter with an integration on a real robotic system with some details on the used perception system and the verbalization method.

4.2 Related work

Referring Expression Generation is a classic task in Natural Language Generation [Gatt 2018] that has been studied for decades. It has been defined by Reiter as the concern of "how we produce a description of an entity that enables the hearer to identify that entity in a given context" [Reiter 2000]. Over time, the criteria for a good Referring Expression (RE) have been refined but still take their roots from the Grice's maxims [Grice 1975]:

The maxim of *manner* requires the communication to be unambiguous. It is also the referential success for the target entity to be unambiguously identified by the RE hearer. The maxim of *relation* requires the communication to be relevant regarding the current context both the context of the task to achieve and the current world state. If you are asking someone to give you an object that is in the room where you are, you can reasonably assume that the objects in the rest of the house are not ambiguous with the one you are requesting.

The maxim of *quality* seems to be evident and requires the provided information to be true. If you are asking for a bottle and you do not know if it is full or not, you should not use this information to refer to the bottle.

The maxim of *quantity* requires the communication to be as informative as required but not more informative than required. In simple words, to be brief. In the context of REG, the hearer must understand quickly want you are talking about. Moreover, giving unnecessary information could lead to false implications. Saying "give me the red pen" could imply that at least one other non-red pen exists and thus warns the hearer to not do the mistake to take the wrong one. If no other pen exists regarding the current context, the sentence "give me the pen" is thus sufficient.

Dale and Reiter are considered as being the pioneers of the Referring Expression Generation and have proposed over years three main algorithms solving it. The two first fundamental approaches are the Depth First Search (DFS) [Dale 1989] and the Full Brevity [Dale 1992]. While the first algorithm does not always find an optimal solution in terms of the number of relations used, the second does it at the cost of an exhaustive search. The most notable advance was thus the Incremental Algorithm (IA) first presented in [Reiter 1992] then refined in [Dale 1995]. With this algorithm, the notion of preference over features has been highlighted. This notion aims at representing the fact that some features are easier to understand than others. For example, the color or the shape of an object is easier the understand than spatial relations [Belke 2002]. However, the major limitation of the presented algorithms is the used representation of knowledge. They used a set of attribute-value pairs for each entity. This means that an entity has a set of attributes but no relation to other entities of the representation at the difference of a graph. Consequently, the solutions can only be composed of entity attributes and cannot use relations between entities. To be more precise, the algorithms can give the fact the referred entity is on a table but cannot discriminate the said table among others.

With the introduction of a new representation in the form of a labeled directed multi-graph (also known as the REG graph), Krahmer et al. solved the issue of the reference to other entities [Krahmer 2003]. The related Graph-Based Algorithm (GBA) REG is able to manage relations between entities and, as Dale and Reiter, considers a preference over features. This preference, called Preference Ordering (PO), is represented by a cost assigned to each edge of the graph. The GBA algorithm uses a branch&bound algorithm which allows finding the optimal RE. On this new basis, extensions have been developed or at least discussed. Regarding the thin link with Description Logic, Krahmer raised the problem of the hierarchy of entity types in [Krahmer 2012]. On its side, Li et al. have proposed an optimized version of the GBA [Li 2017] GBA allowing an efficiency gain close to 56%. However, the used task only involved cubes, meaning that their algorithm does not have to take into account the entities' types, which were just added as a post-process. A last interesting GBA is the Longest First (LF) algorithm presented in [Viethen 2013]. However, more than not respecting the maxim of quantity, its exhaustive search entails poor performance when used on larger realistic knowledge bases.

Learning-based approaches have of course been proposed. The belief network-based method presented in [Yamakata 2004] can only work with objects' attributes. Moreover, the authors indicate that a specific belief network should be constructed and therefore trained for each attribute. Such limitation reduces the genericity of the method. With a log-linear model trained from a corpus of the probability distribution of REs [FitzGerald 2013], Fitzgerald et al. face the same problem. Nevertheless, by working on belief bases, Yamakata has highlighted the importance to run the algorithm on the human partner's estimated belief base. It ensures that the robot generates a referring solution compatible with concepts estimated to be known by the human.

All the algorithms presented above are highly dependent on the task to per-

form. Where learning approaches are dependent on their training corpus, the others rely on knowledge bases integrating only relations usable in the context of the task. Williams et al. proposed a hybrid approach between domain-dependent and domain-independent with a distributed Incremental Algorithm (DIST-PIA) [Williams 2017]. The idea besides this algorithm is to make the core Incremental Algorithm independent of the knowledge representation by making it querying domain-dependent consultants [Williams 2016]. A consultant is an interface of a knowledge base and each knowledge base of the distributed architecture owns one. Each consultant is thus dedicated to a specific set of properties and can be queried regarding these properties. To get relations about the location of entities, the Incremental Algorithm can thus query the consultant associated with the knowledge base of locations. While this solution is interesting for distributed architectures, we can ask ourselves about the domain-independence of the core Incremental Algorithm. Indeed, the ordering of the consultants to query in the algorithm can have an impact on the found solution. However, it is worth mentioning that this method has been successfully integrated into a robotic architecture [Williams 2019].

At the date, the closest work to the one presented in this chapter is introduced in [Ros 2010]. This method uses ontology as a knowledge base. As explained earlier, such knowledge representation is suitable for domain-independent applications. However, here again, the used algorithm takes as a hypothesis that only relations useful for the REG task are present in it. Moreover, in the same way as the IA-based algorithm, their method only supports entities' attributes and not relations between entities. This method has still been integrated into a robotic system that can take advantage of perspective-taking to construct an estimated knowledge base of the human partner to give pertinent RE [Lemaignan 2011].

Even if all the presented algorithms rely on different kinds of knowledge representations and have non-equivalent abilities, they all consider a perfect linguistic realisation [Krahmer 2012]. We mean here that they all consider that any concept of their knowledge bases has a word in natural language and can thus be verbalized. Wanting to run on the same knowledge base as the other component of the robotic architecture, we do not want to make this assumption. Even if our contribution is focused on content determination, we aim with this contribution to make a first step in the linguistic realisation by not considering these to sub-task as being independent of one the other. We thus assume that not all the concepts in the knowledge base can be used in natural language.

To give a better overview of the progress in the REG field, the most representative contributions presented above are summarized in Table. 4.1. The contributions are organized chronologically and around six major features that we have mentioned throughout this section. These desired features are:

- **Domain independent**: The knowledge base used by the REG must be able to be used by other components of a robotic architecture. The REG algorithm must not consider that all the knowledge represented can be used for this task.
- **Representation type**: The used knowledge representation must be able

to be updated all along an interaction to deal with the dynamics of robotic applications.

- **Use of types**: The type of an entity is the minimal information to use to refer to an entity. Without type, linguistic realisation can not be done.
- **Preference ordering (PO)**: Some properties are easier to understand than others. Ordering the properties according to this preference allows finding efficient referring expressions.
- **Referring to other entities**: Entities attributes are not sufficient to find referring expressions in realistic situations. Being able to refer to an entity by referring to another one is thus mandatory.
- **Natural language**: Considering the linguistic realisation during the content generation could prevent the appearance of ambiguity at the linguistic realisation or even the incapacity to perform it.

Table 4.1: Summary of the most representative contributions in the REG field regarding the six major features of the problem. The contributions are listed in chronological order to give a better overview of progress in the field.

Contributions	Domain independent	Rep. Type	Use of types	PO	Referring to other entities	Natural language
[Dale 1989]	No	Knowledge base entity	No	No	No	No
[Dale 1992]	No	-	-	No	No	No
[Reiter 1992]	No	attribute-value pairs	Yes	Yes	No	No
[Krahmer 2003]	No	REG graph	No	Yes	Yes	No
[Yamakata 2004]	No	Belief Network	No	Yes	No	No
[Ros 2010]	Yes	Ontology	No	Yes	No	No
[Viethen 2013]	No	REG graph	No	Yes	Yes	No
[Williams 2017]	Yes	Distributed KBs	No	Yes	No	No
[Buisan 2020b]	Yes	Ontology	Yes	Yes	Yes	Yes

The literature presented here before is focused on REG in its nominal form. Some researches have however addressed side problems that we do not aim to tackle here. Not entering much in the details, we mention them to give a more global picture of the field. The use of spatial relations is not trivial as these relations can differ for certain entities taking the RE emitter's or receiver's point of view while for other entities, having a clear orientation system (e.g. a car), the relations remain unchanged [Kelleher 2006, dos Santos Silva 2015]. Spatial relations

can also be expressed not only based on a single entity but also according to a set of entity [Fang 2013]. While a RE is often considered as being a single sentence referring to an entity without ambiguity, some see it as a more collaborative task where the RE is provided step by step, allowing to catch acknowledgement and to refine it according to the receiver understanding [Fang 2014, Wallbridge 2019]. Finally, some research tries to integrate REG in a more global interaction where several agents refer to entities. The robot thus tries to reuse properties previously used by the partner ensuring that these properties are known [Williams 2020]. Limitations about this work will be discussed later in this thesis.

4.3 Define the REG problem

In this section, we first present an example ontology that we use all along this chapter to illustrate the algorithm. We then discuss how the upper-task in which a REG could be performed can be used to constrain the search among the entire knowledge base. Finally, we formally define the expected form for a solution to a REG problem and the constraints it must respect.

4.3.1 The knowledge representation

As presented earlier, ontology is a way to represent knowledge that is now largely used in many fields. It allows standardization of the representation, easy extension of an existing knowledge base, and the use of inference engines to enrich the knowledge base. For these reasons, we choose to use a knowledge base in the form of an ontology as input of our algorithm for the REG problem. Moreover, we saw that a number of recent REG algorithms tend to use graph representation as it allows to refer to an entity through relations to other entities. Since an ontology can be seen as a more complex and more expressive graph, this representation seems to be adequate to use for the REG problem.

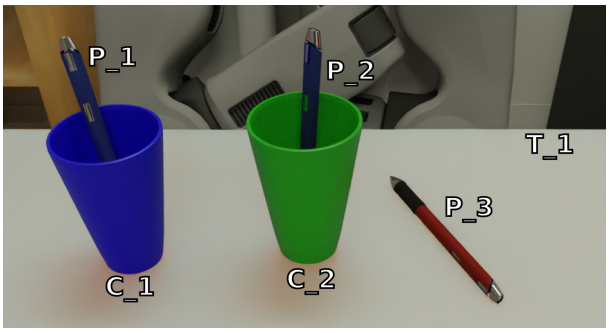

Figure 4.2: A situation view from the hearer perspective, with the robot placed on the other side of the table. Three pens and two cups are on a table. The two blue pens are each in a cup.

In this chapter, we take as an example the situation represented in Figure 4.2.

The situation is assumed to be perceived by the robot[2] and represented in its semantic knowledge base. This knowledge base as an ontology is of the form $K_S^R = \langle \mathbb{A}^R, \mathbb{T}^R, \mathbb{R}^R \rangle$. $\mathbb{A}$, $\mathbb{T}$ like presented in Section 2.1.3. The estimated knowledge base of the human partner K_S^H is here assumed to be the same as that of the robot in the way that $K_S^R \equiv K_S^H \equiv K_S$[3].

The TBox used to describe the situation of this example is represented in Figure 4.3. The class IkeaLisabo represents a precise model of tables and does not have any label. The class Pen is specified through two classes the ClickingPen and the TurningPen. These two classes aim at representing the pens you need to click on top to get the tip of the pen out and the pens you need to turn to get the tip of the pen out. These classes do not have any label to directly speak about them. They are only used by the robot to know how to use them. For sure a more precise ontology could be drawn but we try to keep it simple for the purpose of this chapter.

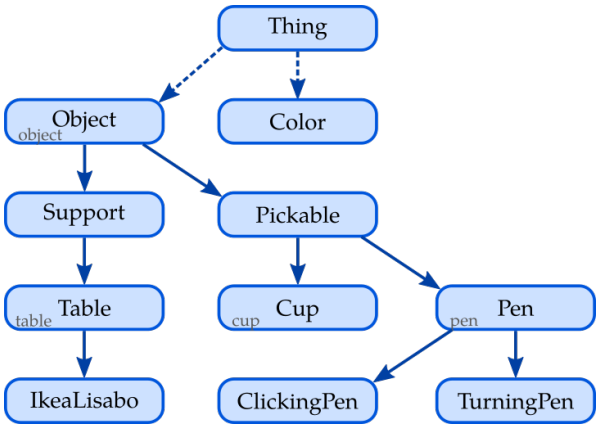

Figure 4.3: Representation of the TBox (classes hierarchy) used to describe the situation of Figure 4.2.

The ABox used to describe the situation is represented in Figure 4.4. The two cups C_1 and C_2 are on table T_1. The two pens P_1 and P_2 are respectively in the cups C_1 and C_2 while P_3 is directly on the table. The pen P_4 is another pen, not on the table. Other objects could be represented as the robot and the human could know other objects being in the room. The pen P_1 is the only pen for which the agent has to click to get the tip of the pen out.

The RBox is not represented but the properties composing it are the ones used in the ABox. In addition we define the properties *hasIn* being the inverse of *isIn* and *isUnder* being the inverse of *isOn* $(\{(isIn,\ hasIn), (isOn,\ isUnder)\} \subset Inv)$. Moreover, the property *isOn* inherits of the upper property *isAbove* $(\{(isOn,\ isABove), (isUnder,\ isBelow)\} \subset Incl)$. While the first defines a direct contact between two entities, the other does not. Finally, we declare the chain axiom: $isIn \bullet isOn \Rightarrow isABove$. This axiom allows to reason on the ontology and

[2]Or estimated to be perceived by the human partner through perspective-taking.
[3]Meaning that no object is hidden to the human partner.

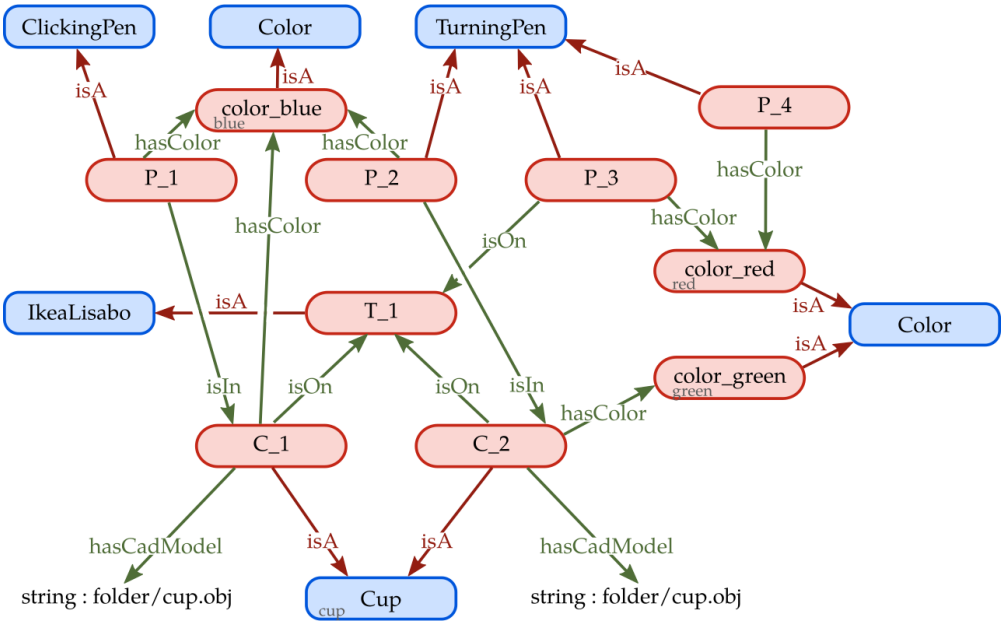

Figure 4.4: Representation of the ABox (relation graph) used to describe the situation of Figure 4.2.

declare that if a first entity is in a second one and that this second is on a third one, the first entity is above the third. Taking our example, since P_1 is in C_1 which is on T_1, the pen P_1 is above the table T_1.

4.3.2 Contextualization and restriction for situated REG

The human (let us name him Tony) and the robot are involved in a shared task around a table. During the task, the robot needs a blue pen to write. However, it can not take one by itself as the blue pens are in cups. Moreover, with its huge gripper, it can not use the kind of pens where you have to click. Our robot thus precisely need the pen P_2 of our example and has to ask it to its human partner[4].

The robot is thus aiming to unambiguously designate a specific entity $a_t \in A$, called the **target entity**, through its attributes and relations to other entities. However, as explained, the REG is meant to be used in the context of an upper task that has to be taken into account. In our example, the collaborative task concerns objects on the table so that the other entities in the room are clearly out of context. Asking for a pen, P_4 will not lead to an ambiguous situation as it is not on the table around which the interaction is performed. To represent this restriction, we provide the problem a **context** $Ctx = (R_{ctx}, C_{ctx})$. It is a set of relations and direct types that are implicit in the current communication

[4]Some could tell that if it is not the first time that our robot and this human collaborate, the human could be aware of the robot capabilities. In this case, the robot would just have to ask for a blue pen and the task would be over. We thus consider that our robot never had interacted with this particular human. For sure it could explain its capabilities but the purpose is not there.

with regard to the task. This context is used to find a reference to a_t, but has not to be included in the generated RE. For our example the context could be $Ctx = (\{(a_t,\ isAbove,\ T_1), (a_t,\ isVisibleBy,\ Tony)\},\ \emptyset)$. With this context, we state that the entity to refer to (i.e. a_t) is known to be above the table T_1 and visible by the human partner, Tony. Since *isAbove* is an upper property of *isOn*, all entities on the table are concerned. Moreover, thanks to the chain axiom, any entity being in an entity on the table are also concerned. The direct types in the context could be used in a more complex communication in which we already know that we are speaking about a pen.

In addition to the entities being out of the context and thus not taken into account, some relations might be present in the knowledge base, but cannot be used in a communication process. In our example, the relations using the *hasCadModel* property should not be used as the robot cannot communicate them verbally. To represent this restriction, we provide the problem a set of so-called **usable properties** $U \subseteq P$. Any relation involving as predicate a property that is not part of the usable properties set can, therefore, not be used in the solution. Because of properties inheritance, *Incl* all the properties inheriting from the ones in U are consequently usable to solve the problem.

With regard to the presented restrictions, the REG problem is defined as follow:

Definition 1 (Referring Expression Generation problem) *The REG problem is a tuple $\mathcal{P} = \langle a_t, K, Ctx, U \rangle$ with $a_t \in A$ the target entity, K_S the hearer semantic knowledge base, Ctx the RE context, and $U \subset P$ the usable properties.*

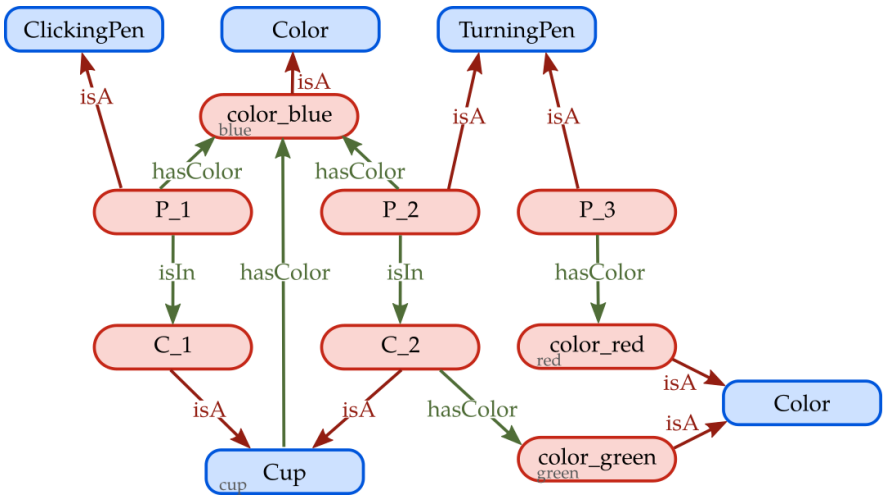

Figure 4.5: Representation of the ABox (relation graph) on which the context of the problem and the usable properties have been applied.

A representation of the ABox on which the restrictions have been applied[5] is represented in Figure 4.5.

4.3.3 Expected solution: structure and validity criteria

What we might expect from a REG solver would be to generate a natural language sentence. However, as explained earlier, we focus here on the sub-task that is the content generation rather than the linguistic realization. For the content generation, the attempted solution is often a set of relations like $\{(a_t,\ hasColor,\ red)\}$. The issue with such a set is that it can not be verbalized afterwards. Indeed, some entities are not labelled and thus can not be used directly in the solution. In our example, it the case for the three pens P_1, P_2, and P_3. They are what we called **anonymous entities**. To refer to an anonymous entity and be able to speak about it we must resort at least to its type. This is what we naturally do when we say *"the red pen"* where the concept *"pen"* does not directly refer to the entity but rather to its type. Before going ahead in the definition of the form of the solution, we can already set a constraint on its content through **the naming need**.

Theorem 1 (The naming need) *For any entity appearing in a RE, exactly one naming relation must be added.*

What we call a naming relation is in its simplest form the presence of a label. The relation to add is thus of the form: $(a_t,\ hasLabel,\ \text{“}tony''\text{)}$. In the case an entity does not have any label, we use one of its types which has a label: $\{(a_t,\ isA,\ Pen), (Pen,\ hasLabel,\ \text{“}pen''\text{)}\}$. This type is not necessarily the direct type of an entity. It could be an inherited one if needed.

Let us now introduce X, a set of variables. Since anonymous entities are not directly present in the RE sentence (hence the presence of ambiguities), we will replace all of them with variables. They are prefixed with a question mark. The set X thus keeps track of these variables. Taking the example of the red pen and replacing the anonymous entities by their corresponding variable, the set of relation become: *{(?0, isA, Pen), (Pen, hasLabel, "pen"), (?0, hasColor, red), (red, hasLabel, "red")}*. Since anonymous entities are now represented by variables, the relations representing the labels can be removed as it is implicit that the others have one. The reference is thus *{(?0, isA, Pen), (?0, hasColor, red)}* that is exactly of the form of a SPARQL query.

Definition 2 (Referring expression) *A referring expression is a set of under-specified relations of the form of* $(s, p, o) \in (X \cup A) \times P \times (X \cup A)$ *or* $(s, isA, o) \in X \times \text{“}isA'' \times T$ *for type ascription.*

The form of the wanted solution would thus be similar to a SPARQL query. It can easily be built from a set of relations. All the content information are present in it,

[5] We do not create a second ABox with the only elements that can be used. The REG algorithm will have to manage the restrictions during the search process. Otherwise, we lose the interest to run on a knowledge base common to the entire robotic architecture.

in addition to the representation that some entities can not be verbalized directly. The way to resolve a SPARQL query can be compared to the process we perform as humans to understand a RE. We search all the combinations of entities matching the RE. We can thus define the **correct instantiation** constraint.

Theorem 2 (The correct instantiation) *For any variable appearing in a RE, at least one substitution function $f : X \rightarrow A$ must exist.*

	Target	**Relations set**	**sparql query**	**Sentence**	**Instantiations**
a)	P_3	{(P_3, isA, Pen), (P_3, hasColor, red)}	{(?0, isA, Pen), (?0, hasColor, red)}	The red pen	[?0 =>P_3]
b)	P_2	{(P_2, isA, Pen), (P_2, isIn, C_2) (C_2, isA, Cup)}	{(?0, isA, Pen), (?0, isIn, ?1) (?1, isA, Cup)}	The pen in the cup	[?0 =>P_2, ?1 =>C_2], [?0 =>P_1, ?1 =>C_1]
c)	P_2	{(P_2, isA, Pen), (P_2, isIn, C_2), (C_2, isA, Cup), (C_2, hasColor, green)}	{(?0, isA, Pen), (?0, isIn, ?1), (?1, isA, Cup), (?1, hasColor, green)}	The pen in the green cup	[?0 =>P_2, ?1 =>C_2]

Table 4.2: Three referring expressions extracted from the example of Figure 4.1. For each target, is represented the relations set, the equivalent SPARQL query, the corresponding sentence in natural language, and the possible instantiations.

Examples of REs base on the Figure 4.1 are presented in Table 4.2. All three respect the constraints of theorems 1 and 2. However, the example b) does not refer in an unambiguous way to the entity P_2. We thus define the two RE validity constraints below.

Theorem 3 (The RE minimal validity) *A RE is said to be minimally valid iff the variable $x_t \in X$ representing the target entity a_t has only one possible instantiation being a_t itself.*

Theorem 4 (The RE complet validity) *A RE is said to be completely valid iff it is minimally valid and if for all the variables $x \in X$ involved in the RE, each has only one possible instantiation.*

Taking the example of Figure 4.6 where the goal of the situation is to refer to cup B, it can be achieved in two manners. Asking for the cup with a pen inside is a referring expression said to be minimally valid as the referred pen is ambiguous. However, not solving this ambiguity between the two pens in the cup still allows identifying the referred cup. To ensure the solution to be completely valid, we should ask for the cup with the red pen inside or the cup with the blue pen inside[6].

[6]A minimally valid reference can thus use references to sets of entities. This is not an issue but the linguistic realisation should take it into account using “a” rather than “the” (e.g. “with a pen inside” rather than “with the pen inside”).

Figure 4.6: A situation where referring to the cup B through relations to the pens can be done either by leaving the ambiguity on the pen (minimally valid RE) or also disambiguating the color of the pen (completely valid RE).

With regard to the presented constraints, a solution to a REG problem is defined as follow:

Definition 3 (Referring Expression Generation solution) *A solution to the REG problem is thus $\mathcal{S} = \langle E, x_t \rangle$, with E a valid referring expression in the form of a set of under-specified relations representing the* SPARQL *query and x_t the variable designating the target entity a_t.*

Finally, considering the fact that some relations are easier to understand than others, we can define the relations cost function[7] $\mathcal{C} : R \to \mathbb{R}^+$. Thanks to it, we can now define an optimal solution to a REG problem.

Definition 4 (Optimal Referring Expression Generation solution) *The optimal solution $\mathcal{S}^* = \langle E^*, x_t \rangle$ is thus the solution minimizing $\sum_{r \in E^*} \mathcal{C}(r)$ over the set of all the possible solutions for a REG problem.*

4.4 Uniform Cost Search REG

In the previous section, we have defined a Referring Expression Generation problem as well as its solution. In this section, we first formalise this problem as a search problem and we then present an algorithm able to solve it in an efficient way.

4.4.1 Formalisation as a search problem

In the REG problem, we can consider a **candidate Referring Expression** as a **state** $\mathbb{s}$ of a search algorithm. A candidate RE is a RE under construction that does not respect all the constraints to make a valid RE. It is a set $\mathcal{T} \subseteq R \cup C$ of relations r representing relations present in the referenced knowledge base K_S. The **initial state** is specified by the context of the problem that is a set of relations that we assume to be already known by the hearer.

To find all the entities referred by a candidate RE, we transform it into a SPARQL query by replacing the anonymous entities with variables. We then submit it to

[7] The cost function is here seen as a black-box which can be a static map, the result of a learning process or other.

the ontology to know which entities can be bound to each variable of the query. Depending on whether the result of the query gives the correct matches between variables and entities or not, we can know if a state is a **target state** or not. A state is a target state if the matching between the variables and the entities make the RE valid (minimally or completely regarding our needs).

An **action** $\mathbb{a}$ in the REG problem consists in the insertion of a new triplet (s, p, o) to the set $\mathcal{T}$ of a state $\mathbb{s}$. Performing an action on state results in the creation of a new state $\mathbb{s}'$. The inserted triplet can represent an entity type (coming from C) or a relation between entities (coming from R). When it represents an entity type, we call the action a **typing action** with $p \equiv isA$. When it is a relation coming from the relation set R, we want it to reduce the existing ambiguity in a given state. We thus expect the relation to differ between ambiguous entities in $\mathbb{s}$. In this case, we call the action a **differentiation action**. To consider the open-world assumption, we can refine this difference in two categories:

Definition 5 (Hard difference) *A hard difference exists when two ambiguous entities own the same property towards a different entity:* $(a_i, p, o_i) \in R \wedge (a_j, p, o_j) \in R | o_i \neq o_j$. *We note this difference between two entities* (a_i, Δ, a_j).

Definition 6 (Soft difference) *A soft difference exists when an entity owns a property that is not owned by another ambiguous entity:* $(a_i, p, o_i) \in R \wedge (a_j, p, \cdot) \notin R$. *We note this difference between two entities* (a_i, δ, a_j).

Considering Figure 4.6 with the two pens, there exists a hard difference between the two pens regarding their color as one is blue and the other is red. The two pens own a relation with the same property *hasColor* but on different entities, red and blue. A soft difference exists between the two cups as cup B has something inside and not cup A.

To represent the fact that some referring expressions are easier to understand than others depending on the property they use, we define a **cost** for each state. It is the sum of the cost of the properties used in the relations composing the candidate RE. The cost can thus be regardless assigned to the property of the relation or the relation itself and the cost of an action is the cost of the relation it inserts. If we assume a state $\mathbb{s}$ with a cost $\mathbb{c}$, performing an action $\mathbb{a}_j$ corresponds to the fact of adding a relation r_j and creates a new state $\mathbb{s}'$. The cost of this new state can be calculated either from the cost of the previous state $\mathbb{c}' = \mathbb{c} + \mathcal{C}(r_j)$ or independently $\mathbb{c}' = \sum_{r \in \mathcal{T}} \mathcal{C}(r)$. In addition, as the hard differences respect the open-world assumption while the soft differences do not, we propose to promote the use of hard differences when possible by adding an extra cost to relations coming from soft differences.

To manage the REG problem as a search problem, we finally define a search **node** $\mathbb{n}$ that is composed of a candidate RE (thus a state) $\mathbb{s}$ and a cost $\mathbb{c}$.

4.4.2 Algorithm choice

We consider the REG problem to be a graph search problem when we perform it on an ontology. If we had considered a state $\mathbb{s}'$ as being composed only of the relation provided by the action $\mathbb{a}$ from state $\mathbb{s}$, a tree search should have been used because state $\mathbb{s}'$ would depend on state $\mathbb{s}$. Considering a state as being a set $\mathcal{T}$ of relations r, the states become independent of each other and can be compared as $\mathcal{T}_x = \mathcal{T}_y$ iff $\forall z, (z \in \mathcal{T}_x \Leftrightarrow z \in \mathcal{T}_y)$.

From any state, we can compute the number of pending ambiguous entities. We could think that this knowledge beyond the problem definition could be used to drive the search. However, when we say that it still has X ambiguous individuals this means that the algorithm will have to perform at most X actions to reach a target state. This means that the heuristic function would overestimate the cost to reach the goal. In this sense, such a heuristic is not admissible and informed search is not applicable.

Although we could define the reverse actions that are to remove relations from a state, we do not know any solution to the problem and therefore we cannot do a backward search from it.

Taking a look at the breath-first search, it is optimal when all steps costs are equal because it always expands the shallowest unexpanded node. However, in the REG problem, actions do not have the same cost as explained previously to represent the preference over features. In this case, the breadth-first search is not optimal and is therefore not suited to our problem.

At the difference of the breadth-first search, the **Uniform-Cost Seach (UCS)** expands the node with the lowest cost. This allows us to take advantage of the preference over features to find efficiently the optimal solution. Indeed, it is **optimal** and **complete** with positive action costs and a finite number of entities and properties in K.

4.4.3 Algorithm presentation

Our REG algorithm performs a graph search in the space of states, using the Uniform-Cost Seach algorithm presented in Algorithm 3. From an initial state, built from the context of the query, the algorithm generates a set of actions that add relations to the current state and creates new states to explore. Just like Dijkstra's algorithm, it expands the states in increasing cost order until a solution is discovered or the search space is exhausted. We can however note a slight difference with the original algorithm. In our case, we know that two candidates RE said to be equivalent (i.e. owning the same relations regardless of their order) always have the same cost. Consequently, we do not need to replace in the frontier a node with an equivalent state but having a higher cost than the newly created child node.

Algorithm 3 Uniform-Cost Search algorithm for Referring Expression Generation

```
function UCS_REG(problem)
  node ← a node with RE = CREATE-INITIAL-RE(problem.context), cost = 0
  frontier ← a priority queue of nodes ordered by their cost
  frontier ← INSERT(node, frontier)
  explored ← an empty set

  loop
    if EMPTY(frontier) then
      return failure

    node ← POP(frontier)
    if GOALTEST(problem, TOQUERY(node)) then
      return SOLUTION(node)

    add node.RE to explored
    for all action in ACTIONS(node) do
      child ← CREATECHILD(node, action)
      if child.RE is not in explored or frontier then
        frontier ← INSERT(child, frontier)
```

We will now detail the functions specific to the UCS used in the REG algorithm.

ToVariable: We globally define a symbol table $\mathcal{S}$ to keep track of the variables assigned to the anonymous entities. When a new anonymous entity is found, it is inserted in this symbol table with a unique variable identifier. We note $\mathcal{S}^{-1}$ the inverse table, allowing us to retrieve the entity represented by an existing variable.

ToQuery: It performs the translation of a set of triplets into a SPARQL query. To do so, each anonymous entity has to be replaced by its corresponding variable. For each entity involved in the set of triplet to convert, they are given a UTF8 string representation with the function $v : A \mapsto str$:

$$v(a) = \begin{cases} str(a), & \text{if } \mathcal{L}_a(a) \neq \emptyset \\ \mathcal{S}(a), & \text{otherwise} \end{cases}$$

SparqlResult: It takes a SPARQL query as input and returns a match table $\mathcal{M}$ in the way that $\mathcal{M}(x)$ is the set of entities matching the variable x in the given query. While usually the matching entities of a given variable depend on the matching entities of another one[8], we do not need to keep this link in our application.

[8]See the example b of Table 4.2 for an illustration of this link.

GoalTest: The test first fails if not all the anonymous entities involved in the tested candidate RE are typed. If this first test succeeds, the function succeeds, for a minimally valid solution, if the target entity a_t is the only solution to the variable $v(a_t)$ in $\mathcal{M}$. For a completely valid solution, the test succeeds if the previous test succeeds and if all the others variables of $\mathcal{M}$ have exactly one solution.

CreateChild: Creating a child node from a parent node and an action is easy to see. The CreateChildfunction is specified in the pseudo-code of Algorithm 4.

Algorithm 4 Child node function pseudocode

```
function CreateChild(action, parent)
  return a node with
  RE = parent.RE ∪ action
  cost = parent.cost + C(action)
```

Actions: For each explored state, we consider two kinds of possible actions that we can perform on it. First, the TypingActions aims at proposing the addition of inheritance relations for all the anonymous entities for which no inheritance relation exists in $\mathcal{T}$. Second, the DifferentiationActions, divided into **hard differentiation actions** and **soft differentiation actions**, consists in the addition of relations that differ between the ambiguous entity for each variable in $\mathcal{M}$.

In the TypingActions function (Alg. 5), the function UsableClasses returns the most specific *labeled* classes of an entity a. In other words, it returns the set of classes $t \in T$ such that $(a, t) \in C$, $\mathcal{L}_t(t) \neq \emptyset$, and $\nexists t'$ *s.t.* $(t', t) \in H \wedge \mathcal{L}_t(t') \neq \emptyset$.

Algorithm 5 Typing actions pseudocode

```
function TypingActions(state)
  for all (s, p, o) in state do
    if L_a(s) = ∅ ∧ ∄t s.t. (s,"isA", t) ∈ state then
      return { (s, "isA", t) | t ∈ UsableClasses(s) }
    else if L_a(o) = ∅ ∧ ∄t s.t. (o,"isA", t) ∈ state then
      return { (o, "isA", t) | t ∈ UsableClasses(o) }
  return OK                                  ▷ every anonymous entity is typed
```

The choice to select the most specific class differs from the one of [Dale 1995] that prefers the least specific types also called the basic-level classes. However, using a knowledge base not specific to the REG task, it would always return the classes *Object* or *Thing*. This could lead to confusion for the hearer and would not help to reduce the ambiguity with other entities. Selecting the most specific thus reduces the branching factor of the algorithm as soon as possible and does not impact the completeness. Moreover, working on the estimated knowledge base of the RE hearer, we can assume that the used labels and thus concepts are known

to the hearer.

The TypingActions is always performed before the DifferentiationActions and this second is even not called if the first has found possible actions. We thus ensure that we do not have more than one untyped anonymous entity at the time. This particularity explains why the TypingActions stops at the first untyped anonymous entity. Besides, it reduces once again the branching factor. Because of the naming need constraint, we knew that any untyped anonymous entity has to be typed to find a valid solution. It is thus useless to try to add other relations while such an entity still exists in a candidate RE. Even if, for any reason, we found ourselves in a case with multiple untyped anonymous entities at the time, each of them will be typed after the other[9].

Algorithm 6 Hard differentiation actions pseudocode

```
1: function HardDifferenceActions(state)
2:     actions ← an empty set of actions
3:     matches ← SparqlResult(ToQuery(state))
4:     for all (x, a) in matches do
5:         if a ≠ S^-1(x) then
6:             for all r = (S^-1(x), p, o) in S^-1(x) Δ a do
7:                 r_inv = (o, Inv(p), S^-1(x))
8:                 if r ∉ state ∧ r_inv ∉ state ∧ p ∈ U then
9:                     actions ← actions ∪ {r}
    return actions
```

The DifferentiationActions function call consecutively the hard then soft differences actions functions and merge their results. The pseudo-code of the hard differentiation actions function is provided in Algorithm 6. The Δ at line 6 (resp. δ) operator returns all the relations that are hard differences (resp. soft) between two entities. In both cases, an action can be added only once and must not be present in the current state to avoid redundancy. Moreover, an action can not be added if the inverse relation to the one added is already present in the current state or proposed actions. We use the Inv set defined in the knowledge base to check it. It once again avoids any redundancy. Taking the example of Figure 4.2, the relation $(C_1, isOn, T_1)$ would be redundant if the relation $(T_1, isUnder, C_1)$ has been already used in the current candidate RE.

4.4.4 Replanning and explaining failures

When the hearer does not understand the referring expression, we could either repeat exactly the same sentence or be smarter and add more information than needed to help him. To do so, slight modifications have to be made to the current

[9]Typing them at once would reduce the execution time but would necessitate actions to be the insertion of a set of relations which adds complexity to the algorithm for cases that should not happen.

algorithm. In the case of replanning, the frontier and the explored set does not have to be re-initialized and in case of success, the current node has to be inserted in the explored set and the next possible actions to be found before exiting. These simple modifications are sufficient to refine a not understood RE.

When no referring expression can be found, it could be interesting to give more information about the failure to the upper process. In the case of the REG problem, it could be to give the entities still in ambiguity. With such information, an upper deliberative process could in this way act on these entities to remove the ambiguity. To do so, we just have at each step of the algorithm, to keep a track of the node having the minimum of ambiguous entities. This information is present in the matching table $\mathcal{M}$ for each node.

4.5 Results

In this section, we present the solution found by our algorithm on the example presented in section 4.3.1. We then challenge our algorithm on a large scale ontology representing a three-room apartment to analyse it in terms of time execution, solution length and composition. Finally, we provide comparisons with two state-of-the-art methods on their own domains.

4.5.1 Solution analysis: The pen in the cup

To familiarize with the form of the solutions and better understand how the algorithm constructs them, we first solve the example of Figure 4.2. The target entity a_t is P_2. The corresponding variable x_t in the SPARQL query is denoted ?0. We assume the general knowledge to be the Tbox of Figure 4.3 and the perceived relations to be the ABox of Figure 4.4.

The context of the problem is that we are speaking about objects being above the table T_1: $Ctx = (\{(?0,\ isAbove,\ T_1)\}, \emptyset)$. The usable properties are $U = \{visualProp,\ geometricProp\}$. Because of the properties inclusions, the properties *isIn*, *hasIn*, or *hasColor* are usable. The problem is thus defined by $\mathcal{P} = \langle P_2, K_S, Ctx, U\rangle$.

Since the classes *ClinkingPen* and *TurningPen* do not have labels, the lowest labelled class of P_1, P_2, and P_3 is *Pen*. A recall of the situation and the available knowledge base on which the context and the usable properties have been applied is represented on the top of Figure 4.7.

The search process is represented at the bottom of Figure 4.7. It starts with node 0 in which we know that P_2 is above the table. In this node, P_2 is an anonymous entity and is not typed. The only possible action is thus to type it, creating node 1 with the candidate RE $\{(?0,\ isA,\ Pen)\}$. From this node, two differentiation actions are created and two new nodes are generated. Since the action leading to node 3 introduces a new anonymous entity, it also introduces a new variable to represent it. The candidate RE related to node 3 is thus *{(?0, isA, Pen), (?0, isIn, ?1)}*. The node with the lowest cost is explored first. In our

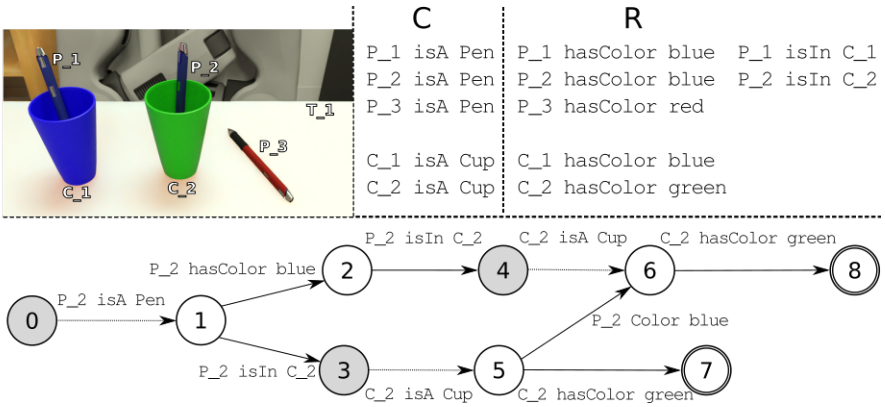

Figure 4.7: On top, a recall of the situation of Figure 4.2 and the entities types and relations. At the bottom, a graphical representation of the search progress to generate a referring expression to the entity P_2. Numbers represent the order in which the nodes are explored. Arrows are the actions performed on the nodes. Hashed arrows correspond to typing actions and greyed nodes do not respect the naming need constraint. Double circled nodes are valid nodes.

example, all properties have a unary cost so the node to be explored is selected randomly. Considering node 2 to be explored first, the entity *blue* has a label so no anonymous entity has to be typed. The only possible action is that P_2 is in cup C_2. Between nodes 3 and 4, node 3 has the lowest cost and is explored first. In this node, C_2 is anonymous and not typed. The only possible action is to type C_2. The same is done for node 4, resulting in node 6. When node 5 is explored, two differentiation actions are possible. However, adding the relation (*P_2*, *hasColor*, *blue*) to the candidate RE of node 5 results in the same state as node 6. The newly created node is thus discarded as already existing. It can also be seen as a merge of the two nodes with the same state. The second action coming from node 5 gives node 7 with the candidate RE *{(?0, isA, Pen), (?0, isIn, ?1), (?1, isA, Cup), (?1, hasColor, green)}*. At this stage, the new node is just created but not evaluated. We do not know at the moment that it is a goal node. Node 6 is thus explored first and gives node 8. Node 7 is then tested as being a goal node as the variable 0 has for only matches P_2 that is the target entity and variable 1 can only be bound to C_2. The algorithm does not test node 8 since a goal node has been found.

The final solution to refer to the entity P_2 is thus *{(?0, isA, Pen), (?0, isIn, ?1), (?1, isA, Cup), (?1, hasColor, green)}*. It could be read as *"The pen in the green cup"*. Here, we see how referring to another entity lead to interesting solutions.

4.5.2 Scaling up: The three-room apartment

The previous example was a simple situation with few entities. To assess the relevance of our approach and evaluate it in terms of performance and solution size, we have created a larger, more realistically-sized, situation. It is the description

of a three-room apartment with several types of furniture (tables, chairs, shelves) and objects (cups, boxes, books) linked through geometrical relations (isAtLeftOf, isOn, isIn) and attributes (color, weight). The resulting ontology is composed of 101 entities, 36 classes, 40 properties and 497 relations once the reasoners have been applied (i.e. the inverse relations, reflective relations, and the relations coming from chain axioms have been computed). The algorithm has been run over the 77 physical entities, inheriting from the *Object* class.

For all the presented measures, we used the Ontologenius software [Sarthou 2019b] to manage the ontology. To avoid taking the ROS communication into account in the measures, we used a direct connection by running the core of Ontologenius in the same process as the algorithm.

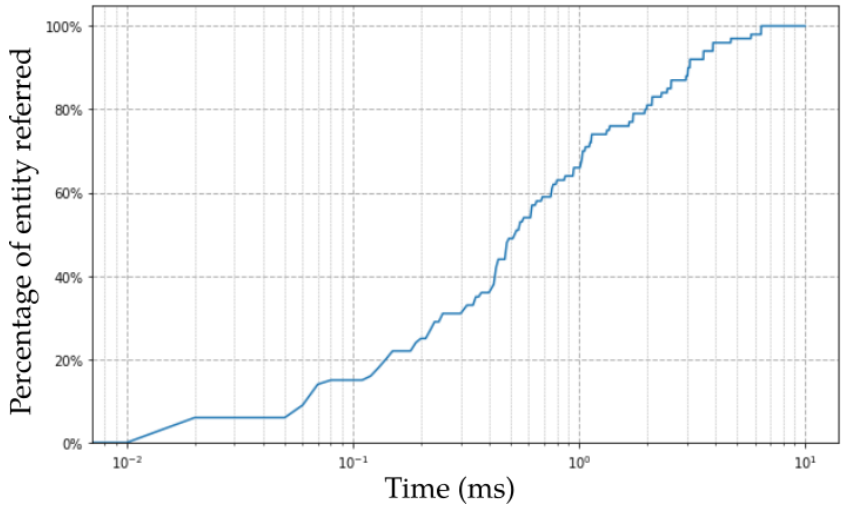

Figure 4.8: Percentage of entities referred successfully over time using a logarithmic timescale.

During a human-robot interaction, a too-long execution time to generate a reference to an entity could spoil the interaction. To assess it we measure the resolution time[10] of the 77 objects. These measures are reported in Figure 4.8[11] and on Table B.1 in the appendix. All the 77 entities (100%) have been referred in under 6.40ms. This is well below 100ms which is the maximum system response time for the user to get a feeling of instantaneity [Miller 1968]. Moreover, 50% are referred under 516μs and 75% under 1.32ms. On average, the algorithm needs to explore 10.6 nodes to refer to an object. It gives an exploration time of 101.88μs/node on average. All the results over the 77 entities are available in appendix B.1.

Regarding the complexity of the solutions, we found that 32 (41.56%) need at least two relations to be referred to. As the type count for a relation, this means the second relation is one of the entity attributes. Among the others, 25 entities

[10]Times reported are run on a CPU Intel Core i7-7700 CPU @ 3.60GHz with 32 Go RAM

[11]The results differ from the original paper because here we are using the latest version of each software to be comparable with the other chapters.

(32.46%) are referred to through the use of four or more relations. The maximum was six relations for only one of the objects. Moreover, 49.4% need a reference to another entity in their solution and two objects need references to two other entities in their RE. This means that 49.4% of the objects of our situation can not be referred using approaches like [Ros 2010] or [Dale 1995].

These results over a larger knowledge base highlight the need for a REG algorithm to use relations and references to other entities to be able to generate suitable RE. They also show the importance of the entities type that is often sufficient with only one attribute to generate an optimal RE. Finally, we demonstrate over these results that our algorithm is suitable for use in human-robot interaction, even with a "large scale" knowledge base. With such results, we can even think that it is suitable to be integrated into a planning process needing to request it at high frequency. Such application will be presented in the next chapter.

4.5.3 Comparisons with other state-of-the-art algorithms

The previous situation has been designed by us and could be adapted for our algorithm. Moreover, existing methods could be more efficient than ours. To compare with other algorithms, we choose to perform two comparisons with state-of-the-art algorithms on their domains. Since we saw the importance for a REG algorithm to be able to refer to entities through relations to other entities and thus through references to other entities, we have only selected methods having this ability.

4.5.3.1 The Longest First algorithm

The Longest First (LF)[12] algorithm [Viethen 2013] is a graph-based algorithm and it has been tested on the GRE3D3 Corpus. This corpus is composed of 20 scenes. As illustrated in Figure 4.9 with the eleventh scene, each is composed of three objects with different spatial relations relative to one another (isOn, isAtLeftOf). Each object has three attributes being a color, a size (small or large) and a type (ball or cube). The entity to refer to is marked by an arrow. This entity has always a direct adjacency relation (e.g. the ball is on the red cube).

Analysing the corpus, we find that 8 target entities can be referenced using only their type (e.g. the ball). Seven entities can be optimally referred to with the use of their type and an attribute (their size or color). In the last five scenes, the target entity can be referred with their types, color and size attributes (e.g. the small green cube). This means that spatial relations are not needed to refer to the target entity.

To perform our analysis, we choose only one scene among the ones needing the most attributes. This scene is the 19^{th} of the corpus, illustrated on Figure 4.9. Since the LF does not consider the types[13], the chosen scene is only composed of

[12] `http://www.m-mitchell.com/code`

[13] The objects type are only added as a post-process. This can explain why they use the spatial relations while it is not necessary.

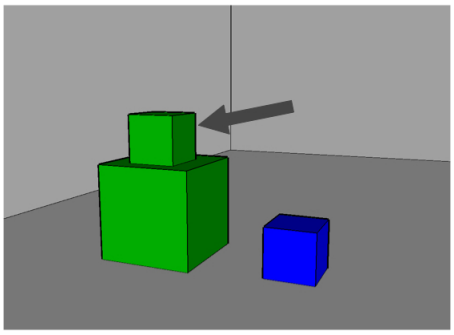

Figure 4.9: The 19$^{\text{th}}$ scene of the GRED3 corpus [Dale 2009]. The target entity is the one marked by the arrow.

cubes to give comparable solutions. Even if we do not compare with the simple cases, we can note that our algorithm resolves the scene requiring only the type in less than 100μs and those with the types and one attribute in less than 250μs.

The LF algorithm has the objective to generate over-specified REs. To limit the solution length, a maximum length parameter is available on the queries. Setting it to the recommended value, that is 4, we get in 311ms the result which we can read as *"The small green cube on top of a cube"*. Setting it to 3, which is the optimal value for this scene, we get in 109ms the result *"The small green cube"*. This optimal solution is the one found by our algorithm in 0.87ms. The difference factor is thus 125.

For all the 20 scenes, setting the maximum length parameter to the optimal length (the minimum of relations allowing the hearer to distinguish the referred entity without ambiguity), both the LF and our algorithm find the same solutions. However, this parameter raises an issue. In the previous situation of the three-room apartment, 13% of the entities need more than 4 relations to be referred to. This means that with the recommended value, all these entities would not be referred to while it is possible. On the other hand, we saw that this parameter has a huge impact on the execution time and set it to a higher value would thus require more computational time and give too complex RE.

4.5.3.2 The optimized Graph-Based Algorithm

The GBA [Viethen 2013] has been a significant turning point in the REG field. We thus compare our algorithm with its more recent optimized version presented in [Li 2017]. From an entities relations graph dedicated to the task, it aims at finding the lowest cost isomorphic subgraph containing the target entity. To do so, the GBA approach is based on a branch and bound algorithm. Once a first solution is found, any other branch that exceeds the cost of the current best-found solution is bounded. Such an algorithm thus needs to explore a large part of the graph if the optimal solution is not found early in the search.

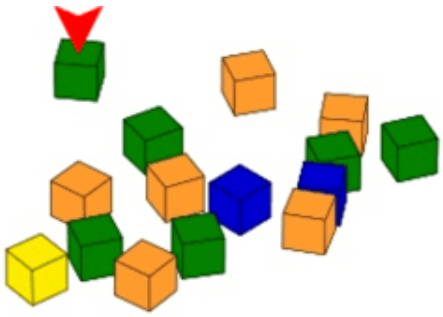

Figure 4.10: The 4th scene used in the user-study in [Li 2016]. The target entity is the one marked by the red arrow.

Their algorithm has been tested on a corpus with 28 tabletop scenes [Li 2016] like the one presented in Figure 4.10. Each scene is composed of 15 cubes of different colors linked through spatial relations and only one of these cubes has to be referred to. To compare our algorithm on their corpus, we selected only one scene (the fourth represented in Figure 4.10) and we have converted their graph into an ontology. On this unique scene, we have tested RE generation for each of the 15 cubes composing it. The GBA, as well as our algorithm, have found a solution for 10 cubes. Since the same costs on relations have been used, both algorithms have found the same solutions. The only difference is that our algorithm has added the objects types. In these 10 cases, our algorithm has performed faster with a factor of 29.4 on average. Because of their use of a branch and bound algorithm, the speed increase is more important for cases with many solutions. We go from a speed factor of 4 for 50% of the cases, growing to 50 for 25% of the cases, and reaching 130 at the maximum. For the other 5 cases where no solution is found, both algorithms detect the absence of a solution in a few milliseconds.

The difference with the GBA is that the Uniform-Cost Seach ensures the first found solution to be the optimal one. Moreover, because their approach does not use the entities types, we think that our approach should work even faster on knowledge bases containing entities with different types since we prioritize the use of the type. Such a test has not been performed since their algorithm can not manage other data than their corpus.

4.6 Proof of concept integration on a robotic system

To assess the usability of our method, we have integrated it on a PR2 robotic platform and used it in a “real” tabletop scenario. The used architecture is represented in Figure 4.11.

The geometrical situation assessment is based on the Robosherlock [Beetz 2015a] perception system. This software does not require any a priori on the environment such as CAD model, mesh or training. With the used pipeline, it detects objects on a surface and gives information about their position, shape (“circular” or “rect-

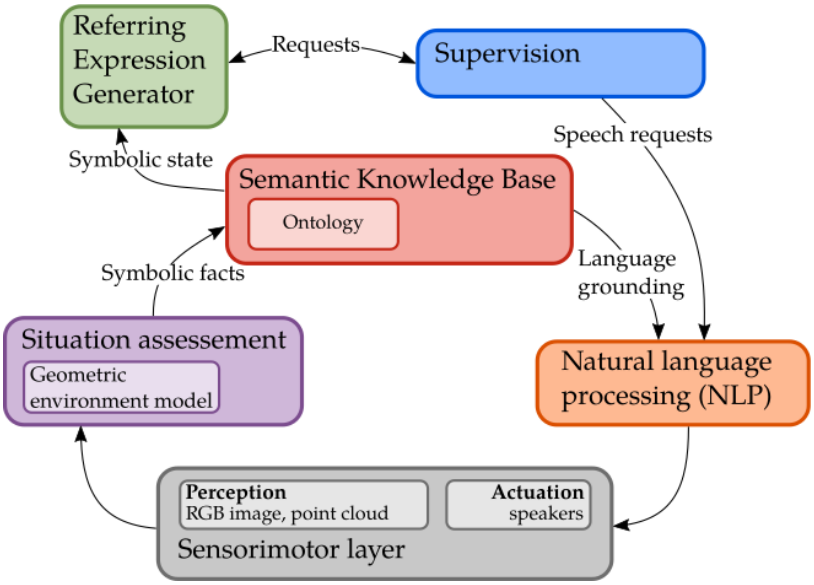

Figure 4.11: The architecture used to validate the method. The knowledge base is continuously kept up to date through the situation assessment.

angular"), color, and size ("large", "medium" or "small"). On top of that, we have applied simple filtering and tracking based on the previous information. This information is then continuously added to the ontology to keep it up to date with the current state of the environment. Since the objects' types are not determined by the system, all the detected objects are thus added as inheriting from the "Object" class. To go further, we could have used the SPARK/TOASTER [Milliez 2014] framework to extract higher-level relationships such as "isOnTopOf" or "isIn". However, the limited information provided by Robosherlock, with the used pipeline, is already interesting to test our algorithm since the robot is not able to use high-level concepts and resulting in various ambiguities.

The knowledge base as an ontology is managed using the software Ontologenius [Sarthou 2019b]. It is used as a server sharing knowledge about the environment with the entire architecture. Since no perspective-taking is performed in this example, it manages a single instance representing the robot's knowledge. The REG is thus performed on this instance even if it should rather be run on the human partner's estimated knowledge.

An ad hoc linguistic realisation has been made, based on simple grammar and the labels present in the ontology. It takes as input a SPARQL query and is able to generate a sentence in English or in French depending on the ontology configuration. For example, it is able to transform the query "?0 isA Cup, ?0 isOn ?1, ?1 isA Table, ?1 hasColor black" into "*the cup on the black table*". The relations in the query do not have to be ordered as the link between the variable can be analysed and a simple language model provides the attribute order for each language.

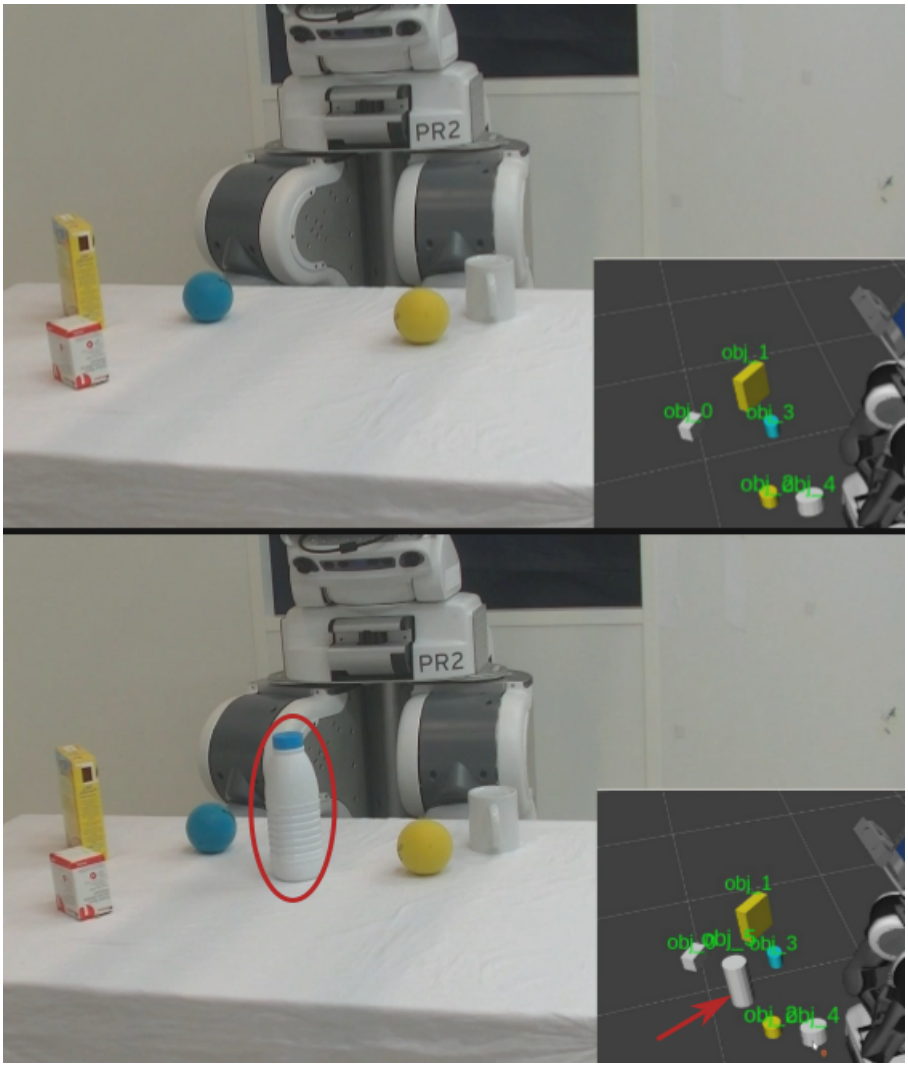

Figure 4.12: Two situations where the robot has to refer to the cup on the right of the image. The presence of the milk bottle (with the red circle on the image on the right and the red arrow in the RVIZ view) changes the generated RE. On the right-bottom part of each image, the geometrical representation of the scene is displayed as perceived by the robot.

The scenario [14] consists of several objects on a table. As illustrated on the left part of Figure 4.12, the initial situation is composed of five objects. The robot has to generate a RE for the cup. The found solution can be verbalized as *"The white circular object"* since there are other non-white circular objects and other white non-circular objects. In a second time, a human adds a milk bottle on the table (right part of Figure 4.12). Since the added bottle is circular and white, the previous RE is no more valid. When the robot is asked to describe the cup, it

[14] Commented video available at: `https://youtu.be/mKDLvDbHfvkit`

generates this time the sentence *"The white small circular object"*. The addition of the size attribute solves the introduced ambiguity.

With this simple task, we show that our UCS ontology-based REG algorithm is usable within a robotic architecture, can deal with a dynamic environment, and can adapt its explanation to the current situation.

Chapter 5

Estimating communication feasibility and cost at task planning

Contents

5.1 Introduction 102
5.2 Related work 104
5.3 The involved components 106
5.3.1 The Hierarchical Task Planner 107
5.3.2 The Semantic Knowledge Base 108
5.4 Integrating planners 108
5.4.1 The representation of the communication action 110
5.4.2 Maintaining the right knowledge base, at the right time . . . 111
5.4.3 Reducing the number of updates 112
5.5 Results 113
5.5.1 Prevent execution dead-end 113
5.5.2 Reduce the overall communication complexity 115
5.5.3 Compare with other communication means 115
5.6 Integration in a robotic system 116

In this chapter, we take advantage of the REG solver to link it with a symbolic task planner, in order to take into account the feasibility and cost of referring communication actions at the task planning level in the context of human-robot interaction. With this method, we endow the task planner with the ability to avoid deadlocked situations where referring is not feasible, to reduce the task overall communication complexity, and to evaluate different communications strategies.

The contribution presented in this chapter is excerpted from our work, published in the proceedings of the ICSR 2020 conference [Buisan 2020a]. In this manuscript, the contribution is more detailed and discussed. In the continuity of the previous chapter, the presented work has been achieved in collaboration with Guilhem Buisan. While his focus was on task planning, mine was on the link between the Knowledge Base as an ontology and the task planner. In this thesis, we will deepen this link and discuss possible improvements to the one initially presented.

5.1 Introduction

It is well established that a key aspect of the success of collaborative tasks is based on clear and fluent communication grounded in the context of the interaction. Focusing on verbal communication, in the Natural Language Processing (NLP) research field and by extension in the Human Robot Interaction (HRI) field, it has been divided into two dual problems [Tellex 2020]. On one hand, the Natural Language Understanding (NLU) aims the robot to interpret and grounds human's utterances with regard to the current situation and to react according to it [Brawer 2018]. In another hand, the Natural Language Generation (NLG) aims the robot to produce language. It could either be to ask for help [Tellex 2014], to align knowledge [Devin 2016], or to explain its decision to its partner [Roncone 2017].

In the previous chapter, we have introduced an algorithm able to generate the content of a referring expression. Such contribution thus falls in the NLG problem. Considering the REG as an action that can be performed by the robot means that the robot could plan such communication in terms of **when** and **what** to communicate. While the "when" is directly handled by the task planner, the "what" in terms of content is provided by the REG[1]. However, the REG does not only provide the content but is also able to state if such communication is feasible or not and give information about its cost. In the context of the REG, communication feasibility is related to the ability to generate an unambiguous RE while the cost depends on the number of relations to communicate. Because the REG algorithm work on a knowledge base representing the current state of the environment, maintaining a comparable representation of the environment for the future states of the task (as it is done in symbolic task planning) would allow the robot to estimate the **feasibility** and the **cost** of the verbal communication actions all along with a task.

With these two pieces of information, being the cost and the feasibility, a task planner could compare verbal communication with one another, compare with other means of communication, minimize the overall communication complexity, and prevent some plan failures. This approach to estimating the communication during task planning can be compared to the one proposed in [Lallement 2016]. In the latter, motion actions were evaluated at task planning to estimate their feasibility, costs, and indirect effects. With both approaches, the symbolic plans can be optimized and can be more reliable in preventing execution failures and thus the need for reparation.

To better understand the advantage to consider the communication at the task planning level, consider the situations of Figure 5.1. The robot has to arrange RFID tags on three areas on a table. The robot can identify them with their unique id but being too small, it can not grasp them. On the contrary, the human partner can not identify them uniquely but can grasp them. For this example, we also assume that the robot cannot point to the tags. The robot must therefore communicate

[1]The REG does not determine the entity to refer to but only how it will be referred to. We could say that the "what" is chosen by a higher decision-making component, while the "how" is determined by the REG. To fit the usual definition we assume this linguistic simplification.

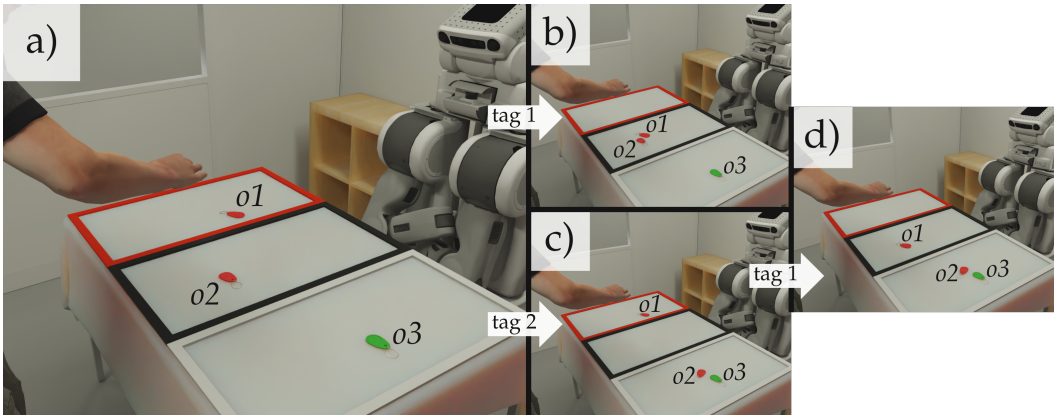

Figure 5.1: A Human-Robot collaborative task with three colored areas and three RFID tags (situation a). The robot has to explain to its human partner to put the tag *o1* in the black area and the tag *o2* in the white area, to reach the situation d. The objects identifiers' are only known to the robot. If all the communications of the task are not planned ahead, a deadlocked situation could appear if the robot first asks to move the tag *o1* before *o2* (situation b).

the successive actions that the human will have to perform to go from the inial configuration (5.1 a) to the goal configuration (5.1 d). Between both configurations, only the tags *o1* and *o2* have to be moved. The tag *o1* has to be moved from the red area to the black and *o2* from the black area to the white. While the tag *o3* can be referred to unambiguously thanks to its color, the two others can not. However, they can be referred thanks to the area they are in (e.g. *"the tag is the red area"*).

If the content of the communications is only refined at execution, two equivalent solutions can be planned (5.1 sequence a-b-d and a-c-d). At execution, the first solution begins by asking the human to move *o1* in the black area resulting in the instruction *"take the tag that is in the red area and put it in the black area"*. In this new situation where both red tags are now in the black area (Figure 5.1b). The robot has no way to designate the tag *o2* without ambiguity. Hence, the task is blocked[2]. Estimating the communication feasibility and cost during the planning process would result in the second possible solution. The robot first ask to move the tag *o2* (Figure 5.1 c) and then the tag *o1* (Figure 5.1 d). If the robot could have pointed, the deadlock of the first solution can be avoided with a pointing action and nevertheless, thanks to the communication cost estimation, the least expensive solution can be selected[3].

[2]The robot could use spatial relations like right, left, or the closest to me. However, the generation of such RE is not an easy job and the understanding of it neither. Even if the situation is not really blocked, the required communication can be complex.

[3]Plenty of other solutions could exist but depend on the robot capability. Giving the two instructions in the initial state before the human act solve also the problem for example. Nevertheless, if the robot cannot compare these different solutions regarding its current capability, non-desirable situations could still appear.

The main contribution presented in this chapter is an approach to **estimate communication feasibility and cost at task planning level**. It implies a fine **link between a planner and an ontology** to estimate communication grounding in the future estimated state of the environment.

First, we briefly review the literature concerning the task planning problem and discuss the issues we aim to tackle. Then we, give an overview of the involved components with a focus on the task planner while the others have been detailed in the previous chapters. We then present how the fine integration of the components allows us to take estimation the communication at task planning and discuss possible improvement. We end this chapter with three case studies, to show how this approach can be used to prevent deadlocked situations at execution, how it can reduce the global communication complexity during a Human-Robot collaborative task and how it can be used to balance between different communication means.

5.2 Related work: The need to plan communication

A significant amount of research has been dedicated to Human-Robot verbal communication, especially to answer the questions of *what* and *when* to communicate [Mavridis 2015]. A lot of early works address these questions at execution time, with a fixed plan in which the robot inserts verbal communication afterwards when needed. The communication can be used to share and negotiate plans [Sebastiani 2017], to ask for or give specific information [Shah 2011], to repair errors [Tellex 2014], align knowledge [Devin 2016], or increase trust [Schaefer 2017]

In the work [Devin 2016] the robot is provided with a shared plan for both itself and its human partner. On top of that, they use a theory-of-mind enabled framework to estimate, throughout the interaction, the partner's mental state about the current state of the environment and the performed actions. When the robot performs an action while its partner performs another one in a different room, the robot can detect a belief divergence due to the fact that the partner can not know if the robot has acted or not[4]. When such divergence is detected, if it can endanger the overall plan, leading the human to perform a wrong action, or block the interaction, verbal communication can be inserted at execution time. It goes the same if the human wait for an action already performed by the robot. The content of the communication is determined with regard to the divergences that can break the shared task.

However, in most cases, deciding communication at execution time is not enough and more recent works deal with communication actions at the planning level. Nikolaidis et al. [Nikolaidis 2018] identify two types of communication: *commands*, where the robot ask for an action to be performed by its partner and *state-conveying* to inform about its internal state. They use a Mixed Observability Markov Decision Processes (MOMDP) to determine the need for communication and its type, return-

[4]This is the case when the action performed by the robot has no observable effects on the environment like scanning object.

ing a policy capturing the probability of the human to take a given action based on the performed communication. A comparable approach is presented by Roncone et al. [Roncone 2017] with three types of verbal communication action: *command* to provide instruction to the human to perform an action, *ask* to be informed if the human current action is over, and *inform* to communicate an intent. These communications are integrated with others actions into a Partially Observable Markov Decision Process (POMDP) which returns a policy integrating communication actions. However, for both presented approaches, the communication complexity and thus costs are not taken into account. Moreover, while for the first the content is pre-generated, for the second it is not specified at the planning level. This can cause non-achievable communication in some situations.

A similar approach is proposed by Unhelkar et al. [Unhelkar 2020] with more communication types considered: *command*, *ask*, *inform* and *answer*. This time, a communication cost is explicitly considered. However, the cost is related to the *when* to communicate and not on the *what*. It is represented by a function penalizing temporally close communication actions. Concerning the content, it is in the form of patterns including parameters replaced at execution time. For example, in the sentence *"Please make the next sandwich at -landmark-."* the sentence representing landmark will be resolved at execution. In their examples, every landmark is assumed to be easily referred to the human, but this is not always the case. Using the REG at task planning, our approach addresses two of the five challenges identified by Unhelkar et al.: "estimating benefit of communication" and "quantifying cost of communication" [Unhelkar 2017].

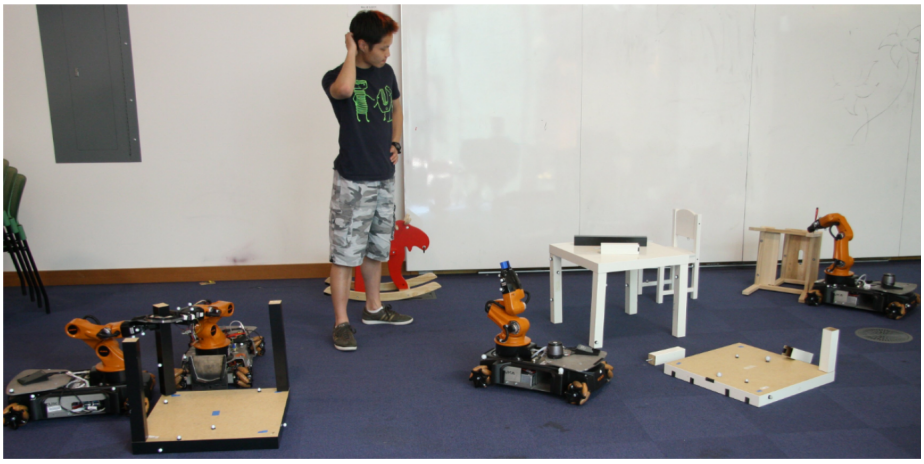

Figure 5.2: Illustration from [Tellex 2014]. A robot engaged in assembling a table requests help using natural language with targeted requests such as "Please hand me the white table leg."

To better understand the difference of our approach regarding existing works, we use the example depicted by Tellex et al. [Tellex 2014] and illustrated in Figure 5.2. In this situation, robots, following a precomputed plan, are assembling

furniture. During the task, the robot assembling the white table encounter a failure because it can not reach the needed table leg (on the other white table). When such a failure occurs, the robot asks a human for help by referring to the object at the origin of the failure. By doing so, the robot performs a plan reparation with the help of the human and thanks to an object referring communication action. However, if the leg has not been move since the beginning of the task, the non-reachability of the leg could have been known by the robot during the planning process. The non-reachability is not really of failure and such reparation could be avoided. Considering the task as a shared task, the assembly of the leg could be assigned to the human. Keeping the human in the role of a helper, verbal communication still could be planned either to group multiple communications reducing the human disturbance or to perform it in order to make the communication easier. The robot would have assembly the other white leg to only refer to the last one as "the white leg" not leading to any ambiguity with the other ones.

5.3 The involved components

The type of communication actions we want to manage in this chapter is commands using Referring Expression presented in the previous chapter. Typical commands will be composed of a static part and a situation-dependent one like *"Take X"*, *"Put it in Y"*, or *"Take X and put it in Y"*. The variable part depends on the state of the situation when the communication is performed and must be solved by a REG. The communication feasibility and cost thus depend on this variable part and by extension of the moment where it is used.

As explained previously, the REG aims to be run on the human partner estimated knowledge base to ensure that all the concepts and relations used in the generated RE are known to him. To be able to estimate communication about the future states of the environment and keeping this principle to run on the estimated KB, we need a symbolic task planner already suitable for HRI. It has to be able to distinguish between the different agents involved in the task and to maintain an independent representation of the environment for each of them.

To resolve a specific task, a planner does not necessarily need to be aware of all the elements present in the current environment. It simply needs a representation of the entity that can be used to solve the task. To solve the task of assembling a table, it only needs the table elements for this particular one even if others are present in the environment. Even if we could represent all the elements, it would be counterproductive by not helping to solve the task but adding exploration complexity. In the same way, it does not need a fine representation of these elements. Even if the task is to create a cube tower with alternating colors, the color information is not necessarily useful. In the introduction example (Figure 5.1) the color of the RFID tags do not matter for the task and the type of the objects are also useless. Representing them as movable objects[5] could be sufficient. Moreover,

[5]To not move the areas around the tags instead of moving the tags.

doing so makes the planner more generic as not being restricted to arrange RFID tags. However, we saw that for the REG, the more the situation will be described precisely (both in terms of types and relation), the more accurate the solution will be. Furthermore, if another tag, which is not part of the task and thus not part of the planner internal representation, is present on the table, it will also impact the REG and thus the complexity and feasibility of the communication action.

This difference of representation requirement between the task planner and the REG lead to the fact that the REG can not be performed on the planner internal representation. To solve this issue, we have to endow the planner with the ability to maintain a semantic KB that is used by the REG. Before going further in the way to solve this challenge, we will first present the newly introduced component as a task planner. We then give more detail about the knowledge base we will consider for this application.

5.3.1 The Hierarchical Task Planner

To implement our approach, we just see that we need a task planner able to maintain an independent estimated knowledge base for each agent involved in the task. We chose the Hierarchical Agent-based Task Planner (HATP)[6] [Lallement 2014]. HATP extends the classical Hierarchical Task Network (HTN) planning by being able to produce **shared plans** to reach a joint goal. A HATP planning domain describes how to decompose tasks into subtasks down to atomic symbolic actions. Both the robot and human feasible tasks and actions are described in the domain. A context-dependent cost function is associated with each action.

During the task decomposition, HATP will explore several applicable sub-tasks until the global task is totally refined into feasible actions, and will return the minimal cost plan. HATP also supports *social rules*, allowing to balance the effort of involved agents depending on human preferences and to penalize plans presenting certain undesirable sequences of actions. We will not use these social rules in what follows, but our approach stays compatible with them.

Moreover, during the exploration of the task tree, HATP will assign actions to available agents, the robot or the human (when an action can be done by both). By doing so, HATP can elaborate one **action stream** per agent, together with causality and synchronization links. Besides, HATP domain syntax supports Multiple Values State Variables (MVSV) [Guitton 2012] which is used to represent and reason about each agent mental state. In this way, a given variable can have different values depending on the represented agent. This allows to represent action preconditions depending on the knowledge of the agent performing the action and also to represent their effect on each agent mental state which can depend on the agent perspective.

Finally, the last argument which motivated our choice was the use of HATP in previous work: the Geometrical Task Planning (GTP) [Gharbi 2015]. This work aimed at refining into motion planning requests the symbolic motion actions explored by HATP during the task planning process. The motion planner would then

[6] Also called Human-Aware Task Planner

returns the feasibility and the cost of the action, but was also able to inform HATP about why the motion action would not be possible (e.g. the object with which collision would occur). The task planner would then, backtrack to choose a different action to remove the colliding object. This method also shows how to update the geometrically planned environment to match the symbolic one to run the motion planning phase. This approach also needs to update the geometrical planned world to match the symbolic planned knowledge base before running the motion planning phase. This work greatly inspired us, and our approach is similar at the difference that we run a REG when a communication action is explored, instead of a motion planning request on a symbolic motion task exploration.

5.3.2 The Semantic Knowledge Base

In the previous applications of this thesis, we could use only one knowledge base representing both the robot's and human's knowledge about the environment. This time, because the task planner explicitly manages an independent world state for each agent, we will fully take advantage of Ontologenius to manage several ontology instances at the time. We will thus note the robot's semantic knowledge base K_S^R and, considering only one partner, the human estimated knowledge base K_S^H. Both will be kept up to date at the same frequency. This means that both represent the knowledge of each agent at the current time. However, as explained in the introduction of this section, we need to run the REG on a knowledge base that will represent what the robot believes that its partner will know about the future states of the environment. Let us consider the initial state of the introduction example of Figure 5.1. The robot plans to remove the tag *o1* from the table and wants to estimate the feasibility of referring the tag *o2* once the first action is performed. It thus has to remove *o1* from the table in the estimated knowledge base of the human to evaluate this future communication. Nevertheless, it can not modify K_S^H as it represents the current estimated knowledge of its partner. Performing such modifications could have side effects on the entire robotic architecture. To deal with that we will use the Ontologenius feature that consists of the copy of an existing ontology instance. As a reminder, a copied instance then become independent from the original ontology. The instances representing a future possible mental state of a human will be noted $K_S^{H_i}$. In this way, the REG can run on a $K_S^{H_i}$ for the planning process and on K_S^H at execution.

5.4 Integrating task and communication planners

The general scheme of our approach to enable a symbolic task planner to estimate the feasibility and cost of future communication is the one presented in Figure 5.3. On the top of the figure, we have large, complete semantic knowledge bases representing the robot knowledge and the other agents estimated knowledge. At the bottom of the figure, we have the planning process with reduced knowledge bases dedicated to task planning. As explained earlier, because the planning process will

need to represent the future estimate mental state of its partner without altering the original estimation, we first perform a copy of the human estimated ontology. From there, it becomes an independent knowledge base, capturing the human knowledge about the environment at a given instant. We call this new ontology the human planned ontology and denote it $K_S^{H_0}$. At the initialization of the planning process, once the copy is performed, the planner extract from the robot ontology and the human planned ontology the symbolic facts it needs. To do so, every entity type declared in the planning domain are retrieved from the ontologies by their name, and entities inheriting from these types in the ontologies are created in the planning knowledge base. Then, each attribute (both static and dynamic) of every entity declared in the domain has its value updated. If the attribute is a set, multiple relations with the same name originating from the same entity and pointing to different ones can be found in the ontologies. If so, all the pointed entities are added to the set.

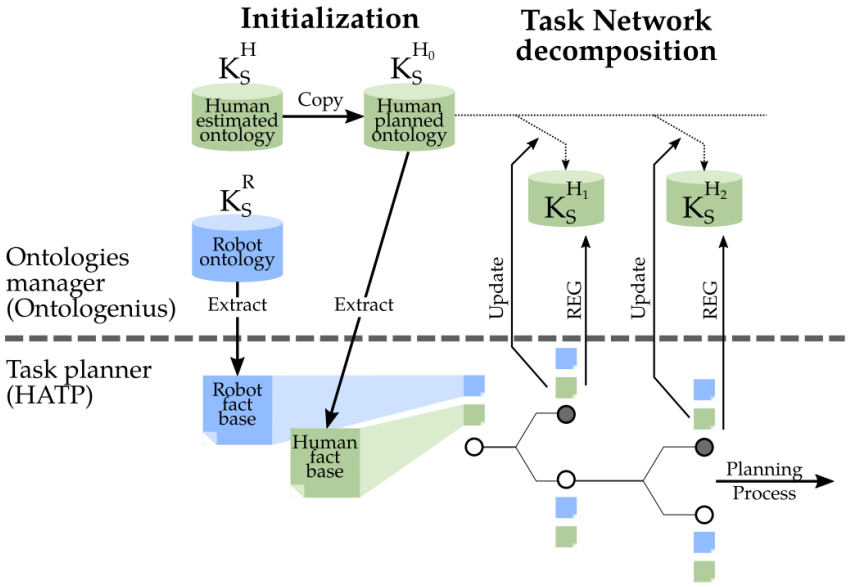

Figure 5.3: A representation of the planning process with the estimation of future human mental states to perform REG. The ontology representing estimated human knowledge is first copied to plan it without altering the original one. The human and robot planning symbolic facts are extracted from their respective ontologies. During the HTN decomposition, for each verbal communication action encounter, the planned human ontology is updated with the current explored state and the REG is executed on it.

During the task network decomposition, the general workflow executed for each communication action encountered consists of 1) updating the human planned ontology with the expected world state 2) identifying the objects involved in the communication 3) executing the REG for each of these objects 4) calculating the feasibility and the cost of the communication action according to the feasibility and

the cost of each individual RE involved in the planned communication. Note that if multiple objects have to be referred to in single communication (e.g. *"give me X and Y"*), the human planned ontology is only updated once as the estimated state in the same for both REG.

In this section, we explain how a communication action is represented in the HTN and how the task planner can easily update the human planned ontology. We will even go further with an unimplemented update solution that could reduce the number of updates to be performed.

5.4.1 The representation of the communication action

To focus ourselves on communication actions, we have thus designed simple planning problems where only the robot knows the goal of a joint task and issues command to its human partner one at a time when the human has to do an action. Communications related to entities of the scene are thus needed in each step of the plan because the human has no way to guess the actions he/she has to perform. Even if these problems present major limitations regarding the *when* to communicate it allows us to simply present our approach. However, the method is still applicable to more general problems which need to estimate the *what* of communications and ensure their pertinence in future states. Moreover, the presented method is compatible with others, focused on the estimation of the *when* to communicate [Devin 2016], [Unhelkar 2020].

In the HATP domain it is represented in the way that when an abstract or a primitive task is only feasible by a human and requires to designate a specific entity, a decomposition is added. In this decomposition, we specify that if the task is assigned to a human, a referring communication action must be done before by the robot in the direction of this human. This kind of decomposition is needed because we place ourselves in cases where the human partner does not know the global objectives of the task, and thus, all the instructions must be given by the robot. With this approach, we consider that the human needs communication from the robot to act, and does not plan by himself. In a more general case, the communication action from the robot would only appear if humans do not have the necessary information [Devin 2016]. An example of a plan is represented in Figure 5.4.

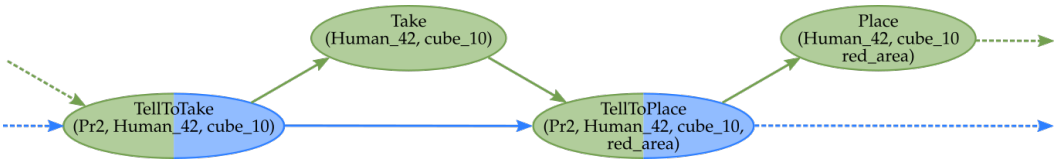

Figure 5.4: An example of a plan generated by HATP where the robot instructs to take an object then place it in an area. Green actions are performed by the human while the bicolor actions involve both agents.

The communication action feasibility is then determined by both symbolic preconditions (e.g. the human and the robot are in the same room) and REG result (whether a solution is found or not). If the communication action is feasible, the cost of the communication action is then computed as the sum of a fixed cost depending on the type of communication and the REG solution cost. If multiple entities would have to be referred in a single communication, the cost would be the sum of each REG solution cost and the fixed cost.

5.4.2 Maintaining the right knowledge base, at the right time

We previously saw the need to keep an ontology updated with the estimated beliefs at the time of the communication to run the REG on it and the necessity to create the human planned ontology with a copy of the human estimated one to not create side effect on the rest of the architecture.

Since maintaining this external representation can be a heavy process, we first choose to only update it when a communication action has to be evaluated. However, having a single planned ontology, knowing the update to perform on it to reflect the current explored state can be an intractable problem. The planner could perform backtracking during the exploration in order to determine the modifications to perform to represent the state of the currently evaluated communication. Consequently, the modifications would depend on the previous communication action explored. Because the previous communication could come from a side decomposition, the planner would have to either compare each relation between its representation and the ontology or keep a trace of all the modifications between two successive evaluations.

A first solution would be to create a copy of the planned ontology representing the initial state in order to have one ontology for each communication evaluation, all created from the initial state. Such a solution would reduce the planner complexity but as the ontology represents a superset of the planner facts base, it would be too time-consuming for the ontologies manager. For huge ontologies, analyzing the modification between two updates would be faster than an entire copy.

To solve this problem we thus choose to use a versioning system on the ontology. This versioning system is the one proposed by the software Ontologenius presented in Chapter 2[7]. We thus keep the first step that is to create the planned ontology with a copy then perform a commit on it to mark this initial state. From there, when a communication action is found, the planner goes back to the initial commit and update the ontology with the difference from the initial state. However, HATP is not adapted to keep such trace and it would require too many modifications in its internal structure. To pass over this problem, the planner performs a first commit to represent the initial state and extract the necessary knowledge from it. It then removes all the dynamic relations from the ontology and performs a second commit. It will represent a kind of pattern for each state involving a communication action.

[7]For the story of this thesis, this functionality was initially developed especially for this application.

For each of them, the planner just has to go back to the commit representing this pattern and update all the dynamic relations with their values in the evaluated state. An example of a commit-graph related to this solution is represented in Figure 5.5. Each commit thus represents a $K_S^{H_i}$ on which a REG can be performed. The advantage here is that the knowledge backtracking is internally managed by Ontologenius and thus optimized.

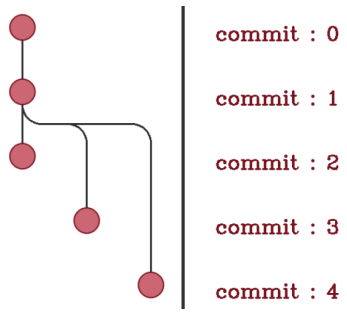

Figure 5.5: The commit-graph generated by Ontologenius for a plan with three communication action evaluated. Each evaluation generates a specific commit representing the state of the world at the moment of the communication.

5.4.3 Reducing the number of updates

A major limitation of the previous solutions is the need to update all the dynamic relations even if few changes have been made between communications. An unimplemented but feasible solution would be to update the ontology for each explored state during the planning process and create a commit for each of them. The advantage is that the backtracking has not to be managed by the task planner and only the modified relations have to be updated in the ontology. A commit-graph of such solution is represented in Figure 5.6. Assuming this graph to represent the same task decomposition of the previous one, we see however that more commits have to be performed even if fewer modifications are performed between the two. For tasks with many communication actions, it could lead to a performance gain but for tasks with less communication, it would require more updates than needed as even decompositions without communication have to update the ontology.

From the planner point of view, it would just have to assign UID to each state and created a commit using these UIDs. When backtracking, it would just have to checkout the ontology with the UID of the state to explore.

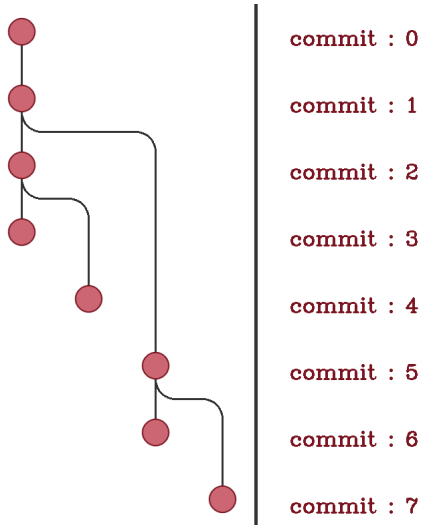

Figure 5.6: The commit-graph generated by Ontologenius for a plan with three communication action evaluated. Each explored state, with ou without communication action, generates a specific commit representing the state of the world at the moment of the action.

5.5 Results

In this section, we present three case studies. The two first ones are run in simulation with only minimalist setups. They respectively show that the estimation of the communication content during the task planning 1) can prevent execution dead-end and 2) can reduce the global communication complexity during the task. The third case study is run on a PR2 robot. It shows that our method makes it possible to compare different means of communication and to choose the most appropriate. The integration in the robotic architecture is presented in the next section.

All three cases studies are based on a cube arrangement task in the same principle of the RFID tag arrangement presented in the introduction. The human can only distinguish the cubes by their color and the digit written on them (one or two) if there is one. Three colored storage areas cover the entire surface of the table. An object is thus always in one of the areas. The area in which the cube is, is an additional piece of information usable for communication. The most complete RE is of the kind of: *"the green cube with the number 2 which is in the black area"*. For the three cases, only the robot knows the goal configuration but can not manipulate the cubes. It thus has to guide the human in the arrangement task. The robot can only point the cubes in the third case study. In the first two, it can only use verbal communication.

5.5.1 Prevent execution dead-end

In this first case study, we consider the introduction example recall in Figure 5.7 with the initial state (left) and final state (right). The cube C1 has to be moved

from the red area to the black area. The cube C2 has to be moved from the black area to the white area. The cube C3 does not have to be moved.

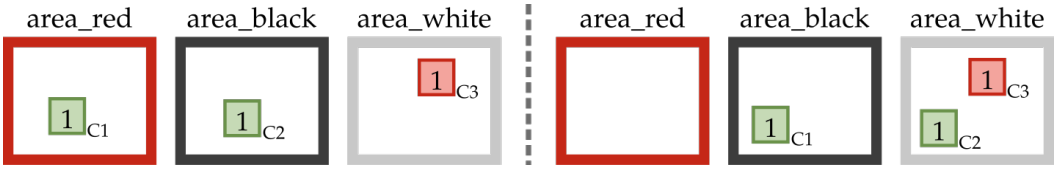

Figure 5.7: The initial state (left) and final state (right) of a task where the robot has to explain to the human partner how to move the cubes to complete the task. In this situation, explaining C2 first then C1 avoids a dead-end.

Taking into account the cost and the feasibility of the communication, we found with our method the plan of Listing 5.1. Cube C2 is moved first then cube C1. Doing the inverse order, after moving C1, the two cubes would be in the black area at the same time. Such a situation would cause a dead-end during the execution of the plan or at least require complex communication. Actions prefixed with HR are performed simultaneously by the robot and the human while the actions prefixed by H are only performed by the human. As a comment of each action are the relations to communicate.

```
HR - TellHumanToTake(C2) // (C2, isA, Cube), (C2, isIn, area_black),
                         // (area_black, isA, Area),
                         // (area_black, hasColor, black)
H  - Take(C2)
HR - TellHumanToPlace(C2, area_white) // (area_white, isA, Area),
                                      // (area_white, hasColor, white)
H  - Place(C2, area_white)
HR - TellHumanToTake(C1) // (C1, isA, Cube), (C1, isIn, area_red),
                         // (area_red, isA, Area), (area_red, hasColor, red)
H  - Take(C1)
HR - TellHumanToPlace(C1, area_black)  // (area_black, isA, Area),
                                       // (area_black, hasColor, black)
H  - Place(C1, area_black)
```

Listing 5.1: The obtained plan for the first case study where cube C1 must be moved from the red to the black area and cube C2 moved from the black to the white area. The lines beginning with H represent the actions of the human and the lines beginning with HR represent actions involving the human and the robot (communication actions). In green are the REG results for each communication action.

Considering once again the same initial state but with the goal to invert the positions of the two cubes, if the communication cost and feasibility are not taken into account during planning, both actions directly leading to the goal state (*i.e.* cube C1 moved to the black area or cube C2 to the red area) will lead to a dead-end at plan execution. To solve this situation, the task planner chooses to add a supplementary action. It consists of putting cube C1 in the white area that then leads to a problem similar to the previous one. This additional action avoids a dead-

end by making communication about cube C2 feasible. Another solution could have been to move C2 in the white area first, leading once again to a situation comparable to the previous one.

5.5.2 Reduce the overall communication complexity

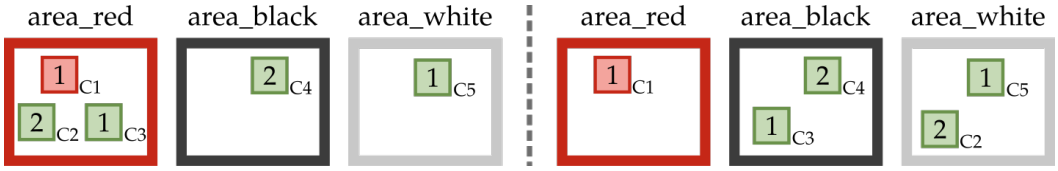

Figure 5.8: The initial state (left) and final state (right) of a task where the robot has to explain to the human partner how to move the cubes to complete the task. In this situation, explaining C2 first then C3 is easier than the inverse.

In this second case study, we show how the estimation of communication can be used to reduce the complexity of global communication. We consider the initial state and the goal state represented in Figure 5.8. Only cubes C2 and C3 should be moved. Our method finds the solution consisting of moving cube C2 first, then cube C3. With this order, cube C2 is referred by three relations: its type (*i.e.* cube), the number on it, and the colored area in which it is located. After that, cube C3 can also be referred to using only three relations being its type, its color and the colored area in which it is located. Considering the reverse order, this would have generated a more complex RE first for cube C3 with four relations: its type, its color, the number on it and the colored area in which it is located. The solution chosen by our method communicates a sum of six relations rather than seven with the reverse order.

5.5.3 Compare with other communication means

In this last case study, we show how the estimation of verbal designation communication cost can be used to compare it with other communication means, here pointing. Now, we consider twelve cubes. The initial state and the goal state are represented in Figure 5.9. Such a number of similar objects leads to long explanations to refer to certain cubes. Therefore, we aim the task planner to choose another means of communication to refer to cubes too difficult to explain verbally. The pointing action has a constant cost which is higher than simple verbal communication but lower than a complex one (with three or more relations to verbalize). To exemplify the comparison with other communication means, the arrangement order is predefined in this setup.

The generated plan is available in appendix B.2. The cubes C5 and C7 are chosen to be pointed instead of verbalized. Indeed, in the world states where these cubes need to be moved, verbal referring is considered to be too costly, thus a

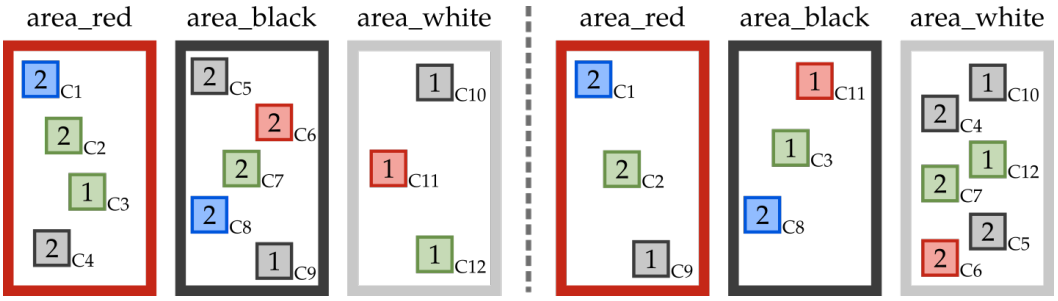

Figure 5.9: The initial state (left) and final state (right) of a task where the robot has to explain to the human partner how to move the cubes to complete the task. In this situation, some cubes are too complex to explain. Pointing them could help in some cases.

pointing motion is preferred. For example, the cube C5 in the initial situation needs a long and complex explanation that is: *"take the black cube with the number two which is in the black area"*. Even in the cases where the pointing action takes more execution time, it could require less cognitive load for the human partner and so make the human action faster.

Here, we see another benefit of our approach, it allows the planner to balance between the use of verbal communication actions, which can become complex in some states (hard to predict without a task planner), and other communication modalities. Here, balancing was done with other means of communication, but it could also be done with other actions such as a pick and place by the robot. In the task presented here, this has no advantage, because a pick and place by the robot would be slower than an explanation or a pointing and is more likely to fail.

5.6 Integration in a robotic system

The third case study has been implemented and integrated on a PR2 robotic platform. We extend the previously used architecture and integrate two new components as illustrated in Figure 5.10. The execution of the third case can be found in the video available at `https://youtu.be/3YnGh_t-UpY`.

The Symbolic task planner is HATP. It is requested by the supervision component which is able to manage the execution of the generated shared plan. The task planner can recover the initial state from the semantic knowledge base and update an ontology instance to represent the human future estimated knowledge base. In addition, it can request the generation of referring expression on a human planned ontology.

The second newly integrated component is a motion planner and execution component. For now, it only provides to the supervision a pointing service and automatically chooses the arm to point with. However, it does not consider the obstacles.

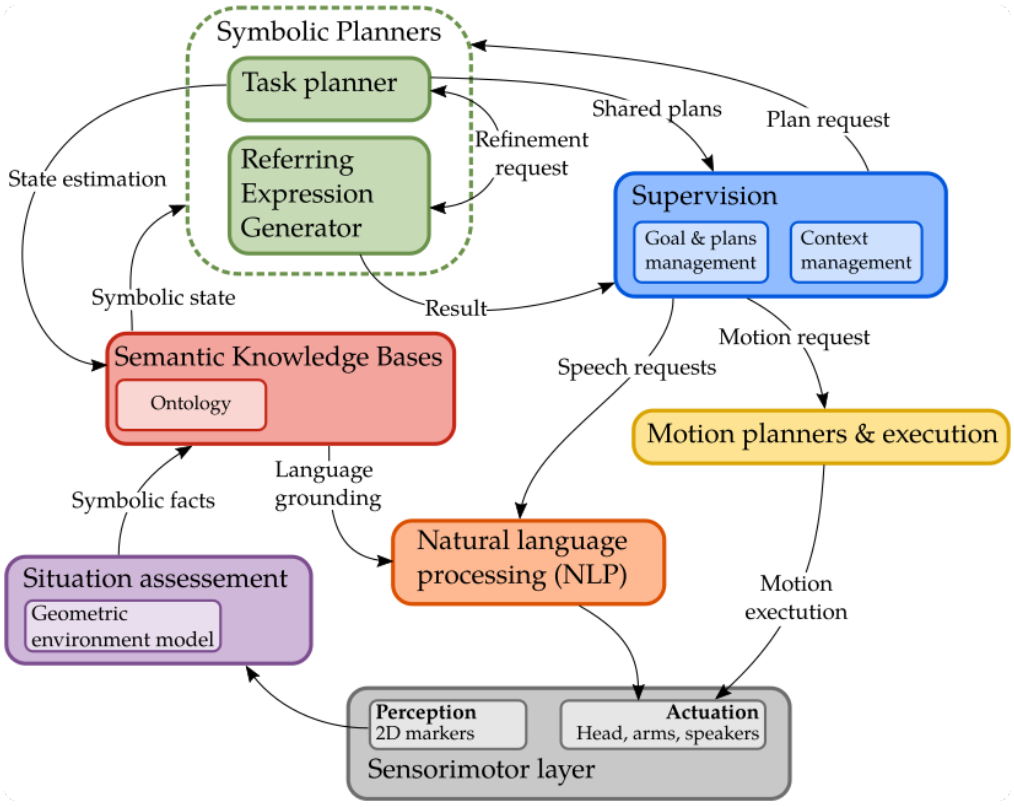

Figure 5.10: The architecture used to validate the method. The knowledge bases are continuously kept up to date through the situation assessment. The task planner can query the REG to estimate the feasibility and cost of future communications.

The geometrical situation assessment component has been changed from Robosherlock to a custom version of Toaster[8]. It is the successor of the software SPARK [Milliez 2014]. It can take as inputs several perception modalities and merge them in a coherent geometric representation. For our implementation, we simply use AR-tags which give precise enough position and allow us to identify objects with UIDs. The table and other static elements of the environment are not perceived and provided as static elements. The three storage areas are neither perceived and described as 3D virtual areas. Thanks to these areas, Toaster is able to compute the fact *isIn* for each of the cubes. From there, Toaster has been linked to Ontologenius to update the ontologies continuously. A representation of Toaster geometric environment is represented in Figure 5.11.

A major limitation of the current situation assessment component is that it can not perform perspective-taking. This means that even if it can represent the human as a particular entity, it can not estimate the state of the world from the human point of view[9]. It is not a problem for our application since all the

[8]https://github.com/sarthou/toaster

[9]We can compute if an object is in the field of view of the human but not working with a

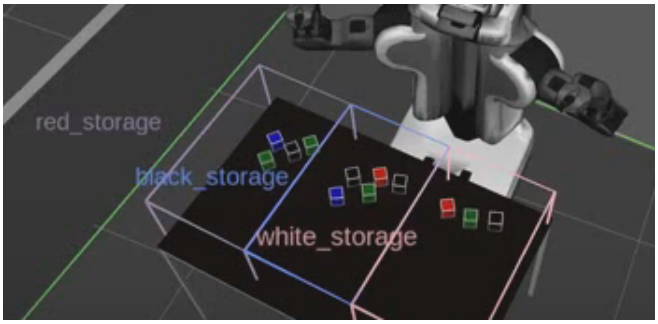

Figure 5.11: A visual representation in Rviz of the geometric state of the managed by Toaster. While the cubes are perceived with tags, the other elements are static.

elements are visible for both agents and the entire interaction is performed around the table. Toaster thus updates the robot and human ontologies with the same facts.

With this integration, the robot is thus able to analyse the current situation, and then to generate a plan to choose the more adapted communication mean for each cube to be moved. Consequently, at execution, at each step of the plan, the robot either verbally explains the cube to take or point to it.

graphical engine, it can not take into account the occlusions.

Chapter 6

Extending the REG with knowledge about past activities

Contents

6.1 Introduction 120

6.2 Related work 122

6.2.1 Interaction based Referring expression 122

6.2.2 HTN-based tasks representation in ontology 123

6.3 Structuring and gathering the knowledge 124

6.3.1 The three knowledge representations 124

6.3.2 The knowledge gathering scheme 126

6.3.3 Building the ontology 128

6.4 REG algorithm modifications 132

6.5 Results 134

6.5.1 One execution trace for five referring expressions 134

6.5.2 Impact of the extension on the performances 138

6.6 Discussion 139

Taking advantage of the link previously made between the ontology, the human-aware symbolic task planner, and the REG solver, we propose in this chapter a new way to generate Referring Expression based on agents past shared experience. Among this chapter, we first propose a representation of HTN as well as execution trace using an ontology. Then we extend the UCS-based REG algorithm to consider this new knowledge as a piece of information, usable to generate Referring Expression.

The contribution presented in this chapter is excerpted from our work, published in the proceedings of the IROS 2021 conference [Sarthou 2021b]. In this manuscript, the contribution is more detailed and discussed. In the continuity of the two previous, the presented work has been achieved in collaboration with Guilhem Buisan. He brought his expertise on HTNs to allow the best possible representation in an ontology.

6.1 Introduction

When two or more agents perform a collaborative task, although they may have a different perception of their shared environment, they can estimate the information they share and can thus use it to communicate about entities they estimate to be known by the others. This assumption is the one commonly used to develop and evaluate Referring Expression Generation (REG) methods through the use of caption of the environment[Duboue 2015]. These captions are images always taken from the hearer point of view. The image, or the related knowledge representation, is provided to the algorithm which has to generate a referring expression. This assumption has also been used when the REG has been applied to Human Robot Interaction (HRI) and can be compared to a robot spawning in an environment and having to designate an object. However, this designation occurs during a joint activity between a robot and a human partner meaning that the designated objects may have been used, moved, or already talking about. All of this information about the performed task can be seen as additional knowledge shared by the involved agents. We can thus refer to the entities through these past actions in addition to their attributes and relations with each other.

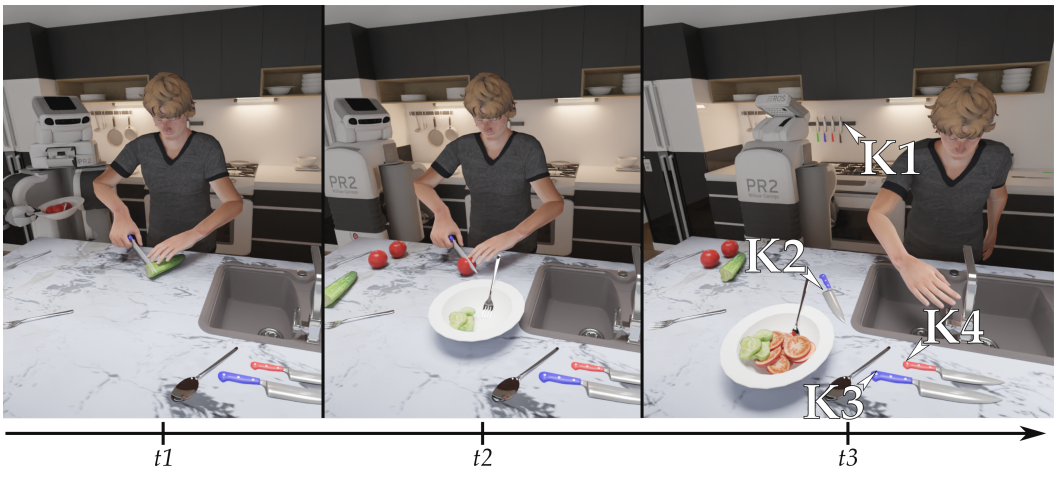

Figure 6.1: Referring to knife *k2* in the current situation (*t3*) is impossible if the robot is performing an action that does not allow it to see what is in front of the human. Considering previous steps of the human's task, the robot can refer to the knife through the action to cut a tomato (*t2*) or to cut a cucumber (*t1*).

Consider the caption of the interaction represented in Figure 6.1 at the current instant *t3*. The robot, in the back of the kitchen, has to ask the human for the knife *k2*. Since the robot is performing another action of the joint task, it cannot see what is in front of the human. Consequently, it cannot know and thus use any spatial relations about *k2*[1]. Therefore, the robot can only use *k2* attributes

[1]We could also consider an object known by the robot but for which it does not have any

(i.e. only its color) to generate an expression referring to it. Still considering only the current instant *t3*, two other blue knives hold in the kitchen being *k1* and *k3*. The knife *k1* is fixed to the wall in front of the robot meaning that it is already accessible to it and not to the human. This knife can thus be considered as being out of context and not leading to any ambiguity with *k2*. The other blue knife *k3* remains ambiguous since it has no perceptible attribute that differs from the one the robot has to refer to.

Until now, we have only considered the current situation *t3* and not the human-robot shared experience about the task they perform. At the previous instant *t2* the human was cutting a tomato with the knife *k2*. It was manifest to the human that the robot was observing the scene while he acted. This new information about the performed action could thus be used by the robot to generate a reference to the wanted knife in the current situation. A possible RE would be "*the knife with which you cut the tomato*".

Consider now the action a step before cutting the tomato at instant *t1*. The human was cutting a cucumber with this same knife. The combination of these two past actions can be seen as the task of preparing vegetables. The robot can thus also use this knowledge to refer to the knife. A possible RE considering the totality of the interaction would be "*the knife with which you prepared the vegetables*". The exploitation of shared knowledge about past activities in addition to the usual attributes and properties could lead to the generation of richer RE that could be easier to understand by the human partner. Besides, it allows to generate REs in contexts where the previous method was not effective.

This chapter is an extension of our previous work [Buisan 2020b] presented in Chapter 4. It has been integrated within a cost-based Hierarchical agent-Base Task Planner to estimate the feasibility and cost of REs during the planning process [Buisan 2020a], presented in Chapter 5. In this chapter we will thus aim to create the inverse link, making the REG able to use execution traces resulting from the execution of hierarchical plans generated by HATP. Like the previous chapters, we only focus on the content determination of the REG problem but continue to consider the need to have names in natural language to enable linguistic realization.

The main contribution of this chapter is an extension of the ontology-based REG algorithm by **considering past agents' activities**. A side contribution of this chapter is a proposal of a formalism to **represent Hierarchical Execution Traces** (executed HTN-based plans) in an ontology. Our previous contribution considered cost functions based on the properties of the used relations to represent the cognitive load required for a human to interpret the RE. In this extension, we propose to add customizable cost functions based on time, to represent the cognitive load required for a human to remember referred activities.

First, we review the literature concerning HTN representation in ontologies and discuss REG-related works that not only consider caption of situations. Then, we

information regarding its new location and searching for it. It would have to refer to it, to ask for the human help, without the possibility to use spatial relations.

describe the used knowledge bases and the usual structure of HTN and shared Hierarchical Execution Trace (HET). We then give an overview of how the knowledge bases should be updated and we describe the content of these updates in terms of how a shared Hierarchical Execution Trace (HET) is represented in an ontology. The extension of the algorithm is then detailed before ending with an efficiency comparison regarding the original version (see Chapter 4) and a discussion around five illustrative cases to show the solutions found by our algorithm depending on the agent's knowledge about past activities.

6.2 Related work

In the previous chapter about Referring Expression Generation we already gave a good overview of the literature of the field. In this chapter, we thus briefly discuss few works trying to consider an interaction. We then move on to a wider part about the representation of HTN and execution traces in ontology to see the kind of information our algorithm could use to generate a new kind of referring expression.

6.2.1 Interaction based Referring expression

In all the previously presented works, the REG is only performed on the current environment state. Williams in [Williams 2020] is the first to add a temporal aspect by considering a sequence of REG. Like others before, he starts from the idea that to designate an entity it is preferable to use properties known by the hearer and that he/she will easily identify. Where other works, our included, represent that with cost on properties that we assume to be representative for the hearer, Williams tries to take advantage of an entire interaction. During such interaction, two partners will generate RE. The presented algorithm thus try to re-use properties used in previous descriptions made by the partner. In addition, he has implemented a forgetting model based on decay or interfering to avoid the use of properties used too long ago. This method has been tested on a "Guess Who"-style game. This kind of game has the advantage that the used properties hold between the REG and thus can be re-used. However, this assumption can no longer be maintained in a real dynamic interaction where objects are manipulated and their properties modified all along the interaction.

Early in the field, Oberlander and Dale already showed that generating references to eventualities (*i.e.* to past activities or past events) can be done in the same way as generating references to physical entities [Oberlander 1991]. However, they never generate references to entities through the use of past actions. To close this short tour, Wiriyathammabhum et al. in [Wiriyathammabhum 2019] use RE involving past actions to identify a referred entity in videos but does not generate them.

6.2.2 HTN-based tasks representation in ontology

In robotic and even more in HRI, storing semantic information about past activities is needed to generate training data [Diab 2020] and learn from experience [Petit 2016], or to speak about what happened [Mealier 2017]. Some approaches represent the past actions using only structured sets of SQL tables [Mealier 2017], but such a representation lacks semantic information both on the involved entities (e.g. a robotic agent is a specific type of agent) and on the actions (e.g. a cut action is part of a salad preparation task). Since ontologies are fully suitable to represent semantic information about entities and their relations, they have been used to represent task planning knowledge. In [Sun 2019], a Robot Task Planning Ontology (RTPO) is proposed but the model does not consider the rich semantics involved by the hierarchical nature and intricacies of human-robot joint activities. For example, they represent the fact that the action "ChargeAction" is a "ChargeTask" while a more correct semantic would be that the "ChargeTask" is composed of a "ChargeAction". However, their representation does not allow such a correct semantic representation.

To represent episodes, the EASE-CRC has put forward the concept of narratively-enabled episodic memories(NEEMs) [Diab 2020]. Build from annotated perception events and sensor data, they provide comprehensive logs of tasks performed by the robot. The annotations are based on the terminology provided by the Socio-physical Model of Activities(SOMA) [Beßler 2020]. It proposes a high-level description of what is an event or an object in addition to the notion of plan. However, a plan is just a succession of actions and this terminology thus does not support the use of HET for the moment. A design pattern for the representation of such NEEMs in ontology has been proposed in [Krieg-Brückner 2020]. However, this pattern is too cumbersome for ontology developers and not practical to use in side-fields for which the safety of data input is not mandatory for the moment.

HTN is a very popular way for representing, planning, and controlling autonomous agents' activities [Ghallab 2004, Ingrand 2017]. It is a tree representing how to decompose abstract tasks into primitive tasks directly applicable by an agent [Erol 1994]. They are widely used for robotic planning as they allow to efficiently find complex plans by choosing between different task decompositions depending on the world state. Unlike more classical state-space search-based planning algorithms like STRIPS [Fikes 1971], HTN planning does not explore applicable actions until a goal is reached, but rather tries to fully decompose an abstract task into applicable primitive tasks. Moreover, HTN planning is often quicker as domains (HTN representations) are provided with expert knowledge through the hierarchical structure and task decomposition alternatives. In HRI scenarios, their usefulness is even more apparent. In [Lallement 2014], an HTN is used to generate human-robot joint plans. Furthermore, the hierarchical structure can be used to negotiate or communicate about high-level plans when details about realisation do not matter [Milliez 2016]. As an example, for a robot equipped with a charge plug and solar panels, an HTN may represent the abstract task of "ChargeTask" as being decomposed into either

"GoToChargeStation" primitive (or abstract) task or "GoOutside" task. An HTN planner would then explore these alternatives and generate the most appropriate plan depending on the current world state, here, the current weather.

To the best of our knowledge, few works exist on the representation of an HTN and their Hierarchical Execution Trace (HET) in ontologies. Umbrico et al. in [Umbrico 2020] describe the notion of complex tasks composed of simple tasks but do not go further in the representation. In [Freitas 2014] only the planning domain is represented. The major issue is that the ontology classes are used to describe the general HTN concepts (i.e. action, method) while the field concepts (e.g. the cut task in our example) are described using the ontology individuals. Hence, this representation does not make it possible to represent instantiations of abstract or primitive tasks. This distinction between the domain as a high-level knowledge and thus represented in the ontology classes and properties on one side, and the HET representation as an instantiation of this domain on the other side is important to us. It allows to both represent how a given task could be done and how it has been done during execution. Our work is closer to BOWL [Ko 2011], an HTN ontology for business process representation. Even if they have defined some specific relations for the Business-to-Business field, the general tasks, decomposition, and tasks links representation are interesting. However, BOWL only represents the HTN and not HETs, but does not preclude it.

6.3 Structuring and gathering the knowledge

In this section, we first present the main knowledge structures necessary to perform the extended REG through shared knowledge about past Human-Robot collaborative activity. We continue with an overview of the robotic architecture allowing us to acquire this knowledge, to understand the knowledge to be acquired offline and those to be acquired during the interaction. We end this section with the proposed terminology to represent HTN and HETs in the ontology.

6.3.1 The three knowledge representations

Three knowledge representations are used and can be grouped into two categories: the dynamic and static ones. The dynamic part is updated all along the course of the interaction. It is defined as $K = \langle K_S, K_E \rangle$ with K_S the semantic part already presented (see Section 2.1.3) and K_E the episodic part. The static knowledge base represents the planning domain as a Hierarchical Task Network (HTN).

6.3.1.1 The Hierarchical Task Network

In the previous chapter, we saw that an HTN is a way of representing a task to be planned, meaning to be fully refined and instantiated. In the present section, we go deeper into the HTN and give a more formal definition based on [Erol 1994].

An HTN, noted Pl, is defined as a set of tasks N. A task n is represented by a task name associated with a list of typed arguments. Taking the HTN illustrated in Figure 6.2, the task *cut* has the arguments *A, V, K*, which are respectively an agent, a vegetable, and a knife. The general term task can be refined into **primitive task** ($n \in Pt$ with Pt the set of primitive tasks) or **abstract task** ($n \in At$ with At the set of abstract tasks). A task is said to be primitive if it does not need refinement meaning that it can directly be executed by the robot or by a higher-level component[2]. An abstract task needs further decomposition and can not be executed by the robot as it is. In Figure 6.2, *PrepareSalad()* is an abstract task while *cut(A,V,K)* is a primitive task.

Then, we define the set of decompositions D. A decomposition is a pair $(\upsilon, N) \in D$ where $\upsilon \in At$ is an abstract task and N a set of tasks describing one way of decomposing υ. In Figure 6.2, the abstract task *PrepareVegetable(A,K,V)* has two decompositions *d8* and *d9*. Moreover, among the tasks set of a decomposition (i.e. the tasks used by the decomposition), can be enrich with a precedence order in which the tasks have to be performed. For example, in Figure 6.2, the tasks *peel(A, V, K)* and *peel(A, V, K)* are totally ordered under the decomposition *d8*.

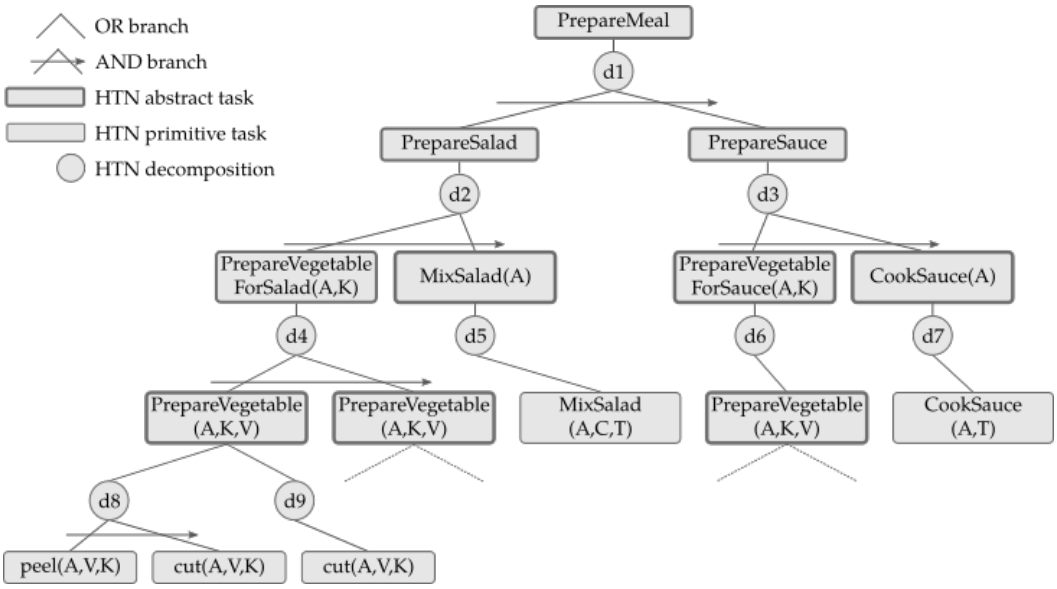

Figure 6.2: The domain of the high-level task *PrepareMeal*. It is used by the planner (HATP) to elaborate a Human-Robot shared plan through a context-based decomposition and parameter instantiation process (which vegetable V, which knife K, ...) including the selection of the agent A (Tony the human, or PR2 the robot) to which the abstract tasks and/or the primitive tasks will be allocated.

Given an initial world state and an initial task to decompose, a planner such

[2]Primitive tasks usually have preconditions and effects. We do not describe them in this thesis as they are not used.

as HATP elaborates a plan through successive decompositions from the initial task and respecting the constraints issued by the initial world state. A resulting plan is a sequence of primitive tasks. The planner thus tries to recursively select a decomposition for each abstract task encountered until it reaches primitive tasks. Besides, the planner has to ground every argument of the tasks into entities of $\mathbb{A}$ (the ABox of the knowledge base K_S) while respecting constraints regarding their types.

6.3.1.2 The semantic and episodic knowledge bases

The semantic knowledge base K_S is an ontology as described earlier. The episodic knowledge base K_E is a timeline also called datebase [Allen 1983]. It maintains temporal reference for every fact that varies in time. Representing only the changes, we can assume that a fact holds between two changes.

We defined it as a pair $K_E = (\{\langle r, \tau \rangle\}, \{\langle \alpha, \mathcal{T} \rangle\})$. The first element is a set of time-stamped relations like a classical datebase. The second element aims at representing the performed tasks. It is a set of pairs with α an instance of a task and $\mathcal{T}$ a temporal interval composed of two numerical values. The tasks defined in K_E are also represented as entities of K_S ($\alpha \in A$). Since they are semantically represented in K_S we can, inter alia, retrieve their types or arguments.

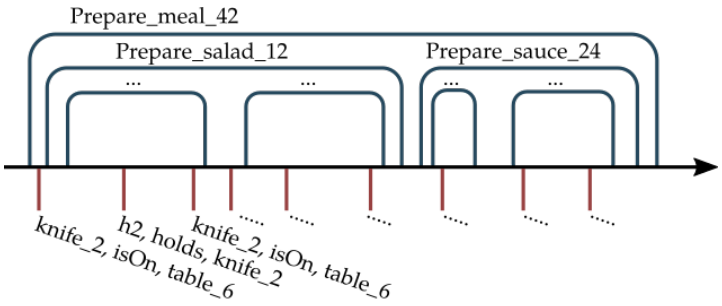

Figure 6.3: An example of timeline: On the upper part of the arrow are the tasks performed with their intervals. On the lower part are the relation changes.

Since we address HRI applications, in the same way, it has been done previously with the semantic knowledge base, we consider that the robot maintains a semantic and episodic knowledge base per human it is interacting with (K_S^{Hi} and K_E^{Hi}) in addition to its own (K_S^R and K_E^R). While the robot's own knowledge base is its personal truth, the agents' knowledge bases represent its estimation of the knowledge of its human partners.

6.3.2 The knowledge gathering scheme

We give now an overview of how the three knowledge bases are interconnected and how the semantic and episodic ones are updated. A minimal robotic architecture allowing the gathering of the necessary information is represented in Figure 6.4.

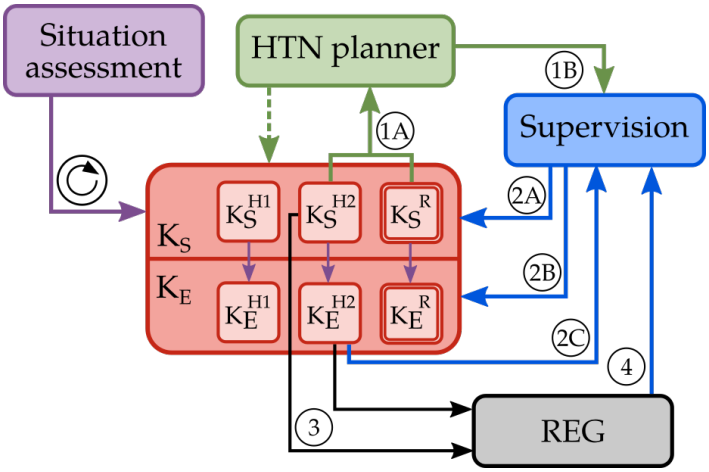

Figure 6.4: A "minimal" robotic architecture (based on the architecture presented in Figure 5.10) allowing to acquire and store the knowledge necessary to perform a REG using the past human-robot activities. The dotted arrow represents an offline acquisition, meaning knowledge acquired before powering on the robot. The other arrows are online interactions between the components. The numbering represents the execution order during the execution of a task. The robot knowledge bases (K_S^R and K_E^R) and the estimated mental states of its human partners (K_S^{Hi} and K_E^{Hi}) are updated permanently by the Situation Assessment component which tracks changes in the environment and by the robot supervisor which controls robot planning and execution activities and monitors humans actions.

First of all, the task decomposition description stored as an HTN is parsed offline. We aim at extracting from it a semantic description of the tasks composing it (abstracts and primitives), their parameters, and their hierarchical links through the decompositions. All these descriptions are then stored into the ontology as classes and properties (dotted arrow on Figure 6.4) in order to ground future traces of execution.

Once the ontology is initialized with a common ground, containing the description of the planning domain, we can start the interaction. During the interaction, the situation assessment updates the semantic KB of each agent with relations to the entities of the environment. Upon receipt of these facts, the semantic knowledge base temporally stamps them and stores them in the episodic knowledge base. It can also infer new facts thanks to the reasoning mechanisms. These inferred facts are also temporally stamped and stored in the episodic knowledge base[3].

On request of the supervision module, the HTN planner takes its initial world state from the ontologies of the involved agents (1A) and generates a shared plan (1B). During the execution of the shared plan, the supervision describes the performed tasks semantically (detailed in the following subsection) (2A) and stamps them in the timeline (2B) with their respective start and end times. While it knows

[3]For now, only the performed tasks will be used to generate the REs but we still wanted to have a more general scheme to understand the context in which we want to use it.

the tasks performed by the robot, it needs to gather data from the episodic knowledge base of the human partner (2C), to monitor their tasks. Descriptions of the tasks are not stored in the episodic knowledge bases as not having any temporal mean. It would be nonsense to say that a task has a given parameter at a given time. We first describe it as having parameters on one side (2A) and then we describe when the task has been performed on the other side (2B).

Upon a REG request from the supervision, the REG component explores the listener's both semantic and episodic knowledge bases (3) and returns the generated RE (4).

When the supervision component inserts the executed tasks in the semantic knowledge base, it thus creates a Hierarchical Execution Trace (HET). The execution trace can differ from the initial plan since it can be the result of plan repair or re-planning steps within the same global task achievement. The HET thus contains the actions which have been performed and their link to the higher level of abstraction. No forgetting mechanism is considered since we focus on "short" interactions (few hours) but adding one would avoid the knowledge bases to grow indefinitely and could also be used to represent the human forgetting mechanism. This could be for future work.

6.3.3 Building the ontology

The aim of the following representation is not for planning per se (since this is done by the human-aware task planner) but rather to allow to store and manipulate a description of the execution of the human-robot shared plans together with their hierarchical structure and the information provided by the situation assessment component. What is provided is the hierarchical task decomposition together with a semantic description of the entities used as tasks parameters, their properties, and relations to other entities in the environment. Moreover, the descriptions presented below are automatically generated from a HATP domain and plan description [Lallement 2014].

6.3.3.1 HTN in ontology

An HTN represents the general knowledge about how to decompose high-level abstract tasks into executable primitive tasks. Since it is a piece of general knowledge, we represent it in the TBox and the RBox of the ontology. It allows instantiating the executed plan as individuals of the ontology. As a reminder, the list of the symbols used to describe an ontology, and thus used in this section, are summarized in Table 2.1.

We first define the classes and properties common to any HTN representation. As shown in Figure 6.5, the upper class in the TBox to describe an HTN is **HtnConcept** from which the **HtnDecomposition** and **HtnTask** classes inherit. The HtnTask class is then refined into **HtnAbstractTask** and **HtnPrimitiveTask**. The RBox is composed of the properties **hasDecomposition**, **hasSubtask**,

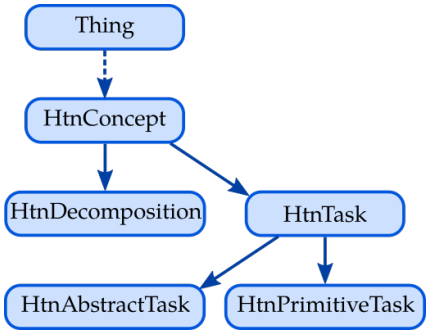

Figure 6.5: The upper classes used to decribe an HTN.

hasParameter, and their inverse (e.g. $(hasParameter, isParameterOf) \in Inv$). The property *hasDecomposition* links an abstract task to its decompositions. The property *hasSubtask* links a decomposition to the tasks (primitive or abstract) composing it. The property *hasParameter* links a task to any other entity.

To represent an HTN Pl we thus create in the ontology a new class t for each :

1. primitive task n such as $n \in Pt \Leftrightarrow t \in T \wedge (t,\ HtnPrimitiveTask) \in H$.
2. abstract task v such as $v \in At \Leftrightarrow t \in T \wedge (t,\ HtnAbstractTask) \in H$.
3. decomposition d such as $d \in D \Leftrightarrow t \in T \wedge (t,\ HtnDecomposition) \in H$.

For each decomposition d we describe the following relations using annotation properties:

1. A decomposition comes from an abstract task such as $v \in N \Leftrightarrow (v,\ hasDecomposition,\ d) \in E$
2. A decompositions has sub-tasks such as $d \in D \Leftrightarrow (d,\ hasSubtask,\ n) \in E$.

To put it into practice, let us consider the abstract task *PrepareVegetable* of the domain illustrated in Figure 6.2. The generated OWL representation of Listing 6.1 is first composed of the *PrepareVegetable* class inheriting from the *HtnAbstractTask* concept. Through the use of the property *hasDecomposition*, we state that the task has two decompositions being *PVDecomp_A* and *PVDecomp_B*. To refine these decomposition, we then create the *PrepareVegetableDecomp* class inheriting from the *HtnDecomposition* concept. It will be used to group all the decompositions of the *PrepareVegetable* abstract task. Considering the first decomposition (*d8* on Figure 6.2), we create a new class for it. The class *PVDecomp_A* thus inherits of *PrepareVegetableDecomp*. This decomposition is composed of two sub-tasks *Cut* and *Peel*. However, to keep track of the fact that they have to be executed in the context of a decomposition we refine them. We define *Cut_PVDecomp_A* and *Peel_PVDecomp_A* respectively inheriting of the primitive tasks *Cut* and *Peel*. With the property *hasSubtask* we then describe that *PVDecomp_A* has two sub-tasks *Cut_PVDecomp_A* and *Peel_PVDecomp_A*.

Even if a task is used several times in the same HTN, it will be described only once.

```
:PrepareVegetable rdf:type owl:Class ;
                  rdfs:subClassOf :HtnAbstractTask ;
                  :hasDecomposition :PVDecomp_A ;
                  :hasDecomposition :PVDecomp_B .

:PrepareVegetableDecomp rdf:type owl:Class ;
                        rdfs:subClassOf :HtnDecomposition .

:PVDecomp_A rdf:type owl:Class ;
            rdfs:subClassOf :PrepareVegetableDecomp ;
            :hasSubtask :Cut_PvDecomp_A ;
            :hasSubtask :Peel_PvDecomp_A .

:Cut_PvDecomp_A rdf:type owl:Class ;
                rdfs:subClassOf :Cut .
```

Listing 6.1: Description of the abstract task PrepareVegetable and one of its decompositions in the OWL language using the Turtle syntax.

In the latter description, we have only represented the hierarchy of the tasks. We now need to specify the parameters of each task. They are represented using the upper property *hasParameter*. Unlike the properties used to describe the hierarchy, these are intended to be used to instantiate the executed tasks. Indeed, this property is first refined into several ones, being *hasParameter.i* with $i \in \mathbb{N}$. This level of refinement describes the position of each parameter in the list of task arguments. We thus have hasParameter.0 the property for the first parameter of an argument list, hasParameter.1 the property for the second one, etc. These refinements are independent of the translated HTN.

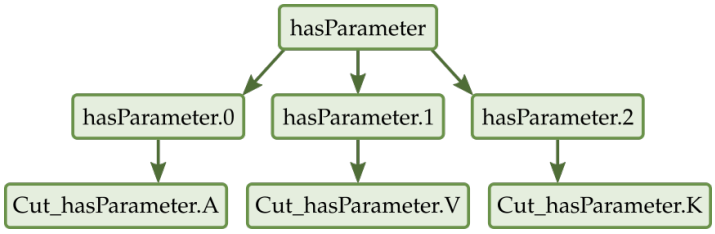

Figure 6.6: The properties hierarchy for the parameters of the *Cut* task.

Each of these properties is then refined and specified for every task of the HTN to describe. This second specification aims at representing the parameters with their name in the list of task parameters, their position in the parameters list, and the type of entities they can be bounded to. Taking the example of the primitive task *Cut* of Figure 6.2, the generated description is presented in Listing 6.2. The parameter *A* of the *Cut* task is at position 0 of the list of parameters. We thus define the property *Cut_hasParameter.A* as a refinement of the property *hasParameter.0*. The resulting hierarchy is illustrated on Figure 6.6 for the parameters of the *Cut* task. The parameter *Cut_hasParameter.A* aims at representing the agent performing the task. Consequently, the property has to link a *Cut* task to an agent.

We represent it respectively with the domain and the range of the property. The same process is performed for every parameter of each task.

```
:Cut_hasParameter.A rdf:type owl:ObjectProperty ;
                    rdfs:subPropertyOf :hasParameter.0 ;
                    rdfs:domain :Cut ;
                    rdfs:range :Agent .

:Cut_hasParameter.V rdf:type owl:ObjectProperty ;
                    rdfs:subPropertyOf :hasParameter.1 ;
                    rdfs:domain :Cut ;
                    rdfs:range :Vegetable .

:Cut_hasParameter.K rdf:type owl:ObjectProperty ;
                    rdfs:subPropertyOf :hasParameter.2 ;
                    rdfs:domain :Cut ;
                    rdfs:range :Knife .
```

Listing 6.2: Description of the *hasParameter* property specifications for the parameters (resp. the agent performing the task, the cut vegetable, and the used knife) of the *Cut* primitive task in the OWL language using the Turtle syntax.

6.3.3.2 HET in ontology

As explained with the gathering scheme, the HTN planner (here HATP) generates a hierarchical plan for a given human-robot collaborative task that is then executed by the robot and its human partner. Whenever a task (abstract or primitive) is executed by the robot or its execution by the human is perceived, its description is inserted in the agents' KBs. These executed tasks are thus instances of tasks and have to be grounded to the HTN described in the ontology.

```
:pv_3 rdf:type owl:NamedIndividual ;
      rdf:type :PrepareVegetable ;
      :hasDecomposition :decomp_pv_3 .

:decomp_pv_3 rdf:type owl:NamedIndividual ;
             rdf:type :PVDecomp_A ;
             :hasSubtask :peel_5 ;
             :hasSubtask :cut_7 .

:cut_7 rdf:type owl:NamedIndividual ;
       rdf:type :Cut_PVDecomp_A ;
       :Cut_hasParameter.A :tony ;
       :Cut_hasParameter.V :c1 ;
       :Cut_hasParameter.K :k2 .
```

Listing 6.3: A partial description of the initiation of a decomposition of a PrepareVegetable task and its primitive Cut task resulting from a plan and linked to the description of the domain. The description is provided in the OWL language using the Turtle syntax.

To explain how the task instances are described in the ontology, let us take the example of an agent executing a decomposition of the abstract task *PrepareVegetable*. A partial representation of this instance is represented in Listing 6.3 and drawn on Figure 6.7.

The human partner has performed *decomp_pv_3*, an instance of the decomposition *PVDecomp_A* being the first decomposition of the abstract task *pv_3* that is a *PrepareVegetable* abstract task. *pv_3* is thus an instance of the class *PrepareVegetable* and *decomp_pv_3* an instance of the class *PVDecomp_A*. We use the property *hasDecomposition* to state that *pv_3* has been achieved with the decomposition *decomp_pv_3*. Then, with the property *hasSubtask*, we describe that *decomp_pv_3* is composed of two instantiated sub-tasks *peel_5* and *cut_7*.

The primitive task *cut_7* is a *Cut* task but has been achieved in the context of the first decomposition of the abstract task *pv_3*. The individual *cut_7* thus inherits of the class *Cut_PVDecomp_A*. Because the *Cut* task has three parameters, they have to be instantiated. To do so, we use the refinements of the property *hasParameter*. Here the task has been performed by the human *tony* who has cut the cucumber *c1* with the knife *k2*. The instance *cut_7* is linked with *tony* through the property *Cut_hasParameter.A*, with *c1* the property *Cut_hasParameter.V*, and with *k2* the property *Cut_hasParameter.K*. All these instances correspond to the range of each property being respectively an agent, a vegetable, and a knife. The same process should be done for the primitive task *peel_5*. Moreover, the abstract task *PrepareVegetable* having also three parameters, we should also link *pv_3* with its instantiated parameters.

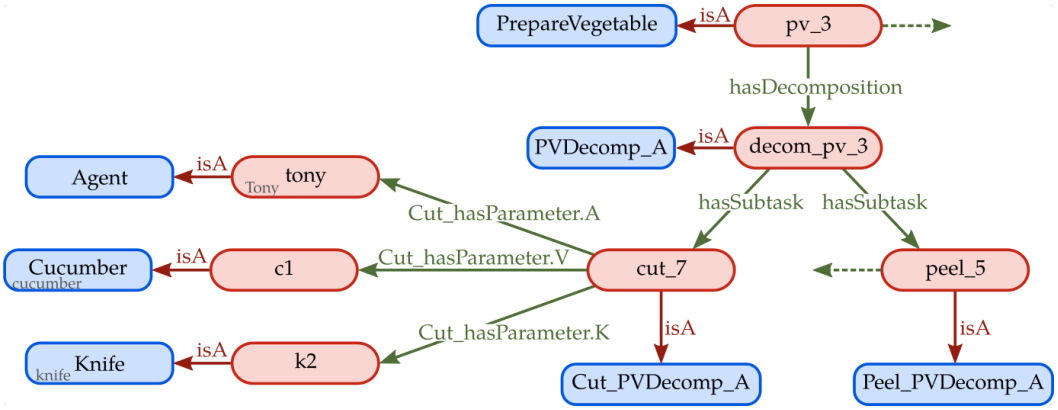

Figure 6.7: A graphical representation of the instance of a decomposition of the abstract task *PrepareVegetable*.

6.4 Modifying the REG algorithm to support the past experiences

Now the Hierarchical Execution Traces are described in the ontology, we present in this section how we have modified the original Uniform-Cost Seach (UCS) REG algorithm to make it able to use this new knowledge.

The first assumption we made to use tasks in a REG solution is that for each

task involved in the solution, all its parameters must be part of the solution. This constraint to get all the parameters of a task is important for linguistic realization. If this constraint were not satisfied, we could have results where even the agent having performed the task would not be referred[4].

Theorem 5 (The complete instantiation) *For any task n appearing in a RE, all its parameters* $p \in P | (p,\ hasParameter) \in Incl \wedge (p, n) \in Dom$ *must exist in the solution.*

The GoalTest function which assesses if a node is a goal is modified according to this constraint. In addition to test if $\mathcal{M}(x_t) = a_t$, it checks if every past tasks used in $\mathcal{T}$ are assigned with all their parameters.

Now we can test if a goal node involves all the parameters of a task, we expand the function finding new actions for the UCS algorithm. To not confuse the reader between the actions of the UCS and the actions that a robot can perform, we will rather use the term "addition" to speak about relations that can be added to a candidate RE in order to find a solution. We thus adapt the CreateAddition function to explore past tasks and their parameters (Alg. 7).

Algorithm 7 The modified CreateAddition function with the two newly introduced sub-functions

```
function CreateAddition(node)
  sucess, additions ← TypingAdditions(node)
  if sucess = True and additions ≠ ∅ then return additions
  additions ← CompletionAdditions(node)          ▷ new introduced function
  if additions ≠ ∅ then return additions
  additions ← ActingAdditions(node)              ▷ new introduced function
  additions ← additions ∪ DifferentiationAdditions(node)
    return additions
```

While the DifferentiationAdditions aims at exploring relations that differ from other ambiguous entities, the ActingAdditions complements it by exploring the tasks in which the ambiguous entities involved in the candidate RE are part of. In the latter, for each variable x_i of $\mathcal{M}$ having several substitutions, we search in K_S all relations involving the anonymous entity a_i and an abtract or primitive task. This means that we are looking at all the relations $r = (a_i,\ p,\ a_{task}) \in R \mid (p,\ isParameterOf) \in Incl \wedge (a_{task}, HtnTask) \in C$.

The second added function, CompletionAdditions, aims at making the RE valid. It ensures that any task used in a candidate RE has all its parameters inserted in that RE. For every task a_{task} used in a relation of $\mathcal{T}$ we search in R all the relations having for subject a_{task} and for predicate a property inheriting from $hasParameter$. Among all these relations, the ones not present in $\mathcal{T}$ are added as possible additions of the current node.

[4]This constraint could be an important limitation. It will be discussed in an exploratory work in the next chapter.

Regarding the order in which the addition functions are performed, we always start with the ones aiming to make the candidate RE valid. First, we try to type any anonymous entity. If typing additions have been found or some should have been found but were not, the function stops here. In the case where some have been found, we thus type the anonymous entities. In the other cases, no additions will be generated and the explored branch will be disused. If every anonymous entity is already typed, we try to complete the tasks involved in the candidate RE. If additions have to be done, we stop here and return them. By the way, this means that at the next evaluation of this candidate RE, the newly introduced entity will be typed first before continuing to complete the involved tasks. If every involved task has all its parameters, then and only then, we try to find new actions based on differences or on past tasks.

The cost of an addition representing a task (i.e. an addition involving the property $hasParameter$ or $isParameterOf$) is the cost of the task itself divided by the number of parameters. We chose to process in this way to avoid zero-cost addition and because every inserted parameter will already have a cost due to at least their typing. Thanks to the episodic KB, the cost of these additions can also be weighted depending on the amount of time passed since the task has been performed. This meets the decay theory used in [Williams 2020] and makes a preference over the more recent which could be easier to remember for the RE receiver. The determination of the properties' costs and tasks' costs will not be discussed here but we can mention [Belke 2002] and [Koolen 2012] which use learning techniques to estimate them.

Using the presented algorithm, a reference solution to refer to a knife through the primitive task *peel(A,V,K)* of the HTN of Figure 6.2 could be : *(?0 isA Knife), (?0 isParameterOf ?1), (?1 isA peel), (?1 hasParameter Tony), (?1 hasParameter ?2), (?2 isA Cucumber)*. The variable ?0 represents the referred knife and the variable ?1 represents the task to which the knife is associated. It could be verbalized as *"The knife with which Tony has peeled the cucumber"*.

6.5 Results

We present hereafter the solutions found by our algorithm on an illustrative example and show how the tasks estimated to be known by the RE hearer can impact the algorithm solution. Then, we discuss execution time measures to analyze the impact of such an extension regarding the original version of our UCS-REG algorithm.

6.5.1 One execution trace for five referring expressions

Let us take activities of the introduction example where PR2 and Tony were preparing a meal (Figure 6.1). We introduce a third agent, the human Bob, as illustrated in Figure 6.8. He is a spectator of the shared activity.

The shared activity is related to the HTN of Figure 6.2. With this HTN, a planner has elaborated a shared hierarchical plan illustrated in Figure 6.9. The

Figure 6.8: A PR2 robot and its human partner Tony, preparing a meal. Bob, a third agent, is looking at them.

primitive tasks are listed and organized according to a timeline. The abstract tasks hierarchy is shown on the right. The planner has assigned the tasks to two agents: a robot (PR2) and a human (Tony).

We define five cases depending on when Bob (the spectator) is in the kitchen where PR2 and Tony are collaborating for the meal preparation. The colored lines, next to the timeline, represent the moments where Bob is in the kitchen for each case. For example, in case 1 (black line), Bob is only aware of the primitive tasks *Cut(tony,t2,k1)* and *CookSauce(pr2, t2)*. The primitive tasks the robot estimated Bob to be aware of are thus added to his knowledge bases, both semantic and episodic. Moreover, if an agent is aware of all the tasks (primitive or abstract) composing an abstract task, he is also aware of the abstract task. In the first case, Bob is thus aware of the abstract tasks *PrepareVegetable(tony, t2, k1)*,*PrepareVegetableForSauce(tony, k1)*, *CookSauce(pr2)*, and *PrepareSauce()*.

In all five cases, the goal is for PR2 to refer to the knife *k2* to Bob. We assume that the communication about it occurs out of the kitchen. This means that no spatial knowledge is available to generate the RE. We present hereafter the solutions found for each of the cases and detail how they have been found.

Case 1: The algorithm first types *k2* as being a Knife with the function TypingAdditions. Since no task is involved for now in the candidate RE,

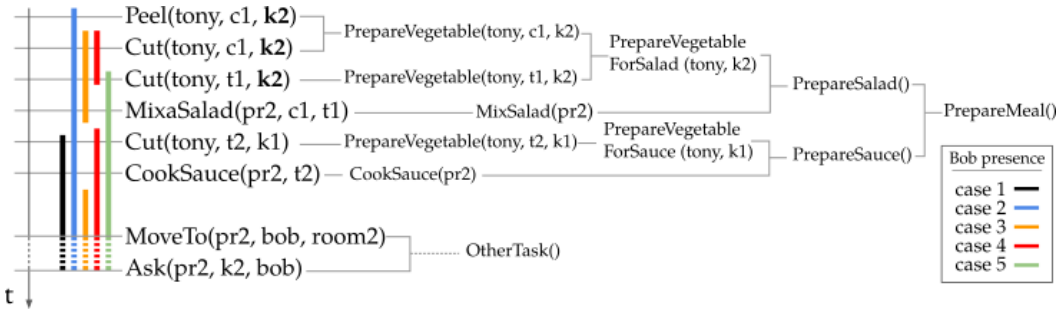

Figure 6.9: The hierarchical execution trace to prepare the meal and a trace for another subsequent high-level task, organized according to a timeline. In the current instant, PR2 is asking Bob, a second human, for knife *k2*. The five cases are represented by five colors at the left of the tasks and correspond to tasks seen by Bob, the human spectator. In case 1, Bob did not see any tasks involved in the preparation of the salad but was present during the sauce preparation. In the second case, he observed all the activities.

the CompletionAdditions does not find any additions. Then, because Bob is not aware of any past task involving *k2*, the ActingAdditions does not find addition either. Other relations, such as its color, can be added by the DifferentiationAdditions but none allow to solve the ambiguity. As a result, no solution can be found in this case.

Case 2: The TypingAdditions function first types *k2*. Then, the CompletionAdditions function has nothing to complete. This time all the tasks (primitive and abstract) performed by PR2 and Tony are known by Bob. Among them, three primitive tasks and three abstract tasks involve *k2*. The ActingAdditions function thus propose six additions of the form $(k2,\ isParameterOf,\ a_t)$ with a_t being the instances of the tasks. The DifferentiationAdditions function has only one possible addition being the color of $k2$. However, it does not help to solve the ambiguity and will give the same solutions as the other candidate RE, with one more relation making it more costly than the others at each step. In the five new nodes to explore involving tasks, the tasks are not labelled and are thus typed. At the next loop, because every candidate RE under exploration involves a task, the CompletionAdditions function adds one new parameter for each using relation with the *hasParameter* property. Considering the addition of the agent having performed the tasks, they both have a label and do not need to be typed. Because the abstract task *PrepareVegetableForSalad* has only two parameters while the five others have three parameters, the node exploring it is found to be valid at this step while the others are not. The parameter of the knife has been added by the relation $(k2,\ isParameterOf,\ a_t)$ and the parameter of the agent has been added by the relation $(a_t,\ hasParameter,\ tony)$. The solution RE is: *(?0 isA Knife), (?0 isParameterOf ?1), (?1 isA PrepareVegetable-*

ForSalad), (?1 hasParameter tony). Matching it in the knowledge base gives only one substitution for the variable ?0 being the target entity *k2*. The verbalization[5] would be *"the knife with which Tony prepared the salad vegetables"*.

Case 3: In this situation, Bob is only aware of two primitive tasks involving *k2* being cutting tasks and one abstract task being a *PrepareVegetable*. Because all have three parameters, at the difference of the previous case, the third parameter of each task is also explored and typed in a second time. At this step, all active nodes are valid. Because all have the same number of parameters, their cost difference depends on the time. The node with the lowest cost is the one with the more recent task. It could be the second cut task of the abstract task of the vegetable preparation. We could prefer the most precise one being the cut task. The solution RE is: *(?0 isA Knife), (?0 isParameterOf ?1), (?1 isA Cut), (?1 hasParameter tony), (?1 hasParameter ?2), (?2 isA tomato).* Only one substitution exists for the variable ?0 being the target entity. The verbalization would be *"the knife with which Tony cut the tomato"*.

Case 4: Previously, even if Tony had cut another tomato with another knife, it did not lead to any ambiguity since it was estimated as unknown by Bob. This time, Bob knows that Tony cut two different tomatoes with two knives so the previous solution is not a goal node anymore since two substitutions exist for the variable ?0. The algorithm thus refers to the knife through the cut task on the cucumber *c1*. The verbalization would be *"the knife with which Tony cut the cucumber"*.

Case 5: In this last case, the only task involving the knife *k2* estimated to be known by Bob is Alex cutting the tomato *t1*. However, Bob knew another cutting task with the tomato *t2* but with the knife *k1* which leads to ambiguity. The algorithm chose anyway the cut task as it is the only possibility. Despite this, our algorithm is able to find a solution by specifying the diverging argument that is the tomato. The tomato *t1* being involved in the *MixSalad* abstract task, the algorithm is able to use this second task to specify the argument of the first one. The solution RE is: *(?0 isA Knife), (?0 isParameterOf ?1), (?1 isA Cut), (?1 hasParameter tony), (?1 hasParameter ?2), (?2 isA tomato), (?2 isParameterOf ?3), (?3 isA MixSalad), (?3 hasParameter pr2), (?3 hasParameter ?4), (?4 isA Cucumber).* Matching it in the knowledge base gives only one substitution for the variable ?0 being the target entity *k2*. The verbalization would be *"the knife with which Tony cut the tomato that I mixed with the cucumber"*.

Over these five case studies, we saw that the algorithm is able to use either primitive or abstract tasks, to consider the moment where the tasks have been performed to choose the most appropriate one, and to use several tasks in a single RE.

[5]The proposed verbalization is automatically generated as proof of usability but the algorithm is not presented here.

6.5.2 Impact of the extension on the performances

With the previous results, we saw the kind of solution the algorithm can generate. With this new test, we want to assess the impact of the proposed extension, in terms of execution time, regarding the original version. To do so, we have chosen the realistically-sized knowledge base (101 entities, 36 classes, 40 properties, and 497 relations) used to evaluate the original version. For recall, it describes an apartment with three rooms including several pieces of furniture (tables, shelves) and objects (cups, boxes) linked through spatial relations (isAtLeftOf, isOn) and attributes (hasColor, hasWeight). Both the original algorithm and its extended version have been run over all the 77 entities inheriting from the "Object" class, representing physical entities. The knowledge base is still managed using the Ontologenius system and we do not pass by the ROS services to not be impacted by the communication time in our measures.

On this ontology, not having tasks described in it, the extension has a negligible impact. The average execution time is about 1.04ms for the original algorithm versus 1.16ms for the extension. Even for the most complex case, we pass from 6.25ms to 6.86ms. This difference can be due to the search of tasks even if none is present in the ontology. Indeed, for each encountered individual, the algorithm has to perform a query to the ontology to check if tasks exist or not.

To estimate the impact when tasks are present in the ontology, we chose to put ourselves in the worst case where tasks are described but can not be used to find a solution. To create this worst case, we add two actions, each having three parameters, to each object described in the apartment. Even if each action is unique (an individual per action), all are of the same type and have the same entities as parameters. In this way, the tasks will be explored but will lead to ambiguity, creating only an overload for the extended algorithm. Consequently, the solutions of both algorithms should be the same. The tasks description leads to an addition of 144 entities to the ontology (the 144 tasks) and 432 relations (three parameters per task). We estimated that such an amount of tasks could be the result of several hours of interaction. With this new setup, the impact of the extension is much more noticeable. For the most complex entity needing six relations in its solution, we pass from 6.40ms to 70.57ms. We note that 75% of the entities are solved under 5.96ms versus 1.32, 50% under 1.53ms versus 0.51ms, and 25% under 0.50ms versus 0.19ms. In addition, we observe that the longer the RE is to compute with the original version, the more noticeable the impact of the extension is. Moreover, because the more relations is needed, the longer the RE is to compute, we find here a part of the explanation. For each node coming from a difference addition (i.e. coming from the function DifferentiationAdditions), two additions representing tasks are proposed at the next step. We thus add two to the original branching factor, having an exponential impact on the number of generated nodes and thus on the execution time. However, in an HRI context, even the worst case is still acceptable and will not spoil the interaction. Moreover, we have put ourselves in the worst case where the added tasks can not be used to generate the RE. In more realistic

cases, we would have a diversity of tasks and their use in the REG may allow us to reduce the solution length for the most complex cases and hence reduce the average execution time.

6.6 Discussion

The contribution presented in this section allows the generation of a new kind of Referring Expression but at the cost of a longer computational time and with the constraint of a precise task description. Moreover, due to the need to use all the parameters of a task, the HTN has to be designed with the goal to be used for REG.

We can identify three major limitations of the current contribution:

- Because we consider the linguistic realization without any consideration of the combination of parameters allowing the generation of a sentence, all the parameters have to be used.
- Because the additions consist of the addition of a single relation at a time, all the parameters can not be inserted in a single addition. The consequence is an increase of the branching factor and thus of the execution time.
- Because the algorithm search for relations with precise properties in it, the method can not be applied to any task description such as the SOMA [Beßler 2020] one.

These limitations will be discussed in detail in the next chapter through a preliminary work aiming to generalize the Referring Expression Generation that supports the use of past shared tasks.

Chapter 7

Beyond binary relations in the REG

Contents

7.1 Introduction 142
7.2 Related work 143
7.3 Through the use of compound relations 145
7.3.1 Defining a compound relation 146
7.3.2 A lightweight representation of the verbal link 147
7.3.3 A strategy to explore compound relations 148
7.4 REG with compound relations 152
7.4.1 Exploring the compound relations 152
7.4.2 Determining a referring expression validity 154
7.4.3 From tree to radix tree 155
7.5 Results 155
7.5.1 The actor playing James Bond 155
7.5.2 The description of past activities as compound relations . . . 157
7.5.3 Assessing compound relations impact on performance 159

A number of works trend to semantically represent and describe robots' actions and tasks using ontologies. However, each representation has its own specificities, depending on the system it is used on and the exploitation it aims to enable. While our description of past activities fits our needs and our task planner, having a REG algorithm constrained by this latter is problematic. In this chapter we explore the use of n-ary relation, a common, even not yet standard, way to represent complex knowledge as actions. With the analysis of such a pattern, we present a novel REG approach, supporting the use of past activities, but also any complex knowledge represented through n-ary relations.

The contribution presented in this chapter is a preliminary work aiming to consider the limitations encountered by the contribution of the previous chapter. The proposed algorithm remains fully compatible with the previous one and hence we directly compare their performance. While it has not been the subject of direct experimentation on the robot, it should be usable as a drop-in replacement of the previous one. This section deviates a little from the field of HRI to be more anchored

in artificial intelligence. However, the ability to generate entity referencing in a more generic way is paramount for a robot to interact with humans.

7.1 Introduction

Representing the whole complexity of the knowledge composing our world into a machine-readable language is a central issue in artificial intelligence. Coming from the Semantic Web, we saw that the use of an ontology through RDF-based languages succeeded in establishing itself in the field of artificial intelligence and therefore robotics. However, what is often viewed as a limitation of ontologies is its capability to only represent unary and binary relations. Binary relations such as *"Sean Connery has the British nationality"* are described through the form of triplets *(sean_connery, hasNationality, british)*. Unary relations such as *"Sean Connery is an actor"* can then be transformed into binary relations through the addition of dedicated predicates *(sean_connery, isA, Actor)*. However, the description of more complex relations involving more than two entities is much more challenging using this kind of representation.

Taking the example of Sean Connery[1], if we want to refer to him[2], we could state that he is the actor playing the role of James Bond *(sean_connery, hasPlayedRole, james_bond)*. However, other actors played this role. We could also say that he is the actor playing in the film Goldfinger *(sean_connery, hasPlayedIn, goldfinger)* but once again others do. We could finally explain that he is the actor playing the role of James Bond and playing in the film Gold Finger. However, limiting us to the use of binary relations modify the exact information. A more accurate description would be that he is the actor playing the role of James Bond in the film Gold Finger. Here we see the necessity of relations involving more than two entities. In our example, we need to link the three entities that are the actor "Sean Connery", the role "James Bond", and the film "Gold Finger". Together, they describe a performance. Without being explicitly linked, these three pieces of information would not represent the performance. Moreover, without these links, we could give an explanation such as the actor playing the role of James Bond and playing in the film Rising Sun. Both pieces of information are true but do not make sense together.

To refer to an entity, being an object or an agent, such complex relations could be useful but have to be managed carefully to keep the link between each binary relation composing it. In the light of this consideration, we can observe that the description of past agent tasks used in the previous chapter is based on the same principle. Where we refer to Sean Connery through his role and the film he plays in, we have described the knife through the vegetable it was used to cut and the agent who used it. However, depending on the context of the conversation, it is

[1] In the case you do not know who is Sean Connery feel free to take another actor that you like but you will have to adapt the entire example.

[2] Obviously we want to refer to him without his name since we consider a person having recognized himself in the previous note.

not always necessary to use all the binary relations of such a complex relation. We may need only one. Trying to list the actors having the honorific title of "Sir", the referring expression *"The man having played James Bond"* could be sufficient. In the same way, to designate the knife, the sentence *"The knife you cut with"* could also be sufficient in some context.

In this chapter, we will try to generalise the REG algorithm to deal with non-binary relations and pass over the limitations of the previous chapter. The algorithm has to be more generic and should no more have any apriori of concepts and properties of the used ontology. First, we review the literature concerning the representation of non-binary relations in ontology. Then, we define what we will call a Compound Relation and how we can represent it. The modified algorithm is then detailed before ending with an efficiency comparison regarding the original version and the extended one.

7.2 Related work: A richer knowledge representation with n-ary relations

A fundamental feature of relations is their **arity**. It is the number of individuals they involve [Giunti 2019]. In this sense, unary relations involve only one entity while binary relation involves two entities[3]. What interests us here is n-ary relations with arity $n > 2$.

Well before being treated for ontology usage, that it is in RDF or OWL, several approaches have been proposed in the field of Artificial Intelligence through the use of semantic network[4] [Brachman 1979, Sowa 2014]. Deliyanni and Kowalski [Deliyanni 1979] were the first to explicitly treat the representation of n-ary relations with arity $n > 2$. They propose a semantic network composed of an element representing the relation itself. In this way, they have represented the assertion "*John gives the book to Mary*" with five nodes and four arrows as illustrated in Figure 7.1. The central node *el* allows linking the four elements of the assertion using only binary relations. The approach is today know as **relation reification** and has been used in many applications [Gangemi 2008, Welty 2006].

The general idea used in all the proposed approaches since Deliyanni is thus the creation of a **relation-class**. From this class, we create an instance of it that represents a specific n-ary relation. Then n binary relations are created to link the n entities to the relation-class instance. For example, in Figure 7.1, the instance of the relation-class is *el* and the n-ary relation is the set of the four relations *el* is involved in. For a more global view of the different proposed patterns, you can refer to the survey [Gangemi 2013].

For the use in an ontology, no standard pattern has been approved so far by the

[3]The entities involved in a relation do not have to be distinct. If the same entity is used twice in the same relation the relation is still a binary relation

[4]Here we close the loop with the hypothesis by Collins and Quillian of the structure of the semantic memory to be like a semantic network.

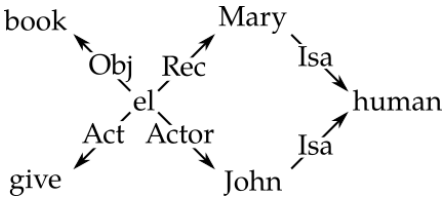

Figure 7.1: The semantic network used to represent the assertion "*John gives the book to Mary*". The element *el* is used to represent the global event.

W3C. However, a Working Group Note has been proposed for the standardisation of such relations in RDF and OWL [W3C 2006]. In the note, two patterns are introduced with two variants for the first one.

Pattern 1 (without subject): The first pattern is based on the introduction of a new class for relation and is illustrated by the relation reification in Figure 7.2. The class *Purchase* is a relation-class and its instance *purchase_1* is linked to the entities of the relation. This pattern is said to be **without subject** as all the relations are oriented from the relation-class instance to the other entities.

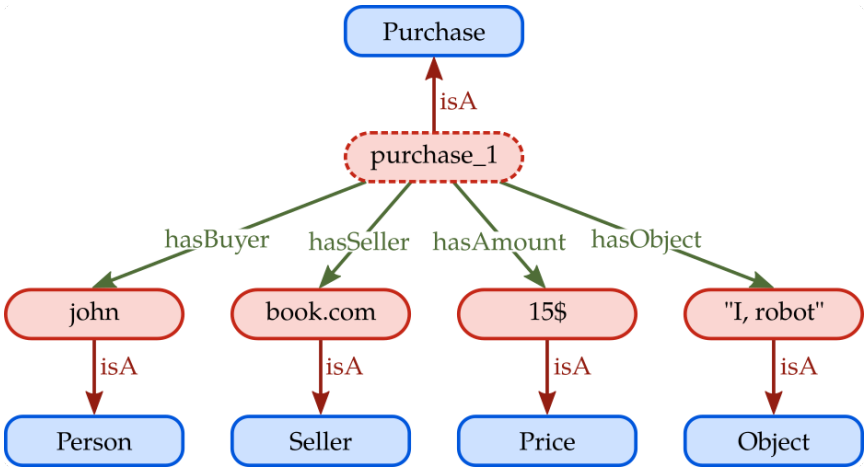

Figure 7.2: Ontological pattern 1 without subject proposed by the W3C Working Group. The described assertion is "John buys "I, robot" from books.com for $15".

Pattern 1 (with subject): A variation of the first pattern is illustrated in Figure 7.3. This variation is said to be **with subject**. The assertion described here is "Christine has a tumor with high probability". Here the subject of the relation is Christine. It is represented in the pattern by a relation oriented from Christine to the instance of the relation-class while the others are in the usual orientation. Such variation can be reproduced with the previous one by defining inverse relations. Defining the relation *isBuyer* and *isObject*, either John or the book can be the subject of the relation represented by *purchase_1*.

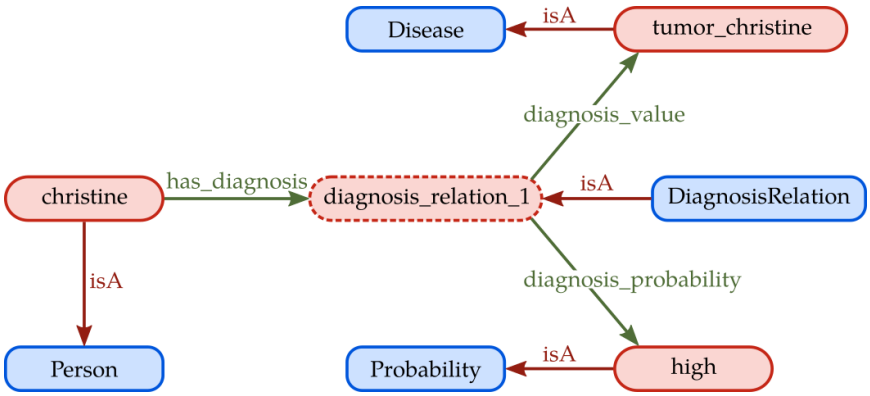

Figure 7.3: Ontological pattern 1 with subject proposed by the W3C Working Group. The describe assertion is "Christine has tumor with high probability".

Pattern 2: The second pattern aims at representing lists, which was not possible with the first pattern. With the previous pattern, it is assumed that the properties involved in the binary relation are only used once to identify uniquely each element of the relation. Wanting to represent the assertion "United Airlines flight 3177 visits the following airports: LAX, DFW, and JFK" the first pattern would not be adapted. With this other pattern, we create several instances of the relation-class each linked to the next and to an entity of the n-ary relation as illustrated in Figure 7.4. Because one of the binary relations go from an entity of the relation to an instance of the relation-class, it is said to be with a subject. This second pattern is dedicated to the description of lists.

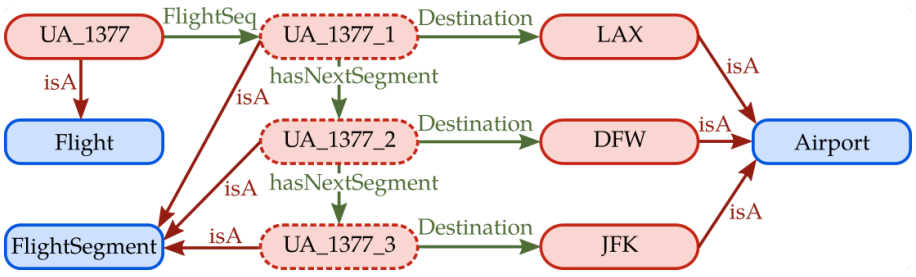

Figure 7.4: Ontological pattern 2 with subject proposed by the W3C Working Group. The describe assertion is "United Airlines flight 1377 visits the following airports: LAX, DFW, and JFK".

7.3 Through the use of compound relations

In the rest of the chapter, we consider the n-ary relations with arity $n > 2$ under the name **Compound Relation (CR)** because of the composition of binary relations to represent them on the principle of reification. The term relation will be used to speak about binary relations. We first define what is a CR with respect to ontology

definition and based on the first pattern without subject, proposed by the Working Group Note. Then, we present an algorithm to pre-process Compound Relation (CR)s with the objective to facilitate their use in the REG algorithm.

7.3.1 Defining a compound relation

To define the structure of a Compound Relation we take the example of the purchase made by John on the website book.com to buy the book "I, Robot" at 15$, illustrated in Figure 7.2. This statement is graphically represented in Figure 7.5a and the underlying pattern in Figure 7.5b. To represent the compound relation, we start by creating a virtual entity (the instance of the relation-class) that will be the common link for all the relations involved in the compound relation. We call this entity the **Compound Entity (CE)**. It is the dotted entity on Figure 7.5, respectively *purchase_1* and a_c. We consider as being part of the CR all the relations for which the CE is the subject: $(purchase_1, has_buyer, john)$.

Definition 7 (Compound Relation) *For any a_c being a Coumpound Entity, a Coumpound Relation is defined by $R_c = \{r_i \mid r_i = (a_c, p_i, a_i) \in R\}$ meaning the set of relations composing it.*

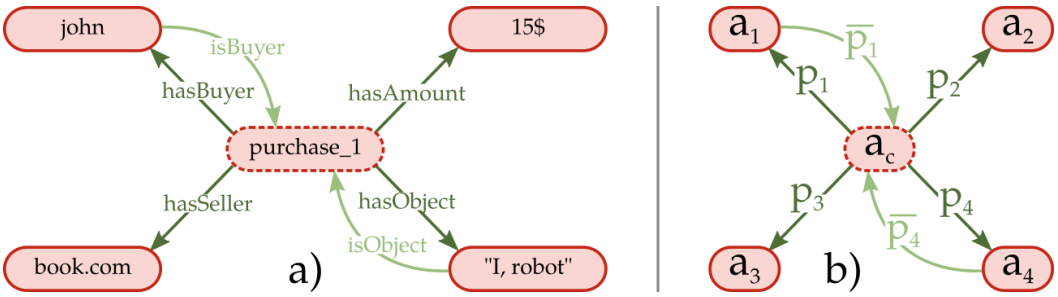

Figure 7.5: The graphical representation of compound relations. The dotted entity at the center of each representation is the so-called compound entity. The outgoing edges are the properties involved in the compound relation. The entering and faded edges are the corresponding inverse properties if any. The compound relation a) describes the purchase made by John on the website book.com of the book "I, Robot" at 15$. The compound relation b) is the underlying pattern of the previous example.

Regarding the previous definition, because any entity of an ontology could be considered as a CE, many sets of relations without a real link could be considered as a CR. To solve it, we could define an upper class common to all the CE, meaning the upper *RelationClass*. However, in the context of the REG, what better defines a CR is that to speak about one of its involved entities using the CR, we have to use other relation of the CR. In other words, to speak about Sean Connery using the role of James Bond, we have to speak about "Gold Finger" rather than "Murder on the

Orient Express" because even if he played in both films, he played the said role only in the first-mentioned film. Conversely, the relation representing his nationality can be used independently to other relations. To represent this verbal link, Giunti et .al [Giunti 2019] introduce a *parametric pattern* on top of n-ary relations (Compound Relations). Their parametric pattern for the purchase example is the following : *"() bought () from () for ()"*. While as humans we easily identify the place of each entity in the pattern, it is a more complex task for machines. This choice of pattern is explained by their complex representation where they assign a position to each involved relation. Regardless of the representation complexity, this kind of pattern raises two issues. First, the pattern describes the entire CR and does not aim to describe one of the involved entity through the CR (e.g. "*() who bought () from () for ()*"). Second, the pattern necessarily involves all the relations composing the CR, while in the context of the REG, we could only need a part of them (e.g. *"() who bought ()*" if John is the only one who bought this book in the present context).

7.3.2 A lightweight representation of the verbal link

To represent the verbal link, we also choose to use parametric patterns, patterns for short, defined as labels in the ontology. However, to know the position of each binary relation among the place-holders, we choose to integrate into the patterns the properties which can be used to form the relations composing the CR. Considering the example of Figure 7.5, our patterns have the following form : $\{?hasObject\}$ *bought on* $\{hasSeller\}$ *by* $\{hasBuyer\}$. In the following, we will prefer the more generic patterns $\{?p_4\}$ *bought on* $\{p_3\}$ *by* $\{p_1\}$.

Given a compound entity a_c with the previous label, to generate a referring expression using it, the place-holder $\{p_3\}$ should be replaced by a referring expression of an entity a_i where a_i is the object of a triple $(a_c,\ p3,\ a_i)$. Because we assume that a property can only appear once in a CR, we know that there is only one such object a_i. In our example, we have $a_i = a_3$ for the a_c CE. In this way, without predefined order, an algorithm can easily replace the place-holders by the RE of the entities a_i of the relations (a_c, p_i, a_i) of the CR. With the example pattern, the resulting sentence would be: *"The book bought on book.com by John"*.

Since we are in the context of REG, the CR will be used as a reference to one of the entities involved in it. This specific entity is called the **subject entity** of the CR. In our example pattern, the subject entity is a_4, meaning the bought book. A CR can thus have multiple labels (i.e. patterns) depending on the subject of the pattern and the involved relations in the verbal link. With our example, we might also want to refer to the website or the buyer. Moreover, if John purchased multiple books on the given website, we might need to refer to the book's price to refer to the book.

For a subject entity to exist, an inverse property $\overline{\mathrm{p_i}}$ must exist in the way that $(p_i, \overline{p_i}) \in Inv$ and $r_i = (a_i, \overline{p_i}, a_c) \in R$. If a_i is the subject entity, p_i is thus the **subject property** of the CR and is prefixed with a question mark in the pattern. In the example of Figure 7.5, only a_1 and a_4 (resp. John and the book) can be

subject of the CR. In other words, only these entities can be referred through the use of this CR. Among all the labels available to speak about the CR, the usable ones to speak about an entity are the ones for which the corresponding property is the subject property (i.e. prefixed by a question mark in the patterns). This choice to not consider the first in the pattern has been made to be adapted to any language. Among the possible labels of Listing 7.1, the patterns L1 to L5 could thus be used as a reference for a_4 (the book in our example) while the patterns L6 and L7 could be used as a reference for a_1 (John in our example).

```
L1 - {?p4} bought on {p3} at {p2}
L2 - {?p4} bought by {p1}
L3 - {?p4} bought on {p3} by {p1}
L4 - {?p4} bought at {p2} on {p3} by {p1}
L5 - {?p4} bought at {p2} by {p1}
L6 - {?p1} who bought {p4}
L7 - {?p1} who bought {p4} on {p3}
```

Listing 7.1: A part of the label set of the purchase compound relation.

The labels are not directly applied to the CE but to the class it inherits. This means that all the entities inheriting from a class having at least one label of the form of the previously introduced pattern, are CE. In a way, we define here a relation-class but only through the use of labels.

Definition 8 (Compound Entity) *Given a pattern ω, an entity a_c is a Compound Entity iff $\exists t \in T | (a_c, t) \in C \wedge \omega \in \mathcal{L}_t(t)$*

An advantage of this solution is that we do not define any new specific concepts or properties in the ontology meaning that any pre-existing ontology can be updated to be used in the REG process with CR only by adding labels, as n-ary relations are already used.

7.3.3 A strategy to explore compound relations

To use Compound Relations in the generation of Referring Expressions, the REG algorithm will have to explore each binary relation composing it. Moreover, to create a valid RE, the algorithm will have to find a combination of binary relations usable in one of the patterns of the CR. However, such a number of possibilities to test with the original algorithm can engender combinatorial explosion. In a graph exploration, an important parameter to avoid a combinatorial explosion is the branching factor. For the REG problem, an advantage of CR is that once we introduce a CR in the search algorithm we can directly know the relations involved in it. In this sub-section, our goal is thus to analyze the labels in the way to define the order in which the relations of the CR will be explored by the REG algorithm. By doing so, we will reduce its branching factor and thus avoid any combinatorial explosion of the REG algorithm.

7.3.3.1 A naive strategy to explore compound relations

The general strategy we want to adopt is to create a sub exploration tree, representing the exploration of a CR, that we will graft to the original REG exploration tree. The resulting constraints we will see in the following, are that we have to insert only one relation a the time (like the REG algorithm) and that the terminal nodes of the sub-trees have to be verbalisable.

Suppose we want to refer to the entity a_4 using the compound relation embodied by the compound entity a_c of Figure 7.5. This is made possible by the triplet (a_c, p_4, a_4) in the knowledge-base and its inverse $(a_4, \overline{p_4}, a_c)$. Listing 7.1 presents some alternative ways in which we can verbalise entities through the CR. In order to refer to a_4, we can only use the ones where p_4 is the subject property. Thus the labels L1 to L5 are the only ones that we can use. The subsets of the compound relation used by labels L1 and L3 are illustrated in Figure 7.6.

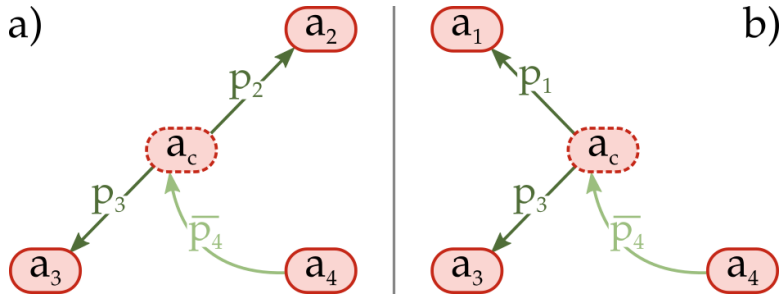

Figure 7.6: The parts of the compound relation used in the label patterns L1 (a) and L3 (b).

What interest us in these labels are the involved properties. Indeed, these properties will inform us about the relations to explore. From the patterns in Listing 7.1, we know that we can use label L1 to verbalize the triple set *{(a_c,p_4,a_4) (a_c,p_3,a_3) (a_c,p_2,a_2)}* and label L2 to verbalise the triple set *{(a_c,p_4,a_4) (a_c,p_1,a_1)}*. On the other hand, other combinations of triplets involving a_c cannot be used to refer to a_4. Therefore, we can see each label usable to refer a_4 as a set of properties and the collection of usable labels as a family of sets. In our example, the family of sets over S is the collection:

$$F = \{\{p_2\ p_3\ p_4\}, \{p_1\ p_4\}, \{p_1\ p_3\ p_4\}, \{p_1\ p_2\ p_3\ p_4\}, \{p_1\ p_2\ p_4\}\}$$

From there, our goal is thus to create a search-tree that will conduct the exploration of the different labels of a CR through the exploration of the properties composing them. This search-tree has few constraints and optimisation criteria:

1. The tree must be composed of a single root.
2. All the descendants of a node have a common prefix of the property associated with that node. In this way, the search-tree is more precisely a trie, also called prefix tree.
3. Walking through the tree from its root, we recompose all the subsets of the family F.
4. The width of the tree must be as small as possible.

A naive solution not minimizing the width of the tree is represented in Figure 7.7a) for the purchase example. We consider the root p_4 and create a branch per label. The resulting width is five. In Figure 7.7b) we merge the node representing the same property at an equivalent level. It reduces the branching factor for the beginning of the exploration by the global width is the same. Switching the elements p_2 and p_3 for L4 and keeping the merging principle, we can see in Figure 7.7c) that the width of the trie can be reduced to four. An advanced algorithm to build the graph could thus reduce the width of the trie.

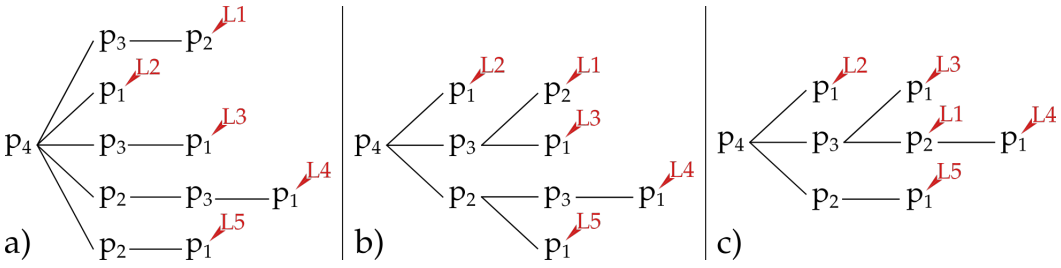

Figure 7.7: Two naive trie representations of the family of subsets extracted from properties involved in the label patterns. The trie a) considers the subject property as the root of the tree and creates a branch for each label pattern respecting the order of apparition of the properties. The trie b) takes the same construction rule as the a) but merges the common children of each node. The trie c) is the same principle of b) but the elements p_2 and p_3 have been switched for L4. While the two first tries have a width of five, the last has a width of four.

7.3.3.2 Advanced strategy to explore compound relations

To find a strategy to create a trie minimizing the width, we first have to study the characteristics of the family of sets. The first axiom that we can do on the subsets of our family is that they are neither totally ordered, as we can explore their element in any order, or partially ordered, as we can not compare their members. Therefore, we cannot use the tools of the order theory such as the Hasse diagram. Taking a look at mathematical approaches of tree-representation of set families [Bui-Xuan 2008], we face the problem that our set family does not respect some essential properties.

For each pair (A, B) of sets of S, we can not ensure that one of the following rules is true : $A \cap B \neq \emptyset$, $A \cap B \neq A$, or $A \cap B \neq B$. This means that we cannot ensure that each pair of sets are either disjoint or related by containment. Wherefore, our family F is not a laminar set family. Moreover, for each pair (A, B) of sets of S, we can not ensure that their intersection is non-empty ($A \cap B \neq \emptyset$), neither that their differences are non-empty ($A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$). Wherefore, our family F is not a cross-free or an overlap-free family but it does not mean that it is an intersecting and crossing family.

To limit our problem, we do a first assumption. Because the subject property p_s is the one having introduced the CR, we can assume that this property has already been selected among the others. Moreover, it will always be the common element of all the sets of the family. We thus consider it as the root node of the exploration tree and remove it from every set of the family S giving a new family:

$$S' = \{A \mid A = X \setminus p_s,\ \forall X \subseteq S\}$$

From there, we need to find the child nodes of the root in a way to minimize their number and that every subset of S' has at least one of their element attached to one of the child nodes. This sub-problem is a specification of the Hitting Set problem. It is defined as follows. Giving $F = \{S_1, S_2, ..., S_m\}$ the collection of subsets of S (i.e. $S_i \subseteq S, \forall i$) and a natural number $k \in \mathbb{N}$, we want to know if exists $S' \subset S$ where $|S'| < k$ such that $S_i \cap S' \neq \emptyset, i = 1, 2, ..., m$. In our case, we are searching k as to be as small as possible. In some way, the Hitting Set problem can be seen as a Set Covering problem, shown to be NP-complete [Karp 1972]. To avoid any combinatorial explosions, we thus propose a greedy algorithm.

Given a node n_i of the tree and its related family of set S, the quantity $|\{S_j \in S \mid x_i \in S_j\}|$ is the frequency of the element x_i in S. Among the elements of the universe of the current node n, we select x_{max}, the element with the highest frequency and create a child node with it. The family related to this new node is computed with the equation (7.1) and the family related to the current node is updated with the equation (7.2). These steps are repeated while S is non-empty to create all the children of the current node and this process is repeated for each created child node until it is possible.

$$S' = \{S_j \setminus x_{max}, S_j \cap \{x_{max}\} \neq \emptyset, \forall S_j \in S\} \tag{7.1}$$

$$S \leftarrow \{S_j, S_j \cap \{x_{max}\} = \emptyset, \forall S_j \in S\} \tag{7.2}$$

The tree resulting from this process is represented in Figure 7.8 for the purchase example. In the root node with the property p_4, the element with the highest frequency is p_1. We thus create a child node with this property and create its family. Updating the family of the root node, the family only contains the set $\{p_2, p_3\}$. Both having the same frequency, one is chosen over the other, here p_3,

and we create a new child node. After an update of the family of the root, it will be empty and all the children of the root have been created. The process is repeated for the two created nodes.

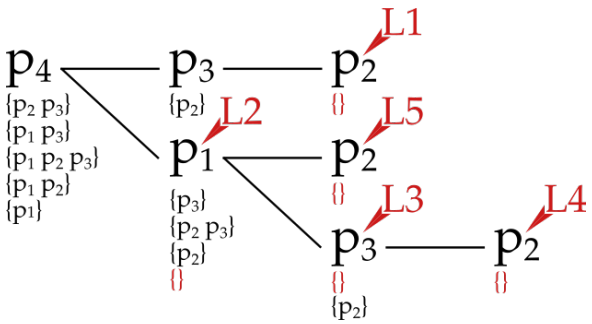

Figure 7.8: The trie with reduced width, representing all the labels of a compound entity in terms of involved properties. Each node is a property to explore. Attached to each node is the family of subsets that has to be decomposed. An empty set in a family related to a node (in red) signified that one of the initial subsets is fully represented in the trie, meaning that all the properties of a pattern will be explored by reaching this node. The width of the trie is three against five for the naive version.

In the following, such search-tree will be referred to as a Compound Tree (CT). Taking a part of it (i.e. taking one of its nodes as a local root) gives a sub-CT.

7.4 REG with compound relations

Thanks to the CE labels analysis, we have created a search-tree to lead the exploration of CR in the REG algorithm. In this section, we present the modification we made to use CR in the REG algorithm. The core of the algorithm based on the Uniform Cost Search algorithm is unchanged and is recalled in Algorithm 8.

7.4.1 Exploring the compound relations

At the difference of the tasks descriptions of the previous chapter, the current algorithm can not have any prior knowledge about the relations leading to the use of CR. A CR can only be discovered if a relation introduces a CE. However, an entity can be said to be a CE only thanks to the labels of one of its upper class. With regard to this information, we cannot have any function dedicated to the addition of CR and each newly introduced entity in a candidate RE has to be tested to assess if it is a CE or not.

The analysis of the labels of entities and their usable classes (i.e. their upper classes having labels) is a process already existing in the REG algorithm. It is performed by the function TypingAdditions of Algorithm 5. It initially aims at satisfying the naming need constraint (theorem 1). With some modifications, the TypingAdditions will be used to detect any newly introduced CE by testing if the

Algorithm 8 Uniform-Cost Search algorithm for Referring Expression Generation

```
function UCS_REG(problem)
  node ← a node with RE = CREATE-INITIAL-RE(problem.context), cost = 0
  frontier ← a priority queue of nodes ordered by their cost
  frontier ← INSERT(node, frontier)
  explored ← an empty set
  loop
    if EMPTY(frontier) then
      return failure
    node ← POP(frontier)
    if GOALTEST(problem, TOQUERY(node)) then
      return SOLUTION(node)
    add node.RE to explored
    for all addition in CREATEADDITION(node) do
      child ← CREATECHILD(node, addition)
      if child.RE is not in explored or frontier then
        frontier ← INSERT(child, frontier)
```

labels of the usable classes are in the form of a pattern. If they are, it returns the detected CE. To do so, the *additions* are now composed of a relation to be added and a CE if one has been found.

Once CEs have been detected, we have to create a Compound Tree (CT) for each one in order to lead the REG search process. To each node of the REG algorithm, in addition to the candidate RE and its associated cost, we introduce a map of CTs. This map links a CE involved in the candidate RE related to the node to its CT or one of its sub-CT. The management of these trees is done by the CREATECHILD that has been modified (see Algorithm 9). If the addition introduces a new CE, we create its related CT[5]. Otherwise, with the function GETSUBTREES we test if the new relation corresponds to one of the branches of one of the CTs of the parent node. If it is, we take the sub-CT corresponding to the relation. Taking the entity a_c as introduced CE through the relation $(a_4, \overline{p_4}, a_c)$, the CT of Figure 7.8 is first

Algorithm 9 Child node function modified to use compound relations.

```
function CREATECHILD(addition, parent)
  return a node with
  RE = parent.RE ∪ addition.relation
  cost = parent.cost + C(addition.relation)
  if addition.CE then
    CTs ← INSERT(CREATETREE(addition.CE), parent.CTs)
  else
    CTs ← GETSUBTREES(addition.relation, parent.CTs)
```

[5]For performance gain, all created CT can be stored in a collection of CT in order to compute them only once even if the same CE is introduced in two distinct branches of the UCS.

created. If in a second step the relation (a_c, p_1, a_1) is inserted, CreateChild would take the sub-CT having p_1 as root.

At this stage, we detect the introduction of CR and manage the CTs. The CreateAddition can now use the CTs to propose new additions. As describe with Algorithm 10, we keep the two functions TypingAdditions and DifferentiationAdditions. We introduce a new function CompoundAdditions aiming to complete the CR having been started. For each CT of the node, it proposes the relation of the form $(a_c, p_i, a_i) \in R$ such that the properties p_i are the branches of the root of the CT related to a_c. With the CT of Figure 7.8, the CompoundAdditions function would generate two additions (a_c, p_3, a_3) and (a_c, p_1, a_1).

Algorithm 10 The CreateAddition function modified to use compound relations.

```
function CreateAddition(node)
  success, additions ← TypingAdditions(node)
  if success = True and additions ≠ ∅ then return additions
  additions ← CompoundAdditions(node)          ▷ new introduced function
  additions ← additions ∪ DifferentiationAdditions(node)
    return additions
```

7.4.2 Determining a referring expression validity

Going back to the original definition of the REG problem, a RE is valid if the naming need is satisfied (theorem 1), all the introduced variables can be intantiated (theorem 2), and the variable representing the target entity can only be bound to the target entity (theorem 3) for a minimal validity.

For the compound relations, their validity criterion is that we can use one of their labels to speak about them. In other words, taking all the relations involving a given CE, we must be able to rebuild one of its family's sets. Each of its CR thus has to be complete (see theorem 6).

Theorem 6 (The CR completion) *A CR of family S is said to be complete iff given its CE a_c we can create, from the set of relations $\mathcal{T}$ representing a candidate RE, a set $v = \{p_i \mid (a_c, p_i, a_i) \in \mathcal{T} \vee (a_i, \overline{p_i}, a_c) \in \mathcal{T}\}$ such that $v \subset S$.*

From a technical point of view, such a constraint can be hard to compute for each node to be tested. However, during the creation of a CT, this information can already be known. During the CT creation, each time a child node is created with an empty set as family of sets, this means that a label of the CR is represented in its entirety at this node. In Figure 7.8, the completed labels are represented by the red arrows. We can thus store this information in the CT nodes. Because during the REG search process we cut down these trees taking each time a sub-CT, we just have to test if each current root of the CTs of the node to test can represent

a label or not. If all of them can, the candidate RE of the current node is valid regarding the CR completion.

7.4.3 From tree to radix tree

A limitation identified from the previous chapter was the addition of a single relation at each step while we can know that the candidate RE will not be valid because of the completion constraint. With the present method using multiple labels not involving all the relations, the limitation has been partially solved. In addition, thanks to the CT, the branching factor is limited and even if an addition does not lead to a valid RE, it will be used for several labels. However, in some cases, this limitation still appears. Considering the Compound Tree of Figure 7.8, starting from the root node p_4, we can go to the nodes p_3 and p_1. While the node p_1 creates a complete CR and makes a step toward another complete CR, the node p_3 does not. It makes a step toward the single label $L1$ but does not complete it.

To solve this issue, we can use the radix tree structure, also called compact prefix tree. It consists of merging each node that is the only child with its parent if the parent node does not represent a valid label. In the example of Figure 7.8, the node with the property p_2 could thus be merged with its parent p_3.

The major consequence of such modification is that the additions have no more to represent a single relation but a set of relations. Even if it makes the additions comparison harder to compute it reduces the overall branching factor. Keeping our example, if we add at once the relation involving p_2 and p_3 and that both introduce an anonymous entity, this means that at the next step the TypingAdditions function could type both in one step. Where previously these additions would require four steps, and thus branching at each, with this solution it only requires two.

7.5 Results

In this section, we present some results of the REG with compound relations. We start with the introduction example of Sean Connery's performance. Then we give performance measures using the setup of the previous chapter in order to assess the impact of the proposed modifications.

7.5.1 The actor playing James Bond

For this simple test, we describe actor performance using the pattern of Figure 7.9. Both the actor and the film can be used as a subject of the CR thanks to the presence of inverse properties. The set of labels attached to the *Performance* class is listed in Listing 7.2. Three are available to describe the actor and two (equivalent) for the film.

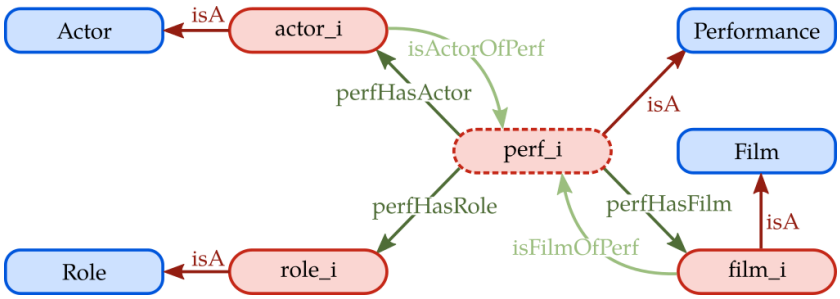

Figure 7.9: The compound relation pattern used to describe a performance of an actor with a role and a film.

```
L1 - {?perfHasActor} who played {perfHasRole} in {perfHasFilm}
L2 - {?perfHasActor} who played {perfHasRole}
L3 - {?perfHasActor} who played in {perfHasFilm}
L4 - {?perfHasFilm} in which {perfHasActor} play {perfHasRole}
L5 - {?perfHasFilm} in which {perfHasRole} is played by {perfHasActor}
```

Listing 7.2: The set of labels usable to describe the performance compound relation.

We first create an ontology describing two performances. The performance *perf_sean* link the actor *sean_connery* with the role of *james_bond* and the film *gold_finger*. The second is *perf_craig* with *daniel_craig* in the role of *james_bond* and in the film *casino_royale*. The individuals representing the films and the roles have labels while the others do not. Running our algorithm on this ontology we get the result:

$$(?0,\ isA,\ Actor),$$
$$(?0,\ isActorOfPerf,\ ?1),$$
$$(?1,\ isA,\ Performance),$$
$$(?1,\ perfHasFilm,\ gold_finger)$$

Matching it in the ontology, the variable *?0* matchs *sean_connery* and the variable *?1* matchs the performance *perf_sean*. Adding the performance *perf_gert* linking the actor *gert_frobe* with the role of *auric_finger* and the film *gold_finger* the previous solution is no more valid. Running the algorithm on the new ontology, we get the result:

$$(?0,\ isA,\ Actor),$$
$$(?0,\ isActorOfPerf,\ ?1),$$
$$(?1,\ isA,\ Performance),$$
$$(?1,\ perfHasFilm,\ gold_finger),$$
$$(?1,\ perfHasRole,\ james_bond)$$

With this simple example we see that with a single compound relation, the algorithm is able to find a solution by selecting the necessary information in it depending on the situation, while keeping the link between each of them.

7.5.2 The description of past activities as compound relations

From now, we see that Compound Relation can be used to represent complex knowledge with various entities linked together. Considering now the representation of an agent's past activities, presented in the previous chapter, such a complex knowledge could thus be represented using CRs. Figure 6.7 represented a part of an ABox of an instance of a decomposition of the abstract task *PrepareVegetable*. Focusing on the primitive task *Cut* and its instance *cut_7*, we can observe that the underlying pattern already was a CR. The graphical representation of this description is reported in Figure 7.10.

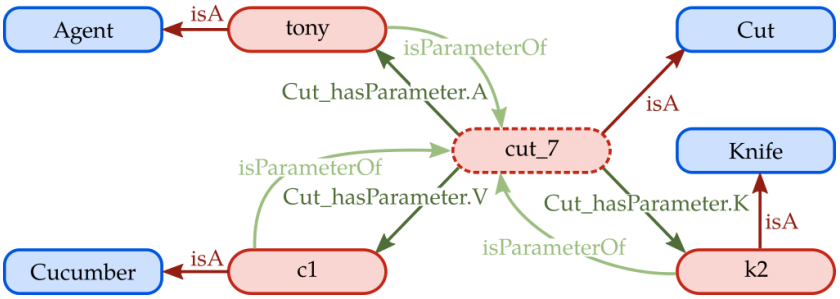

Figure 7.10: The underlined Compound Relation pattern used to describe a past activity. The entity *cut_7* can thus be considered as a Compound Entity.

At the light of the presentation of the CR, we see that the entity *cut_7* is the Compound Entity of the Compound Relation and that the class *Cut* is a relation class. Moreover, due to the presence of the inverse property *isParameterOf*, common to all the used properties, the three involved entities could be used as a subject entity. The only element to add to the previous representation is the labels, in the form of patterns. For the cut task, the proposed labels are listed in Listing 7.3. Among these labels, we see that for all the possible subject entities, two labels are available, involving one or two additional entities.

```
L1 - {?Cut_hasParameter.A} who cut {Cut_hasParameter.V}
L2 - {?Cut_hasParameter.A} who cut {Cut_hasParameter.V}
     with {Cut_hasParameter.K}
L3 - {?Cut_hasParameter.V} cut by {Cut_hasParameter.A}
L4 - {?Cut_hasParameter.V} cut by {Cut_hasParameter.A}
     with {Cut_hasParameter.K}
L5 - {?Cut_hasParameter.K} with which {Cut_hasParameter.A} cut
L6 - {?Cut_hasParameter.K} with which {Cut_hasParameter.A} cut
     {Cut_hasParameter.V}
```

Listing 7.3: The set of labels usable to describe the compound relation representing an instance of a cut primitive task.

To assess the utility to consider the past activities representation as Compound Relation, we take again the execution trace of Figure 6.9 and propose three new cases. The trace and the new cases are illustrated in Figure 7.11. As a reminder, the plan was executed by two agents, Tony, a human, and PR2, a robot. A third agent, Bob, saw part of the execution. Each case corresponds to a set of tasks, seen and thus known by Bob. The goal here is still to refer to the knife *k2*. The usable labels are thus L5 and L6, both referring to the knife used in the task.

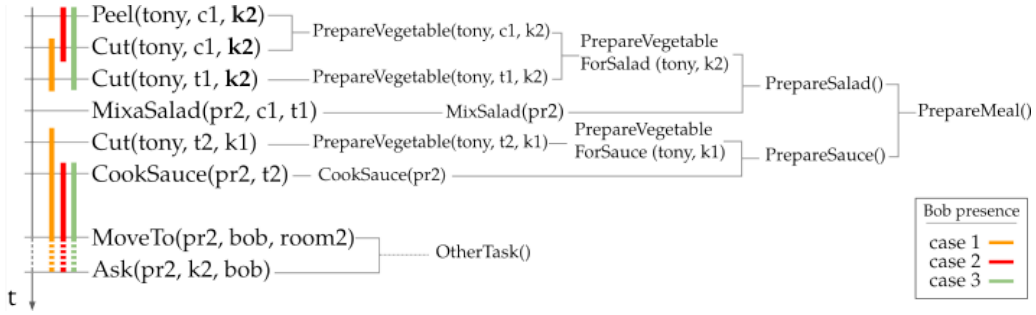

Figure 7.11: The hierarchical execution trace to prepare the meal and a trace for another subsequent high-level task, organized according to a timeline. In the current instant, PR2 is asking Bob, a second human, for knife *k2*. The three cases are represented by three colors at the left of the tasks and correspond to tasks seen by Bob, the human spectator.

Case 1: Bob only saw two tasks involving the knife *k2*. Both are cut tasks but on different vegetables. The algorithm can try to use the label L5, being shorter than L6. However, both tasks have been achieved by Tony. Consequently, the algorithm selects a RE using L6. The solution RE is: *(?0 isA Knife), (?0 isParameterOf ?1), (?1 isA Cut), (?1 Cut_hasParameter.A tony), (?1 Cut_hasParameter.V ?2), (?2 isA tomato)*. The algorithm not based on CR, would have given the same solution. The only difference is in its form. Here, the triplets representing the parameter use the properties really used to describe the CR, where the previous algorithm would have used the more abstract property *hasParameter*.

Case 2: In the second case, Bob only saw two tasks involving the knife *k2*. This time, it is two different tasks as one is a cutting task while the other is a peeling task. In this situation, the algorithm can use the label L5 and provide the solution: *(?0 isA Knife), (?0 isParameterOf ?1), (?1 isA Cut), (?1 Cut_hasParameter.A tony)*. We note that the cut vegetable is not present in the solution since it does not provide discriminative information. Bob saw Tony cut only a cucumber so specifying the vegetable would be useless. The algorithm not based on CR would have provided all the parameters, even if some are useless. The CR-based algorithm is thus able to find shorter RE when the current situation allows it.

Case 3: In the third case, Bob saw three tasks. Two of them are cutting tasks while the other is a peeling task. Even if one of these tasks is different from the others, for the previous algorithm all of them involved three parameters. Consequently, they would have had all the same cost. The selection would have been based on the time, selecting the most recent one. For the CR-based algorithm, the two cutting tasks need all of their parameters to be referred to but the peeling task can be used without reference to the peeled vegetable. Using the latter task allows to generate a shorter RE. Considering the peel task as having the same kind of labels than the cut task, the final solution RE is: *(?0 isA Knife), (?0 isParameterOf ?1), (?1 isA Peel), (?1 Peel_hasParameter.A tony).*

Over these three cases, we saw the CR-based algorithm is suitable to be used with descriptions of past activities. The advantage of this algorithm regarding the previously made is the form of the generated RE with the use of precise properties, and the ability to generate shorter RE in some situations. The most important point is that the algorithm is less dependant on the knowledge representation. It does not need a priori information about the properties of the representation. By simply adding labels to the existing representation, the current algorithm can be used with other task representations.

Finally, even if we will not illustrate it, the use of CR to represent past activities could allow us to be restricted by the information provided by a task planned. Taking the example of the cut task, it is performed on a support, a work plan. Even if this piece of information is not necessarily provided by a task planner, when the task is performed, the work plan on which the task is performed can be perceived by the robot. This additional information could be used to generate RE. To support this new information, we could add a label L7:

{?Cut_hasParameter.K} with which {Cut_hasParameter.A} cut on {TaskHasPlace}

With this new label, when a relation involving the property *TaskHasPlace* exists, it would be explored and when it does not exist it would simply be discarded. This means that we can add additional and optional information to any CR, and thus task representation.

7.5.3 Assessing compound relations impact on performance

Since the presented algorithm is able to manage the past tasks description, we present a comparison in terms of execution time with both the original algorithm and the one using past tasks. To do so, we take the Knowledge Base of the previous chapter (see Chapter 6) containing two tasks descriptions per entity inheriting from the *Object* class. To be used with compound relations, we add a label to each class representing a task. The label involves the three parameters of the task. In this way, we reproduce the constraint to use all the parameters and at the same time take advantage of the radix-tree optimisations. The knowledge base is still managed

using the Ontologenius system and not passing by the ROS services to not be impacted by the communication time in our measures.

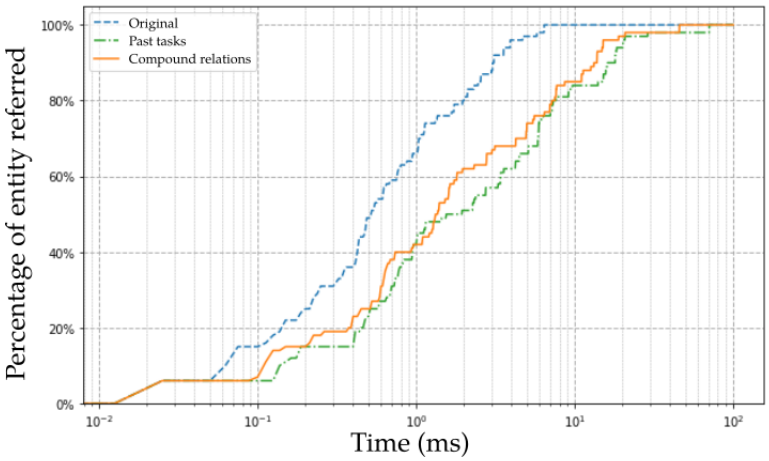

Figure 7.12: Comparison of the three algorithms regarding the percentage of successfully referred entities over time using a logarithmic timescale.

The original algorithm has been run without the possibility to use relations toward task since it is not designed for this use. Its performance is our point of comparison as we expect all algorithms to find the same solutions. For the recall, the described tasks are designed in such a way to not help in the RE generation and but ourselves in the worst case. The measures of the percentage of entities referred over time for all three algorithms are represented in Figure 7.12 and reported on appendix B.3.

With this setup, the current version performs slightly better than the previous one with an average resolution time of 4.17ms versus 5.53ms. It is however more than the original version having an average resolution time of 1.08ms. This difference must be tempered by the exploration of CRs that the original one does not do. While the majority of the entities are referred in a comparable time with the previous version, we can still note that the more complex entity requiring six relations is now solved in 45.71ms versus 70.57ms previously.

Here we have shown that the exploration of CR has an impact, even if being acceptable with regards to HRI applications. However, the major advantage of this solution is that in case no CRs are described in the KB, no extra exploration is needed. This is a difference with the previous solution (see Chapter 6) where even if no tasks were described, the algorithm was trying to find some at each step. This means that, unlike the previous solution, if no CRs are described in the KB, the current solution has the same performance as the original version.

Chapter 8

A robot in the mall: The MuMMER project

Contents

8.1 Introduction 162
8.2 Related work 162
8.3 Learning from exploratory studies 163
8.4 The deliberative architecture 164
8.4.1 Environment representation 165
8.4.2 Perceiving the partner 168
8.4.3 Managing the robot resources 168
8.4.4 Describing the route to follow 170
8.4.5 Planning a shared visual perspective 170
8.4.6 Navigating close to human 173
8.4.7 Executing and controlling the task 173
8.5 Embody architecture in a physical robot 174
8.5.1 Pepper in Ideapark 174
8.5.2 Pepper "in the wild" 175

Algorithms, representations, or pieces of software allow to study and exhibit precise features for robotic applications. It is however through their integration into a more global system and their application to a realistic task that we can assess their usefulness, their limits, and their possible extensions. In this chapter, we present the MuMMER project, aiming to develop a robot guide in a mall, and the resulting robotic architecture. We show how the semantic Knowledge Base (KB) and the route description contribution have been integrated and used by other components.

This chapter is a sum-up of an article submitted to the User Modeling and User-Adapted Interaction (UMUAI) Journal. This work has been achieved in collaboration with Amandine Mayima, Guilhem Buisan, Phani-Teja Singamaneni, Yoan Sallami, Kathleen Belhassein, and Jules Waldhart. In this chapter, we first give an overview of the European H2020 Project MultiModal Mall Entertainment Robot (MuMMER)[1], in the context of which the contribution of Chapter 3 was made. We then present the components developed by the LAAS-RIS team with a focus

[1] http://mummer-project.eu/

on the components in which I participated as well as their integration in a robotic architecture.

8.1 Introduction

In large scale indoor environments, like museums, shopping malls, or airports, the presence of large interactive screens, maps, or signs underline the importance of providing information on itineraries. However, reading such maps can be challenging and some information can be missing like the location of the shops selling a given product. To bring such new information and help people to find their itinerary in large indoor environments such as shopping malls, robots can be used.

To study this challenge and the underlined Human-Robot Interaction requirement, in the context of the European H2020 Project MuMMER [Foster 2016], we have developed and deployed a social service robot in one of the largest malls of Finland, Ideapark in the city of Lemp`a´alä. The resulting robot was able to chat with customers and guide them. The chatting has been brought by a partner of the project [Papaioannou 2018]. The contribution of the LAAS-RIS team, and thus the focus of this chapter, was on the direction-giving task.

With a mall having approximately 1.2 kilometers of pedestrian streets and more than 150 shops, having a robot accompanying customers would be time-consuming. Taking inspiration from the mall employees, we chose to verbally describe the route while grounding it with pointing gestures. The robot can however move a few meters if needed. The output of this project is a complete robot architecture that integrates a number of components. Each of them makes use of various models and decisional algorithms, all integrating explicitly human models.

First, we provide background information about robot guides and discuss how the human partner has been considered. Then, we present the human-human exploratory studies used to identify the required abilities for a guide robot. We then present the developed architecture and its components. We end this chapter with integration on a real robotic system with some details on its deployment “into the wild”.

8.2 Related work

A number of contributions have proposed robot guides, from the first museum guides [Burgard 1999, Siegwart 2003, Clodic 2006] to more recent robot guides in large areas [Bauer 2009, Triebel 2016]. A recent example is presented in [Chen 2017]. The developed robot is able to accompany the customer to its destination, then to point at it. Another robot presented in [Gross 2009] can help the customer to find specific products among all the shops of a mall. Most of these works are focused on the navigation aspect of the task. It requires environment mapping, localisation, and social navigation to take into account the presence of many humans.

Where previous contributions were mostly focused on navigation, others have

investigated the direction-giving task, meaning the fact to not accompany the customer but to describe the route to the goal. For example, [Cassell 2007] describes an embodied conversational agent giving route directions using deictic gestures. Within the Robovie project, the ATR-IRC laboratories have developed a robot providing route description through the use of utterances and gestures, and have highlighted the importance of their timing [Okuno 2009]. Kanda et al. in [Kanda 2009] and [Kanda 2010] have divided the direction giving into two steps. First, the robot points toward the direction to take, then it explains the full route. In addition, the robot can give recommendations for restaurants and shops. Finally, [Satake 2015b] showed a complete architecture of an information-providing robot able to move around a square in a mall. It embedded a map, an ontology, a speech recognition system, a dialog manager, a localization module, and a people tracker. As in their previous works, the robot verbalized utterances and used deictic gestures to give route directions. Numerous other contributions can be found but, only a few of them propose full architectures for an autonomous direction-providing robot, the most complete one being the Robovie robot presented above.

To the best of our knowledge, no system tackles the guiding-task by reasoning about the shared perspective. We mean that if the robot has to point a landmark not visible by the human at its current position, we want the robot to pro-actively propose to the human a pertinent placement. This is one of the basic bricks of our system and it is strongly linked to the key principles of Joint Action which involve the ability to establish and monitor joint attention [Pacherie 2012].

8.3 Learning from exploratory studies

To lead to robot abilities design and implementation, two human-human exploratory studies were conducted in collaboration with VTT Technical Research Centre of Finland. In addition to the current literature, it allows us to enrich our knowledge with effective route descriptions in the robot deployment environment.

The first pilot study consisted of a human guide providing route information. It consisted of one participant asking for shop directions to a guide working at the mall information booth. The analysis focused on gestures used to give guidance, the positions of the two protagonists in relation to the target shop and their interlocutor, and the gazes alternation. [Belhassein 2017] gave the first indications to consider resulting from this pilot study. Among these results, we can note a preference over the ipsilateral hand to the visual field of the target. This study also provides numbers of dialogue transcriptions use to study how guides effectively provide route descriptions and give examples of description.

A second exploratory study was then carried out to focus on more complex situations. Among them, we can note situations with two customers requesting directions simultaneously, a customer requesting for two shops at the time, or someone interrupting an ongoing interaction. Once again the formations formed by the guide, the customer, and the landmark were analyzed. An example of such a formation

Figure 8.1: Picture from the second Human-Human study [Belhassein 2017]. Here, the guide is giving the route description to reach a given shop by pointing at it. The formation formed by the guide, the customer, and the target was analyzed.

seen during the study can be seen in Figure 8.1. The full results can be found in [Heikkilä 2018] and [Heikkilä 2019]. Among these results, the study has shown that the guide usually points to the location of the target first and in a second step explains and points the different stages of the route.

8.4 The deliberative architecture

In this section, we present the robotic architecture developed to handle the direction-giving task. This architecture relates to Beliefs, Desires, Intentions (BDI) architectures. As explain by [Wooldridge 1999], such a kind of architecture is primarily focused on practical reasoning, meaning the process of deciding step by step which action to perform to reach a goal.

Figure 8.2 represents the architecture, its components, and their interconnections. Communication between components relies on ROS. In this chapter, we only present the components developed by the LAAS-RIS team, represented by the colored blocks on the architecture. First, we present the two knowledge representations in the form of geometric and semantic representations. Next, we introduce the components related to the sensorimotor layer. It is the situation assessment and the physical resource manager. Then, we present the components related to the deliberative layer. They are the Human-Aware Navigation, the SVP (Shared Visual Perspective) planner, the Route Handler, and finally, the one linking all the components, the Supervision. The Route Handler, part of the deliberative layer, having been presented in detail in Chapter 3, will not be detailed in this chapter.

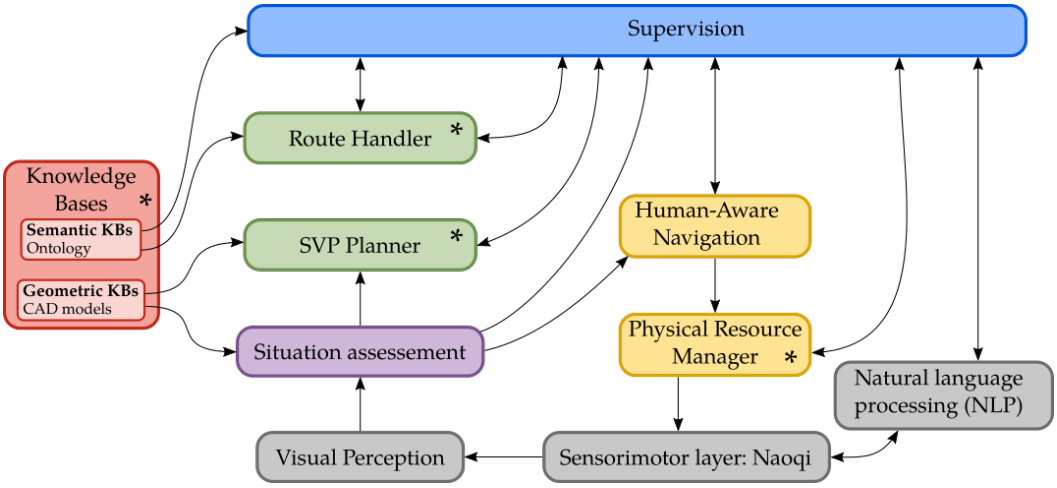

Figure 8.2: The general architecture developed for the robot guide. The components presented in this chapter are the colored blocks. The red components with the symbol * are the ones on which I participate. The visual perception and dialogue components have been respectively developed by IDIAP and HWU and are described by [Foster 2019]. Naoqi is a Softbank Robotics software.

8.4.1 Environment representation

For a service robot providing directions to people, we need information to understand humans' needs, information to compute the route to the goal, and information to compute the visibility of both agents to plan the pointing position. To understand the needs of a human wanted to be guided, we need information about the type of stores and the sold items. To provide so, [Satake 2015a, Satake 2015b] used an ontology. To compute the route to the final destination, [Matsumoto 2012] or [Okuno 2009] used a topological map. Each node of the graph is related to a 2D position of the environment. To estimate the human visibility of elements anywhere in the environment, [Matsumoto 2012] used a simplified 3D model where shops are represented by 3D polygons. In our implementation, we only used two types of representation of the environment: a **geometric** and a **semantic**.

Since the final deployment of the robot was in a Finland mall, we have built a mockup mall in our lab for development purposes. By mockup, we mean that shops signs have been displayed in the laboratory to create configuration similar to the real mall. The representations describe hereafter have thus been created both for the real mall and the mockup one.

8.4.1.1 Geometric representation

The geometric representation is used to compute the visibility of elements of the environment from different positions needed for the pointing of landmarks. However, because the robot does not accompany the person to the final destination and there-

fore does not move much, the possible visibility of the two agents is limited to their immediate environment. For this reason and due to the large scale of the Finland mall, we chose to geometrically describe only the subpart of the global environment that could be visible from the interaction area. For the rest of the environment, we represented the shops with 3D points only. These points are enough to point in the right direction. The resulting geometrical representation is a three-dimensional mesh model, as shown in Figure 8.3 for the mockup mall and in Figure 8.4 for the real one. We have represented in the 3D model all the elements that could hinder visibility, such as poles or panels. In this way, we can precisely emulate human visibility. The model was created from the architectural plans first and then refined with measurements in the mall.

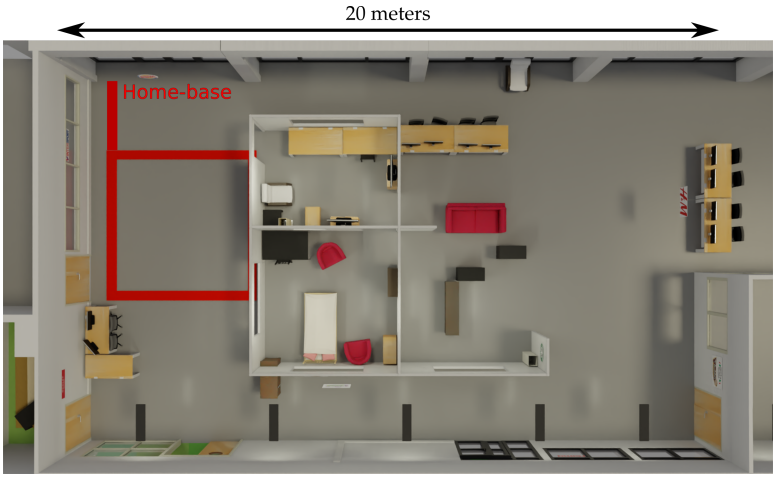

Figure 8.3: The 3D mesh model of the mockup mall at laboratory. The red square represent the interaction area as a square of 4 meters per 4 meters. Signs representing the shops have been place all around the environment.

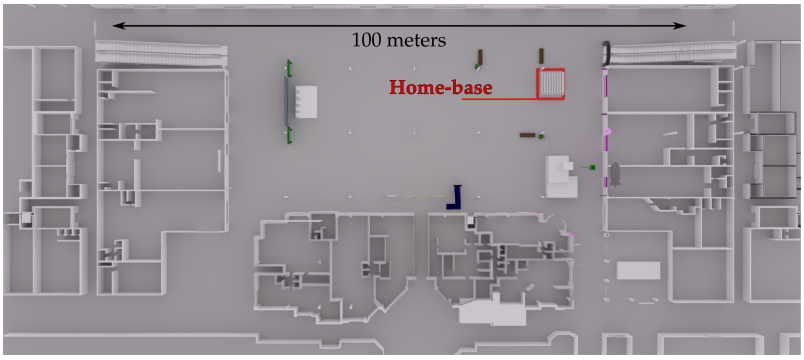

Figure 8.4: The 3D mesh model of the real mall in Finland. The entire mall having a size of 528.6 meters per 247.5 meters on two levels, we have only modelled the part which can be visible from the interaction area. It results in a model of 150 meters per 69 meters.

In order for the pointing planner to compute the visibility of the landmarks used for the route description, stairs, escalators, elevators, and store signs are represented each by a single mesh while the rest of the building is a unique 3D mesh. This means that a store is said to be visible if we can see its sign, which we think to be the most relevant element to see to recognize a shop.

The 3D model is also used to generate a navigation map, constraining the robot to move in the interaction area while avoiding obstacles in it.

8.4.1.2 Semantic representation

The environment semantic representation is oriented toward communication and human navigation in a mall. It makes use of the terms, affordances and actions that are needed to walk around in the mall. It represents the needed human-robot common ground knowledge and is used by the robot to perform dialog acts concerning shop names and categories a well as the types of products sold.

This representation takes the form of an ontology. It uses the Semantic Spatial Representation (SSR) (see Chapter 3) to describe the environment topology. The notion of place is then extended to represent information about the stores. It allows to define and refine the shared goal of the task by understanding the client's desired destination. We thus represented in it the stores' types, their names, and the items they sell with a richer semantic. It allows for example to represent that both soda and hamburgers are sold in fast-foods, which are types of restaurants, but that soda can also be found in a supermarket. Thanks to Ontologenius, the names of concepts are defined in different languages and with synonyms for these names. It allows the robot to adapt itself to the human partner language. Moreover, with Ontologenius, we endow the robot with the ability to recognize a set of names in natural language but that it will be prevented to use. For example, the robot can understand a reference to "bank" when a human says it but only refers to it as "ATM" or "cash machine" since there was no bank office in the mall. In addition, we used the provided fuzzy matching service to help the supervision system to handle ambiguities coming from the speech-to-text component. For example, if the language recognition module catches the word "Juwelsport", we can match it with "Juvesport", being a shop in the mall. This set of functionalities around the concepts' names facilitates the understanding of the partner's need and thus helps at increasing the quality of interaction.

An example of the final semantic knowledge represented in the ontology for a given shop is presented in Figure 8.5 for a clothes shop called "H&M". We find in this description the identifier of the shop, the category to which each store belongs, the topological information, the items sold and the names and synonyms in natural language and that for different languages.

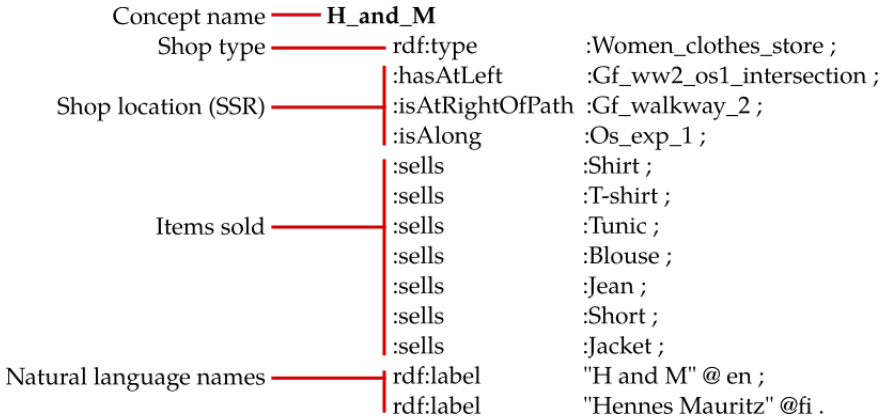

Figure 8.5: Representation of the knowledge about the shop H&M stored in the ontology. We have both purely semantic knowledge and topological information.

8.4.2 Perceiving the partner

The situation assessment component is based on the Underworld framework [Lemaignan 2018]. It aims at gathering perception information in the form of 3D position and orientation of human faces, with the 3D model and the robot state. With this information, it is able to generate the symbolics facts listed in Table 8.1.

Predicate	**Description**
isPerceiving	The robot is perceiving a human
isCloseTo	The human is within a distance of 0 to 1 meter of the robot
isLookingAt	The human is looking at the robot
isInArea	The human is in the interaction area
isEngagingWith	The human is close to the robot and is looking at it

Table 8.1: Facts computed and monitored during the direction-giving task.

8.4.3 Managing the robot resources

A humanoid robot such as Pepper can be seen as a composition of multiple physical components that can act independently of each other. For the pointing task, we identified four resources: the head, both arms, and the base. If we consider the head for example, at the beginning of the interaction, it is used to find people to interact with, but later it is used to track the human with the gaze. It means that during an interaction, several components would need to grant access to this resource. We thus need to manage the global picture of the ongoing task to enable this access in a safe way and to avoid failures (whenever two components would try to use the very same resource at the same time).

Moreover, in some cases, several resources have to be used simultaneously to perform a high-level action. To point to a landmark, one arm is selected to point

while the other has to be lowered. The base is then rotated if the arm reaches the joint limit to point a target on its back. If at least one of the involved resources is simultaneously used to perform another action, the overall high-level action will fail as the global posture will no more be clear. For example, if the human gets too close to the robot and a component tries to move away from a little, the arm would no more point in the right direction.

Thus, the correct handling of all the resources is critical for performing the task, but it can be cumbersome for a deliberative component, such as the Supervision, to do all the micro-management required. To tackle this issue, we designed a physical resource management system to provide an abstraction of each resource. For each of the identified resources, we instantiated a component called **Resource Manager (RM)**, having multiple inputs. It is endowed with a low-level decision-making ability allowing it to vote for the next command from a given input to be executed. Its choice is based on events, priorities, and commands importance. Inputs are divided into two types: **atemporal inputs** and **finite state machine inputs**. Atemporal inputs are unit overwriting buffers receiving commands which can be run continuously and preempted at any time. Each atemporal input has semantic meaning. For example, the head manager has an atemporal input dedicated to the monitoring of the interacting human. This input thus receives permanently a command to look at the head of the human to be monitored, even if we do not have to look at him. When this input is voted, the robot looks at this point. Finite state machine inputs are prioritized queues of state machines. Each state corresponds to a

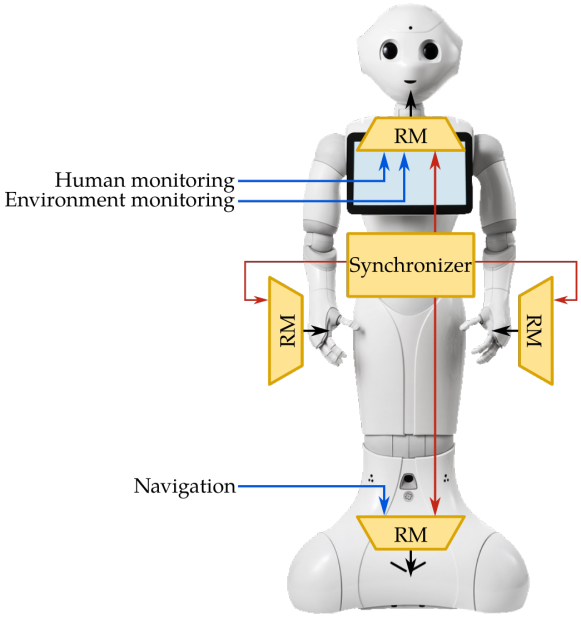

Figure 8.6: Representation of the resource management system with four resource managers and a synchronizer. The red arrows represent the state machines inputs and the blue arrows represent the atemporal inputs.

command to be executed. Transitions can be events received from other components or durations. In this scheme, a state machine cannot be preempted, as it is seen as a set of commands being part of the same high-level action, like a pointing.

To deal with high-level actions requiring multiple resources, we created a **Resource Synchronizer**. It does not have atemporal inputs but only one finite state machine inputs. It can thus receive finite state machines handling multiple resources, so-called coordination signals. The synchronizer checks if the needed resource managers are free or preemptable, if so, it dispatches the state machines to the proper resource managers and ensures their synchronicity when needed. The synchronizer also reports the status of the ongoing coordination signal to the Supervision component to monitor the progress of the action. The global resource management scheme is illustrated in Figure 8.6 with four resource managers and one synchronizer.

8.4.4 Describing the route to follow

The route description process has already been presented in Chapter 3. However, in the context of the MuMMER project, the robot has to be deployed in a Finnish mall. Consequently, the description of the route to follow has to be in Finnish.

The Finnish language is not suitable for pattern-based solutions with placeholders to fill. Indeed, nouns and verbs have a large number of inflection types, some of which are more common than others. Since it would be too difficult to build sentences as we did for the English version, we chose to generate the explanation in English and then translate it using the Google Translate API. An issue with this solution is that even the names of the stores are translated by giving incorrect sentences. We could think to fill the placeholders after the translation but in some types of sentences, store names would have to be declined. To pass over this later issue, we have limited the variety of sentences to only keep the ones for which it is not necessary to decline store names. This gives less diversity in the way the robot expresses itself but allows more reliable translations. Finally, in some cases, we had to degrade the English quality, creating syntactically incorrect English sentences, so the Finnish translation can be right.

8.4.5 Planning a shared visual perspective

When the robot has to point to a target, two criteria have to be respected. First, the human has to be able to see the target. Second, the human has to be able to look at the pointed target and at the robot without turning the head too much. It goes the same for the robot as it has to see the pointed target, meaning not to point toward a wall and be able to simultaneously point at the target and look at the human. Consequently, to point a target in its back, it has to move. The robot and the human can thus move in the interaction area during the direction-giving task, to move to a better position for pointing at the target. To find the robot and human possible positions we designed a component called the SVP (Shared

Visual Perspective) Planner, presented in [Waldhart 2019]. For the purpose of the deployment, the presented version is an adapted and slightly simplified version.

To compute the visibility of both agents, the planner has access to the geometrical representation of the environment and the agents current positions. In addition, it considers an estimated agent's maximal speed to move and a visibility threshold.

When the robot explains the route to the human and points to a landmark, they form what is called an F-formation. Kendon explains that *"An F-formation arises whenever two or more people sustain a spatial and orientational relationship in which the space between them is one to which they have equal, direct and exclusive access"* [Kendon 1990]. This F-formation has been decomposed in [McNeill 2005] into two types: the social formation and the instrumental formation. While the first type corresponds to the original definition, the instrumental formation includes a physical object that all the agents can gaze at. This means that once the robot will have moved, the human will come in front of it creating a social formation in the form of a vis-a-vis (each facing the other) and when the robot will point they will change for an instrumental formation. Indeed, when both agents will reach their position computed by the planner, we want them to be able to go from one formation to the other with only a rotation; the human will not need to move again

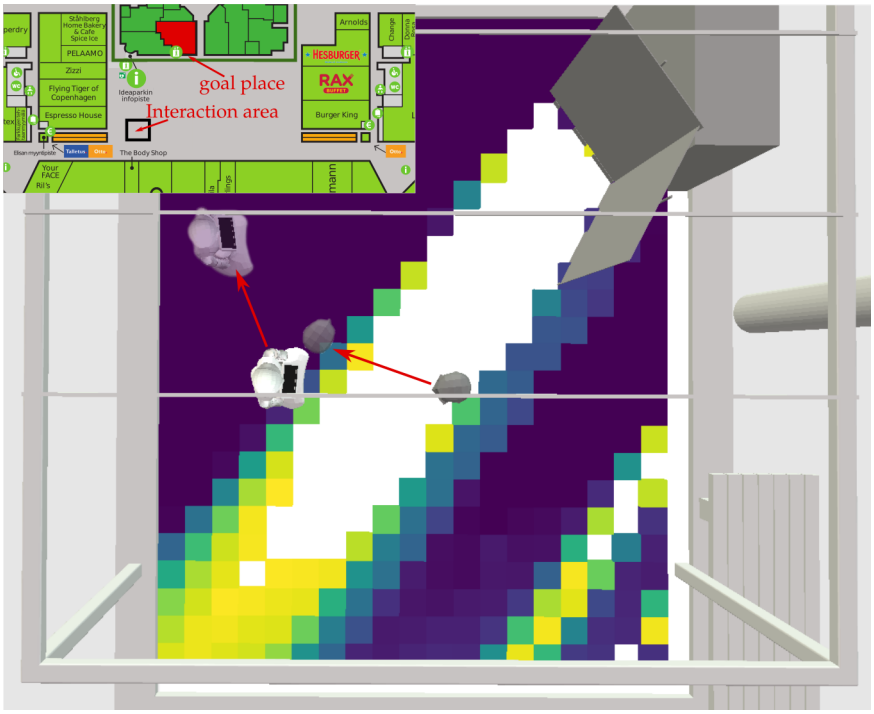

Figure 8.7: Visibility grid for a target located at the top right. The uncoloured areas represent an absence of visibility and the others represent the cost of visibility ranging from yellow for low visibility to purple for good visibility. The robot and the human in transparency on the image represent the final calculated positions while the others are the initial positions.

from their arriving position to see what the robot will point.

To search for better positions to reach in order to point a landmark, the planner takes three main parameters into account:

- Visibility constraint: The two agents can see either the target shop (when it is the only element of the route) or the passage.
- Navigation distance cost: The agents do not have to move too much.
- F-formation cost: The human-robot-target angle and a robot-human-target have to be less than 90°.

To compute the positions, the interaction area is firstly decomposed into a weighted three-dimensional (x,y for the possible positions in the area and z for the human height) grid representing the estimated human visibility of the target. The target visibility is computed offline for each position of the grid. It is based on the part that the target takes in the 360° field of view of the environment. Such grid is represented in Figure 8.7 for a given human height. The white cells are positions from which the human cannot see the pointed target. The other colored cells represent the degree of visibility from the poor in yellow to the good in purple.

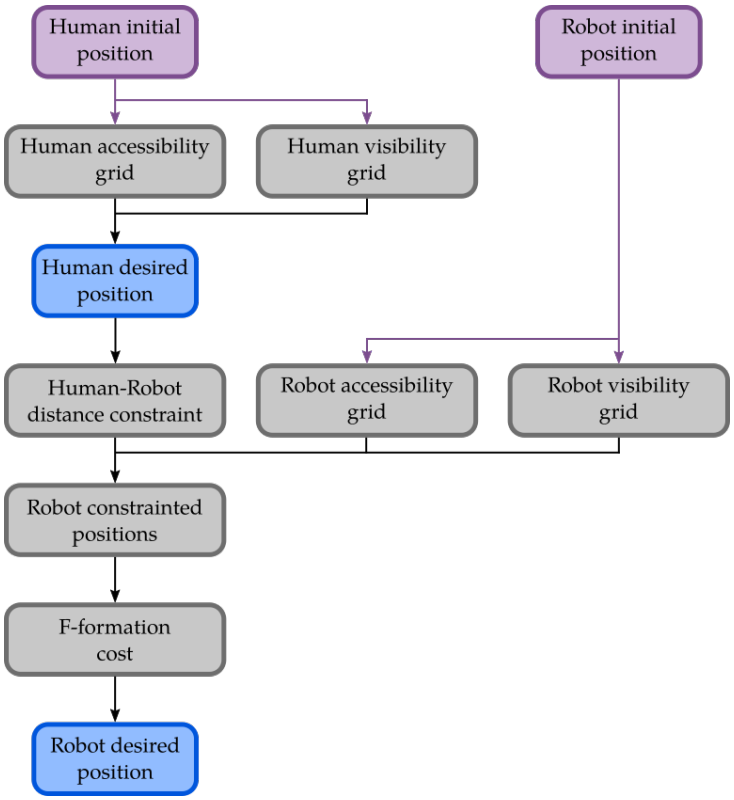

Figure 8.8: Human and robot positions computation flow. The purple blocks represent the initial positions while the blue ones represent the planned positions.

Having the human visibility grid, the goal position is computed using a weighted cost function between good visibility and restricted distance to cross. In the example of Figure 8.7, the transparent human head is the human goal position while the other is the initial position. From the initial position, the human was not able to the pointed target.

The overall computation flow is illustrated in Figure 8.8. The robot position is computed in a second time, according to the human planned position. Divided the search into two steps allows reducing the search complexity. The robot position is thus constrained by the human one. It has also to respect a minimal and maximal distance to the human and minimal visibility of the target from it. Finally, the robot position is also determined regarding a cost preferring an F-formation limiting the robot reorientation, meaning that it can point to the target keeping its torso and its chest oriented towards the human.

8.4.6 Navigating close to human

The Human-Aware Navigation component aims at moving the robot while avoiding dynamic and static obstacles in addition to proposing a socially acceptable navigation solution for the robot. For example, the robot should not pass too close to the human and should not show its back while navigating around the human. A full presentation of the planner is available in [Singamaneni 2020].

8.4.7 Executing and controlling the task

The Supervision component aims at implementing the joint action guidelines to manage the direction-giving task as a human-robot joint activity. It makes use of all the other components of the architecture to step-by-step establish the shared goal, plan by considering the human preferences, adapt to the human perspective of the scene, and monitor human commitment. The action sequencing and the incremental task-refinement process follows the steps and the main decisions exhibited in human-human studies.

First, when the Supervision receives for example a request for a restaurant from the Dialogue, it asks Ontologenius for all the existing restaurant types. This list of restaurant types is sent to the Dialogue whose role is to return to the Supervision with the type selected by the human. It then continues to obtain a more precise restaurant type from the human to finally get a restaurant name. It goes the same if the human asks for a product. The Supervision gets from Ontologenius a list of shops selling the requested item and passes it to the Dialogue. The Dialogue returns the name of the shop chosen by the person. When it directly receive as a goal a shop name, it queries the ontology to know if the given shop exists in the mall. If it does not, it can be misunderstood. To test it, the Supervision can query the Ontologenius using the fuzzy match feature. For example, when a person asks to go to “jewelsport”, the system can make the assumption that the person actually asked for “Juvesport”.

Once the shared goal is determined, the Supervision queries the Route Handler which returns a list of routes of the form $place - path - place - ... - place$. Among these routes, it selects the one with fewer elements. If it is composed of only one path, it means that the goal place can be visible from there, since it is along the same path as the robot. Otherwise, the robot will have to explain the route. Before that, it tests if the route has stairs along with it, getting the types of the elements with the ontology. If so, the supervision is in charge of ensuring the human is able to climb stairs. If he/she cannot, the supervision selects another route, without stairs if any.

The robot's role is not only to give verbal route directions but also to point to the target or the passage the person should take. We call the passage the third element of the route. It is the first element the human has to go through to start the route[2]. Pointing increases the chances to find the destination as it helps to orient oneself in space. Before pointing, the supervision requests the SVP planner for a new position for itself and the human. If the current positions do not fit the visibility and formation constraints, the supervision uses the Human-Aware navigation module to go to its new position. We expect the human to come in front of it. If he/she does not, the supervision is able to verbally inform him about movements to do (i.e. to come in front of it, to move a bit aside, etc).

Finally, once the human is at the desired position, the supervision requests the Physical Resource Manager for pointing and simultaneously explains the route to take.

In addition to the management of the direction-giving task itself, the supervision component is also in charge of the beginning of the interaction, the greeting, and its end, the goodbyes. All along with the interaction, it also ensures the human understanding of the instructions, and adapt itself if needed.

8.5 Embody architecture in a physical robot

The architecture presented in the previous chapter has been embedded in an upgraded, custom version, of the Pepper platform [Caniot 2020], which is equipped with an Intel D435 camera and an NVIDIA Jetson TX2. It has been deployed multiple times in a mall in Finland, Ideapark.

8.5.1 Pepper in Ideapark

For availability for as many customers as possible, the robot was contained in a defined place in the mall as shown in Figure 8.9. A home base was designed with the participation of all the project partners. It was a 4 per 4 meters area with a 2.5m high frame structure on it. The home base included a non-reflecting carpet on the floor and an acoustic ceiling surface on the roof.

[2]The first element of the route returned by Route Handler is their current location.

Figure 8.9: The pepper robot in its interaction area in the Finalnd mall, Ideapark.

During the first deployment in the real mall, we have updated both the Geometric Representation with actual measurements and the Semantic Spatial Representation (SSR) by making sure the regions, interfaces, corridors and intersections were represented reflecting the actual mall topology. To ensure the correctness of the instructions given by the route handler, we generated routes from the deployment location to several shops in the mall and followed them to the destination. Inaccuracies, as well as algorithmic flaws, have been fixed using this method. We also tested the interaction in the Finnish language with our native Finnish partners and corrected some mistakes in the route verbalization.

8.5.2 Pepper "in the wild"

The robot has been installed for a long-term 14 weeks deployment from September 2019 to December 2019. During this period, the robot interacted with everyday clients of the mall, who may never have had the chance to interact with a robot before. The robot was active for 3 hours per day, three days a week. In total, the robot ran the direction-giving task for approximately 96 hours "in the wild". Out of these 96 hours, it was interacting with customers for 45 hours.

Chapter 9

The Director Task: Assessing cognitive architectures

Contents

9.1 Introduction 178
9.2 From psychology to Human-Robot Interaction 180
9.2.1 The original task 180
9.2.2 The Director Task setup 182
9.2.3 The adapted task 184
9.2.4 Additional rules for the first implementation 184
9.2.5 Additional abilities 186
9.3 The cognitive architecture 187
9.3.1 Storing and reasoning on symbolic statements 188
9.3.2 Assessing the world: from geometry to symbols 188
9.3.3 Planning with symbolic facts 191
9.3.4 Managing the interaction 191
9.3.5 Speaking and understanding 192
9.4 Experiments 194
9.4.1 PR2 as the director 195
9.4.2 PR2 as the receiver 196
9.5 Open challenges for the community 198
9.5.1 Challenges to take up 199
9.5.2 User studies to perform 201

In this chapter, we propose a new psychology-inspired task, gathering perspective-taking, planning, knowledge representation with theory of mind, object manipulation, and Human-Robot communication. Along with a precise description of the task allowing its replication, we present a cognitive robot architecture able to perform it in its nominal cases. In addition, we suggest some challenges and evaluations for the Human-Robot Interaction research community, all derived from this easy-to-replicate task.

The contribution presented in this chapter is excerpted from our work, published in the proceedings of the RO-MAN 2021 conference [Sarthou 2021a]. This contribution closes this thesis and has been achieved in collaboration with other PhD

students of the HRI team. Guilhem Buisan was concerned about the task planning part. Amandine Mayima worked on the supervision component. Kathleen Belhassein has designed the presented task with us giving her psychologist point of view to create a task on which user studies could be performed. The engineer Yannick Riou worked on the motion planning component allowing us to develop a task where the robot acts on its environment. My responsibility for this task has been the integration of my previous contributions about Ontology and the REG. It has also been the opportunity to create an entire architecture extending the ones presented all along with this thesis and linked with the contributions of the team. Finally, I have contributed to the Situation Assessment component and on the Language understanding part.

The components related to my teammates will be briefly described to give an overview of the architecture. The newly introduced capabilities on which I work will be more detailed to explain the links I make between all my contributions, centred on the knowledge representation.

9.1 Introduction

Developing robotic architectures adapted to Human-Robot Interaction and thus able to carry out interactions in an acceptable way is still today a real challenge. The complexity comes, among other things, from the number of capabilities that the robot must be endowed with and therefore from the number of software components which must be integrated in a consistent manner. Such architectures should provide the robot with the capability to perceive its environment and its partners, to merge and interpret this perceptual information, to communicate about it, to plan tasks with its partner, to estimate the others' perspective and mental state, etc. Once developed, the evaluation of these architectures can be difficult because all these components are grouped into a single system. The tasks we usually want the robot to handle must highlight a maximum of abilities, while still being simple enough to be reproduced by the community. Moreover, we should be able to conduct user studies with it to validate choices regarding naive users.

Since a long term goal of the robotic field is to see robots acting in our daily life, many tasks and scenarios have been inspired by everyday activities. Even if these tasks offer a large variety of situations to be handle, since the human partner is not limited in his actions, they have the disadvantage of not highlighting some subtle abilities which are nevertheless necessary for good interaction. The robot guide task [Satake 2015b] in mall, museum, or airport, requires high communication skills to understand free queries (possibly involving chatting) and respond to them, whether to indicate a direction or to give advice. However, the perception needs can be limited due to the vast environments, as well as the perspective-taking needs due to the same perception of the environment by the robot and the human[1]. Finally, even

[1]For sure we can find some tricky cases where it could help but they do not reflect common situations.

if the human should contribute to the problem, with such a task the human partner is not necessarily an actor of the task and can just listen to the robot once their question is asked. Even if being in more constrained environments, bartender-like tasks [Petrick 2012] have the same disadvantages. Indeed, the human is considered as a customer, and as such, the interaction with the robot is limited. The robot will never ask the human to help it for performing a task and its actions do not require coordination either full collaboration.

To involve the human partner in the task and requiring him to act with the robot, assembly-like tasks [Tellex 2014][2] can be used. Nevertheless, in most cases, the human acts as an assistant rather than as a partner as full collaboration can be challenging to perform. The robot thus elaborates a plan and performs the assemble, then asks for help when detecting errors during the execution (e.g., when it cannot reach some pieces). Here the task leads to unidirectional communication. Moreover, because in such a task both the robot and the human have equivalent knowledge about the environment, it can be hard to design situations where belief divergence appears and thus perspective-taking would be required.

Scaling down an everyday task to transform it into a toy task around a table can reduce the task complexity and allow easy reproducibility. Moreover, it allows the robot and the human to work in the vicinity of each other, with smaller robots for example. With the toy version of the assembly task presented in [Brawer 2018], the human is more involved in the task. They ask the robot to take pieces and to hold them to help them assemble a chair. Even if the communication is unidirectional, we could imagine inverting the roles to test different abilities. Moreover, communication implies objects referring with the use of various visual features about the entities. Even if both agents have the same knowledge about the environment, the communication is grounded according to the current state of the world. In this task, no decision has to be made by the robot but once again, inverting the roles could open other challenges.

To focus studies around perspective-taking and belief management, the Sally and Anne scenario, coming from a psychology test, has been studied in robotic [Milliez 2014]. In this scenario, the robot is an observer of a situation where two humans come and go from a room, and move an object from a box to another. Since a human is in the room when the other is acting, a belief divergence appears between the two humans and the robot has to understand it. While the task highlights the belief management, it is first limited regarding the perspective-taking since the human presence or not could be sufficient to estimate the humans beliefs[3]. Moreover, the humans do not act with the robot since it is just an observer of the scene. In addition, no goal is formulated and the human neither interacts with one another. Finally, no communication is needed in the task. The scenario is thus focussed on the analysis of a situation.

[2]This task is not explicitly intended to be replicated by the community.

[3]When both humans are in the room they have the same perception of the scene but have different beliefs about hidden objects. Perspective-taking would be required if the humans could lean over the boxes to check what is inside.

In this chapter, we first propose a new psychology-inspired task that we think to be challenging for the Human-Robot Interaction community and rich enough to be extended: the Director Task. Inter alia, it requires perspective-taking, planning, knowledge representation with theory of mind, manipulation, communication, and decision-making. Then, we present the robotic cognitive architecture that we develop to perform the task in its nominal cases. Finally, on the basis of the presented task and what has been developed, we present a discussion about the possible future challenges and evaluations for the research community, with possible extensions of the task.

9.2 The Director Task: From psychology to Human-Robot Interaction

In this section, we present the origins of the Director Task and the needs it aims to respond to regarding other tasks from the psychology. We then detail the setup we have designed in terms of objects characteristics and organisation in the environment. We end this section with our adaptation and the required abilities we have identified.

9.2.1 The original task

The Director Task has been mainly used in psychology as a test of the Theory-of-Mind usage in referential communication. This task originates from a referential communication game from [Krauss 1977]. In this game, two participants are one in front of the other with an opaque panel between them. A speaker has to describe odd designs to a listener, either to number them for the adults or create a stack of cubes for the children. To refer to the odd figures, participants have to use images (e.g. “it looks like a plane”).

This game was then adapted by Keysar et al. [Keysar 2000] and became the Director Task. It has been used to study the influence of mutual knowledge in language comprehension. In this task, two people are placed one in front of the other but instead of an opaque panel between them, they place a vertical grid composed of different cells and objects in some of them. The **director**, a participant or in most cases an accomplice, instructs the **receiver**, a participant, about objects to move in the grid. The receiver thus follows the director’s instructions about objects to move. The particularity of the task is that some cells are hidden from the director, meaning that the receiver, being on the other side of this grid, does not have the same perspective as the director. He/she thus knows the content of more cells than the director and consequently sees more objects. When the director instructs the receiver to move an object, for a successful performance, participants must take the shared perspective of the director to move the right one. Because the configuration varies all along with the task, he/she has to update this estimated perspective all along with the interaction.

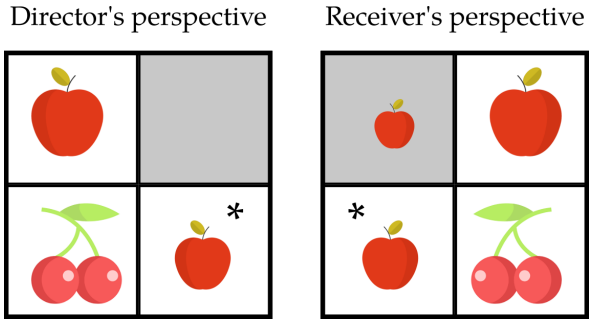

Figure 9.1: Sample display from the director's and the receiver's perspectives. The asterisk indicates the target object. Giving the sentence "the smallest apple" the receiver should find the good one even if he/she can see a smaller one in its perspective.

Taking the example of Figure 9.1, if the director instructs the receiver to take the smallest apple, the target object in its perspective is the one marked with the symbol *. However, for the receiver, in its perspective, the target object is not the smallest apple since the smallest one (called the distractor) is only visible by the participant and not by the director. The participant then must understand the director's perspective to take the target apple and not the distractor. Some studies showed that for their first attempt, participants took the smallest apple from their own point of view and only after, the target one. These results were interpreted in [Keysar 1994, Keysar 1998, Keysar 2002, Keysar 2003] as the participants understanding language in an egocentric way. Some social cognition studies used a computer-version of the Director Task [Dumontheil 2010] whose results are consistent with the ones mentioned previously, namely that participants do not use Theory-of-Mind inferences in language interpretation.

Although Theory-of-Mind and perspective-taking both require the attribution of mental states to others, some authors trend at distinguishing Theory-of-Mind tasks and perspective-taking tasks as involving distinct although related mechanisms. In [Santiesteban 2012], they consider that perspective-taking abilities were measured by the Director Task whereas Theory-of-Mind usage was investigated through another task called "strange stories" [Happé 1994]. This Theory-of-Mind task requires the attribution of mental states to a story protagonist, meaning to maintain an estimation of others' mental states. At the difference, the Director Task requires the adoption of the perspective of the director in order to follow the instructions, meaning to use this knowledge in order to execute the task properly. In this way, the authors estimated that the Director Task requires a higher degree of self-other distinction by continuously isolating our own perspective from the director one, in order to use it to act. In addition to perspective-taking abilities, the Director Task makes use of executive functions [Rubio-Fernández 2017] (i.e. vary the processing of information according to current goals in an adaptive manner) and attentional resources [Lin 2010].

To summarize, the Director Task has been used to study referential communication, language comprehension, and perspective-taking abilities. However, to our knowledge, it has never been exploited in the context of a HRI although this task presents interesting challenges for this field. More than technical challenges, it provides a way to investigate the different cognitive and behavioral processes involved in such a cooperative Human-Robot task.

9.2.2 The Director Task setup

The material used in this task has been chosen to be easily acquired and can be hand-built. It is composed of blocks, compartments, and a storage area. Each element is equipped with AR-tags allowing the robot to perceive them without advanced perception algorithms.

Figure 9.2: Part of the material used for the Director Task. Each element is equipped with AR-tags allowing their detection by the robot. Each block has four visual characteristics: a main color, a border color, a geometric figure and a figure color.

Three types of compartments exist and are illustrated on the right part of Figure 9.2. The basic ones are open on two of their opposite sides (d). They allow both the receiver and director to see the content and to reach it for manipulation. Others are open only on one of their sides (e). With such a compartment, only one of the participants can see and take what is inside. The other participant can neither know if a block is inside or not. The last compartment type (not used in the implemented version) has an open side and the opposite one equipped with a wired mesh (c). Thanks to the wire mesh, both participants can see what is inside but only one of them can take it. Thanks to these three types, we will be able to vary the awareness of the blocks (e.g., a block is known to be present but not necessarily visible), the visibility of the blocks, and their reachability (e.g., a block can be visible but not reachable). While the original Director Task uses a vertical grid, we prefer here to use several compartments to create the grid. Compartments can be stacked one on top of the other, allowing more modularity to create different situations.

While the tasks used in psychology use everyday objects, we rather choose blocks

that can easily be manipulated by robots and on which we can fix tags for their localisation and identification (a-b on Figure 9.2). The blocks have a primary color covering them all. On two opposite faces, additional visual features are drawn. The top part of these faces is dedicated to the robot's perception with a unique AR-tag on each face[4]. The bottom part is the same on both faces and is dedicated to human perception. In addition to the primary color, three visual features are available for the human to distinguish them: a colored border, a colored geometric figure (both the color and the figure can change making two features). Every visual feature (the colors and the forms) has exactly two variants. The colors are either blue or green and the figures are either a triangle or a circle. We can thus have 16 unique blocks.

The agents can use the four visual features to refer to a specific block and the complexity of the description depends on the used features. While the main color is directly related to a block, the other colors are respectively related to the border and the figure. In this way, for two blocks for which the only difference is the color of one of these elements, the said element has to be referred to in order to refer to the divergent color. A description of a block involving all its four features would be "the [color] block with the [color] border and the [color] [figure]".

The figures and colors have been chosen in such a way to allow the emergence of "coded words" between the participant to identify a block. With a bit of imagination, some could refer to the left-most block (a) through the sentence "the mountain in the sea" or the other (b) by "the puddle"[5].

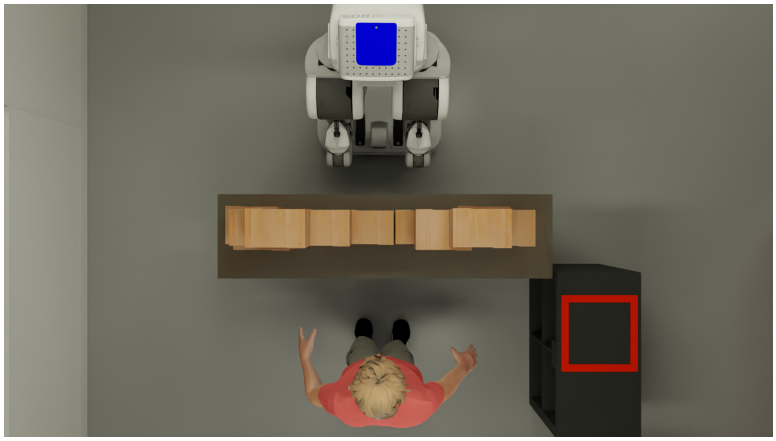

Figure 9.3: The Director Task setup with the robot and the human partner one in front of the other and a piece of furniture between them. Compartments are placed on top of the furniture and blocks are placed in the compartments. Next to the agent having the receiver role, here the human, a storage area is placed to drop the removed blocks.

Regarding the configuration, the compartments are stacked on a piece of fur-

[4]Since the tags are different on each side, the director cannot refer to them as the receiver does not see the same ones

[5]This is not generated or understood by the robot in the current version.

niture to create a kind of grid. The blocks can be put inside a compartment. As illustrated in Figure 9.3, the two agents are placed one in front of the other with the furniture and thus the compartments between them. Finally, one storage area, corresponding to the place where the receiver has to store the blocks, is delimited by a rectangle on a shelf next to the receiver. In Figure, the human would be the receiver since he/she has the storage area on his right.

9.2.3 The adapted task

Now we explained the Director Task setup and the available material, we present the rules we have adapted for HRI applications. First, the high-level goal of the task is known by both agents: to put a set of blocks away. The precise goal is given by the experimenter to the director, either the robot or the human. It corresponds to a subset of the blocks presents in the compartment that the receiver should remove from and put in the storage area. This choice, to remove the objects instead of moving them in the grid, induces changes in the situation over time. It thus requires a constant adaptation during the interaction. The goal can be given on a sheet of paper, a screen behind the receiver, or marks on the blocks on the director side. No block order is required in the formulation of the goal. The director is thus allowed to elaborate a strategy if needed.

As mentioned previously, the Director Task characteristics bring a number of interesting challenges for a collaborative robot to solve. Because this is a task with roles, one of the first challenges is to build a robotic architecture that gives the robot the ability to play both roles. Then, each role brings some specific problems to solve from a robotic point of view.

In order to enrich the task with perspective-taking, we adapted the task so that both the director and the receiver have to use perspective-taking. Since in the original task, the director knows he/she has a subset of the receiver's perspective, he/she can consider all the objects when communicating. Thus, only the receiver has to reason about the other's perspective, taking into account that some objects are not visible by the director. For HRI applications, we use the one side hidden compartments in a way to also have objects hidden from the receiver and visible by the director. Therefore, both roles have to perform perspective-taking, whether to give instructions or to understand them. On the illustration of Figure 9.4, the director (left image) can instruct the receiver to take the blue blocks as the other blue blocks in his perspective is hidden from the receiver. From the receiver point of view (right image), he/she can find the instructed block as the other blue block is hidden from the director.

9.2.4 Additional rules for the first implementation

To be able to study precise skills, such as verbal communication, perspective-taking, and adaptation, we defined a set of rules for both roles. First, the agents are not allowed to point to objects, either with their hands or gaze. They thus have to

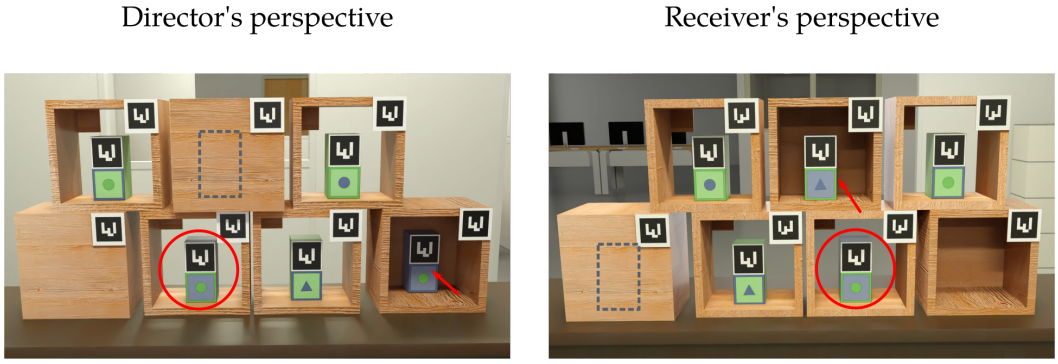

Figure 9.4: A director task setup adapted to the HRI with the director's and receiver's perspectives. For the material, each element (blocks and compartment) is equipped with AR-tags allowing their detection by the robot. Each block has four visual characteristics: a main color, a border color, a geometric figure, and a figure color. Compartments can be hidden for the director or the receiver. For the director to designate the block marked with a red circle, estimating the receiver's perspective, he/she can refer to it by its main color (blue) because he/she estimates the other blue block is not visible by the receiver. For the receiver, by taking into account the director's perspective, he/she can understand the referred block as he/she estimates the other blue block to not be visible by the director.

verbally describe the objects, focusing the task on verbal communication. However, to avoid too easy description of the kind "the fully green block", we remove the four uni-color variants[6]. In addition, to not fall into a simple referential communication task, participants are not allowed to use spatial relations in their verbal communications. They cannot, for example, say "the leftmost block" or "the block to the right of the green one". In this way, they are limited to few visual features, with high ambiguity. Since a description of a block using its four visual features can be hard for the human to process, we first expect the participants to minimize the complexity of their communication by referring to the blocks only using the features distinguishing them from other blocks. Moreover, we also expect the participant to take into account the other perspective allowing once again to minimize the complexity of the communication.

Over these elements, we can see that the task can easily be replicated and offer a controlled setup, making it a good task for human-robot user studies. Moreover, due to the number of involved processes and the number of situations that can be made, there are a lot of elements that can be analyzed and explored. Also, with the same setup, it is possible to perform human-human studies or human-robot studies which can be interesting to compare.

[6]When we said too easy it is from the human point of view, generating and understanding such description can be challenging for a robot.

9.2.5 Additional abilities

More than being an easily reproducible scenario to perform user studies on human-robot interactions in a controlled environment, the Director Task allows to demonstrate the abilities of a robotic system. We discuss here some additional abilities for which the task has been designed.

Planning When a large number of blocks have to be considered to achieve the goal, it quickly becomes complicated to communicate about some of them as the director would have to add a lot of adjectives to be able to refer to one block. Therefore when the robot is the director, it becomes interesting to integrate the communication and the task planning. Indeed, depending on the order in which the blocks are designated, the complexity of instructions, and thus their ambiguity, can decrease or increase over time. Then, the planner can provide an optimal order in which the robot has to give the instructions to the human.

Contingencies handling While performing the Director Task, errors can easily happen. Either because the director gives a wrong instruction or the receiver misinterprets the instruction and takes the wrong block. In both cases, it can be because of a wrong consideration of the other agent's perspective or simply inattention. Moreover, because some instructions might be right but hard to interpret by the receiver leading also to an error from them. Finally, errors can happen because of failures of the robotic system, as a failed action execution leading to a block to fall on the floor. A robot with a robust decision-making system will be able to analyze, try to determine their origin, and handle a number of these contingencies. For example, if the human takes the wrong block, the robot can react in different ways, either by asking the human to put it back if this block is not part of the goal, or saying nothing and re-planning if this block was among the ones to take. If errors happen repeatedly, the robot can also react differently than for a punctual error and maybe try to modify its behavior.

Communication We saw that the task requires to put a focus on communications. Communication about an object can be more or less efficient, depending on the number of characteristics given about the object or the pertinence of these characteristics. Instructing for the blue block with a circle in Figure 9.4, the geometrical figure information is not mandatory. Thus, the robot needs to be able to give proper instructions but also to understand the human ones. Moreover, in complementarity with the error management, the robot can communicate to help to solve the detected contingency. Taking a situation (with the configuration of Figure 9.4) where the human as director instructs the robot to remove the green block with a circle. This instruction matching two blocks, the robot could say that it does not find the instructed block. A preferable reaction would be to help the human to refine the instruction and say "the one with a blue circle or a green circle ?".

9.3 The cognitive architecture

In this section, we present the architecture developed to handle the Director Task in its nominal case for both roles. The architecture aims at being extending but already endows the robot with the abilities listed previously even if there are not mandatory to achieve the task. This architecture is the final version of the ones presented all along this thesis. It can also be seen as a whole new instantiation of the deliberative architecture for Human-Robot Interaction presented in [Lemaignan 2017]. The seven identified modules are represented in Figure 9.5 with their respective communication links. In the rest of this section, we detail each module and how we have refined them in terms of functionality and links to others. The modules already presented in this thesis will be briefly recalled but not detailed in-depth.

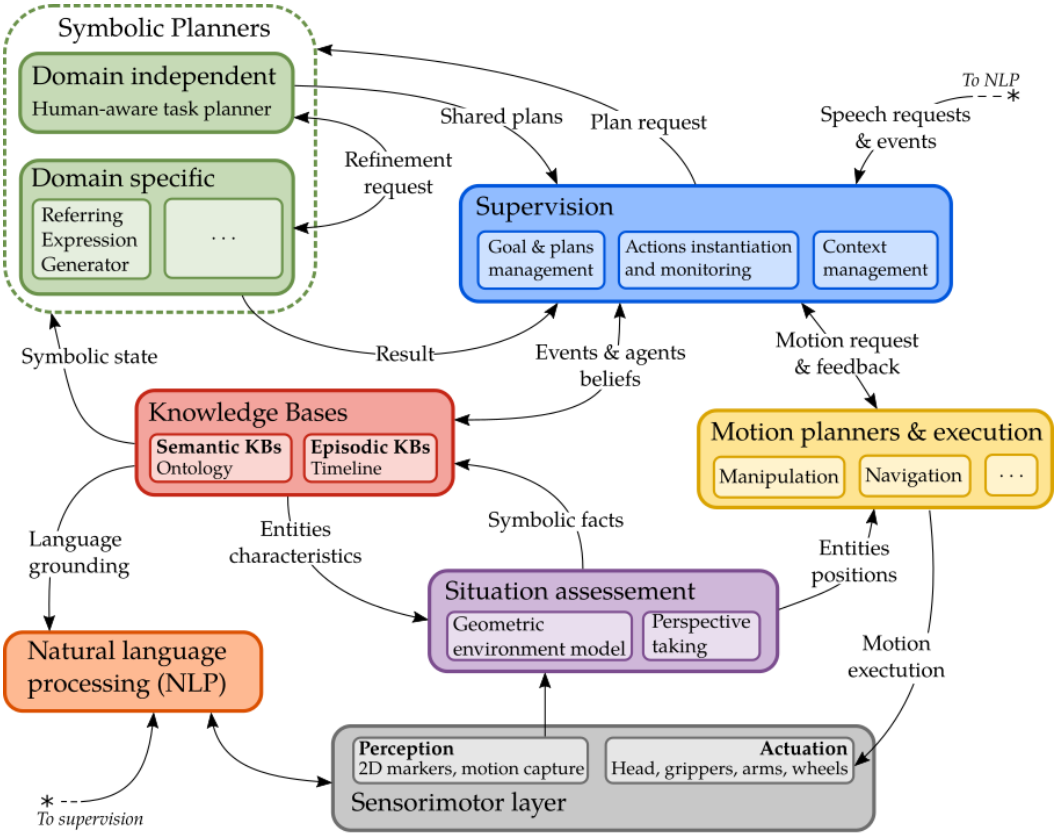

Figure 9.5: An overview of the cognitive architecture developed to handle the Director Task. Each block does not necessarily represent one software component but rather an architectural module (in terms of the features it implements). The arrows represent the type of information exchanged between the modules. This architecture extends the ones presented all along with this thesis.

9.3.1 Storing and reasoning on symbolic statements

The knowledge representation is always a core component of cognitive architectures as organising the knowledge allowing the robot to better understand the environment it evolves in. Moreover, it is on the basis of this knowledge that a robot can communicate with its human partner about the current state of the world and ground the partner's utterance regarding this world state.

Some architectures propagate knowledge all along their components [Hawes 2007], each of them enriching knowledge at each stage before providing it to the next ones. Others consider their knowledge base as an active server, activating perception processes when needed, depending on the information we are looking for [Beetz 2018]. For our architecture, we remain on the principle of a central, server-based knowledge base. It is refined into two distinct sub-modules, the semantic knowledge base and the episodic one. The semantic part is in charge of representing the environment elements: the objects' and agents' types, their applicable properties, the descriptions and parameters of the actions, a part of the language model with verbs or pronouns, and their names in natural language. This part is common to the robot's Knowledge Base and the human's estimated one. We consider it as the common ground, known a priori. Besides, we also use it to represent the current symbolic world-state (the computed facts) and thus the instantiation of the concepts in terms of physical (e.g. this particular block) or abstract (e.g. this particular action instance) entities. This part can be acquired during the interaction, through perception or communications. Among these instantiations, we have a part used for the interaction in itself, like the blocks' visual features, and others for the robot programming, like the objects' computer-aided design (CAD) models or tags ids. The episodic knowledge base aims at keeping a trace of the symbolic transitions of the world over time. It is strongly linked to the semantic knowledge base as it allows to semantically interpret these transitions.

The semantic knowledge base is still an ontology managed by the software Ontologenius. The episodic one is in the form of a timeline, managed by the software Mementar[7].

9.3.2 Assessing the world: from geometry to symbols

The role of the geometrical Situation Assessment module is first to gather different perceptual information and build an internal geometric representation of the world, composed of objects and agents. From this world representation, the module runs reasoning processes to interpret it in terms of symbolic statements between the objects themselves and between the involved agents and the objects. Doing so, the module only builds the robot's representation. However, it does not necessarily reflect what the human partner believes about the world. This is the case with the occluded compartments of the task. If a block is present in a compartment

[7] `https://github.com/sarthou/mementar`

occluded from the human perspective, this block is not visible and thus unknown to the human. Consequently, it should not exist in the human representation of the world. Here is the second role of the Situation Assessment module, estimate the human's perspective and build an estimation of their world representation. It is the first step allowing to implement the theory of mind principles [Baron-Cohen 1985].

To implement this module, we have chosen the Underworld framework [Lemaignan 2018]. Its advantage is to not be monolithic[8]. It works on the principle of a set of worlds, each working at a different granularity and providing specific features, links to create a so-called cascading structure. In the idea, it can be compared to a perception pipeline like [Beetz 2015a]. It allows easy reuse of existing modules and makes the core reasoning capabilities independent of the used perception modalities. Even if we choose to use tags for objects detection in this implementation, we could easily switch to machine learning approaches. In the same way, we could use the module with simulations or Virtual Reality systems.

The four worlds we create for the Director Task and their connexions are represented in Figure 9.6. At the top (a), we have the perception modalities. For the objects we use AR-tags [Fiala 2005]. For humans, we use a motion capture (mocap) system with helmets equipped with reflectors. For now, only the head is tracked. From each perception input, we create a dedicated world. In these worlds, we can filter the perception data depending on the used system. For the mocap, the data is clean enough. For the AR-tags we apply first a motion filter to discard data acquired when the robot moves. In addition, we apply a field of view (FOV) filter to discard data from the border of the camera because of distortions giving wrong positions even with camera calibration. To know to which object correspond a given tag unique identifiers (UID), the worlds have access to the ontology and can query it to get the UID related to. In the same principle, they can get the objects CAD model. As the output of these worlds, we ensure to have stable data with UID related to the knowledge base.

The world of the middle (b) is the robot's world representation. Information from the perception worlds is merged along with the static elements, like the building walls, and the robot model. From this world, additional perception reasoning processes are applied for the objects that are no more visible in the way of [Milliez 2014]. If an entity is no more perceived in one of the previous worlds, we first test if it should be in the robot's FOV. If so, the robot should see it. To get an explanation of this absence, we test if another entity could hide it. If not, the object is removed from the world representation. Otherwise, we keep it as we have found an explanation. Once the entities are stabilised, geometric reasoners are applied to them to extract symbolic facts. In the current version of the system, the computed facts are *isOnTopOf*, for an object on top of another with a direct contact, *isInside*, for a block in a compartment, *isVisibleBy*, assessing if an agent could see the object or not from his position, and *isReachableBy*, assessing if an object can be taken by

[8]It can however be a disadvantage in terms of performance but for research purposes, it allows more flexibility.

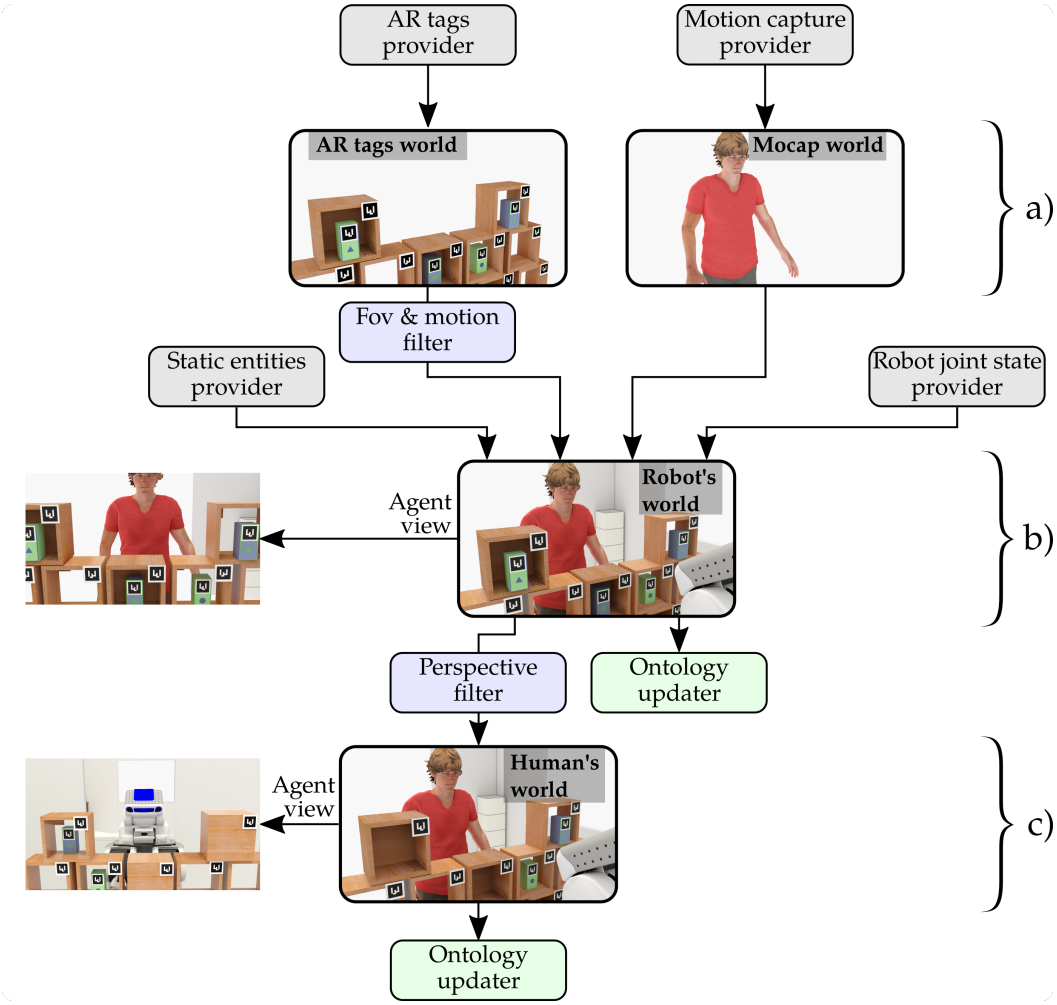

Figure 9.6: The world cascading structure of the geometrical situation assessment system. The two worlds at the top (a) are build from the perception systems and filtered. The world of the middle (b) merges the different perception information and computes symbolic facts on it. The world at the bottom (c) is the estimation of the human world representation and is computed from perspective-taking in the robot's world. Like for the world of the middle, symbolic facts are computed and sent to the semantic knowledge base.

an agent. All these facts are sent to the robot's semantic knowledge base, where reasoners will deduce further facts. For example, if a block is in a compartment, thanks to inverse property *hasInside* the fact that the compartment has the block inside is computed. In the same way, if this compartment is on top of the table, the block inside is computed to be above the table (*isAbove)* thanks to chain axiom.

While the previous world corresponds to the robot's representation, the human partner cannot have the same because of the occluded compartments. The world c)

thus aims at estimating the representation of the world from the partner's perspective. From the robot's world, we compute a segmentation image from the human point of view and use it as a filtered perception world. This allows us to instantiate the same world management process we used for the robot but this time for the human. In this way, we emulate their perception capability and geometric reasoning process. Symbolic facts are thus computed and sent to the human's semantic knowledge base. In the world of the bottom (c) on Figure 9.6, we can see that the two blocks in the occluded compartments are not present in the human world. Here we make explicit the difference between an object that is unknown and an object that is known but not visible. We could have an interaction where the human goes to see the robot side and the robot would consequently estimate the blocks in the occluded compartments as known to the human but not visible.

9.3.3 Planning with symbolic facts

The symbolic planners are divided into two categories: the domain-independent one, planning high-level tasks, and the domain-dependant one, specialized in solving precise problems. For the Director Task, the only domain-specific planner used is the Referring Expression Generator presented all along with this thesis. More precisely, we integrate the algorithm presented in Chapter 7.

Where we previously used HATP [Lallement 2014] as task planner, for this task we used its next-generation presented in [Buisan 2021]. In the same way, as HATP, the new planner aims at taking into account the human's contribution to planning how to perform a high-level task. To do so, it can generate a shared plan in which parts of the task are assigned to the human partner and others to the robot itself, depending on some criteria. However, the robot's partner is not an agent that the planner can directly control. Indeed, it must sometimes communicate about the plan to inform the human about their next actions. The new planner rather trends at emulating the human decision, action, and reaction processes to generate a shared plan. For the Director Task, emulating the human reaction to a given instruction enables the comparison between multiple blocks order, the communication of higher-level instructions to the human and the balance between multiple communication modalities.

The REG planner has been successfully integrated with the new planner allowing it to estimate the cost and the feasibility of referring communication at the task planning level. The initial world state is fetched from the ontology leading to a uniformity of the knowledge among the architecture.

9.3.4 Managing the interaction

The supervision component aims at managing the overall interaction. In this architecture, we use JAHRVIS (Joint Action-based Human-aware supeRVISor) which constitutes the decisional kernel of this cognitive architecture. Like its predecessors, SHARY [Clodic 2009] and its extensions [Fiore 2016, Devin 2016], it is designed for

a human-aware robot. It has to not only handle the robot's action execution but also to estimate the human mental state, monitoring his actions, and communicate with him. To handle these features, several processes are needed:

Interaction sessions management: It manages an interaction session that is first refined into tasks, themselves refine into action coming from the task planner. Moreover, it is in charge of the greetings happening at the beginning of an interaction, the goodbyes at the end, and all events and exchanges happening outside tasks (e.g., conversation, goal negotiation) or during a task but not related to it like a human doing a parallel task on its own.

Communication management: Communications are categorized in JAHRVIS either to: give information updating the receiver beliefs; ask a question to update the emitter belief; ask the other agent to perform an action; discuss with dialogue not related to a task or a goal/plan negotiation.

Human management: As the supervision system manages shared plans, it has to make sure the human follows them. Moreover, even if some communications are planned, it also has to make sure that the human has all the knowledge he/she needs for what he/she has to perform and if not, it hence acts or communicates through the other processes. To do so, it monitors the human beliefs about the ongoing task and plan.

Task management Even if the human has also the necessary information about the plan, contingencies can happen. The supervision can react and perform a repair thanks to action or communication.

Quality of Interaction management Even if a task is achieved, it could be done more or less efficiently and smoothly. All along an interaction session and a task, the supervision system thus estimates in real-time the Quality of Interaction (QoI) [Mayima 2020]. It measures the human engagement and the effectiveness of collaborative task performance. This information can then be used by the decision-making process to tune dynamically other processes such as the cost of properties for the REG.

9.3.5 Speaking and understanding

The Natural Language Generation is made of two parts, a static one for action verbs and communications to signify a lack of understanding and a dynamic part for the referring expressions. The content is determined by the REG and the linguistic realisation is done on the basis of concepts' labels in the ontology and a simple grammar model to know in which order the adjectives have to be sorted depending on the language.

Natural Language Understanding is more difficult due to the variety of ways the same information can be communicated. Moreover, in a given communication, we have different information. In the Director Task, we have the action to perform and the object on which the action has to be performed. First, we use the Google Speech To Text (STT) API to pass from an audio stream to a string of characters. Even if such technology is now well mastered, mistakes still appear in the transcription[9]. On the string, we perform a first analysis trying to match words and groups of words with labels of the ontology. We used sliding windows limited on the length and the fuzzy match technique available with Ontologenius. To cover a maximum of possibilities, several action verbs are described as well as synonyms for the concepts. We also tried to have a good hierarchy in the ontology types for the robot to better catch the concepts depending on the abstraction level used by the human. To refer to the blocks, some only use the terms "object" as they are the only ones involved in the task. At the end of this analysis, we have a list of concepts. Depending on the number of uncaught words (the words unknown in the ontology), we can already know if the understanding is poor or not. On the concept list, we first extract the action verb to know the instructed action (e.g. take, place, remove). The rest of the sentence is analysed thanks to the inverse grammar model for one part but also thanks to the properties ranges and domains. When we said "the red apple", we do not have any word representing the used property[10]. With the analysis of the usable properties linking color to an apple (and thus to a vegetable and so on), we are able to find the corresponding property. The result of this analysis is a SPARQL query in the same way such query is used for the NLU. Depending on the number of concepts successfully linked we can estimate the comprehension quality. The SPARQL query describing the entity to act upon is then merged with the context of the task and sent to the ontology to find the target entity. In our case, the context would be the same as for the generation meaning that we are speaking about an object being above the table of interaction.

In the case the human gives an accurate description, we should have only one match for the target entity. However, we cannot consider that the human will never do a mistake or that the robot will fully understand the instruction. In this case, we run a REG on all the ambiguous entities. The context of these generations is the SPARQL query coming from the understanding process. If we know that we are already speaking of a green block, we do not have to recall it. We fall back into Natural Language Generation and generate sentences like "do you mean the block with a circle or a triangle ?". When the human responds, we use again the SPARQL query coming from the first utterance and merge it with the newly understood.

For the Natural Language Understanding part, we could use machine learning approaches based on sequence-to-sequence (seq2seq) models like [Panchbhai 2020]. However, by doing so we duplicate the knowledge already existing in the ontology to put it in a neural network. Unless creating a standard of concept identifier, such

[9] And this, even more, depending on our English accent and the quality of the microphone used
[10] It is often the case of the attributes where relations between entities are more explicit.

model should be trained for each used knowledge base in order to be compatible with it and use the same symbols. Having different symbols would lead to failure, having more symbols in the trained model would lead to failure (queries that could not match), and having fewer symbols in the trained model would lead to a lack of understanding. Moreover, in addition, to create the ontology, we would have to create the corresponding training dataset that is a huge amount of work even if artificially augmented dataset creation techniques exist.

Even if our method can be seen as being ha-doc, we ensure uniformity of the knowledge among the architecture. Moreover, it can be easily extended and even dynamically extended during an interaction.

9.4 Experiments

The architecture has been successfully implemented on a PR2 robotic platform. The robot is thus able to play both roles, the director and the receiver. In this section, we comment and analyse a video[11] of two experiments. For both experiments, the initial state is the same and are represented in Figure 9.7. The only emulated element is the human action recognition to trigger the next actions of the robot when it holds the director role.

Figure 9.7: Initial configuration for both case studies. The top-right block is not visible, and thus unknown, by the human partner. The robot can not know if there is a block in the bottom-left compartment. All other blocks are known by both the robot and the human.

[11] https://youtu.be/jtSyZeqBkp0

9.4.1 PR2 as the director

We start this section with a PR2 in the role of the director (0:21 in the video). The setup is composed of six compartments including two compartments with a hidden face. One of these compartments is hidden from the human (the receiver) and one from the robot (the director). One block has been placed in each compartment. Consequently, only four blocks are known by both the human and the robot. Figure 9.8 is a visualization of the estimated geometric world of the human, maintained by the situation assessment component. Even if a block is present in each compartment, the leftmost one is not present in the estimation of the human's world. This absence comes from the fact that the human can not see what is in the compartment and thus can not know this block.

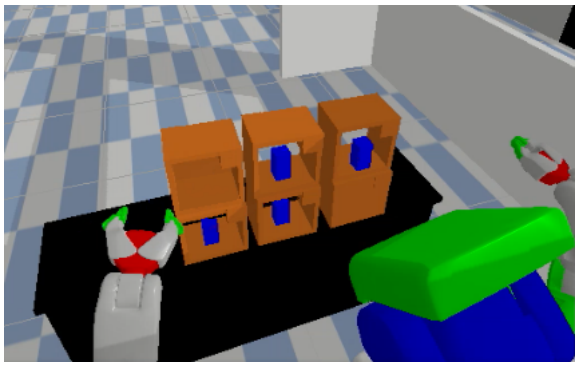

Figure 9.8: A visualization of the human's estimated geometric world from a third-person view. Even if a block is present in each compartment, the right most one is not present in this worls since the human can not see this block.

Figure 9.9 represents the entire interaction when the robot is the director. At the initial state, four blocks are visible from both agents. Describing them with all their visual features, they are:

- A blue block with a blue border and a green triangle
- A blue block with a blue border and a green circle
- A blue block with a green border and a blue triangle
- A green block with a green border and a blue circle

Thanks to the estimation of the communication cost at task planning using the results of the REG, the robot is able to find the optimal sequence of blocks to instruct. The overall communication is thus minimized and the RE is unambiguous in each situation. In the initial state (a to b), the robot asks for the green block as only one of the visible blocks is green. Since the green block has a circle on it, removing it, only one of the remaining blocks has a circle on it. The robot can thus use this feature to refer to the next block (b to c). Without communication cost

estimation during the task planning, such a simple situation would not necessarily appear.

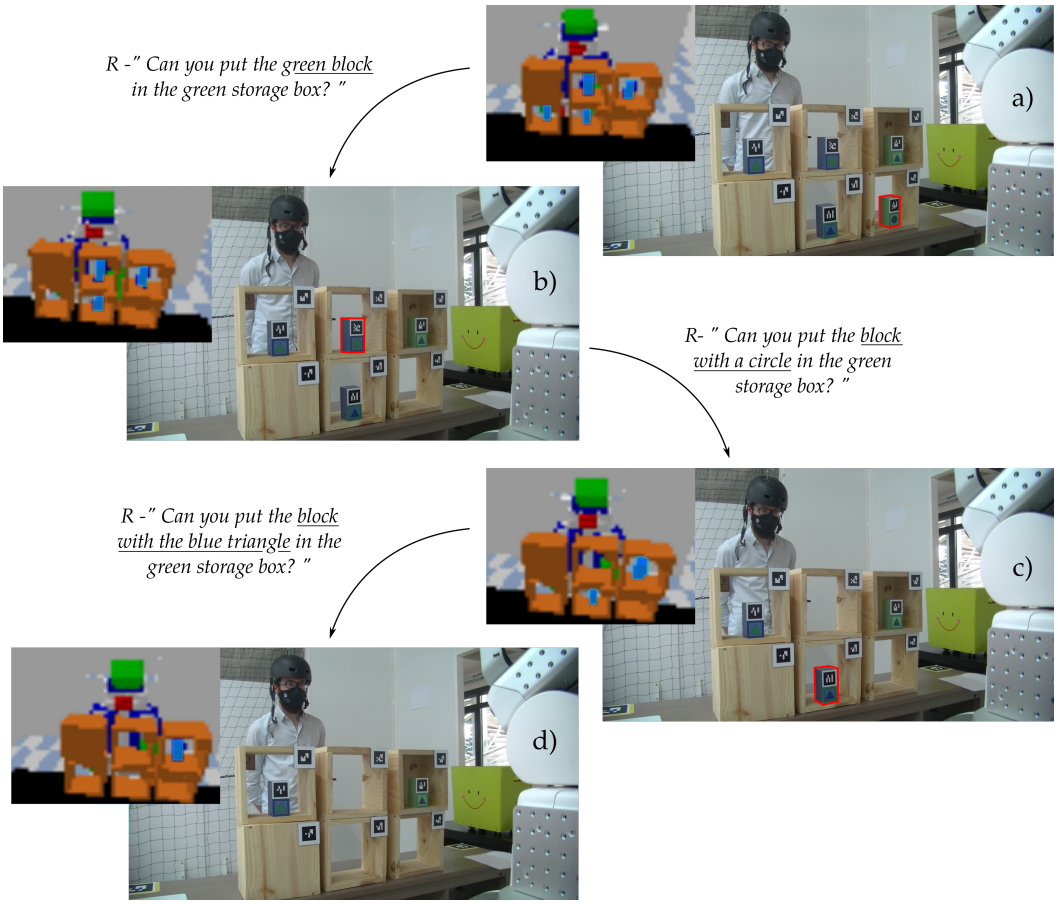

Figure 9.9: The director task handled by an autonomous PR2 robot in the role of the director. Each picture represents a step toward the achievement of the task. The estimated human perspective is displayed in the top left-hand corner of each picture. On top of the arrows leading to a new state are the sentences said by the robot to the human. The block outlined in red are the blocks referred to at each step.

9.4.2 PR2 as the receiver

While in its previous role the robot just had to instruct the human, when the robot is the receiver (1:33 in the video) more reasoning is needed. A retranscription of part of the interaction is represented in Figure 9.10. In the initial state, the same four blocks as previously are visible by both the agents. The robot is able to understand three actions: take, drop, and remove. The latter action is a combination of the two others.

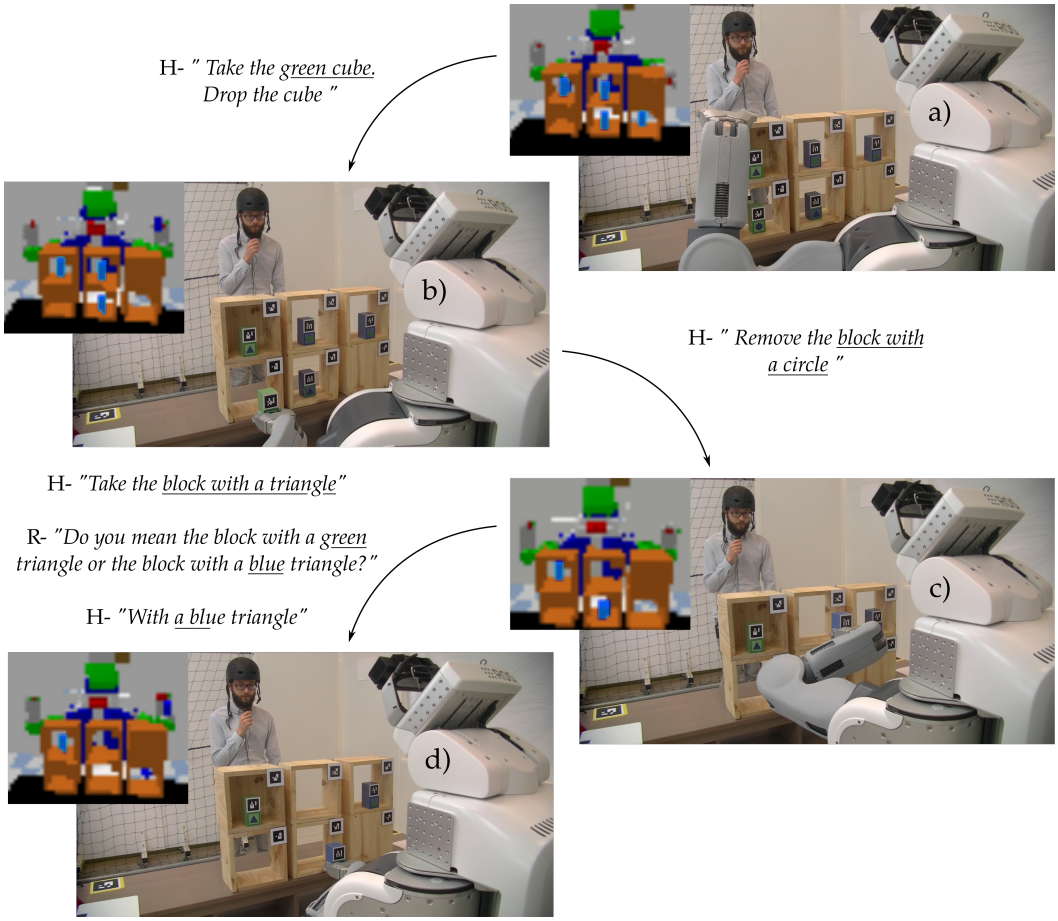

Figure 9.10: The director task is handled by an autonomous PR2 robot in the role of the receiver. Each picture represents a step toward the achievement of the task. The estimated human perspective is displayed in the top left-hand corner of each picture. On top of the arrows leading to a new state are the sentences said by the human to the robot and for the last situation the refinement query from the robot to the human, followed by the answer of the human.

For the first block (a to b on the figure), the human instructs the robot for the green block. The natural language understanding module returns the SPARQL query:

(?0, isA, Block), (?0, hasColor, green)

Since the robot assumes the human to speak about objects on the table, the understood query is merged with another one representing the context of the task: (?0, isAbove, table_1). Querying the human estimated ontology with the merged query, only one entity matches. There is no ambiguity in human instruction. The robot takes the instructed block then drop it. If the query was applied to the robot

ontology, two blocks would have matched since the block unknown by the human is also green. It goes the same for, the second instruction. There is no ambiguity. The SPARQL query related to this second block is:

(?0, isA, Block), (?0, hasFigure, ?1), (?1, isA, Circle)

The third instruction given by the human as the director is the most interesting for us. The human asks for *"The block with a triangle"*. However, the speech to text returns *"take is about to whip a triangle"*. With this sentence, the NLU module can only extract two known concepts being "take" and "triangle". Due to the limited amount of words understood, it does not try to generate a SPARQL query. The robot thus informs the human about its incapacity and repeat the heard sentence as a back loop for the human. At the second try, the sentence is understood and gives the query:

(?0, isA, Block), (?0, hasFigure, ?1), (?1, isA, Triangle)

However, matching this query to the human's estimated ontology, we get two results. Once again, matching it to the robot's ontology would give three results but the third one is not visible from the human. Since all the concepts of the sentence have been understood and linked together to create the query, the human should have made a mistake, providing an ambiguous referring expression.

To be proactive, we want the robot to ask precision about the block to take by proposing visual features to distinguish them. To do so, we use the REG algorithm on each ambiguous block. As a context for the REG, we pass the previously merged SPARQL query. It represents what has already been understood by the robot. In the current situation, the robot thus performs two REG and their results are used to generate the disambiguation sentence:

"Do you mean the block with a green triangle or the block with a blue triangle?"

When the human responds, for sure it does not generate a complete description f the block to be taken. It rather answers the question. The query extracted from his answer is thus combined with the previously understood one in case some information is missing. Matching this last query to the human's estimated ontology, the robot finally get the block to remove.

With this latter case, we saw how the robot can react to a human's mistake and use the REG to help the progress of the task, even if it is the receiver.

9.5 Open challenges for the community

So far, we have described the main abilities a robot has to be endowed with to perform the Director Task. Then, we have proposed a cognitive robot architecture handling the Director Task in its simplest form, both for the director and receiver

roles. However, we have only tackled the regular cases that the task offers. In this section, we now present some open challenges that we have identified around the task. In addition, since we see that the environment of the task can be controlled, we also propose some user studies to investigate the ways of sharing information.

9.5.1 Challenges to take up

The components or abilities related to each challenge are reported in the following table. The list of challenges is not exhaustive. Moreover, even if some challenges have already been mentioned among the presentation of the components, they are here reported as requiring finer and more generic management.

Challenged abilities / components	Challenges
Perspective-taking	1
Communication	4, 6, 7
Task planning	2, 3, 4
Reference generation	4, 5, 8
Contingencies handling	1, 2, 3, 4

1. **Finer contingency analysis:** In this task, failures can easily arise due to the high ambiguity between the blocks and the difference of perspective. Such failures have to be handled by the robot and to do so their origin has to be understood to react to them in an appropriate way. In the case the human, as the receiver, does not take the instructed block, the failure can have different origins. First, it could come from a perspective not taken into account. However, this lack of perspective-taking can be assigned either to the director or the receiver. Another origin can be a description not clear enough or correct but too complex. Finally, it can just be an error of inattention. Each of these origins has to be handled in a different way.

2. **Handling contingencies as errors:** When the receiver takes another block than the one instructed, has to fix the error through communication and negotiation. First, the wrong block has to be put back in its original compartment. Then, the robot has to adapt its original instruction to make it clearer and improve the chances to have the receiver taking the right one.

3. **Not handling contingencies as errors:** When the receiver takes the wrong block, even if it is the instructed one, it can however be part of the goal. In this case, the robot not necessarily has to repair the plan, asking the human to put it back as no order is required for the task. It can thus re-plan or re-instruct the human for the same block without further information. It may also mention to the receiver for the mistake and explain that it does not matter because this one is also part of the goal. Rather than re-planning, the robot could use a conditional plan, anticipating possible confusions, and adapt according to the human's actions.

4. **Adapting to recurrent failures:** In case of recurrent failures by the partner or degradation of the Quality of interaction with a number of latencies, the robot could try to analyse the origin of the problems and determine if a common point exists. If so, it can adapt itself to increase the QoI and reduce the failures. For example, if the partner is found to have difficulties with certain visual features, the robot can react through properties' cost adaptation. If the partner still consider the removed blocks, it can react through communication context adaptation.

5. **Allowing spatial references:** As explained in section in the origins of the task, the Director Task is originally a task to test referential communication. Even if the present version asks the participants to not use spatial reference, this rule could be relaxed to study perspective-corrected spatial Referring Expression Generation.

6. **Understanding the human instructions:** In the current implemented version, the robot can only understand a limited vocabulary that is restricted to the context of the task. In this way, the robot only understands descriptions of blocks. In a more natural interaction, humans could use a richer vocabulary, give a single instruction in multiple steps, or have communications not directly linked to the task. During tests for designing the task, it was common to have instructions like "take the block with a ... triangle. No, rather the one with a green border". Such complex communications where the director corrects his explanations should have to be managed by the robot.

7. **Introducing code words:** As presented through the design of the used material, the visual features on the blocks have been chosen in a way to allow the visualisation of landscapes on them, with a little imagination. Considering multiple tasks with the same robot and human, alternating the roles if needed, the introduction of coded words could be interesting to reduce the communication complexity and thus the overall efficiency. The robot could thus try to negotiate some coded words. Once introduced, it would also have to remember them and understand them as being part of a description.

8. **Communicating about multiple blocks:** With the currently implemented system, the director only instructs one block at a time. It can either be through a reference matching all of them, like "Take all the blocks with a triangle on them", or multiple descriptions in a raw. The latter method could bring different kinds of communications such as "I do not remember the instruction for the last block" when the human is the receiver. For the first method, when the robot is the receiver, it would also be a different kind of instructions to interpret.

9.5.2 User studies to perform

Some robot behaviours, mainly about the referring expression generation, have been designed with regard to the current literature. However, the Director Task could be used to refine them thanks to user studies. More than providing a controlled task and environment, this task has the advantage to hide the real goal of the study. From the participant point of view, the goal is to remove blocks from compartments. The goal of the study can be focused on other aspects and could help the community in the design of architectures applied to more realistic scenarios.

Currently, the references to the blocks are made in such a way as to minimize the number of visual features used while staying discriminative. Such implementation fit Grice's Maxim of Quantity [Grice 1975]. However, due to all the cognitive mechanisms to use in this task (e.g., perspective-taking) and the high ambiguity among the blocks, evaluating such behaviour compared to a full explanation could be interesting. Indeed, giving a reference with more information than needed would ensure to not match blocks being only visible by the receiver, which could help them to select the right block. In a way, it could allow to not use perspective-taking at the cost of complex communications.

During the material presentation, we have introduced a special compartment equipped with a wire mesh. Because a block in such a compartment is visible from the receiver but not accessible, referring to a block matching also this one could disturb the receiver. We could expect such a situation to require a higher cognitive load to determine the right block to take. Such behavior could also be interesting to evaluate as even if the human receiver is able to take the right block it could also decrease the Quality of Interaction. In the same way, a block previously visible by the receiver and that the director moves in a hidden compartment could disturb the receiver to interpret a description.

Conclusion: represent, store, explore, communicate

In this thesis, we presented several contributions around the use of ontology as a way to represent knowledge for the robot as well as to represent an estimation of the robot's partners knowledge. An ontology is a knowledge graph with on top a formal and explicit specification of the shared meaning of the concepts used in it. In robotic, the use of such a formal and explicit representation provides a unification of the knowledge among an entire architecture. The knowledge is no more ubiquitous among the components, each owning the part it needs without a global consensus about who owned the truth (if any). Instead, the knowledge becomes a shared resource taking advantage of each component's inputs. This notion of shared knowledge is also important in multi-robot systems. Using an ontology allows all the robots to communicate using the same vocabulary. Consequently, it can facilitate their interaction, even if they do not rely on the same architecture.

Extending multi-robot applications, we reach multi-agent applications and consequently Human Robot Interaction (HRI). The knowledge an ontology represents is the knowledge of how we, as humans, perceive our environment and how we interpret it. For sure an ontology is machine-understandable but especially, at the base, human understandable, at the difference of neural networks for example. Thanks to all these characteristics and because it comes from the human cognition, ontology can be suitable to represent an estimation of the human knowledge. In addition, trying to align it with the human knowledge, a robot could use it to communicate with a human, that it is to interpret a communication act or to produce one.

In Chapter 2, we presented Ontologenius, an open-source and lightweight software to maintain knowledge graphs using ontology. It aims to be the semantic memory of the robot. This software had been developed for HRI application with the ability to maintain several Knowledge Bases at the time, one representing the robot's knowledge and the others being the estimation of the robot's partners knowledge. In addition, regarding the management of several instances at the time, Ontologenious comes with a deep-copy feature to catch the state of a KB at a given moment and to modify it freely. To represent several knowledge states of the same agent and to avoid the creation of too many instances that could slow down the CPU, with Ontologenius we propose a kind of versioning system, keeping a trace of the changes. Regarding knowledge retrieval, with Ontologenius we choose to provide a set of precise queries working at the semantic level but allowing the exploration of the structure of the knowledge rather than simply the relations between entities.

On the basis of Ontologenius, we have presented several contributions. In Chapter 3, we proposed a way to describe the topology of indoor environments using an ontology. The resulting representation was called the Semantic Spatial Represen-

tation (SSR). In the context of a route description task and using the SSR, we presented a combination of two algorithms able to find several routes leading to a destination. Using the same representation, we presented a third algorithm to generate the route explanation, in the form of a sentence, by respecting the three good practices identified by Allen in [Allen 2000]. The way the explanation is generated allows the guided human to perform an imaginary tour of the environment, increasing the chances to reach the requested destination.

Continuing on the knowledge exploitation and more precisely on spatial communication, from Chapter 4 to Chapter 7, we presented contributions around the Referring Expression Generation (REG) task. As explained in [Reiter 2000], it is the concern of "how we produce a description of an entity that enables the hearer to identify that entity in a given context". While this task has been studied for decades, none of the existing methods has attempted to use an ontology as KB. In addition, because their KB are dedicated to the task, it is only composed of relations usable to communicate with a human. However, considering a shared KB among the architecture, some knowledge can have a purely technical use for the robot. More importantly, none of the existing works really considered the notion of context. Their only goal was to designate an entity in a given state of an environment. However, in HRI use, the REG is just an action of a wider task in which a context exists. It could be already known information about the entity to refer to or implicit restriction about the entities in concern. In Chapter 4, we thus presented a new algorithm managing all the previously evoked issues and using an ontology as KB. In addition, through comparisons with other algorithms, we showed that our contribution is, to the date, the most efficient to our knowledge, solving most of the problems in less than a millisecond. Finally, as a proof of usability, the algorithm has been integrated into a robotic architecture with a KB continuously updated through perception.

Taking advantage of the high performance of our REG algorithm, in Chapter 5, we proposed a method integrating it with a task planner. The goal of this method was to endow the task planner with the ability to estimate the feasibility and the cost of communication. Such estimations thus allow to avoid deadlock at execution and to find plans minimizing the overall communication complexity. In this contribution, the communication was limited to the reference to entities but could be extended to other communication contents. The used task planner was HATP, a human-aware task planner. This means that the planner is able to plan for the robot and its partner, estimating their future mental state and their abilities in order to assigned tasks to one or another. To estimate the future mental states, HATP has a dedicated internal representation, limited to the entities used in the task to plan. However, to work, our REG algorithm needs an ontology as KB and needs to take into account all the entities of the environment. To make it works, we have thus presented a scheme allowing the planner to update an ontology, representing the future estimated knowledge of the human partner, in order to be able to run the algorithm. This contribution fully takes advantage of the features brought by Ontologenius, that is the ability to maintain an instance per agent, and

to copy an instance at a given moment to freely modify it. The resulting method was implemented into a robotic architecture as a proof of concept.

With the integration of the REG algorithm with a task planner, we highlight the fact that the act to refer to an entity, appears most of the time in the context of a task. During this task, agents act with and manipulate the entities of the environment. On this basis, in Chapter 6, we proposed to use this additional knowledge about the activities of the agents with the entities of the environment as a new piece of information, usable to refer to them. Continuing to use HATP as task planner, we had first proposed a way to represent the task planning domain into an ontology, as well as the execution of this plan, that we called the Hierarchical Execution Trace (HET). We had then presented modifications to our original REG algorithm, allowing it to use the past activities representation to generate a new kind of Referring Expression (RE).

Even if the adaptation of the REG algorithm of Chapter 6 brings new possibilities, in Chapter 7 we shown that it had some limitations. The major one is the apriori knowledge it needs about the representation. It is thus restricted to a unique representation of past activities. However, we saw in the literature that many representations of activities exist, each created for particular applications. In Chapter 7, we thus explored common ontology patterns used to represent such complex relations being in the form of n-ary relations. Adding to them a set of simple parametric patterns as labels, we created what we called Compound Relation (CR). With this new type of relation, we had adapted our original REG algorithm to provide a more generic way to generate RE. The resulting algorithm had been shown to still work with a description of past activities but also any representation using n-ary relation. In addition, we have shown that the performances of the new algorithm are better than the algorithm of Chapter 6 and equal to the ones of the original algorithm when no CR is used in the representation.

Through the chapters 8 to 9, we presented two robotic architectures involving the contributions presented all along this thesis. In Chapter 8, we presented the MultiModal Mall Entertainment Robot (MuMMER) project and the resulting architecture. The goal of the MuMMER project was to develop a robotic architecture allowing a robot to guide a customer in a mall and to chat with them. By guiding, we do not mean accompanying the customer but rather to explain the route to follow. To be able to find the route to explain and generate the explanation sentence, we used the contribution of Chapter 3. To maintain the representation of the environment and share it with other components, we used Ontologenius, presented in Chapter 2. The resulting architecture, embodied in a Pepper, has run for 14 weeks in a mall in Finland.

Finally, in Chapter 9, we presented an integration of the REG algorithm into a complete robotic architecture dedicated to HRI applications. In addition, with this new architecture, we showed how we had put the knowledge at the center of the architecture, used by all the components. Moreover, in this chapter we introduced a new task, inspired by psychology experiments, to assess cognitive architecture for HRI. Taking this task as a basis and with regard to the implemented architecture,

we ended this chapter with a set of challenges we think to be interesting for the community.

Future work

To end this thesis, we introduce some potential future work, focussing on two main topics studied during these three years: the knowledge need for HRI and the REG.

Knowledge need for HRI

In the introduction of this thesis, we saw the main memory system that we estimate a human to own. In this thesis, we focused on semantic memory, to endow the robot with the meaning about the elements of its environment. While we managed the procedural memory which could be assimilated to the task planning domain, we have not managed the episodic memory, even if we have started to draw some of its aspects through the description of past activities. The ability of remembering is however a key aspect for HRI. It allows a robot to speak about it as story-telling, as we saw to refer to entities, to learn acting policies, or to learn others preferences. One of the actual most advanced systems managing such knowledge is today the openEASE system [Beetz 2015b]. However, how could it be extended to HRI application? How could we represent the estimate of the other's experiences? How should we make the difference between a real experience and a past situation others tell us they have experienced?

Finally, for me, the more complex and still open question, would be: how to make a clear distinction between the semantic system and the episodic one? However, asking this question, leads to another being: do we need a clear distinction? On one hand, they are two different types of knowledge but on the other hand, one is nothing without the other. We need meaning to understand the memories (in the sense of past experiences) meaning that we need a link between them. Wanting to create two distinct systems, in my opinion, we arrive at two solutions, which do not seem to me to be suitable. First, we could keep in the semantic system a trace of the experiences with their meanings (e.g. cut_7 is a cutting action) while the episodic system would only temporally order them (e.g. cut_7 holds between t1 and t2). The second solution would be to do not keep any trace of the experience in the semantic system but would require to keep a part of the meaning in the episodic one (e.g. cut_7, being a cutting action, holds between t1 and t2).

Referring Expression Generation

In the last year of this thesis, we gain an interest toward the Referring Expression Generation (REG) problem. This problem is interesting as it requires a rich representation of an environment, a fine exploration of it, the verbal communication of the inherent knowledge, the consideration of the problem into a task, and the consideration of the partner. Even if we proposed four contributions around this

topic, we think that number of challenges has still to be investigated. In the continuation of our works, we could first use the REG algorithm based on past activities at task planning as we did for the original version. In this way, the robot could plan communication-based on future past activities.

A second study track will be about spatial relations like left and right. During this thesis, we tried to void such relations, even if our algorithm could manage them. The reason is that such relations depends on the perspective and thus on the reference frame, meaning the frame in which the relations are computed. For example, we could say "the pen at your left" where the reference frame is the hearer. We could also say "my car is the one at the left of the red car" where the reference frame is the red car. However, saying "look the squirrel at the right of the tree" is more tricky. At the difference of a car, a tree does not have any orientation. You and the hearer being at the same place and having the same perspective, this sentence does not lead to any ambiguity as you and the hearer could be the reference frame. However, if you are one on each side of the tree, the used reference frame is unclear, and the resulting RE is ambiguous. With these examples, we see that both the perspective and the characteristics of the involved entities have to be used. Some works start to study it [Kelleher 2006, dos Santos Silva 2015]. However, they are mainly based on Incremental Algorithm which is not adapted and the used of an ontology could help at generalizing the process.

Finally, another interesting study track would be the reference to sets. An approach supporting this feature would support two new kinds of RE. First, we could say "give me the screws that are in the red bin" or "give me the red pens" where we ask for a set of entities. Nevertheless, we saw that the REG is a kind of recursive process in the way that to refer to a given entity we may need to generate a RE to another entity. With RE supporting references to sets, we could thus generate sentences like "give me the bottle which is next to the sheets of paper". Here the target entity is not a set but it owns relation toward a set. Where the few existing methods own the set representation into the Knowledge Base in a static way [Fang 2013], it would be more interesting to compute them dynamically depending on the current situation and the target entity. An algorithm and the underlying knowledge representation to support such a feature could be highly challenging, and even more if we managed to consider spatial references at the same time.

Appendix A

Route description supplementary material

A.1 Routes verbalization patterns

```
void Sentences::createEnd()
{
  end_.push_back({end_side,
    {{",and ", ",then ", ". ", ". After that ", ". Finally, "},
    {"you will "}, {"see ", "find "},
    {"/X "}, {"on "}, {"your ", "the "}, {"/D "},
    {"side ", "when you walk ", ""},
    {", straight after /Y ", ""}} });

  end_.push_back({end_side,
    {{",and /X is ", ",then ", ",/X is then ", ". After that, "},
    {"on "}, {"the ", "your "},
    {"/D "}, {"side ", "when you walk"},
    {", straight after /Y ", ""}} });

  end_.push_back({end_side,
    {{",then ", ". After that, ", ". ", ",and ", ". Finally, ",
    ". From there on, "},
    {"/X will be on your /D "},
    {"side ", "when you walk", ""},
    {", straight after /Y ", ""}} });

  end_.push_back({end_side,
    {{",then ", ". After that, ", ",and ", ". Finally, ",
    ". From there on, "},
    {"/X is on the /D there "},
    {", straight after /Y ", ""}} });

  end_.push_back({end_side,
    {{",then ", ". After that, ", ",and ", ". Finally, "},
    {"it is on the /D, you will see /X "},
    {", straight after /Y ", ""}} });

  end_.push_back({end_here,
    {{",then ", ". After that, ", ",and ", ". Finally, "},
    {"you see there /X "},
    {", straight after /Y ", ""}} });

  end_.push_back({end_here,
```

```
    {{",then ", ". After that, ", ",and ", ". Finally, "},
    {"you will find /X there "}} });

  end_.push_back({end_here,
    {{",then ", ". After that, ", ",and ", ". Finally, ",
    ". From there on, "},
    {"/X is "}, {"there ", ""},
    {"on the /DY side of /Y "}} });

  end_.push_back({end_in_front,
    {{",then ", ". After that, ", ",and ", ". Finally, ",
    ". From there on, "},
    {"you will "}, {"find ", "see "},
    {"/X right away "}} });
}

void Sentences::createEndBegin()
{
  end_begin_.push_back({end_side,
    {{"you will "}, {"see ", "find "},
    {"it ", "/X "}, {"on "}, {"your ", "the "}, {"/D "},
    {"side ", "when you walk ", ""},
    {", straight after /Y, ", ""}} });

  end_begin_.push_back({end_side,
    {{"it will be on your /D "},
    {"side ", "when you walk", ""},
    {", straight after /Y, ", ""}} });

  end_begin_.push_back({end_side,
    {{"it's on the /D there "},
    {", straight after /Y, ", ""}} });

  end_begin_.push_back({end_side,
    {{"on the /D you will see /X "},
    {", straight after /Y, ", ""}} });

  end_begin_.push_back({end_here,
    {{"you see there /X "}} });

  end_begin_.push_back({end_here,
    {{"you will find it there "}} });

  end_begin_.push_back({end_here,
    {{"it's "}, {"there ", ""},
    {"on the /DY side of /Y "}} });

  end_begin_.push_back({end_in_front,
    {{"you will "}, {"find ", "see "},
    {"it ", "/X "}, {"right away "}} });
}

void Sentences::createBegin()
{
```

```
  begin_.push_back({start_corridor,
    {{"just ", ""},
    {"go ", "walk ", "go straight "},
    {"across ", "through ", "on ", "down "},
    {"that ", "this "}, {"corridor "}} });

  begin_.push_back({start_corridor,
    {{"go to this "}, {"corridor "}} });

  begin_.push_back({start_end_of_corridor,
    {{"/X "}, {"is "}, {"just straight ", ""},
    {"down "},
    {"this ", "that "}, {"corridor "}} });

  begin_.push_back({start_end_of_corridor,
    {{"just ", ""}, {"walk ", "go almost "},
    {"until the end of "}, {"this ", "that "},
    {"corridor "}} });

  begin_.push_back({start_end_of_corridor,
    {{"/X is at the end of "},
    {"this ", "that "}, {"corridor "}} });

  begin_.push_back({start_interface,
    {{"/I "}} });
}

void Sentences::createDuring()
{
  during_.push_back({during_end_of_corridor,
    {{"walk to ", "go almost at ", "walk until "},
    {"the "}, {"very ", ""}, {"end of the corridor "}} });

  during_.push_back({during_turn_continu_corridor,
    {{"turn /D "}, {"at ", "straight after "},
    {"/Y "}} });

  during_.push_back({during_turn_continu_corridor,
    {{"turn /D "}} });

  during_.push_back({during_turn,
    {{"turn /D "}} });

  during_.push_back({during_turn,
    {{"turn /D "}, {"at ", "straight after "},
    {"/Y "}} });

  during_.push_back({during_reference,
    {{"you will "}, {"see ", "find "}, {"/Y "}} });

  during_.push_back({during_interface_font,
    {{"/I "}, {"just in front ", "right away "}} });

  during_.push_back({during_interface_side,
```

```
    {{"/I "}, {"at "}, {"the ", "your "}, {"/D "},
    {"side ", ""}} });
}
```

A.2 Routes search solutions

Table A.1: Performance analysis and solutions characteristics for the route search algorithm based on the Semantic Spatial Description. The algorithm has been run on 26 places being on distinct paths and each being on a unique path. The table presents the resolution time, the number of regions used, the number of routes found, the number of paths explored, and the length of the longest route. Resolution times include the ROS communications to query the ontology.

	Place name	**Resolution time(ms)**	**Used regions**	**Found routes**	**Expl. paths**	**Max. length**
1	Espresso_House	0.434841	1	1	1	1
2	Brother_Clothing	1.289182	1	1	2	2
3	only_2	1.251221	1	1	2	2
4	Gant	1.369534	1	1	2	2
5	Donna_Rosa	2.751091	1	1	3	3
6	River_and_Co	2.940316	1	1	3	3
7	Jesper_Junior	2.674137	1	2	6	3
8	Guess	2.984623	1	1	3	3
9	gf_atm_east	2.953073	1	1	3	3
10	Kauneus	3.344798	1	1	3	3
11	HairStore	3.110017	1	1	3	3
12	gf_toilet_west	2.640851	1	1	3	3
13	Kalastus_Suomi	3.622685	1	1	4	4
14	Juvesport	3.021612	1	1	4	4
15	Cafe_de_Lisa	12.298473	2	3	11	4
16	Ballot	13.682847	2	3	12	5
17	Hairlekiini	13.668867	2	3	12	5
18	Pancho_Villa	46.16124	2	11	32	5
19	LahiTapiola_desk	33.773911	2	16	94	8
20	Ti_Ti_Nallen	33.954918	2	11	63	8
21	Masku	42.927055	2	11	70	9
22	Beefking	30.531876	2	14	68	9
23	ff_toilet_west	31.247139	2	12	65	9
24	ff_toilet_east	37.755016	2	11	71	9
25	Sticky_Wingers	35.685139	2	11	57	10
26	Coyote_Grill	33.838421	2	11	63	11

Appendix B

Referring Expression Generation suplementary material

B.1 Referring Expression Generation solutions

Table B.1: Performance analysis and solutions characteristics for the UCS-based REG algorithm on the three-room apartment ontology. The algorithm has been run on the 77 entities inheriting the upper class "Object". Resolution times do not include the ROS communications to query the ontology.

	Entity name	Resolution time (ms)	Result size	Number of entity
1	ball_l_0	0.067859	2	1
2	ball_O_0	0.059308	2	1
3	bottle_l_0	0.139627	4	2
4	bottle_l_1	0.068305	2	1
5	box_l_0	0.193782	2	1
6	box_l_1	0.217469	2	1
7	box_l_2	1.361858	4	2
8	box_l_3	1.1367	4	2
9	box_l_4	0.617972	3	2
10	box_l_5	0.564576	3	2
11	box_O_5	0.619296	4	2
12	can_l_0	0.014065	1	1
13	chair_l_0	0.226579	2	1
14	chair_O_0	0.149172	2	1
15	chair_O_1	0.317905	3	1
16	coat_hanger_l_0	0.066575	2	1
17	couch_l_0	0.013508	1	1
18	cup_O_0	0.764456	3	2
19	cup_O_2	1.064564	4	2
20	cup_O_3	3.034582	4	2
21	cup_O_4	2.105825	4	2
22	cup_O_5	1.018651	3	2

Table B.1 continued from previous page

	Entity name	Resolution time (ms)	Result size	Number of entity
23	drawer_O_0	0.013237	1	1
24	fan_O_0	0.014185	1	1
25	food_container_1_0	1.322064	2	1
26	lamp_O_0	0.685341	5	2
27	lamp_O_1	0.185361	2	1
28	lamp_O_2	0.343764	3	2
29	room_B_bed_0	0.013346	1	1
30	room_B_book_0	6.405807	6	3
31	room_B_book_1	5.771545	4	2
32	room_B_book_10	1.668816	2	1
33	room_B_book_11	0.869473	2	1
34	room_B_book_12	3.559328	5	2
35	room_B_book_13	2.979439	5	2
36	room_B_book_2	1.035305	3	1
37	room_B_book_3	4.698687	4	2
38	room_B_book_4	3.922041	5	2
39	room_B_book_5	3.558096	5	2
40	room_B_book_6	0.799378	2	1
41	room_B_book_7	3.029112	5	2
42	room_B_book_8	0.641222	2	1
43	room_B_book_9	0.758528	2	1
44	room_B_box_0	0.18311	2	1
45	room_B_box_1	0.361131	2	1
46	room_B_box_2	0.407418	3	1
47	room_B_chair_0	0.546668	4	2
48	room_B_chair_1	0.474871	3	2
49	room_B_chair_2	0.435847	3	1
50	room_B_coat_hanger_0	0.073927	2	1
51	room_B_cube_0	0.058384	2	1
52	room_B_cube_1	0.060926	2	1
53	room_B_cup_0	1.1184	4	2
54	room_B_cup_1	1.030775	4	2
55	room_B_cup_2	2.454008	5	2
56	room_B_cup_3	3.096267	5	2
57	room_B_cup_4	2.55846	4	2
58	room_B_cup_5	0.5239	2	1
59	room_B_cup_6	1.952396	3	2
60	room_B_lamp_0	0.305946	3	2
61	room_B_lamp_1	0.111063	2	1
62	room_B_shelf_0	0.485909	2	1
63	room_B_shelf_1	2.316131	3	2

Table B.1 continued from previous page

	Entity name	Resolution time (ms)	Result size	Number of entity
64	room_B_shelf_2	0.220443	2	1
65	room_B_shelf_3	2.001844	4	2
66	room_B_table_0	0.245021	2	1
67	room_l_book_0	0.423708	2	1
68	shelf_l_0	1.733897	4	2
69	shelf_l_1	0.416069	3	2
70	shelf_O_0	0.42869	3	2
71	shelf_O_1	0.478794	2	1
72	table_l_0	0.516371	3	1
73	table_l_1	0.425897	3	2
74	table_O_0	0.478838	3	1
75	table_O_1	0.941871	3	1
76	TV_O_0	0.132627	5	3
77	TV_O_1	0.121472	3	2

B.2 Compare REG with other communication means

Listing B.1: The HATP solution for the third case study of Chapter 5. The robot chooses to point instead of verbalizing to designate the cubes 5 and 7. Please note the order of cube motions is not considered in this problem. The lines beginning with H represent the actions of the human and the lines beginning with HR represent actions involving the human and the robot (communication actions). In green are the REG results for each communication action even if a pointing has been choose.

```
HR - TellHumanToTake(C3) // (C3, isA, Cube), (C3, hasDigit, int:1)
                         // (C3, isIn, area_red), (area_red, isA, Area),
                         // (area_red, hasColor, red)
H  - Take(C3)
HR - TellHumanToPack(C3, area_black)   // (area_black, isA, Area),
                                       // (area_black, hasColor, black)
H  - Pack(C3, area_black)


HR - TellHumanToTake(C4) // (C4, isA, Cube), (C4, hasColor, black)
                         // (C4, isIn, area_red), (area_red, isA, Area),
                         // (area_red, hasColor, red)
H  - Take(C4)
HR - TellHumanToPack(C4, area_white)   // (area_black, isA, Area),
                                       // (area_black, hasColor, white)
H  - Pack(C4, area_white)

HR - PointHumanToTake(C5) // (C5, isA, Cube), (C5, hasDigit, int:2)
                          // (C5, hasColor, black)
                          // (C5, isIn, area_black), (area_black, isA, Area),
                          // (area_black, hasColor, black)
H  - Take(C5)
HR - TellHumanToPack(C5, area_white)   // (area_black, isA, Area),
                                       // (area_black, hasColor, white)
```

```
H  - Pack(C5, area_white)

HR - TellHumanToTake(C6) // (C6, isA, Cube), (C6, hasDigit, int:2)
                         // (C6, hasColor, red)
H  - Take(C6)
HR - TellHumanToPack(C6, area_white)   // (area_black, isA, Area),
                                       // (area_black, hasColor, white)
H  - Pack(C6, area_white)

HR - PointHumanToTake(C7) // (C7, isA, Cube), (C7, hasDigit, int:2)
                          // (C7, hasColor, green)
                          // (C7, isIn, area_black), (area_black, isA, Area),
                          // (area_black, hasColor, black)
H  - Take(C7)
HR - TellHumanToPack(C7, area_white)   // (area_black, isA, Area),
                                       // (area_black, hasColor, white)
H  - Pack(C7, area_white)

HR - TellHumanToTake(C9) // (C9, isA, Cube), (C9, hasColor, black)
                         // (C9, isIn, area_black), (area_black, isA, Area),
                         // (area_black, hasColor, black)
H  - Take(C9)
HR - TellHumanToPack(C9, area_red)     // (area_red, isA, Area),
                                       // (area_red, hasColor, red)
H  - Pack(C9, area_red)

HR - PointHumanToTake(C11) // (C11, isA, Cube), (C11, hasDigit, int:1)
                           // (C11, hasColor, red)
H  - Take(C11)
HR - TellHumanToPack(C11, area_black)   // (area_black, isA, Area),
                                        // (area_black, hasColor, black)
H  - Pack(C11, area_black)
```

B.3 Referring Expression Generation comparisons

Table B.2: Performance comparison between the original REG version, the task-based extension, and the compound relation version. The three algorithms have been run on the three-room apartment ontology over the 77 entities inheriting the upper class "Object". Resolution times do not include the ROS communications to query the ontology.

	Entity name	**Original:** **Resolution time (ms)**	**Task based:** **Resolution time (ms)**	**CR based:** **Resolution time (ms)**
1	ball_l_0	0.067859	0.175575	0.120015
2	ball_O_0	0.059308	0.141727	0.112483
3	bottle_l_0	0.139627	1.119924	0.591754
4	bottle_l_1	0.068305	0.131283	0.099736
5	box_l_0	0.193782	0.476698	0.374592
6	box_l_1	0.217469	0.508015	0.394767
7	box_l_2	1.361858	9.136083	7.154199

Table B.2 continued from previous page

	Entity name	Original: Resolution time (ms)	Task based: Resolution time (ms)	CR based: Resolution time (ms)
8	box_l_3	1.1367	9.653615	4.984035
9	box_l_4	0.617972	4.470673	2.772374
10	box_l_5	0.564576	1.144243	1.514405
11	box_O_5	0.619296	7.610336	7.494356
12	can_l_0	0.014065	0.013077	0.016982
13	chair_l_0	0.226579	0.586629	0.427859
14	chair_O_0	0.149172	0.509738	0.26145
15	chair_O_1	0.317905	1.442332	1.399296
16	coat_hanger_l_0	0.066575	0.178833	0.11939
17	couch_l_0	0.013508	0.013362	0.017672
18	cup_O_0	0.764456	4.2213	2.777132
19	cup_O_2	1.064564	14.022109	8.645581
20	cup_O_3	3.034582	15.155928	18.904949
21	cup_O_4	2.105825	5.927248	5.447168
22	cup_O_5	1.018651	3.575444	1.951084
23	drawer_O_0	0.013237	0.01316	0.016902
24	fan_O_0	0.014185	0.012952	0.017147
25	food_container_l_0	1.322064	2.312981	1.363851
26	lamp_O_0	0.685341	7.014069	3.129251
27	lamp_O_1	0.185361	0.401585	0.207447
28	lamp_O_2	0.343764	0.735108	0.596541
29	room_B_bed_0	0.013346	0.013202	0.017394
30	room_B_book_0	6.405807	70.577606	45.716713
31	room_B_book_1	5.771545	28.865629	20.767796
32	room_B_book_10	1.668816	5.911679	1.63053
33	room_B_book_11	0.869473	0.730637	1.323878
34	room_B_book_12	3.559328	18.389738	13.825704
35	room_B_book_13	2.979439	15.685247	11.037728
36	room_B_book_2	1.035305	5.968753	4.244997
37	room_B_book_3	4.698687	18.002239	12.628333
38	room_B_book_4	3.922041	20.141655	13.855048
39	room_B_book_5	3.558096	16.105089	13.210673
40	room_B_book_6	0.799378	0.694669	1.301659
41	room_B_book_7	3.029112	15.880693	11.307087
42	room_B_book_8	0.641222	0.637827	1.203315
43	room_B_book_9	0.758528	0.685999	1.284837
44	room_B_box_0	0.18311	0.409545	0.578893
45	room_B_box_1	0.361131	0.780212	0.621853
46	room_B_box_2	0.407418	0.940093	0.730397

Table B.2 continued from previous page

	Entity name	Original: Resolution time (ms)	Task based: Resolution time (ms)	CR based: Resolution time (ms)
47	room_B_chair_0	0.546668	3.404545	1.802689
48	room_B_chair_1	0.474871	3.391011	2.316866
49	room_B_chair_2	0.435847	0.82642	1.607748
50	room_B_coat_hanger_0	0.073927	0.154667	0.138173
51	room_B_cube_0	0.058384	0.131024	0.104184
52	room_B_cube_1	0.060926	0.127503	0.104204
53	room_B_cup_0	1.1184	7.298525	4.937895
54	room_B_cup_1	1.030775	5.893492	5.554281
55	room_B_cup_2	2.454008	18.066069	14.860561
56	room_B_cup_3	3.096267	20.715389	15.100975
57	room_B_cup_4	2.55846	5.101856	7.619192
58	room_B_cup_5	0.5239	1.015293	0.932277
59	room_B_cup_6	1.952396	6.329725	6.954328
60	room_B_lamp_0	0.305946	0.772256	1.257307
61	room_B_lamp_1	0.111063	0.464758	0.220344
62	room_B_shelf_0	0.485909	0.941829	0.438766
63	room_B_shelf_1	2.316131	5.849635	6.427223
64	room_B_shelf_2	0.220443	0.446526	0.390474
65	room_B_shelf_3	2.001844	5.099002	4.94336
66	room_B_table_0	0.245021	0.409597	0.628512
67	room_l_book_0	0.423708	0.953376	0.618058
68	shelf_l_0	1.733897	7.198431	7.687705
69	shelf_l_1	0.416069	1.538461	1.598254
70	shelf_O_0	0.42869	4.22667	3.003134
71	shelf_O_1	0.478794	3.277662	0.685916
72	table_l_0	0.516371	2.270701	0.638314
73	table_l_1	0.425897	0.802616	0.658631
74	table_O_0	0.478838	2.374376	0.952758
75	table_O_1	0.941871	1.945553	1.087856
76	TV_O_0	0.132627	2.745053	1.733654
77	TV_O_1	0.121472	1.012234	0.513965

Appendix C

Résumé en Français

Nous fournissons ici un résumé en langue française des travaux présentés dans ce manuscrit de thèse.

Résumé : L'arrivée des robots dans notre vie quotidienne fait émerger le besoin pour ces systèmes d'avoir accès à une représentation poussée des connaissances et des capacités de raisonnements associées. Ainsi, les robots doivent pouvoir comprendre les éléments qui composent l'environnement dans lequel ils évoluent. De plus, la présence d'humains dans ces environnements et donc la nécessité d'interagir avec eux amènent des exigences supplémentaires. Ainsi, les connaissances ne sont plus utilisées par le robot dans le seul but d'agir physiquement sur son environnement mais aussi dans un but de communication et de partage d'information avec les humains. La connaissance ne doit plus être uniquement compréhensible par le robot lui-même mais doit aussi pouvoir être exprimée et partagée.

Dans la première partie de cette thèse, nous présentons Ontologenius, le logiciel développé durant ces trois ans de thèse. C'est un logiciel permettant de maintenir des bases de connaissances sous forme d'ontologies, de raisonner dessus et de les gérer dynamiquement. Nous commençons par expliquer en quoi ce logiciel est adapté aux applications d'interaction humain-robot (HRI), notamment avec la possibilité de représenter la base de connaissances du robot mais aussi une estimation des bases de connaissances des partenaires humains ce qui permet d'implémenter les mécanismes de théorie de l'esprit, telle que la prise de perspective. Nous poursuivons avec une présentation de ses interfaces. Cette partie se termine par une analyse des performances du système ainsi développé.

Dans une seconde partie, cette thèse présente notre contribution à deux problèmes d'exploration des connaissances : l'un ayant trait au référencement spatial et l'autre à l'utilisation de connaissances sémantiques. Nous commençons par une tâche de description d'itinéraires pour laquelle nous proposons une ontologie permettant de décrire la topologie d'environnements intérieurs et deux algorithmes de recherche d'itinéraires. Nous poursuivons avec une tâche de génération d'expression de référence. Cette tâche vise à sélectionner l'ensemble optimal d'informations à communiquer afin de permettre à un auditeur d'identifier l'entité référencée dans un contexte donné. Ce dernier algorithme est ensuite affiné pour y ajouter les informations sur les activités passées provenant d'une action conjointe entre un robot et un humain, afin de générer des expressions encore plus pertinentes. Il est également intégré à un planificateur de tâches symbolique pour estimer la faisabilité et le coût des futures communications.

Cette thèse se termine par la présentation de deux architectures cognitives, la

première utilisant notre contribution concernant la description d'itinéraire et la seconde utilisant nos contributions autour de la Génération d'Expression de Référence. Les deux utilisent Ontologenius pour gérer la base de connaissances sémantique. À travers ces deux architectures, nous présentons comment nos travaux ont amené la base de connaissances a prendre un rôle central, fournissant des connaissances à tous les composants du système.

Introduction

Dans cette thèse, nous présentons plusieurs contributions autour de l'utilisation d'ontologies comme moyen de représenter les connaissances pour le robot ainsi que pour représenter une estimation des connaissances des partenaires du robot. Une ontologie est un graphe de connaissances avec en plus une spécification formelle et explicite de la signification partagée des concepts qui y sont utilisés. En robotique, l'utilisation d'une telle représentation formelle et explicite permet une unification des connaissances au sein d'une architecture entière. La connaissance n'est plus omniprésente parmi les composants, chacun possédant la partie dont il a besoin sans cohérence globale. Avec une ontologie, la connaissance devient une ressource partagée. Cette notion de connaissance partagée est également importante lorsqu'on parle de systèmes multi-robots. L'utilisation d'une ontologie permet à tous les robots de communiquer en utilisant le même vocabulaire. Par conséquent, il peut faciliter leur interaction, même s'ils ne reposent pas sur la même architecture.

En étendant les applications multi-robots, nous atteignons les applications multi-agents et par conséquent les applications pour l'interaction humain-robot. La connaissance qu'une ontologie représente est la connaissance de la façon dont nous, en tant qu'êtres humains, nous percevons notre environnement et comment nous l'interprétons. Certes une ontologie est compréhensible par la machine mais surtout, à la base, compréhensible par l'homme, à la différence des réseaux de neurones par exemple. Grâce à toutes ces caractéristiques et parce qu'elle vient de la connaissance humaine, l'ontologie peut convenir pour représenter une estimation de la connaissance humaine pour un robot. De plus, en essayant de l'aligner sur le savoir humain, de la même manière que plusieurs robots peuvent utiliser le vocabulaire qu'une ontologie contient pour communiquer, un robot pourrait l'utiliser pour communiquer avec un humain, que ce soit pour interpréter un acte de communication ou pour en produire un.

Ontologenius : une mémoire sémantique pour l'interaction Humain-Robot

Dans ce chapitre, nous présentons Ontologenius, un logiciel open source et léger pour maintenir un graphe de connaissances à l'aide d'une ontologie. Ce logiciel a été développé pour des applications d'Interaction Humain-Robot avec la possibilité de maintenir plusieurs bases de connaissances à la fois, l'une représentant les con-

naissances du robot et les autres étant l'estimation des connaissances des partenaires du robot. De plus, concernant la gestion de plusieurs instances à la fois, Ontologenious est livré avec une fonctionnalité de copie profonde pour récupérer l'état d'une base de connaissances à un moment donné, puis pour la modifier librement. Pour représenter plusieurs états de connaissance d'un même agent et éviter la création de trop d'instances qui pourraient ralentir le CPU, avec Ontologenius nous avons proposé une sorte de système de versionnage, gardant une trace des changements. Concernant la récupération de connaissances, avec Ontologenius nous avons choisis de fournir un ensemble de requêtes précises travaillant au niveau sémantique mais permettant l'exploration de la structure de la connaissance plutôt que simplement les relations entre entités.

Conception et fonctionnalités

Dans cette section, nous expliquons d'abord notre choix d'utiliser l'ontologie comme moyen de représentation des connaissances, tant du point de vue de son expressivité que de son utilisation croissante en robotique. Ensuite, nous présentons les fonctionnalités souhaitées et le niveau d'expressivité que nous avons sélectionné pour ce logiciel.

Pourquoi une ontologie ?

En psychologie cognitive, la mémoire sémantique fait référence à la connaissance encyclopédique des mots associée à leur signification. Après plusieurs études sur les temps de réponse des participants aux questions, certains auteurs ont proposé un modèle de cette mémoire sémantique comme étant un réseau sémantique [Collins 1969, Collins 1970]. Avec ce modèle, ils posent l'hypothèse que les connaissances sont organisées de manière hiérarchique, respectant un principe d'inclusion entre les classes. Par exemple, une classe représentant le concept de chat hériterait d'une classe supérieure, représentant le concept d'animal. De plus, les instances de ces classes seraient liées aux autres par des propriétés, et de la même manière que pour les classes, une notion de hiérarchie sur les propriétés existerait. Une telle structure des connaissances chez l'homme permettrait une économie cognitive ainsi qu'un stockage efficace de ces connaissances. Même s'ils n'ont pas été les premiers à formaliser le principe d'un réseau sémantique, Collins et Quillian ont fourni des travaux et des implémentations informatiques de premier plan.

Comme indiqué dans [Prasad 2020], un tel réseau sémantique, également appelé graphe sémantique ou graphe de connaissances, est aujourd'hui fréquemment utilisé comme représentation de connaissances dans les applications robotiques pour représenter entre autres :

- les catégories d'entités à différents niveaux d'abstraction [Bálint-Benczédi 2018], par ex. une poignée est un objet physique

- les caractéristiques des entités [Tenorth 2017], par ex. un réfrigérateur a une poignée
- la fonction ou le but des entités [Paulius 2019], par ex. la poignée du réfrigérateur permet d'ouvrir le réfrigérateur
- l'emplacement d'une entité par rapport à une autre (c'est-à-dire des relations spatiales) [Singh 2020], par ex. la bouteille de lait est au frigo

C'est sur la base de ces connaissances que nous pouvons ensuite créer des programmes pour fournir des capacités cognitives aux robots. Voici un point important, cette connaissance doit supporter les composants de l'architecture mais n'en fait pas forcément partie. C'est ce qu'on appelle le paradigme de la programmation robotique basée sur la connaissance [Beetz 2012], où la connaissance est séparée du programme. Il permet d'avoir plusieurs sous-systèmes d'une même architecture robotique, chacun apportant une capacité cognitive spécifique mais partageant tous les mêmes connaissances. Le choix de partager des connaissances communes entraîne des défis importants nécessitant d'abord de se mettre d'accord sur le contenu de la connaissance et ensuite de le fournir d'une manière compréhensible par la machine. Considérant l'utilisation d'un réseau sémantique comme représentation des connaissances, l'utilisation de l'ontologie est destinée à répondre à ces défis.

La première définition d'une ontologie a été proposée par Guber dans [Guber 1993] comme suit : *"une ontologie est une spécification explicite d'une conceptualisation"*. Plus tard, Borst [Borst 1999], a introduit deux nouveaux concepts définissant une ontologie comme une *"spécification formelle d'une conceptualisation partagée"*. Les nouveaux concepts sont les notions de « formel » et de « partagé ». En fusionnant ces deux définitions, Studer et al. [Studer 1998] sont parvenus à la définition : *"Une ontologie est une spécification formelle et explicite d'une conceptualisation partagée"*. Avec cette dernière définition, on commence à voir qu'une ontologie peut être un outil puissant pour créer une base de connaissance commune à toute une architecture robotique et utilisée par des sous-systèmes fournissant des capacités cognitives spécifiques.

Même si toutes les définitions précédentes tendent à affiner ce qu'est une ontologie, elles reposent toutes sur la notion commune de conceptualisation pour laquelle aucune définition formelle n'est fournie. Une ancienne définition, présentée dans [Genesereth 1987], était :

> *« Un corps de connaissances formellement représenté est basé sur une conceptualisation : les objets, concepts et autres entités qui sont supposés exister dans un domaine d'intérêt et les relations qui existent entre eux. Une conceptualisation est une vue abstraite et simplifiée du monde que nous souhaitons représenter dans un but précis. Chaque base de connaissances, système à base de connaissances ou agent de niveau de connaissance est engagé dans une certaine conceptualisation, explicitement ou implicitement."*

Cependant, comme l'explique Guarino dans [Guarino 2009], une telle définition d'une conceptualisation ne correspond ni à notre intuition ni à notre besoin d'une ontologie. En effet, nous ne voulons pas que la conceptualisation dépende de la situation actuelle. Il doit plutôt être ce qui est commun à toute situation, permettant de les représenter de manière uniforme. Comme expliqué dans [Guarino 1995], lorsque le monde est modifié, la conceptualisation doit rester la même. On s'intéresse donc au sens des concepts utilisés pour décrire un état du monde. Cependant, la définition finale d'une conceptualisation fournie par Guarino va difficilement dans ce sens malgré son objectif initial. Pour la simplicité du propos actuel, la conceptualisation devrait être le sens de tout concept permettant de représenter une situation ou un état du monde donné. Par conséquent, une ontologie pourrait être définie comme :

une spécification formelle et explicite du sens partagé de tout concept permettant de représenter une situation donnée.

En d'autres termes, une ontologie vise à contraindre l'interprétation d'un vocabulaire utilisé. C'est donc une théorie logique. Cependant, en regardant la littérature, nous remarquons que la définition d'une ontologie est encore aujourd'hui floue. En effet, même si elle représente le sens des concepts utilisables, nous souhaitons l'appliquer à la base de connaissance représentant un état du monde donné, même s'il évolue dans le temps. Là où un réseau sémantique représenterait cette instanciation, car la signification de ses arêtes pourrait être définie par l'utilisation d'une ontologie, dans cette thèse, nous utiliserons le terme ontologie pour désigner **un réseau sémantique possédant une restriction sur l'interprétation du vocabulaire utilisé**. Le but de cette thèse n'étant pas de modéliser ces règles de restriction mais plutôt de les utiliser, cela ne doit pas prêter à confusion.

Considérant l'étude de l'ontologie en robotique, et plus généralement en informatique, on peut distinguer trois types d'apports :

- création d'ontologie,
- stockage et inférence à base d'ontologies,
- système utilisant une ontologie

La création d'ontologies se concentre sur la définition du vocabulaire et des règles. Les ontologies créées sont souvent divisées en quatre catégories : niveau supérieur, référence, domaine et application. Les ontologies de niveau supérieur définissent des concepts largement applicables, transversaux à plusieurs disciplines. Les plus utilisées sont DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [Masolo 2003], Cyc [Lenat 1989], ou SUMO (Suggested Upper Merged Ontology) [Niles 2001]. Les ontologies de référence sont basées sur une ontologie supérieure et décrivent le vocabulaire d'une discipline, comme l'ingénierie ou la médecine. Par exemple, [Schlenoff 2015] couvre la robotique et l'automatisation, décrivant le robot d'un point de vue technique, avec ses capteurs, ses processus et sa

pose. Les ontologies de domaine visent à affiner une discipline, en se concentrant sur un domaine plus restreint, comme l'ontologie SOMA [Beßler 2020] se concentrant sur la représentation des activités. Enfin, les ontologies d'applications étendent les ontologies de domaine pour des applications précises. La conception de l'ontologie utilise des langages tels que RDF (Resources Description Framework), FOL (First Order Logic) ou OWL (Web Ontology Language). Chaque langage est livré avec des caractéristiques telles que le formalisme ou la calculabilité.

À l'aide de ces ontologies, de nombreux frameworks ont été développés pour prendre en charge les processus de raisonnement de haut niveau dans les applications robotiques. Parmi elles, on trouve des ontologies utilisées pour la prise de décision, l'évaluation de situation, la planification ou le maintien de croyances. Un examen complet de ces capacités de raisonnement et des systèmes correspondants est disponible dans [Olivares-Alarcos 2019]. Parmi les systèmes revus, les emblématiques sont KnowRob [Tenorth 2013], ROSETTA [Stenmark 2013], CARESSES [Bruno 2017], RoboBrain [Saxena 2014], ou encore ORO [Lemaignan 2010].

Le dernier type de contribution est le stockage et l'inférence à base d'ontologies pour former une base de connaissance sur laquelle le framework peut s'appuyer pour effectuer un raisonnement de haut niveau. KnowRob utilise Prolog [Wielemaker 2003] avec une bibliothèque pour gérer les triplets RDF. ROSETTA utilise un triple store Sésame [Broekstra 2002]. ORO utilise le système de stockage de triplets Jena en plus du raisonneur Pellet [Sirin 2007]. Enfin, RoboBrain utilise une base de données de graphiques personnalisée. Nous pouvons voir qu'aucune solution standard n'a émergé car elle dépend principalement des besoins de l'application en termes de capacité de requête, de prise en charge de l'expressivité ou de support technique et de compatibilité. Comme d'autres outils, nous pouvons trouver en C++ owlcpp [Levin 2011] basé sur Raptor et Fact++ [Tsarkov 2006], ou en Python RDFlib, OWLReady2 ou AllegroGraph.

En résumé, une base de connaissances sous forme d'ontologie, peut être vue comme un graphe sémantique reposant sur une spécification formelle et explicite du sens partagé des concepts utilisés pour représenter une situation donnée. Afin d'être utilisée dans un système pour effectuer un raisonnement de haut niveau, l'ontologie doit être stockée pour exposer les connaissances qu'elle contient à d'autres composants. Aucune solution de stockage standard n'a vu le jour jusqu'à présent car elle dépend principalement des besoins de l'application. Par conséquent, dans la partie suivante de cette section, nous discutons de nos besoins en nous concentrant sur les applications d'interaction humain-robot et en nous inspirant des solutions existantes, nous définissons les fonctionnalités que nous souhaitons pour notre système de stockage.

Caractéristiques souhaitées

Travailler en serveur : L'architecture robotique que nous visons est composée de plusieurs modules, capables de communiquer. En plus de l'architecture, nous

considérons un système de supervision, étant une sorte de “chef d’orchestre” qui appelle chaque composant en cas de besoin. Un problème commun avec une telle architecture est que chaque composant possède une partie des connaissances générales qui peuvent conduire à des incohérences. Par conséquent, la base de connaissances que nous voulons doit être un serveur, accessible par requête et mise à jour par n’importe quel composant. Il fournit ainsi une connaissance uniforme au sein de l’architecture. Pour créer un serveur, nous pourrions utiliser n’importe quel outil existant et lui attacher une couche de communication. C’est par exemple le cas d’ORO, qui fonctionne comme un serveur, basé sur le triple store Jena et fournissant une interface Telnet pour envoyer les requêtes et les mises à jour.

Supporter plusieurs instances : Puisque nous ciblons l’application HRI, nous devons être en mesure de représenter une estimation des connaissances des partenaires humains. Là où certains pourraient utiliser une seule base de connaissances pour représenter les connaissances des robots ainsi que les connaissances des humains, nous voulons que notre logiciel prenne en charge la "distinction soi-autre". Elle est présentée dans [Pacherie 2012] comme le fait que “pour que les représentations partagées (...) favorisent la coordination plutôt que de créer la confusion, il est important que les agents soient capables de séparer les représentations de leurs propres actions et intentions et de celles des autres ”. A notre connaissance, ORO est le seul logiciel proposant cette capacité. Cependant, il utilise un moyen simple de l’implémenter en exécutant plusieurs instances du même système de stockage de triplets, chacune attachée à un agent. En application robotique une telle solution n’est pas suffisante. Ainsi, pour l’utilisation de la planification des tâches, nous pourrions avoir besoin d’estimer un état futur d’une base de connaissance. Pour implémenter cette fonctionnalité, il faut pouvoir capter l’état d’une base de connaissance à un instant donné puis pouvoir le modifier sans impacter la base de connaissance d’origine, qui est continuellement mise à jour par la perception. A la date de cette thèse, aucun outil à base d’ontologie n’offre cette possibilité.

Requête au niveau sémantique : Comme expliqué dans [Broekstra 2002], pour interroger des graphes RDF, trois niveaux sont possibles. Considérant une ontologie écrite en XML, le niveau syntaxique interrogerait la structure XML. Par conséquent, il interrogerait un arbre plutôt qu’un graphe. Une requête serait “donnez-moi les ressources imbriquées dans un élément *Description* ayant l’attribut *about* avec la valeur X”. Il est donc dépendant de la langue et nécessite de connaître les mots-clés utilisés. Le niveau structurel, tel qu’il est disponible dans [Lassila 1998], fournit une abstraction au langage. Il permet des requêtes du genre “donnez-moi tous les éléments de type A” quelle que soit la façon dont il est représenté dans le langage. Cependant, le niveau structurel ne regarde que les triplets explicites. Si B est un sous-type de A et que l’élément x est décrit comme étant de type B, demandant les éléments de type A, x ne sera pas renvoyé. Le dernier niveau de requête est le niveau sémantique qui ne se limite pas à des connaissances explicites. À ce niveau,

en demandant les éléments de type A, x serait renvoyé car x est de type B et B est une spécification de A. Cela signifie que nous ne considérons pas un simple système de stockage de triplets essayant de faire correspondre des modèles. Pour supporter ce niveau de requête, deux solutions peuvent être utilisées : calculer la fermeture du graphe (ajouter A comme type de x et le stocker), ou l'inférer à la requête. Alors que la première solution supprime une partie de la sémantique en aplatissant la hiérarchie[1], la seconde peut être chronophage pour les requêtes.

Requêtes spécifiques : Nous visons à utiliser notre logiciel avec des algorithmes de solveur. Cela signifie que nous voulons fournir un accès fin aux connaissances stockées à haute fréquence. A la différence de Prolog ayant son propre algorithme de recherche, nous avons préféré des requêtes de niveau inférieur telles que : "quels sont les types directs de X ?", "quelles sont les propriétés inverses de Y ?", "quelles les propriétés relient C et D?", ou "E est-il dans le domaine d'application de F?", et cela au niveau sémantique. Néanmoins, pour les requêtes rapides, nous devons faire attention au nombre d'inférences nécessaires au moment de la requête.

Raisonnement à la mise à jour : Pour éviter trop d'inférences au moment de la requête, nous voulons que certaines inférences soient appliquées lors de la mise à jour. Bien que le calcul de la fermeture entière aplatirait la base de connaissance, certaines relations peuvent encore être calculées lors de mises à jour comme celles provenant de relations inverses ou d'axiomes en chaîne.

Thread safe : Puisque le logiciel doit fonctionner comme un serveur, nous voulons qu'il supporte plusieurs requêtes en parallèle mais qu'il soit sécurisé lors de la mise à jour.

Niveau d'expressivité : Considérant le langage d'ontologie Web (OWL), il existe différents niveaux d'expressivité selon les besoins. Le plus expressif est OWL-full, cependant, il est dit qu'il n'est pas informatique, ce qui signifie que les inférences qu'il permet ne peuvent pas être résolues au moment de la requête. Ensuite, OWL-DL (Description Logic) prend en charge la hiérarchie des propriétés et des classes, la restriction de valeur d'énumération, les propriétés inverses ou la restriction de cardinalité. Enfin, OWL-lite est le moins expressif, ne supportant pas la restriction de cardinalité et la restriction de valeur. Même si OWL-DL conviendrait, dans une première version OWL-lite pourrait suffire et sera donc le niveau minimal à atteindre.

Raisonnement personnalisé : À des fins de recherche, nous ne voulons pas nous limiter au raisonnement logique du premier ordre. Nous voulons pouvoir intégrer des processus de raisonnement applicatifs ayant un accès direct à la structure interne. Une telle fonctionnalité pourrait être fournie par le support de plugins

[1] On perd le fait que x est de type A car il est de type B.

par exemple. L'avantage des plugins est que n'importe qui peut ajouter des capacités de raisonnement sans modifier le noyau du logiciel ou posséder une version personnalisée.

Au vu de nos fonctionnalités souhaitées, du nombre d'outils disponibles mais plus maintenus ou sans documentation, et de notre besoin de pouvoir mettre en œuvre de nouvelles capacités en fonction de nos besoins de recherche évoluant dans le temps, nous choisissons de développer de nouveaux logiciels en partant de zéro. Le logiciel résultant est Ontologenius. Un tel choix est risqué car il sera difficile d'atteindre le niveau de logiciels beaucoup plus matures. En revanche, cela nous a permis au cours de cette thèse d'avoir la maîtrise de son fonctionnement interne et de pouvoir le faire évoluer facilement au cours des différents projets dans lesquels nous l'avons utilisé.

Ontologenius s'inscrit dans la continuité du logiciel ORO mais en proposant une gestion multi-instances plus poussée et un système de requêtes plus adapté à l'intégration dans des algorithmes.

Recherche d'un itinéraire avec des connaissances sémantiques

Sur la base d'Ontologenius, nous présentons plusieurs contributions utilisant les connaissances stockées.

Nous avons tous déjà été sollicités, ou avons nous-mêmes demandé, notre chemin en ville, dans un centre commercial, ou plus simplement dans une maison. Lorsque nous fournissons de telles informations à une personne, nous effectuons ce que l'on appelle communément une tâche d'orientation. Même si cela peut sembler anodin pour nous, développer un robot capable de l'exécuter peut être un défi. Dans ce chapitre, nous choisissons de nous concentrer sur la sous-tâche consistant à générer la phrase d'explication. Cette sous-tâche s'appelle la description de l'itinéraire. Pour l'effectuer, nous avons d'abord besoin d'un ensemble de connaissances sur l'environnement dans lequel la personne guidée va se déplacer, comme les chemins, les intersections des chemins, ou les éléments qui les côtoient. Ensuite, nous avons besoin d'un ensemble de « bonnes pratiques » pour fournir un itinéraire assez facile à suivre et à retenir.

Dans le domaine de l'Interaction Humain-Robot (HRI), des robots guides ont été étudiés intensivement et déployés dans des centres commerciaux [Okuno 2009], des musées [Burgard 1999, Clodic 2006, Siegwart 2003], ou des aéroports [Triebel 2016]. D'un point de vue représentation des connaissances, on peut remarquer l'utilisation de représentations métriques [Thrun 2007] ou topologiques [Morales Saiki 2011] pour représenter l'environnement dans lequel le robot agit. Puisque nous nous concentrons sur la tâche de description d'itinéraire, nous considérons que le robot n'accompagne pas l'humain jusqu'à sa destination finale mais explique plutôt comment l'atteindre. Par conséquent, la représentation métrique ne sera pas considérée comme étant principalement utilisée à des fins de navigation [Thrun 2007]. Pour ef-

fectuer plus spécifiquement une description d'itinéraire, la connaissance topologique n'est pas suffisante. En plus de la topologie de l'environnement, le robot a besoin de connaître les types des éléments composant l'environnement et leurs noms en langage naturel. Certaines contributions ont ainsi tenté de mélanger des représentations métriques ou topologiques avec des représentations sémantiques pour contenir ces connaissances supplémentaires [Satake 2015b, Chrastil 2014, Zender 2008]. Cependant, les mélanger peut créer un manque d'uniformité dans la représentation globale des connaissances. De cette façon, créer une représentation unique permettant à un robot de calculer des itinéraires et de les exprimer pourrait assurer l'uniformité des connaissances.

Même muni d'une représentation cohérente de son environnement, le robot doit trouver un itinéraire non pas pour lui-même mais pour l'humain guidé. Un robot accompagnant l'humain n'a qu'à déterminer un chemin, adapté à ses capacités et interprétable uniquement par lui-même. Fournissant un itinéraire à un humain, l'itinéraire doit être adapté aux capacités et aux connaissances humaines. Par exemple, dans un environnement extérieur, on ne donnera pas le même itinéraire pour un automobiliste ou un cycliste. Dans le cadre d'un centre commercial, nous ne donnerons pas de parcours avec escalier à une personne à mobilité réduite ou à une personne avec un caddie. Une fois calculé, le robot doit expliquer l'itinéraire. Là où les cartes interactives n'ont qu'à mettre en évidence un chemin, ici, le robot doit générer une phrase que l'humain pourra mémoriser. Bien sûr, le robot n'instruira pas un humain avec une phrase comme « marche de 30 mètres et tourne à -90 degrés ». Ce ne serait pas adapté. L'utilisation de l'orientation et de la référence aux éléments de l'environnement sera nécessaire à travers une phrase comme "marcher jusqu'au fleuriste puis tourner à gauche". Nous voulons donc que le robot génère des plans compréhensibles et exécutables par l'humain.

La première contribution de ce chapitre est une **représentation unifiée** d'un environnement intérieur utilisant une ontologie, pour inclure à la fois des connaissances topologiques et sémantiques. Ensuite, sur la base de cette représentation, nous avons proposé un premier algorithme pour **trouver un itinéraire approprié** à expliquer à un humain et un second algorithme pour **verbaliser un itinéraire** de manière appropriée.

Génération d'expression de référence basée sur une ontologie

Poursuivant sur l'exploitation des connaissances et plus précisément sur la communication spatiale, nous présentons maintenant une contribution autour de la tâche de génération d'expressions de référence.

Faire référence à une entité est l'une des tâches les plus courantes que nous effectuons chaque jour." Pouvez-vous m'apporter ma tasse ? C'est la noire à côté de l'évier". "Je ne me souviens pas du nom de l'homme avec la chemise rouge et les lunettes". "J'ai perdu mes clés, elles sont sur un porte-clés avec une peluche en

forme de licorne". Un tel type de communication, précis et efficace, est un aspect clé pour le succès d'une tâche collaborative. Néanmoins, dans des environnements complexes avec une grande variété d'objets, de lieux ou de personnes, faire référence à une entité spécifique peut devenir un véritable défi pour un robot. Elle doit tenir compte du contexte de la tâche globale, de la diversité des faits qui peuvent être extraits de la situation et qui dépendent des modalités de perception disponibles, et de la connaissance partagée (ou non) entre le robot et son partenaire.

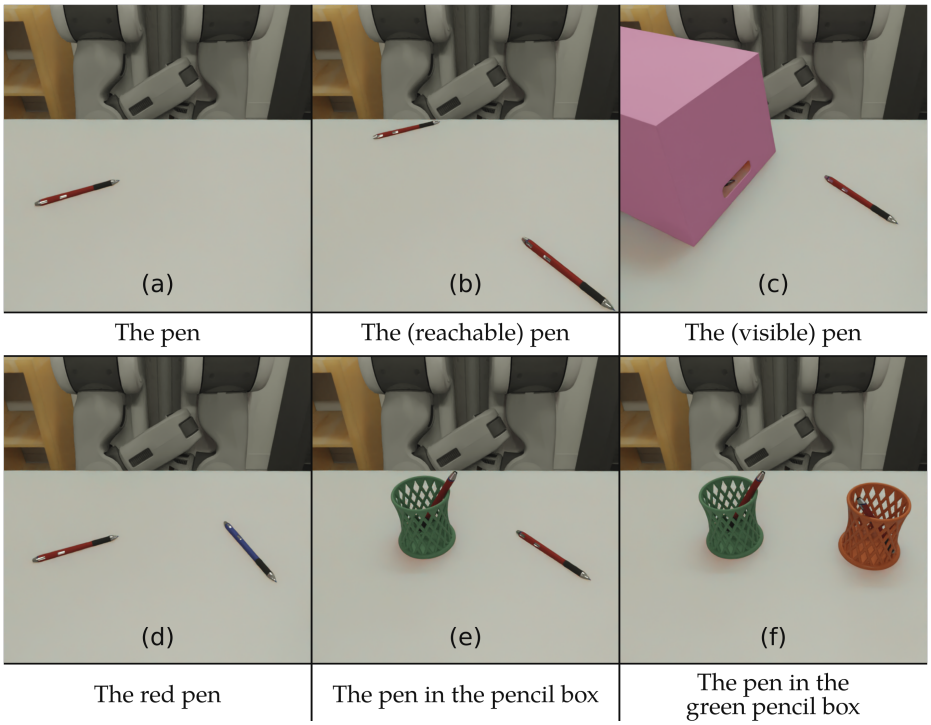

Figure C.1: Six situations vues du point de vue de l'auditeur, avec le robot placé de l'autre côté de la table. Se référer au même stylo implique différents mécanismes pour lever l'ambiguïté, dans chaque situation. La phrase en-dessous de chaque situation est une expression de référence possible pour désigner ledit stylo sans ambiguïté.

Considérez la situation où vous êtes autour d'une table avec votre robot collaboratif et le robot a besoin d'un stylo. La simple instruction *"Donnez-moi le stylo"* peut entraîner plusieurs situations de complexités diverses. Dans le cas où il n'y a qu'un seul stylo (Fig.C.1a), s'y référer est évident. Considérons maintenant deux stylos sur la table. Si l'un est accessible par vous (l'humain) et l'autre ne l'est pas (Fig.C.1b), l'accessibilité peut être utilisée pour désigner le stylo. Si les deux stylos ne sont pas accessibles mais que l'un est visible pour vous et l'autre ne l'est pas (Fig.C.1c où un stylo est caché sous la boîte), la visibilité par prise de perspective peut également être utilisée pour déterminer que le autre stylo ne conduit pas à un ambiguïté avec celui référencé. Maintenant, les deux stylos sont visibles et ac-

cessibles, mais l'un est bleu et l'autre est rouge (Fig.C.1d). L'ajout de la couleur du stylo résout l'ambiguïté. Si les deux stylos ont la même couleur mais que l'un est dans une boîte à crayons (Fig.C.1e), la relation avec la boîte à crayons résout l'ambiguïté. Si malheureusement, les deux sont dans une boîte à crayons mais l'un est vert et l'autre orange (Fig.C.1f), le rapport avec la boîte à crayons et la couleur de cette dernière lève l'ambiguïté. On pourrait continuer ainsi longtemps, en considérant qu'un est à votre gauche et un à votre droite, qu'il n'y a pas de stylo sur la table mais qu'il y en a un sur une étagère et ainsi de suite.

Jusqu'à présent, nous considérions que le robot connaissait les notions de stylo et de boite a crayon ainsi que leurs noms en langage naturel pour en parler. Cependant, imaginez-vous voyager dans un autre pays et devoir parler anglais[2], vous pouvez parfois ne pas connaître certains mots précis et donc en utiliser un plus générique à la place. Cependant, ce faisant, une nouvelle ambiguïté peut être levée car ce mot générique fait également référence à d'autres entités. Il en est de même pour notre robot s'il doit parler français et ne connaît pas la traduction du concept de la boîte à crayons. Il devra utiliser un terme plus générique, tel que « conteneur ». Cependant, ce terme plus générique pourrait également désigner d'autres entités, comme des boîtes.

En plus des noms de concepts, le robot doit faire attention aux relations qu'il utilise. Le poids ou la longueur exacte d'un objet ne serait pas utile car un humain ne peut pas facilement les évaluer. Au contraire, la couleur de l'objet semble être une propriété appropriée à utiliser, sauf si le robot partenaire est daltonien. Cela signifie que le robot doit utiliser des relations qu'il estime connues et facilement observables par son partenaire. Cela peut être fait en considérant la théorie de l'esprit et en effectuant la génération d'expression de référence sur la base de connaissances humaine estimée par le robot.

Cette tâche pour faire référence à une entité précise parmi d'autres est communément appelée la tâche **Referring Expression Generation (REG)**. Elle est souvent décomposée en deux sous-tâches : la **détermination du contenu** et la **réalisation linguistique** [Krahmer 2012]. La détermination de contenu vise à déterminer les relations à utiliser pour se référer à une entité tandis que la réalisation linguistique vise à choisir les mots à utiliser pour communiquer le contenu. Notre contribution dans ce chapitre est centrée sur la détermination du contenu mais nous ne pouvons pas considérer ces deux sous-tâches comme entièrement indépendantes. Comme expliqué précédemment, si le robot ne connaît pas certains noms de concepts en langage naturel, la réalisation linguistique échouera dans le cas où la détermination du contenu les sélectionne. Une autre possibilité pour la réalisation linguistique serait de choisir un mot plus général mais il ne correspondrait pas au contenu déterminé. On pourrait imaginer une base de connaissances dédiée pour le REG avec uniquement des concepts utilisables en langage naturel, mais une telle solution n'est pas adaptée si l'on veut une base de connaissances unique pour tout le système robotique. De plus, il pourrait être difficile de maintenir cette base

[2]Je m'excuse auprès des anglophones natifs qui ne profiteront pas pleinement de cet exemple.

de connaissances dédiée pendant l'interaction, en plus d'autres car il s'agirait d'un dédoublement de connaissances.

La principale contribution présentée dans ce chapitre est un **algorithme basé sur une ontologie et indépendant du domaine pour la génération d'expressions de référence**. Il utilise une fonction de coût personnalisable estimant la charge cognitive nécessaire à un humain pour interpréter l'expression de référencement afin de produire **l'ensemble "optimal" d'assertions verbalisables** qui permet de se référer sans ambiguïté à une entité donnée.

Comme expliqué dans [Reiter 2000], il s'agit de "comment produire une description d'une entité qui permet à l'auditeur d'identifier cette entité dans un contexte donné". Alors que cette tâche est étudiée depuis des décennies, aucune des méthodes existantes n'a tenté d'utiliser une ontologie comme base de connaissances, travaillant la plupart du temps sur une base de connaissances dédiée à la tâche. De plus, parce que leur base de connaissances est dédiée à la tâche, elle n'est composée que de relations utilisables pour communiquer avec un humain. Cependant, si l'on considère une base de connaissances partagée par tous les composants de l'architecture, certaines connaissances peuvent avoir un usage purement technique pour le fonctionnement du robot. Plus important encore, aucun des travaux existants n'a vraiment pris en compte la notion de contexte. Leur seul but était de désigner une entité dans un état donné d'un environnement. Cependant, dans son utilisation pour de l'interaction humain-robot, la génération d'expression de référence n'est qu'une action d'une tâche plus large, dans laquelle un contexte existe, qu'il s'agit d'informations déjà connues sur l'entité à laquelle se référer ou d'une restriction implicite sur les entités concernées. Nous présentons ainsi un nouvel algorithme gérant toutes les problématiques précédemment évoquées et utilisant une ontologie comme base de connaissances. De plus, à travers des comparaisons avec d'autres algorithmes, nous montrons que notre contribution est, à ce jour, la plus efficace, résolvant la plupart des problèmes de génération d'expressions de référence en moins d'une milliseconde. Enfin, comme une preuve d'utilisabilité, l'algorithme a été intégré dans une architecture robotique avec une base de connaissances mise à jour en permanence par la perception.

Estimation de la faisabilité et du coût de la communication lors de la planification des tâches

Profitant des bonnes performances de notre algorithme de génération d'expressions de référence, dans ce chapitre, nous proposons une méthode l'intégrant à un planificateur de tâches.

Il est bien établi qu'un aspect clé du succès des tâches collaboratives repose sur une communication claire et fluide ancrée dans le contexte de l'interaction. Centré sur la communication verbale, dans le domaine de recherche Traitement du langage naturel (TALN) et par extension dans le domaine de l'interaction humain-robot, il a été divisé en deux doubles problèmes [Tellex 2020]. D'une part, la compréhension

du langage naturel (NLU) vise au robot à interpréter et à fonder les énoncés humains par rapport à la situation actuelle et à réagir en fonction de celle-ci [Brawer 2018]. D'un autre côté, la génération de langage naturel (NLG) vise à permettre au robot à produire du langage. Il peut s'agir soit de demander de l'aide [Tellex 2014], d'aligner les connaissances [Devin 2016], soit d'expliquer sa décision à son partenaire [Roncone 2017].

Dans le chapitre précédent, nous avons introduit un algorithme capable de générer le contenu d'une expression référente. Une telle contribution relève donc du problème NLG. Considérer le REG comme une action pouvant être effectuée par le robot signifie que le robot pourrait planifier une telle communication en termes de **quand** et **quoi** pour communiquer. Alors que le "quand" est directement géré par le planificateur de tâches, le "quoi" en termes de contenu est fourni par le REG[3]. Cependant, le REG ne fournit pas seulement le contenu mais est également en mesure d'indiquer si une telle communication est réalisable ou non et de donner des informations sur son coût. Dans le contexte du REG, la faisabilité de la communication est liée à la capacité à générer une expression référente sans ambiguïté alors que le coût dépend du nombre de relations à communiquer. Étant donné que l'algorithme REG fonctionne sur une base de connaissances représentant l'état actuel de l'environnement, le maintien d'une représentation comparable de l'environnement pour les états futurs de la tâche (comme cela se fait dans la planification symbolique des tâches) permettrait au robot d'estimer le **faisabilité** et le **coût** des actions de communication verbale tout au long d'une tâche.

Avec ces deux informations, à savoir le coût et la faisabilité, un planificateur de tâches pourrait comparer les communications verbales entre elles, comparer avec d'autres moyens de communication, minimiser la complexité globale de la communication et éviter certains échecs du plan. Cette approche pour estimer la communication lors de la planification des tâches peut être comparée à celle proposée dans [Lallement 2016]. Dans ce dernier cas, les actions de mouvement ont été évaluées lors de la planification des tâches pour estimer leur faisabilité, leurs coûts et leurs effets indirects. Avec les deux approches, les plans symboliques peuvent être optimisés et peuvent être plus fiables pour prévenir les échecs d'exécution et donc le besoin de réparation.

Pour mieux comprendre l'avantage de considérer la communication au niveau de la planification des tâches, considérons les situations de la Figure C.2. Le robot doit disposer des tags RFID sur trois zones d'une table. Le robot peut les identifier avec leur identifiant unique mais les tags étant trop petits, il ne peut pas les saisir. Au contraire, le partenaire humain ne peut pas les identifier de manière unique mais peut les saisir. Pour cet exemple, nous supposons également que le robot ne peut pas pointer vers les tags. Le robot doit donc communiquer les actions successives que l'humain devra effectuer pour passer de la configuration initiale (C.2 a) à la

[3]Le REG ne détermine pas l'entité à laquelle se référer mais seulement comment elle sera référencée à. On pourrait dire que le « quoi » est choisi par une composante décisionnelle supérieure, tandis que le « comment » est déterminé par le REG. Pour correspondre à la définition habituelle, nous supposons cette simplification linguistique.

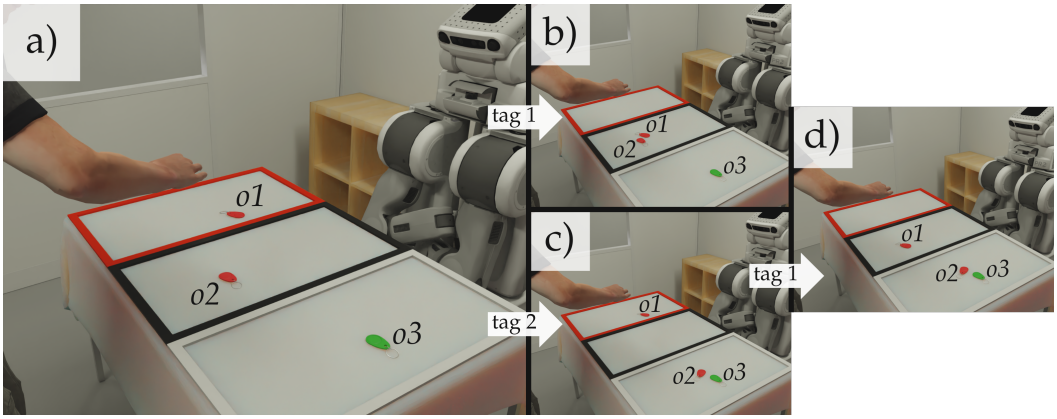

Figure C.2: Une tâche collaborative Humain-Robot avec trois zones colorées et trois tags RFID (situation a). Le robot doit expliquer à son partenaire humain de mettre le tag *o1* dans la zone noire et le tag *o2* dans la zone blanche, pour atteindre la situation d. Les identifiants des objets ne sont connus que du robot. Si toutes les communications de la tâche ne sont pas planifiées à l'avance, une situation de blocage peut apparaître si le robot demande d'abord de déplacer la balise *o1* avant *o2* (situation b).

configuration but (C.2 d). Entre les deux configurations, seules les tags *o1* et *o2* doivent être déplacés. Le tag *o1* doit être déplacé de la zone rouge vers la zone noire et *o2* de la zone noire vers la zone blanche. Alors que le tag *o3* peut être référencé sans ambiguïté grâce à sa couleur, les deux autres ne le peuvent pas. Cependant, ils peuvent être référencés grâce à la zone dans laquelle ils se trouvent (par exemple *"le tag qui est dans la zone rouge"*).

Si le contenu des communications n'est affiné qu'à l'exécution, deux solutions équivalentes peuvent être envisagées (C.2 séquence a-b-d et a-c-d). A l'exécution, la première solution commence par demander à l'humain de déplacer *o1* dans la zone noire résultant en l'instruction *"prendre le tag qui est dans la zone rouge et le mettre dans la zone noire"* . Dans cette nouvelle situation où les deux tags rouges sont maintenant dans la zone noire (Figure C.2b). Le robot n'a aucun moyen de désigner le tag *o2* sans ambiguïté. Par conséquent, la tâche est bloquée[4]. L'estimation de la faisabilité et du coût de la communication au cours du processus de planification aboutirait à la deuxième solution possible. Le robot demande d'abord de déplacer le tag *o2* (Figure C.2 c) puis le tag *o1* (Figure C.2 d). Si le robot avait pu pointer, l'impasse de la première solution aurait pu être évitée avec une action de pointage et néanmoins, grâce à l'estimation des coûts de communication, la solution la moins

[4]Le robot pourrait utiliser des relations spatiales comme la droite, la gauche ou le plus proche de moi. Cependant, la génération d'une telle expression de référence n'est pas une tâche a comfacile et spréhension non plus. Même si la situation n'est pas vraiment bloquée, la communication requise peut être complexe.

chère peut être sélectionnée[5].

La principale contribution présentée dans ce chapitre est une approche pour **estimer la faisabilité et le coût de la communication au niveau de la planification des tâches**. Cela implique un **lien fin entre un planificateur et une ontologie** pour estimer la communication ancrée dans l'état futur estimé de l'environnement.

Le but de cette méthode est de doter le planificateur de tâches de la capacité d'estimer la faisabilité et le coût de la communication, et donc d'éviter les impasses et de trouver des plans minimisant la complexité de la communication. Certes, dans notre contribution, la communication s'est limitée à la génération d'expressions de référence d'entités mais pourrrait s'étendre à d'autres. Premièrement, le planificateur de tâches que nous utilisons est HATP [Lallement 2014], un planificateur de tâches prenant en compte l'humain, ce qui signifie qu'il est capable de planifier pour le robot et son partenaire, en estimant leur état mental futur et leurs capacités à assigner des tâches à l'un ou l'autre. Pour estimer les futurs états mentaux, HATP dispose d'une représentation interne dédiée, limitée aux entités utilisées dans la tâche à planifier. Cependant, pour fonctionner, notre algorithme de génération d'expressions de référence a besoin d'une ontologie comme base de connaissances et doit prendre en compte toutes les entités de l'environnement. Pour le faire fonctionner, nous présentons donc un schéma permettant au planificateur de mettre à jour une ontologie, représentant la future connaissance estimée du partenaire humain, afin de pouvoir exécuter l'algorithme dessus. Cette contribution tire pleinement avantage des fonctionnalités apportées par Ontologenius, à savoir la possibilité de maintenir une instance par agent, et de copier une instance à un moment donné pour la modifier librement. La méthode résultante est implémentée dans une architecture robotique en tant que preuve de concept.

Étendre le REG avec des connaissances sur les activités passées

Lorsque deux ou plusieurs agents effectuent une tâche collaborative, bien qu'ils puissent avoir une perception différente de leur environnement partagé, ils peuvent estimer les informations qu'ils partagent et peuvent ainsi les utiliser pour communiquer sur des entités qu'ils estiment connues des autres. Cette hypothèse est celle couramment utilisée pour développer et évaluer les méthodes de génération d'expressions de référence via l'utilisation de la légende de l'environnement[Duboue 2015]. Ces situations sont des images toujours prises du point de vue de l'auditeur. L'image, ou la représentation des connaissances associée, est fournie à l'algorithme qui doit générer une expression de référence. Cette

[5]Beaucoup d'autres solutions pourraient exister mais dépendent de la capacité robotique. Donner les deux instructions dans l'état initial avant l'acte humain résout aussi le problème par exemple. Néanmoins, si le robot ne peut pas comparer ces différentes solutions au regard de sa capacité actuelle, des situations non souhaitables peuvent encore apparaître.

hypothèse a également été utilisée lorsque le REG a été appliqué à l'interaction Humain-Robot et peut être comparé à un robot apparaissant dans un environnement et devant désigner un objet. Cependant, cette désignation se produit lors d'une activité conjointe entre un robot et un partenaire humain, ce qui signifie que les objets désignés peuvent avoir été utilisés, déplacés ou déjà évoqués. Toutes ces informations sur la tâche effectuée peuvent être considérées comme des connaissances supplémentaires partagées par les agents impliqués. On peut ainsi se référer aux entités à travers ces actions passées en plus de leurs attributs et relations les unes avec les autres.

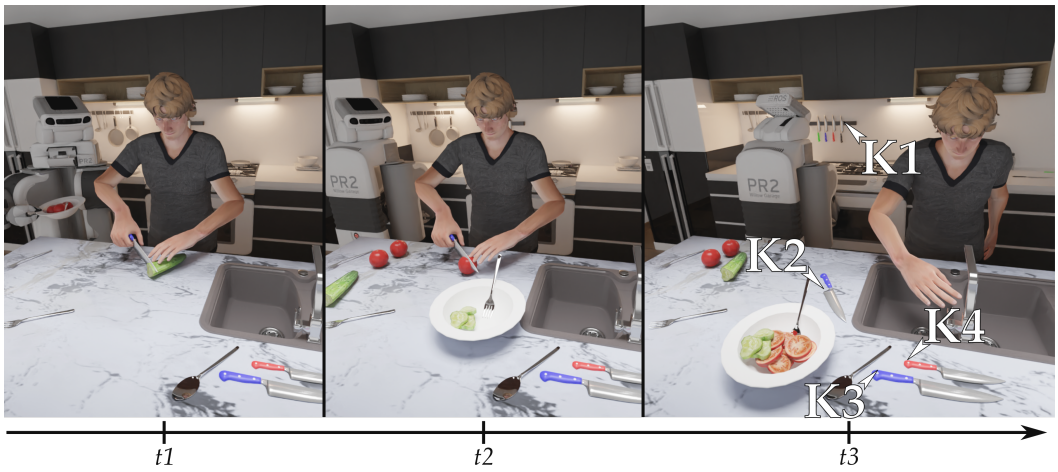

Figure C.3: Se référer au couteau *k2* dans la situation actuelle (*t3*) est impossible si le robot effectue une action qui ne lui permet pas de voir ce qui se trouve devant l'humain. Compte tenu des étapes précédentes de la tâche humaine, le robot peut se référer au couteau à travers l'action pour couper une tomate (*t2*) ou pour couper un concombre (*t1*).

Considérez la légende de l'interaction représentée dans la Figure C.3 à l'instant courant *t3*. Le robot, au fond de la cuisine, doit demander le couteau à l'humain *k2*. Étant donné que le robot effectue une autre action de la tâche conjointe, il ne peut pas voir ce qui se trouve devant l'humain. Par conséquent, il ne peut connaître et donc utiliser de relations spatiales sur *k2*[6]. Par conséquent, le robot ne peut utiliser que les attributs *k2* (c'est-à-dire uniquement sa couleur) pour générer une expression lui faisant référence. Toujours en ne considérant que l'instant courant *t3*, deux autres couteaux bleus se trouvent dans la cuisine étant *k1* et *k3*. Le couteau *k1* est fixé au mur devant le robot ce qui signifie qu'il lui est déjà accessible et non à l'humain. Ce couteau peut donc être considéré comme étant hors contexte et n'entraînant aucune ambiguïté avec *k2*. L'autre couteau bleu *k3* reste ambigu

[6]On pourrait aussi considérer un objet connu du robot mais pour lequel il ne dispose d'aucune information concernant sa nouvelle localisation et sa recherche. Il faudrait s'y référer, demander l'aide humaine, sans possibilité d'utiliser des relations spatiales.

puisqu'il n'a pas d'attribut perceptible qui diffère de celui auquel le robot doit se référer.

Jusqu'à présent, nous n'avons considéré que la situation actuelle *t3* et non l'expérience partagée entre l'homme et le robot sur la tâche qu'ils effectuent. A l'instant précédent *t2* l'humain coupait une tomate avec le couteau *k2*. Il était évident pour l'humain que le robot observait la scène pendant qu'il agissait. Cette nouvelle information sur l'action effectuée pourrait ainsi être utilisée par le robot pour générer une référence au couteau recherché dans la situation actuelle. Une expression de référence possible serait "*le couteau avec lequel vous avez coupé la tomate*".

Considérons maintenant l'action une étape avant de couper la tomate à l'instant *t1*. L'humain coupait un concombre avec ce même couteau. La combinaison de ces deux actions passées peut être considérée comme la tâche de préparer des légumes. Le robot peut donc aussi utiliser cette connaissance pour se référer au couteau. Une expression de référence possible considérant la totalité de l'interaction serait "*le couteau avec lequel vous avez préparé les légumes*". L'exploitation des connaissances partagées sur les activités passées en plus des attributs et propriétés habituels pourrait conduire à la génération d'une expression de référence plus riche qui pourrait être plus facile à comprendre par le partenaire humain. De plus, cela permet de générer des ER dans des contextes où la méthode précédente n'était pas efficace.

Ce chapitre est une extension de notre précédent travail [Buisan 2020b] présenté dans le chapitre 4. Il a été intégré dans un planificateur de tâches de base d'agents hiérarchique basé sur les coûts pour estimer la faisabilité et le coût des ER pendant le processus de planification [Buisan 2020a], présenté au chapitre 5. Dans ce chapitre nous visons donc à créer le lien inverse, rendant le REG capable d'utiliser des traces d'exécution résultant de l'exécution de plans hiérarchiques générés par HATP. Comme les chapitres précédents, nous nous concentrons uniquement sur la détermination du contenu du problème REG mais continuons à considérer la nécessité d'avoir des noms en langage naturel pour permettre la réalisation linguistique.

La principale contribution de ce chapitre est une extension de l'algorithme REG basé sur une ontologie en **considérant les activités passées des agents**. Une contribution secondaire de ce chapitre est une proposition d'un formalisme pour **représenter les Traces d'Exécution Hiérarchiques** (plans exécutés basés sur HTN) dans une ontologie. Notre contribution précédente a considéré des fonctions de coût basées sur les propriétés des relations utilisées pour représenter la charge cognitive requise pour qu'un humain interprète l'expression de référence. Dans cette extension, nous proposons d'ajouter des fonctions de coût personnalisables basées sur le temps, pour représenter la charge cognitive requise pour qu'un humain se souvienne des activités référencées.

Avec l'intégration de l'algorithme de génération d'expressions de référence avec un planificateur de tâches, nous mettons donc en évidence le fait que l'acte de faire référence à une entité, apparaît la plupart du temps dans le contexte d'une tâche. Au cours de cette tâche, les agents agissent avec l'environnement et manipulent les entités de l'environnement. Sur cette base, dans ce chapitre, nous proposons d'utiliser

cette connaissance supplémentaire sur les activités des agents avec les entités de l'environnement comme une nouvelle information, utilisable pour s'y référer. En continuant à utiliser HATP pour le planificateur de tâches, nous proposons d'abord un moyen de représenter le domaine de planification de tâches dans une ontologie, ainsi que l'exécution de ce plan, que nous avons appelé la trace d'exécution hiérarchique. Ensuite, nous présentons les modifications apportées à notre algorithme de génération d'expression de référence d'origine, lui permettant d'utiliser la représentation des activités passées pour générer un nouveau type d'expression de référence.

Aller plus loin que les relations binaires dans le REG

Représenter toute la complexité des connaissances composant notre monde dans un langage lisible par machine est un enjeu central en intelligence artificielle. Venant du Web sémantique, nous avons vu que l'utilisation d'une ontologie à travers des langages basés sur RDF a réussi à s'imposer dans le domaine de l'intelligence artificielle et donc de la robotique. Cependant, ce qui est souvent considéré comme une limitation des ontologies est sa capacité à ne représenter que des relations unaires et binaires. Les relations binaires telles que *"Sean Connery a la nationalité britannique"* sont décrites sous la forme de triplets *(sean_connery, hasNationality, british)*. Les relations unaires telles que *"Sean Connery est un acteur"* peuvent alors être transformées en relations binaires par l'ajout de prédicats dédiés *(sean_connery, isA, Actor)*. Cependant, la description de relations plus complexes impliquant plus de deux entités est beaucoup plus difficile en utilisant ce type de représentation.

Prenons l'exemple de Sean Connery[7], si nous voulons vous référer à lui[8], on pourrait affirmer qu'il s'agit de l'acteur jouant le rôle de James Bond *(sean_connery, hasPlayedRole, James Bond)*. Cependant, d'autres acteurs ont joué ce rôle. On pourrait aussi dire qu'il est l'acteur jouant dans le film Goldfinger *(sean_connery, hasPlayedIn, goldfinger)* mais encore une fois d'autres le font. On pourrait enfin expliquer qu'il s'agit de l'acteur jouant le rôle de James Bond et jouant dans le film Gold Finger. Cependant, nous limiter à l'utilisation de relations binaires modifie l'information exacte. Une description plus précise serait qu'il est l'acteur jouant le rôle de James Bond dans le film Gold Finger. On voit ici la nécessité de relations impliquant plus de deux entités. Dans notre exemple, nous devons lier les trois entités que sont l'acteur "Sean Connery", le rôle "James Bond" et le film "Gold Finger". Ensemble, ils décrivent une performance. Sans être explicitement liées ces trois informations ne représenteraient pas la performance. D'ailleurs, sans ces liens, on pourrait donner une explication comme l'acteur jouant le rôle de James Bond et

[7]Dans le cas où vous ne savez pas qui est Sean Connery, n'hésitez pas à prendre un autre acteur que vous aimez mais vous devrez adapter l'ensemble de l'exemple.

[8]Évidemment on veut faire référence à lui sans son nom puisque l'on considère une personne s'étant reconnue dans la note précédente.

jouant dans le film Rising Sun. Les deux informations sont vraies mais n'ont pas de sens ensemble.

Pour faire référence à une entité, étant un objet ou un agent, de telles relations complexes pourraient être utiles mais doivent être gérées avec soin pour garder le lien entre chaque relation binaire la composant. A la lumière de cette considération, on peut observer que la description des tâches passées des agents utilisée dans le chapitre précédent est basée sur le même principe. Là où nous nous référons à Sean Connery à travers son rôle et le film dans lequel il joue, nous avons décrit le couteau à travers le légume qu'il a servi à couper et l'agent qui l'a utilisé. Cependant, selon le contexte de la conversation, il n'est pas toujours nécessaire d'utiliser toutes les relations binaires d'une relation aussi complexe. Nous n'en avons peut-être besoin que d'une. En essayant de lister les acteurs ayant le titre honorifique de "Sir", l'expression référente *"L'homme ayant joué James Bond"* pourrait suffire. De la même manière, pour désigner le couteau, la phrase *"Le couteau avec lequel vous avez coupé"* pourrait également suffire dans certains contextes.

Même si l'adaptation de l'algorithme de génération d'expressions de référence du chapitre précédent apporte de nouvelles possibilités, nous montrons donc dans ce chapitre qu'il présente certaines limites. La principale est la connaissance a priori dont il a besoin sur la représentation. Elle est ainsi restreinte à une représentation unique des activités passées. Cependant, nous avons vu dans la littérature qu'il existe de nombreuses représentations d'activités, chacune créée pour des applications particulières. Dans ce chapitre, nous explorons donc des modèles d'ontologie communs utilisés pour représenter de telles relations complexes sous la forme de relations n-aires. En leur ajoutant un ensemble de motifs paramétriques simples en tant que label, nous avons créé ce que nous avons appelé des relations composées. Avec ce nouveau type de relation, nous adaptons notre algorithme de génération d'expression de référence original pour fournir un moyen plus générique de générer une expression de référence. L'algorithme résultant fonctionne toujours avec une description des activités passées, mais aussi avec toute représentation utilisant une relation n-aire. De plus, nous montrons que les performances du nouvel algorithme sont meilleures que celles de l'algorithme précédent et égales à celles de l'algorithme d'origine lorsqu'aucune relation composée n'est utilisée dans la représentation.

Un robot dans le centre commercial : le projet MuMMER

A travers les chapitres suivants, nous présentons deux architectures robotiques impliquant les contributions présentées tout au long de cette thèse.

Dans les environnements intérieurs à grande échelle, comme les musées, les centres commerciaux ou les aéroports, la présence de grands écrans interactifs, de cartes ou de panneaux souligne l'importance de fournir des informations sur les itinéraires. Cependant, la lecture de telles cartes peut être difficile et certaines informations peuvent manquer, comme l'emplacement des magasins vendant un produit donné.

Pour apporter ces nouvelles informations et aider les gens à trouver leur itinéraire dans de grands environnements intérieurs tels que les centres commerciaux, des robots peuvent être utilisés.

Pour étudier ce défi et l'exigence d'interaction humain-robot soulignée, dans le cadre du projet européen H2020 MuMMER [Foster 2016], nous avons développé et déployé un robot de service dans l'un des plus grands centres commerciaux de Finlande, Ideapark dans le ville de Lempaalä. Le robot résultant a pu discuter avec les clients et les guider. Le dialogue a été apporté par un partenaire du projet [Papaioannou 2018]. La contribution de l'équipe LAAS-RIS, et donc l'objectif de ce chapitre, portait sur la tâche de direction.

Avec un centre commercial comptant environ 1,2 kilomètre de rues piétonnes et plus de 150 magasins, avoir un robot accompagnant les clients prendrait beaucoup de temps. En nous inspirant des employés du centre commercial, nous avons choisi de décrire verbalement le parcours tout en accompagnant ce discours par des gestes de pointage. Le robot peut cependant se déplacer de quelques mètres si besoin. Le résultat de ce projet est une architecture de robot complète qui intègre un certain nombre de composants. Chacun d'eux utilise des modèles et des algorithmes décisionnels, prenant en compte explicitement les humains.

Dans le chapitre courant, nous présentons donc le projet MuMMER et l'architecture qui en résulte. Le but du projet MuMMER était de développer une architecture robotique permettant à un robot de guider un client dans un centre commercial et de discuter avec lui. Par guider, nous n'entendons pas accompagner le client mais plutôt lui expliquer le parcours à suivre. Pour pouvoir trouver l'itinéraire à expliquer et générer la phrase d'explication, nous utilisons la contribution sur la description de l'itinéraire. Pour conserver la représentation de l'environnement et la partager avec d'autres composants, nous utilisons Ontologenius. L'architecture résultante, incarnée dans un robot Pepper, a fonctionné pendant 14 semaines dans un centre commercial en Finlande.

La tâche du directeur : Évaluer les architectures cognitives

Développer des architectures robotiques adaptées à l'Interaction Humain-Robot et ainsi capables de réaliser des interactions de manière acceptable est encore aujourd'hui un véritable défi. La complexité vient entre autres du nombre de capacités dont doit être doté le robot et donc du nombre de composants logiciels qui doivent être intégrés de manière cohérente. De telles architectures doivent fournir au robot la capacité de percevoir son environnement et les personnes/agents avec lequel il interagit, de fusionner et d'interpréter cette information perceptive, de communiquer à son sujet, de planifier des tâches avec son partenaire, d'estimer la perspective et l'état mental des autres, etc. Une fois développé, l'évaluation de ces architectures peut être difficile car tous ces composants sont regroupés en un seul système. Les tâches que l'on souhaite généralement que le robot exécute doivent mettre en

avant un maximum de capacités, tout en étant suffisamment simples pour être reproduites par la communauté. De plus, nous devrions pouvoir mener avec elle des études d'utilisateurs pour valider les choix concernant les utilisateurs naïfs.

Étant donné qu'un objectif à long terme du domaine de la robotique est de voir des robots agir dans notre vie quotidienne, de nombreuses tâches et scénarios ont été inspirés par les activités quotidiennes. Même si ces tâches offrent une grande variété de situations à gérer, puisque le partenaire humain n'est pas limité dans ses actions, elles ont l'inconvénient de ne pas mettre en avant certaines capacités subtiles pourtant nécessaires à une bonne interaction. La tâche de guidage de robot [Satake 2015b] dans un centre commercial, un musée ou un aéroport, nécessite des compétences de communication élevées pour comprendre les requêtes (éventuellement en discutant) et y répondre, que ce soit pour indiquer une direction ou pour donner des conseils. Cependant, les besoins de perception peuvent être limités en raison des vastes environnements, ainsi que les besoins de prise de perspective dus à la même perception de l'environnement par le robot et l'humain[9]. Enfin, même si l'humain doit contribuer au problème, avec une telle tâche le partenaire humain n'est pas forcément acteur de la tâche et peut juste écouter le robot une fois sa question posée. Même si elles sont dans des environnements plus contraints, les tâches de type barman [Petrick 2012] présentent les mêmes inconvénients. En effet, l'humain est considéré comme un client, et à ce titre, l'interaction avec le robot est limitée. Le robot ne demandera jamais à l'humain de l'aider à effectuer une tâche et ses actions ne nécessitent ni coordination ni collaboration complète.

Pour impliquer le partenaire humain dans la tâche et lui demander d'agir avec le robot, des tâches de type assemblage [Tellex 2014][10] peuvent être utilisées. Néanmoins, dans la plupart des cas, l'humain agit comme un assistant plutôt que comme un partenaire, car une collaboration complète peut être difficile à réaliser. Le robot élabore ainsi un plan et effectue l'assemblage, puis demande de l'aide lorsqu'il détecte des erreurs lors de l'exécution (par exemple, lorsqu'il ne peut pas atteindre certaines pièces). Ici, la tâche conduit à une communication unidirectionnelle. De plus, étant donné que dans une telle tâche, le robot et l'humain ont des connaissances équivalentes sur l'environnement, il peut être difficile de concevoir des situations où une divergence de croyances apparaît et donc une prise de perspective serait nécessaire.

Réduire une tâche quotidienne pour la transformer en une tâche jouet autour d'une table peut réduire la complexité de la tâche et permettre une reproductibilité facile. De plus, cela permet au robot et à l'humain de travailler à proximité l'un de l'autre, avec des robots plus petits par exemple. Avec la version jouet de la tâche d'assemblage présentée dans [Brawer 2018], l'humain est plus impliqué dans la tâche. Cette tâche demande au robot de prélever des morceaux et de les tenir pour les aider à assembler une chaise. Même si la communication est unidirectionnelle, on pourrait imaginer d'inverser les rôles pour tester différentes capacités. De

[9] Bien sûr, nous pouvons trouver des cas délicats où cela pourrait aider mais ils ne reflètent pas les situations courantes.

[10] Cette tâche n'est pas explicitement destinée à être répliquée par la communauté.

plus, la communication implique des objets se référant à l'utilisation de diverses caractéristiques visuelles sur les entités. Même si les deux agents ont les mêmes connaissances sur l'environnement, la communication est fondée sur l'état actuel du monde. Dans cette tâche, aucune décision ne doit être prise par le robot mais encore une fois, l'inversion des rôles pourrait ouvrir d'autres défis.

Pour focaliser les études autour de la prise de perspective et de la gestion des croyances, le scénario de Sally et Anne, issu d'un test de psychologie, a été étudié en robotique [Milliez 2014]. Dans ce scénario, le robot est un observateur d'une situation où deux humains vont et viennent dand une pièce, et déplacent un objet d'une boîte à une autre. Puisqu'un humain est dans une autre pièce quand l'autre agit, une divergence de croyance apparaît entre les deux humains et le robot doit le comprendre. Alors que la tâche met en évidence la gestion des croyances, elle est d'abord limitée en ce qui concerne la prise de perspective puisque la présence humaine ou non pourrait être suffisante pour estimer les croyances humaines[11]. De plus, les humains n'agissent pas avec le robot puisqu'ils ne sont qu'observateurs de la scène. De plus, aucun objectif n'est formulé et les humain n'interagissent pas les uns avec les autres. Enfin, aucune communication n'est nécessaire dans la tâche. Le scénario est ainsi centré sur l'analyse d'une situation.

Pour terminer cette thèse, dans ce chapitre, nous présentons une intégration de l'algorithme de génération d'expressions de référence dans une architecture robotique complète dédiée aux applications d'Interaction Humain-Robot. De plus, avec cette nouvelle architecture, nous montrons comment nous plaçons la connaissance au centre de l'architecture, utilisée par tous les composants. De plus, dans ce chapitre, nous introduisons une nouvelle tâche, inspirée d'expériences de psychologie, pour évaluer l'architecture cognitive de l'interaction Humain-Robot. Partant de cette tâche et eu égard à l'architecture mise en œuvre, nous terminons ce chapitre par un ensemble de défis que nous pensons être intéressants pour la communauté.

Travaux futurs

Pour terminer cette thèse, nous présentons quelques travaux futurs potentiels, en nous concentrant sur deux thèmes principaux étudiés au cours de ces trois années : le besoin de connaissances pour l'HRI et le REG.

Besoin de connaissances pour l'IHR

Dans l'introduction de cette thèse, nous avons vu le système de mémoire principal que nous estimons qu'un humain possède. Dans cette thèse, nous nous sommes concentrés sur la mémoire sémantique, pour doter le robot de la sémantique sur les éléments de son environnement. Si nous avons géré la mémoire procédurale qui pourrait être assimilée au domaine de la planification des tâches, nous n'avons pas

[11] Quand les deux humains sont dans la pièce, ils ont la même perception de la scène mais avoir des croyances différentes sur les objets cachés. Une prise de perspective serait nécessaire si les humains pouvaient se pencher sur les boîtes pour vérifier ce qu'il y a à l'intérieur.

géré la mémoire épisodique, même si nous avons commencé à en dessiner certains aspects à travers la description des activités passées. La capacité de mémorisation est cependant un aspect clé pour HRI. Cela permet à un robot d'en parler en tant que narrateur, comme nous l'avons vu pour se référer à des entités, d'apprendre des politiques d'action, ou d'apprendre les préférences des autres. L'un des systèmes actuels les plus avancés gérant ces connaissances est aujourd'hui le système openEASE [Beetz 2015b]. Mais comment l'étendre à l'application HRI ? Comment représenter l'estimation des expériences de l'autre ? Comment faire la différence entre une expérience réelle et une situation passée que d'autres nous disent avoir vécue ?

Enfin, pour moi, la question la plus complexe et encore ouverte, serait : comment faire une distinction claire entre le système sémantique et l'épisodique ? Cependant, poser cette question conduit à une autre : avons-nous besoin d'une distinction claire ? D'une part, ce sont deux types de savoirs différents mais d'autre part, l'un n'est rien sans l'autre. Nous avons besoin de sens pour comprendre les souvenirs (au sens d'expériences passées) ce qui signifie que nous avons besoin d'un lien entre eux. A vouloir créer deux systèmes distincts, à mon avis, on arrive à deux solutions, qui ne me semblent pas adaptées. Premièrement, nous pourrions garder dans le système sémantique une trace des expériences avec leurs significations (par exemple *cut_7* est une action de coupe) tandis que le système épisodique ne les ordonnerait que temporellement (par exemple *cut_7* tient entre t1 et t2). La seconde solution serait de ne garder aucune trace de l'expérience dans le système sémantique mais nécessiterait de garder une partie du sens dans l'épisodique (par exemple *cut_7*, étant une action coupante, tient entre t1 et t2).

Génération d'expressions de référence

Au cours de la dernière année de cette thèse, nous nous intéressons au problème de génération d'expressions de référence. Ce problème est intéressant car il nécessite une représentation riche d'un environnement, une exploration fine de celle-ci, la communication verbale des connaissances inhérentes, la prise en compte du problème au niveau de la tâche, et la prise en compte du partenaire. Même si nous avons proposé quatre contributions autour de ce sujet, nous pensons que nombre de défis restent à explorer. Dans la suite de nos travaux, nous pourrions d'abord utiliser l'algorithme REG basé sur les activités passées lors de la planification des tâches comme nous l'avons fait pour la version originale. De cette façon, le robot pourrait planifier la communication en fonction des futures activités passées.

Une deuxième piste d'étude pourrait porter sur l'utilisation de relations spatiales comme la gauche et la droite. Au cours de cette thèse, nous avons essayé d'annuler de telles relations, même si notre algorithme pouvait les gérer. La raison en est que de telles relations dépendent de la perspective et donc du cadre de référence, c'est-à-dire du cadre dans lequel les relations sont calculées. Par exemple, nous pourrions dire "le stylo à votre gauche" où le cadre de référence est l'auditeur. On pourrait aussi dire « ma voiture est celle à gauche de la voiture

rouge » où le référentiel est la voiture rouge. Cependant, dire “regarde l'écureuil à droite de l'arbre” est plus délicat. A la différence d'une voiture, un arbre n'a pas d'orientation. Vous et l'auditeur étant au même endroit et ayant la même perspective, cette phrase ne conduit à aucune ambiguïté car vous et l'auditeur pourriez être le cadre de référence. Cependant, si vous êtes un de chaque côté de l'arbre, le cadre de référence utilisé n'est pas clair et l'expression de référence résultante est ambiguë. Avec ces exemples, nous voyons que la perspective et les caractéristiques des entités impliquées doivent être utilisées. Certains travaux commencent à l'étudier [Kelleher 2006, dos Santos Silva 2015]. Cependant, ils sont principalement basés sur l'algorithm incrémental qui n'est pas adapté et l'utilisation d'une ontologie pourrait aider à généraliser le processus.

Enfin, une autre piste d'étude intéressante serait la référence aux ensembles. Une approche prenant en charge cette fonctionnalité prendrait en charge deux nouveaux types d'expression de référence. Premièrement, on pourrait dire «donnez-moi les vis qui sont dans le bac rouge» ou «donnez-moi les stylos rouges» où l'on demande un ensemble d'entités. Néanmoins, nous avons vu que le REG est une sorte de processus récursif dans le sens où pour faire référence à une entité donnée, nous pouvons avoir besoin de générer une expression de référence à une autre entité. Avec des expressions de référence supportant des références à des ensembles, on pourrait ainsi générer des phrases comme “donne-moi la bouteille qui est à côté des feuilles de papier”. Ici l'entité cible n'est pas un ensemble mais elle possède une relation avec un ensemble. Là où les quelques méthodes existantes possèdent la représentation de l'ensemble dans la base de connaissances de manière statique [Fang 2013], il serait plus intéressant de les calculer dynamiquement en fonction de la situation actuelle et de l'entité cible. Un algorithme et la représentation des connaissances sous-jacente pour prendre en charge une telle caractéristique pourraient être très intéressante, et encore plus si nous parvenions à prendre en compte les références spatiales en même temps.

Bibliography

[Allen 1983] James F Allen. *Maintaining knowledge about temporal intervals.* Communications of the ACM, vol. 26, no. 11, pages 832–843, 1983. (Cited in page 126.)

[Allen 2000] Gary L Allen. *Principles and practices for communicating route knowledge.* Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition, vol. 14, no. 4, pages 333–359, 2000. (Cited in pages 53, 69, and 204.)

[Andresen 2016] Erik Andresen, David Haensel, Mohcine Chraibi and Armin Seyfried. *Wayfinding and cognitive maps for pedestrian models.* In Traffic and Granular Flow'15, pages 249–256. Springer, 2016. (Cited in page 57.)

[Atkinson 1966] Richard C Atkinson and Richard M Shiffrin. *Some two-process models for memory(Two-process models for memory and learning).* 1966. (Cited in page 8.)

[Baader 2003] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, Daniele Nardi*et al.* The description logic handbook: Theory, implementation and applications. Cambridge university press, 2003. (Cited in page 22.)

[Baddeley 1986] Alan Baddeley, Robert Logie, S Bressi, S Della Sala and H Spinnler. *Dementia and working memory.* The Quarterly Journal of Experimental Psychology Section A, vol. 38, no. 4, pages 603–618, 1986. (Cited in page 8.)

[Bálint-Benczédi 2018] Ferenc Bálint-Benczédi and Michael Beetz. *Variations on a Theme:"It's a Poor Sort of Memory that Only Works Backwards".* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 8390–8396. IEEE, 2018. (Cited in pages 16 and 221.)

[Baron-Cohen 1985] Simon Baron-Cohen, Alan M Leslie and Uta Frith. *Does the autistic child have a "theory of mind"?* Cognition, vol. 21, no. 1, pages 37–46, 1985. (Cited in page 189.)

[Bauer 2009] Andrea Bauer, Klaas Klasing, Tingting Xu, Stefan Sosnowski, Georgios Lidoris, Quirin Muhlbauer, Tinguang Zhang, Florian Rohrmuller, Dirk Wollherr, Kolja Kuhnlenz*et al. The autonomous city explorer project.* In IEEE International Conference on Robotics and Automation (ICRA), pages 1595–1596. IEEE, 2009. (Cited in page 162.)

[Beetz 2012] Michael Beetz, Dominik Jain, Lorenz Mosenlechner, Moritz Tenorth, Lars Kunze, Nico Blodow and Dejan Pangercic. *Cognition-enabled autonomous robot control for the realization of home chore task intelligence.* Proceedings of the IEEE, vol. 100, no. 8, pages 2454–2471, 2012. (Cited in pages 17 and 222.)

[Beetz 2015a] Michael Beetz, Ferenc Bálint-Benczédi, Nico Blodow, Daniel Nyga, Thiemo Wiedemeyer and Zoltán-Csaba Marton. *Robosherlock: Unstructured information processing for robot perception.* In IEEE International Conference on Robotics and Automation (ICRA), pages 1549–1556, 2015. (Cited in pages 97 and 189.)

[Beetz 2015b] Michael Beetz, Moritz Tenorth and Jan Winkler. *Open-ease.* In IEEE International Conference on Robotics and Automation (ICRA), pages 1983–1990. IEEE, 2015. (Cited in pages 206 and 242.)

[Beetz 2018] Michael Beetz, Daniel Beßler, Andrei Haidu, Mihai Pomarlan, Asil Kaan Bozcuoğlu and Georg Bartels. *Know rob 2.0—a 2nd generation knowledge processing framework for cognition-enabled robotic agents.* In IEEE International Conference on Robotics and Automation (ICRA), pages 512–519. IEEE, 2018. (Cited in page 188.)

[Belhassein 2017] Kathleen Belhassein, Aurélie Clodic, Hélene Cochet, Marketta Niemelä, Päivi Heikkilä, Hanna Lammi and Antti Tammela. *Human-human guidance study.* Technical Report, 2017. (Cited in pages 69, 163, and 164.)

[Belke 2002] Eva Belke and Antje S Meyer. *Tracking the time course of multidimensional stimulus discrimination: Analyses of viewing patterns and processing times during "same"-"different "decisions.* European Journal of Cognitive Psychology, vol. 14, no. 2, pages 237–266, 2002. (Cited in pages 77 and 134.)

[Beßler 2020] Daniel Beßler, Robert Porzel, Mihai Pomarlan, Abhijit Vyas, Sebastian Höffner, Michael Beetz, Rainer Malaka and John Bateman. *Foundations of the Socio-physical Model of Activities (SOMA) for Autonomous Robotic Agents.* arXiv preprint arXiv:2011.11972, 2020. (Cited in pages 18, 123, 139, and 224.)

[Borst 1999] Willem Nico Borst. *Construction of engineering ontologies for knowledge sharing and reuse.* 1999. (Cited in pages 17 and 222.)

[Bovy 2012] Piet H Bovy and Eliahu Stern. Route choice: Wayfinding in transport networks: Wayfinding in transport networks, volume 9. Springer Science & Business Media, 2012. (Cited in page 63.)

[Brachman 1979] Ronald J Brachman. *On the epistemological status of semantic networks.* In Associative networks, pages 3–50. Elsevier, 1979. (Cited in page 143.)

[Brawer 2018] Jake Brawer, Olivier Mangin, Alessandro Roncone, Sarah Widder and Brian Scassellati. *Situated Human–Robot Collaboration: predicting intent from grounded natural language.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 827–833. IEEE, 2018. (Cited in pages 102, 179, 232, and 240.)

[Broekstra 2002] Jeen Broekstra, Arjohn Kampman and Frank Van Harmelen. *Sesame: A generic architecture for storing and querying rdf and rdf schema.* In International semantic web conference, pages 54–68. Springer, 2002. (Cited in pages 18, 20, 224, and 225.)

[Bruno 2017] Barbara Bruno, Nak Young Chong, Hiroko Kamide, Sanjeev Kanoria, Jaeryoung Lee, Yuto Lim, Amit Kumar Pandey, Chris Papadopoulos, Irena Papadopoulos, Federico Pecora*et al.* *The CARESSES EU-Japan project: making assistive robots culturally competent.* In Italian Forum of Ambient Assisted Living, pages 151–169. Springer, 2017. (Cited in pages 18 and 224.)

[Bui-Xuan 2008] Binh-Minh Bui-Xuan. *Tree-representation of set families in graph decompositions and efficient algorithms.* PhD thesis, Université Montpellier II-Sciences et Techniques du Languedoc, 2008. (Cited in page 150.)

[Buisan 2020a] Guilhem Buisan, Guillaume Sarthou and Rachid Alami. *Human aware task planning using verbal communication feasibility and costs.* In International Conference on Social Robotics (ICSR), pages 554–565. Springer, 2020. (Cited in pages 101, 121, and 236.)

[Buisan 2020b] Guilhem Buisan, Guillaume Sarthou, Arthur Bit-Monnot, Aurélie Clodic and Rachid Alami. *Efficient, situated and ontology based referring expression generation for human-robot collaboration.* In 29th International Conference on Robot and Human Interactive Communication (RO-MAN), pages 349–356. IEEE, 2020. (Cited in pages 73, 79, 121, and 236.)

[Buisan 2021] Guilhem Buisan and Rachid Alami. *A Human-Aware Task Planner Explicitly Reasoning About Human and Robot Decision, Action and Reaction.* In Companion of the ACM/IEEE International Conference on Human-Robot Interaction, pages 544–548, 2021. (Cited in page 191.)

[Burgard 1999] Wolfram Burgard, Armin B Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner and Sebastian Thrun. *The museum tour-guide robot RHINO.* In Autonome Mobile Systeme 1998, pages 245–254. Springer, 1999. (Cited in pages 52, 162, and 227.)

[Caniot 2020] Maxime Caniot, Vincent Bonnet, Maxime Busy, Thierry Labaye, Michel Besombes, Sebastien Courtois and Edouard Lagrue. *Adapted Pepper.* arXiv preprint arXiv:2009.03648, 2020. (Cited in page 174.)

[Cassell 2007] Justine Cassell, Stefan Kopp, Paul Tepper, Kim Ferriman and Kristina Striegnitz. *Trading spaces: How humans and humanoids use speech and gesture to give directions*, 2007. (Cited in pages 53, 66, 68, 69, and 163.)

[Chen 2017] Yingfeng Chen, Feng Wu, Wei Shuai and Xiaoping Chen. *Robots serve humans in public places—KeJia robot as a shopping assistant.* International Journal of Advanced Robotic Systems, vol. 14, no. 3, pages 1–20, 2017. (Cited in page 162.)

[Chrastil 2014] Elizabeth R Chrastil and William H Warren. *From cognitive maps to cognitive graphs.* PloS one, vol. 9, no. 11, page e112544, 2014. (Cited in pages 52 and 228.)

[Clodic 2006] Aurélie Clodic, Sara Fleury, Rachid Alami, Raja Chatila, Gérard Bailly, Ludovic Brethes, Maxime Cottret, Patrick Danes, Xavier Dollat, Frédéric Elisei*et al. Rackham: An interactive robot-guide.* In IEEE International Symposium on Robot and Human Interactive Communication (ROMAN), pages 502–509. IEEE, 2006. (Cited in pages 52, 162, and 227.)

[Clodic 2009] Aurélie Clodic, Hung Cao, Samir Alili, Vincent Montreuil, Rachid Alami and Raja Chatila. *Shary: a supervision system adapted to human-robot interaction.* In Experimental robotics, pages 229–238. Springer, 2009. (Cited in page 191.)

[Collins 1969] Allan M Collins and M Ross Quillian. *Retrieval time from semantic memory.* Journal of verbal learning and verbal behavior, vol. 8, no. 2, pages 240–247, 1969. (Cited in pages 16 and 221.)

[Collins 1970] Allan M Collins and M Ross Quillian. *Does category size affect categorization time?* Journal of verbal learning and verbal behavior, vol. 9, no. 4, pages 432–438, 1970. (Cited in pages 16 and 221.)

[Dale 1989] Robert Dale. *Cooking up referring expressions.* In 27th Annual Meeting of the association for Computational Linguistics, pages 68–75, 1989. (Cited in pages 77 and 79.)

[Dale 1992] Robert Dale. Generating referring expressions: Constructing descriptions in a domain of objects and processes. The MIT Press, 1992. (Cited in pages 77 and 79.)

[Dale 1995] Robert Dale and Ehud Reiter. *Computational interpretations of the Gricean maxims in the generation of referring expressions.* Cognitive science, vol. 19, no. 2, pages 233–263, 1995. (Cited in pages 77, 90, and 95.)

[Dale 2009] Robert Dale and Jette Viethen. *Referring expression generation through attribute-based heuristics.* In 12th European workshop on natural language generation (ENLG 2009), pages 58–65, 2009. (Cited in page 96.)

[Deliyanni 1979] Amaryllis Deliyanni and Robert A Kowalski. *Logic and semantic networks.* Communications of the ACM, vol. 22, no. 3, pages 184–192, 1979. (Cited in page 143.)

[Denis 1997] Michel Denis. *The description of routes: A cognitive approach to the production of spatial discourse.* Cahiers de psychologie cognitive, vol. 16, no. 4, pages 409–458, 1997. (Cited in pages 53 and 66.)

[Devin 2016] Sandra Devin and Rachid Alami. *An implemented theory of mind to improve human-robot shared plans execution.* In ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 319–326. IEEE, 2016. (Cited in pages 102, 104, 110, 191, and 232.)

[Diab 2020] Mohammed Diab, Mihai Pomerlan, Daniel Bebler and Jan Rosell Gratacòs. *"Knowing from"–An outlook on ontology enabled knowledge transfer for robotic systems.* In Joint Ontology Workshops (JOWO), pages 1–6, 2020. (Cited in page 123.)

[dos Santos Silva 2015] Diego dos Santos Silva and Ivandré Paraboni. *Generating spatial referring expressions in interactive 3D worlds.* Spatial Cognition & Computation, vol. 15, no. 3, pages 186–225, 2015. (Cited in pages 79, 207, and 243.)

[Duboue 2015] Pablo Ariel Duboue, Martin Ariel Domínguez and Paula Estrella. *Evaluating robustness of referring expression generation algorithms.* In 14th Mexican International Conference on Artificial Intelligence (MICAI), pages 21–27. IEEE, 2015. (Cited in pages 120 and 234.)

[Dumontheil 2010] Iroise Dumontheil, Ian A Apperly and Sarah-Jayne Blakemore. *Online usage of theory of mind continues to develop in late adolescence.* Developmental science, vol. 13, no. 2, pages 331–338, 2010. (Cited in page 181.)

[Ebbinghaus 1885] Hermann Ebbinghaus. Über das gedächtnis: untersuchungen zur experimentellen psychologie. Duncker & Humblot, 1885. (Cited in page 7.)

[Erol 1994] Kutluhan Erol, James Hendler and Dana S Nau. *HTN planning: Complexity and expressivity.* In AAAI, volume 94, pages 1123–1128, 1994. (Cited in pages 123 and 124.)

[Fang 2013] Rui Fang, Changsong Liu, Lanbo She and Joyce Chai. *Towards situated dialogue: Revisiting referring expression generation.* In Conference on Empirical Methods in Natural Language Processing, pages 392–402, 2013. (Cited in pages 80, 207, and 243.)

[Fang 2014] Rui Fang, Malcolm Doering and Joyce Chai. *Collaborative models for referring expression generation in situated dialogue.* In AAAI Conference on Artificial Intelligence, volume 28, 2014. (Cited in page 80.)

[Fiala 2005] Mark Fiala. *ARTag, a fiducial marker system using digital techniques.* In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, pages 590–596. IEEE, 2005. (Cited in page 189.)

[Fikes 1971] Richard E Fikes and Nils J Nilsson. *STRIPS: A new approach to the application of theorem proving to problem solving.* Artificial Intelligence, vol. 2, no. 3-4, pages 189–208, 1971. (Cited in page 123.)

[Fiore 2016] Michelangelo Fiore, Aurélie Clodic and Rachid Alami. *On planning and task achievement modalities for human-robot collaboration.* In Experimental Robotics, pages 293–306. Springer, 2016. (Cited in page 191.)

[FitzGerald 2013] Nicholas FitzGerald, Yoav Artzi and Luke Zettlemoyer. *Learning distributions over logical forms for referring expression generation.* In Conference on empirical methods in natural language processing, pages 1914–1925, 2013. (Cited in page 77.)

[Fokoue 2006] Achille Fokoue, Aaron Kershenbaum, Li Ma, Edith Schonberg and Kavitha Srinivas. *The summary abox: Cutting ontologies down to size.* In International Semantic Web Conference, pages 343–356. Springer, 2006. (Cited in page 21.)

[Foster 2016] Mary Ellen Foster, Rachid Alami, Olli Gestranius, Oliver Lemon, Marketta Niemelä, Jean-Marc Odobez and Amit Kumar Pandey. *The MuMMER project: Engaging human-robot interaction in real-world public spaces.* In International Conference on Social Robotics (ICSR), pages 753–763. Springer, 2016. (Cited in pages 162 and 239.)

[Foster 2019] Mary Ellen Foster, Bart Craenen, Amol Deshmukh, Oliver Lemon, Emanuele Bastianelli, Christian Dondrup, Ioannis Papaioannou, Andrea Vanzo, Jean-Marc Odobez, Olivier Canévet, Yuanzhouhan Cao, Weipeng He, Angel Martínez-González, Petr Motlicek, Rémy Siegfried, Rachid Alami, Kathleen Belhassein, Guilhem Buisan, Aurélie Clodic, Amandine Mayima, Yoan Sallami, Guillaume Sarthou, Phani-Teja Singamaneni, Jules Waldhart, Alexandre Mazel, Maxime Caniot, Marketta Niemelä, Päivi Heikkilä, Hanna Lammi and Antti Tammela. *MuMMER: Socially Intelligent Human-Robot Interaction in Public Spaces.* In Artificial Intelligence for Human-Robot Interaction Symposium (AI-HRI), Arlington, VA, United States, 2019. AAAI Fall Symposium Series 2019. (Cited in page 165.)

[Freitas 2014] Artur Freitas, Daniela Schmidt, Felipe Meneguzzi, Renata Vieira and Rafael H Bordini. *Using ontologies as semantic representations of hierarchical task network planning domains.* In Proceedings of WWW, 2014. (Cited in page 124.)

[Gangemi 2008] Aldo Gangemi. *Norms and plans as unification criteria for social collectives.* Autonomous Agents and Multi-Agent Systems, vol. 17, no. 1, pages 70–112, 2008. (Cited in page 143.)

[Gangemi 2013] Aldo Gangemi and Valentina Presutti. *A multi-dimensional comparison of ontology design patterns for representing n-ary relations.* In International Conference on Current Trends in Theory and Practice of Computer Science, pages 86–105. Springer, 2013. (Cited in page 143.)

[Gatt 2018] Albert Gatt and Emiel Krahmer. *Survey of the state of the art in natural language generation: Core tasks, applications and evaluation.* Journal of Artificial Intelligence Research, vol. 61, pages 65–170, 2018. (Cited in page 76.)

[Genesereth 1987] Michael R Genesereth and Nils Robert Nilsson. *Logical foundations of artificial intelligence.* 1987. (Cited in pages 17 and 222.)

[Ghallab 2004] Malik Ghallab, Dana Nau and Paolo Traverso. Automated planning: theory and practice. Elsevier, 2004. (Cited in page 123.)

[Gharbi 2015] Mamoun Gharbi, Raphaël Lallement and Rachid Alami. *Combining symbolic and geometric planning to synthesize human-aware plans: toward more efficient combined search.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6360–6365. IEEE, 2015. (Cited in page 107.)

[Giunti 2019] Marco Giunti, Giuseppe Sergioli, Giuliano Vivanet and Simone Pinna. *Representing n-ary relations in the Semantic Web.* Logic Journal of the IGPL, 2019. (Cited in pages 143 and 147.)

[Graf 1985] Peter Graf and Daniel L Schacter. *Implicit and explicit memory for new associations in normal and amnesic subjects.* Journal of Experimental Psychology: Learning, memory, and cognition, vol. 11, no. 3, page 501, 1985. (Cited in page 8.)

[Grice 1975] Herbert P Grice. *Logic and conversation.* In Speech acts, pages 41–58. Brill, 1975. (Cited in pages 54, 76, and 201.)

[Gross 2009] H-M Gross, H Boehme, Ch Schroeter, Steffen Müller, Alexander König, Erik Einhorn, Ch Martin, Matthias Merten and Andreas Bley. *TOOMAS: interactive shopping guide robots in everyday use-final implementation and experiences from long-term field trials.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2005–2012. IEEE, 2009. (Cited in page 162.)

[Guarino 1995] Nicola Guarino and Pierdaniele Giaretta. *Towards very large knowledge bases—knowledge building and knowledge sharing.* Ontologies and

knowledge bases: towards a terminological clarification, 1995. (Cited in pages 17 and 223.)

[Guarino 2009] Nicola Guarino, Daniel Oberle and Steffen Staab. *What is an ontology?* In Handbook on ontologies, pages 1–17. Springer, 2009. (Cited in pages 17 and 223.)

[Guber 1993] T Guber. *A translational approach to portable ontologies.* Knowledge Acquisition, vol. 5, no. 2, pages 199–229, 1993. (Cited in pages 17 and 222.)

[Guitton 2012] Julien Guitton, Matthieu Warnier and Rachid Alami. *Belief management for hri planning.* In European Conference on Artificial Intelligence-Workshop on Belief change, Non-monotonic reasoning and Conflict Resolution, 2012. (Cited in page 107.)

[Happé 1994] Francesca GE Happé. *An advanced test of theory of mind: Understanding of story characters' thoughts and feelings by able autistic, mentally handicapped, and normal children and adults.* Journal of autism and Developmental disorders, vol. 24, no. 2, pages 129–154, 1994. (Cited in page 181.)

[Hawes 2007] Nick Hawes, Michael Zillich and Jeremy Wyatt. *BALT & CAST: Middleware for cognitive robotics.* In IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pages 998–1003. IEEE, 2007. (Cited in page 188.)

[Heikkilä 2018] Päivi Heikkilä, Hanna Lammi and Kathleen Belhassein. *Where Can I Find a Pharmacy? -Human-Driven Design of a Service Robot's Guidance Behaviour.* In Workshop on Public Space Human-Robot Interaction (PubRob) as part of the International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI), pages 1–2, 2018. (Cited in page 164.)

[Heikkilä 2019] Päivi Heikkilä, Hanna Lammi, Marketta Niemelä, Kathleen Belhassein, Guillaume Sarthou, Antti Tammela, Aurélie Clodic and Rachid Alami. *Should a robot guide like a human? A qualitative four-phase study of a shopping mall robot.* In International Conference on Social Robotics (ICSR), pages 548–557. Springer, 2019. (Cited in page 164.)

[Hemachandra 2014] Sachithra Hemachandra, Matthew R Walter, Stefanie Tellex and Seth Teller. *Learning spatial-semantic representations from natural language descriptions and scene classifications.* In IEEE International Conference on Robotics and Automation (ICRA), pages 2623–2630. IEEE, 2014. (Cited in page 55.)

[Ingrand 2017] Félix Ingrand and Malik Ghallab. *Deliberation for autonomous robots: A survey.* Artificial Intelligence, vol. 247, pages 10–44, 2017. (Cited in page 123.)

[Kanda 2009] Takayuki Kanda, Masahiro Shiomi, Zenta Miyashita, Hiroshi Ishiguro and Norihiro Hagita. *An affective guide robot in a shopping mall.* In ACM/IEEE international Conference on Human-Robot interaction (HRI), pages 173–180, 2009. (Cited in page 163.)

[Kanda 2010] Takayuki Kanda, Masahiro Shiomi, Zenta Miyashita, Hiroshi Ishiguro and Norihiro Hagita. *A communication robot in a shopping mall.* IEEE Transactions on Robotics, vol. 26, no. 5, pages 897–913, 2010. (Cited in page 163.)

[Karp 1972] Richard M Karp. *Reducibility among combinatorial problems.* In Complexity of computer computations, pages 85–103. Springer, 1972. (Cited in page 151.)

[Kelleher 2006] John Kelleher and Geert-Jan M Kruijff. *Incremental generation of spatial referring expressions in situated dialog.* In 21st International Conference on Computational Linguistics, pages 1041–1048, 2006. (Cited in pages 79, 207, and 243.)

[Kendon 1990] Adam Kendon. Conducting interaction: Patterns of behavior in focused encounters, volume 7. CUP Archive, 1990. (Cited in page 171.)

[Keysar 1994] Boaz Keysar. *The illusory transparency of intention: Linguistic perspective taking in text.* Cognitive psychology, vol. 26, no. 2, pages 165–208, 1994. (Cited in page 181.)

[Keysar 1998] Boaz Keysar, Dale J Barr and William S Horton. *The egocentric basis of language use: Insights from a processing approach.* Current directions in psychological science, vol. 7, no. 2, pages 46–49, 1998. (Cited in page 181.)

[Keysar 2000] Boaz Keysar, Dale J Barr, Jennifer A Balin and Jason S Brauner. *Taking perspective in conversation: The role of mutual knowledge in comprehension.* Psychological Science, vol. 11, no. 1, pages 32–38, 2000. (Cited in page 180.)

[Keysar 2002] Boaz Keysar and Dale J Barr. *Self-anchoring in conversation: Why language users do not do what they'should'.* 2002. (Cited in page 181.)

[Keysar 2003] Boaz Keysar, Shuhong Lin and Dale J Barr. *Limits on theory of mind use in adults.* Cognition, vol. 89, no. 1, pages 25–41, 2003. (Cited in page 181.)

[Ko 2011] Ryan KL Ko, Eng Wah Lee and SG Lee. *Business-OWL (BOWL)—A hierarchical task network ontology for dynamic business process decomposition and formulation.* IEEE Transactions on Services Computing, vol. 5, no. 2, pages 246–259, 2011. (Cited in page 124.)

[Koolen 2012] Ruud Koolen, Emiel Krahmer and Mariët Theune. *Learning preferences for referring expression generation: Effects of domain, language and algorithm.* In INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference, pages 3–11, 2012. (Cited in page 134.)

[Krahmer 2003] Emiel Krahmer, Sebastiaan van Erk and André Verleg. *Graph-based generation of referring expressions.* Computational Linguistics, vol. 29, no. 1, pages 53–72, 2003. (Cited in pages 77 and 79.)

[Krahmer 2012] Emiel Krahmer and Kees Van Deemter. *Computational generation of referring expressions: A survey.* Computational Linguistics, vol. 38, no. 1, pages 173–218, 2012. (Cited in pages 75, 77, 78, and 230.)

[Krauss 1977] Robert M Krauss and Sam Glucksberg. *Social and nonsocial speech.* Scientific American, vol. 236, no. 2, pages 100–105, 1977. (Cited in page 180.)

[Krieg-Brückner 2020] Bernd Krieg-Brückner, Mihai Codescu and Mihai Pomarlan. *Modelling Episodes with Generic Ontology Design Patterns.* In Proceedings of the Joint Ontology Workshops co-located with the Bolzano Summer of Knowledge (BOSK), volume 2708 of *CEUR Workshop Proceedings*, 2020. (Cited in page 123.)

[Krötzsch 2013] Markus Krötzsch, Frantisek Simancik and Ian Horrocks. *Description logics.* IEEE Intelligent Systems, vol. 29, no. 1, pages 12–19, 2013. (Cited in page 21.)

[Kuipers 2000] Benjamin Kuipers. *The spatial semantic hierarchy.* Artificial Intelligence, vol. 119, no. 1-2, pages 191–233, 2000. (Cited in page 55.)

[Kuipers 2004] Benjamin Kuipers, Joseph Modayil, Patrick Beeson, Matt MacMahon and Francesco Savelli. *Local metrical and global topological maps in the hybrid spatial semantic hierarchy.* In IEEE International Conference on Robotics and Automation (ICRA), volume 5, pages 4845–4851. IEEE, 2004. (Cited in page 55.)

[Lallement 2014] Raphaël Lallement, Lavindra De Silva and Rachid Alami. *HATP: An HTN Planner for Robotics.* In ICAPS Workshop on Planning and Robotics, 2014. (Cited in pages 107, 123, 128, 191, and 234.)

[Lallement 2016] Raphaël Lallement. *Symbolic and Geometric Planning for teams of Robots and Humans.* PhD thesis, Toulouse, INSA, 2016. (Cited in pages 102 and 232.)

[Lassila 1998] Ora Lassila, Ralph R Swick*et al.* *Resource description framework (RDF) model and syntax specification.* 1998. (Cited in pages 20 and 225.)

[Lefebvre 2003] Sylvain Lefebvre and Samuel Hornus. *Automatic cell-and-portal decomposition.* PhD thesis, INRIA, 2003. (Cited in page 55.)

[Lemaignan 2010] Séverin Lemaignan, Raquel Ros, Lorenz Mösenlechner, Rachid Alami and Michael Beetz. *ORO, a knowledge management platform for cognitive architectures in robotics.* In 2010 IEEE/RSJ International conference on intelligent robots and systems, pages 3548–3553. IEEE, 2010. (Cited in pages 18, 44, and 224.)

[Lemaignan 2011] Séverin Lemaignan, Raquel Ros, Rachid Alami and Michael Beetz. *What are you talking about? grounding dialogue in a perspective-aware robotic architecture.* In 20th International Symposium in Robot and Human Interactive Communication (RO-MAN), pages 107–112. IEEE, 2011. (Cited in page 78.)

[Lemaignan 2017] Séverin Lemaignan, Mathieu Warnier, E Akin Sisbot, Aurélie Clodic and Rachid Alami. *Artificial cognition for social human–robot interaction: An implementation.* Artificial Intelligence, vol. 247, pages 45–69, 2017. (Cited in page 187.)

[Lemaignan 2018] Séverin Lemaignan, Yoan Sallami, Christopher Wallhridge, Aurélic Clodic, Tony Belpaeme and Rachid Alami. *Underworlds: cascading situation assessment for robots.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7750–7757. IEEE, 2018. (Cited in pages 168 and 189.)

[Lenat 1989] Douglas B Lenat and Ramanathan V Guha. Building large knowledge-based systems; representation and inference in the cyc project. Addison-Wesley Longman Publishing Co., Inc., 1989. (Cited in pages 18 and 223.)

[Levin 2011] Mikhail K Levin, Alan Ruttenberg, Anna Maria Masci and Lindsay G Cowell. *Owl-cpp, a C++ library for working with OWL ontologies.* In International Conference on Biomedical Ontology, ICBO, pages 255–257, 2011. (Cited in pages 19 and 224.)

[Li 2016] Shen Li, Rosario Scalise, Henny Admoni, Stephanie Rosenthal and Siddhartha S Srinivasa. *Spatial references and perspective in natural language instructions for collaborative manipulation.* In 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pages 44–51. IEEE, 2016. (Cited in page 97.)

[Li 2017] Shen Li. *Automatically evaluating and generating clear robot explanations.* Master's thesis, 2017. (Cited in pages 77 and 96.)

[Lin 2010] Shuhong Lin, Boaz Keysar and Nicholas Epley. *Reflexively mindblind: Using theory of mind to interpret behavior requires effortful attention.* Journal of Experimental Social Psychology, vol. 46, no. 3, pages 551–556, 2010. (Cited in page 181.)

[Lorenz 2006] Bernhard Lorenz, Hans Jürgen Ohlbach and Edgar-Philipp Stoffel. *A hybrid spatial model for representing indoor environments.* In International

Symposium on Web and Wireless Geographical Information Systems, pages 102–112. Springer, 2006. (Cited in page 55.)

[Mallot 2009] Hanspeter A Mallot and Kai Basten. *Embodied spatial cognition: Biological and artificial systems.* Image and Vision Computing, vol. 27, no. 11, pages 1658–1670, 2009. (Cited in pages 53 and 68.)

[Masolo 2003] C Masolo, S Borgo, A Gangemi, N Guarino, A Oltramari and L Schneider. *Dolce: a descriptive ontology for linguistic and cognitive engineering.* WonderWeb Project, Deliverable D17 v2, vol. 1, pages 75–105, 2003. (Cited in pages 18 and 223.)

[Matsumoto 2012] Takahiro Matsumoto, Satoru Satake, Takayuki Kanda, Michita Imai and Norihiro Hagita. *Do you remember that shop? computational model of spatial memory for shopping companion robots.* In Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction, pages 447–454, 2012. (Cited in page 165.)

[Mavridis 2015] Nikolaos Mavridis. *A review of verbal and non-verbal human–robot interactive communication.* Robotics and Autonomous Systems, vol. 63, pages 22–35, 2015. (Cited in page 104.)

[Mayima 2020] Amandine Mayima, Aurélie Clodic and Rachid Alami. *Toward a Robot Computing an Online Estimation of the Quality of its Interaction with its Human Partner.* In 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 291–298. IEEE, 2020. (Cited in page 192.)

[McNeill 2005] David McNeill. *Gesture, gaze, and ground.* In International workshop on machine learning for multimodal interaction, pages 1–14. Springer, 2005. (Cited in page 171.)

[Mealier 2017] Anne-Laure Mealier, Gregoire Pointeau, Solène Mirliaz, Kenji Ogawa, Mark Finlayson and Peter F Dominey. *Narrative constructions for the organization of self experience: Proof of concept via embodied robotics.* Frontiers in psychology, vol. 8, page 1331, 2017. (Cited in page 123.)

[Miller 1956] George Miller. *Human memory and the storage of information.* IRE Transactions on Information Theory, vol. 2, no. 3, pages 129–137, 1956. (Cited in page 8.)

[Miller 1968] Robert B Miller. *Response time in man-computer conversational transactions.* In Proceedings of the fall joint computer conference, part I, pages 267–277, 1968. (Cited in page 94.)

[Milliez 2014] Grégoire Milliez, Matthieu Warnier, Aurélie Clodic and Rachid Alami. *A framework for endowing an interactive robot with reasoning capabilities about perspective-taking and belief management.* In 23rd inter-

national symposium on robot and human interactive communication (RO-MAN), pages 1103–1109. IEEE, 2014. (Cited in pages 98, 117, 179, 189, and 241.)

[Milliez 2016] Grégoire Milliez, Raphaël Lallement, Michelangelo Fiore and Rachid Alami. *Using human knowledge awareness to adapt collaborative plan generation, explanation and monitoring.* In ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 43–50. IEEE, 2016. (Cited in page 123.)

[Montello 1993] Daniel R Montello. *Scale and multiple psychologies of space.* In European conference on spatial information theory, pages 312–321. Springer, 1993. (Cited in page 53.)

[Morales Saiki 2011] Luis Yoichi Morales Saiki, Satoru Satake, Takayuki Kanda and Norihiro Hagita. *Modeling environments from a route perspective.* In ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 441–448, 2011. (Cited in pages 52 and 227.)

[Morales 2015] Yoichi Morales, Satoru Satake, Takayuki Kanda and Norihiro Hagita. *Building a model of the environment from a route perspective for human–robot interaction.* International Journal of Social Robotics, vol. 7, no. 2, pages 165–181, 2015. (Cited in pages 54 and 55.)

[Nikolaidis 2018] Stefanos Nikolaidis, Minae Kwon, Jodi Forlizzi and Siddhartha Srinivasa. *Planning with verbal communication for human-robot collaboration.* ACM Transactions on Human-Robot Interaction (THRI), vol. 7, no. 3, pages 1–21, 2018. (Cited in page 104.)

[Niles 2001] Ian Niles and Adam Pease. *Towards a standard upper ontology.* In Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001, pages 2–9, 2001. (Cited in pages 18 and 223.)

[Nothegger 2004] Clemens Nothegger, Stephan Winter and Martin Raubal. *Selection of salient features for route directions.* Spatial cognition and computation, vol. 4, no. 2, pages 113–136, 2004. (Cited in page 54.)

[Oberlander 1991] Jon Oberlander and Robert Dale. *Generating expressions referring to eventualities.* In 13th Annual Conference of the Cognitive Science Society, pages 67–72. Erlbaum Hillsdale, NJ, 1991. (Cited in page 122.)

[Okuno 2009] Yusuke Okuno, Takayuki Kanda, Michita Imai, Hiroshi Ishiguro and Norihiro Hagita. *Providing route directions: design of robot's utterance, gesture, and timing.* In ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 53–60. IEEE, 2009. (Cited in pages 52, 163, 165, and 227.)

[Olivares-Alarcos 2019] Alberto Olivares-Alarcos, Daniel Beßler, Alaa Khamis, Paulo Goncalves, Maki K Habib, Julita Bermejo-Alonso, Marcos Barreto, Mohammed Diab, Jan Rosell, Joao Quintas*et al. A review and comparison of ontology-based approaches to robot autonomy.* 2019. (Cited in pages 18 and 224.)

[Pacherie 2012] Elisabeth Pacherie. *The Phenomenology of Joint Action: Self-Agency vs. Joint-Agency.* In Axel Seemann, editor, Joint Attention: New Developments, pages 343–389. MIT Press, 2012. (Cited in pages 19, 163, and 225.)

[Panchbhai 2020] Anand Panchbhai, Tommaso Soru and Edgard Marx. *Exploring Sequence-to-Sequence Models for SPARQL Pattern Composition.* In Iberoamerican Knowledge Graphs and Semantic Web Conference, pages 158–165. Springer, 2020. (Cited in page 193.)

[Papaioannou 2018] Ioannis Papaioannou, Christian Dondrup and Oliver Lemon. *Human-robot interaction requires more than slot filling-multi-threaded dialogue for collaborative tasks and social conversation.* AI-MHRI, 2018. (Cited in pages 162 and 239.)

[Paulius 2019] David Paulius, Kelvin Sheng Pei Dong and Yu Sun. *Functional object-oriented network: Considering robot's capability in human-robot collaboration.* arXiv preprint arXiv:1905.00502, 2019. (Cited in pages 16 and 222.)

[Petit 2016] Maxime Petit, Grégoire Pointeau and Peter Ford Dominey. *Reasoning based on consolidated real world experience acquired by a humanoid robot.* Interaction Studies, vol. 17, no. 2, pages 248–278, 2016. (Cited in page 123.)

[Petrick 2012] Ronald PA Petrick, Mary Ellen Foster and Amy Isard. *Social state recognition and knowledge-level planning for human-robot interaction in a bartender domain.* In AAAI Workshop on Grounding Language for Physical Systems, Toronto, ON, Canada, July, 2012. (Cited in pages 179 and 240.)

[Prasad 2020] Pranav Krishna Prasad and Wolfgang Ertel. *Knowledge Acquisition and Reasoning Systems for Service Robots: A Short Review of the State of the Art.* In International Conference on Robotics and Automation Engineering (ICRAE), pages 36–45. IEEE, 2020. (Cited in pages 16 and 221.)

[Reiter 1992] Ehud Reiter and Robert Dale. *A fast algorithm for the generation of referring expressions.* In 15th International Conference on Computational Linguistics, volume 1, 1992. (Cited in pages 77 and 79.)

[Reiter 2000] Ehud Reiter and Robert Dale. Building natural language generation systems. Cambridge university press, 2000. (Cited in pages 76, 204, and 231.)

[Roediger 1996] Henry L Roediger and Melissa J Guynn. *Retrieval processes.* Memory, pages 197–236, 1996. (Cited in page 7.)

[Roncone 2017] Alessandro Roncone, Olivier Mangin and Brian Scassellati. *Transparent role assignment and task allocation in human robot collaboration.* In IEEE International Conference on Robotics and Automation (ICRA), pages 1014–1021. IEEE, 2017. (Cited in pages 102, 105, and 232.)

[Ros 2010] Raquel Ros, Séverin Lemaignan, E Akin Sisbot, Rachid Alami, Jasmin Steinwender, Katharina Hamann and Felix Warneken. *Which one? grounding the referent based on efficient human-robot interaction.* In 19th International Symposium in Robot and Human Interactive Communication (RO-MAN), pages 570–575. IEEE, 2010. (Cited in pages 78, 79, and 95.)

[Rubio-Fernández 2017] Paula Rubio-Fernández. *The director task: A test of Theory-of-Mind use or selective attention?* Psychonomic bulletin & review, vol. 24, no. 4, pages 1121–1128, 2017. (Cited in page 181.)

[Santiesteban 2012] Idalmis Santiesteban, Sarah White, Jennifer Cook, Sam J Gilbert, Cecilia Heyes and Geoffrey Bird. *Training social cognition: from imitation to theory of mind.* Cognition, vol. 122, no. 2, pages 228–235, 2012. (Cited in page 181.)

[Sarthou 2019a] Guillaume Sarthou, Rachid Alami and Aurélie Clodic. *Semantic Spatial Representation: a unique representation of an environment based on an ontology for robotic applications.* In Combined Workshop on Spatial Language Understanding (SpLU) and Grounded Communication for Robotics (RoboNLP), 2019. (Cited in page 51.)

[Sarthou 2019b] Guillaume Sarthou, Aurélie Clodic and Rachid Alami. *Ontologenius: A long-term semantic memory for robotic agents.* In 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 1–8. IEEE, 2019. (Cited in pages 16, 94, and 98.)

[Sarthou 2021a] Guillaume Sarthou, Mayima Amandine, Buisan Guilhem, Belhassein Kathleen and Aurélie Clodic. *The Director Task: a Psychology-Inspired Task to Assess Cognitive and Interactive Robot Architectures.* In IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 1–8. IEEE, 2021. (Cited in page 177.)

[Sarthou 2021b] Guillaume Sarthou, Guilhem Buisan, Aurélie Clodic and Rachid Alami. *Extending Referring Expression Generation through shared knowledge about past Human-Robot collaborative activity.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1–8. IEEE, 2021. (Cited in page 119.)

[Satake 2015a] Satoru Satake, Kotaro Hayashi, Keita Nakatani and Takayuki Kanda. *Field trial of an information-providing robot in a shopping mall.*

In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1832–1839. IEEE, 2015. (Cited in pages 55 and 165.)

[Satake 2015b] Satoru Satake, Keita Nakatani, Kotaro Hayashi, Takyuki Kanda and Michita Imai. *What should we know to develop an information robot?* PeerJ Computer Science, vol. 1, page 8, 2015. (Cited in pages 52, 163, 165, 178, 228, and 240.)

[Saxena 2014] Ashutosh Saxena, Ashesh Jain, Ozan Sener, Aditya Jami, Dipendra K Misra and Hema S Koppula. *Robobrain: Large-scale knowledge engine for robots.* arXiv preprint arXiv:1412.0691, 2014. (Cited in pages 18 and 224.)

[Schaefer 2017] Kristin E Schaefer, Edward R Straub, Jessie YC Chen, Joe Putney and Arthur W Evans III. *Communicating intent to develop shared situation awareness and engender trust in human-agent teams.* Cognitive Systems Research, vol. 46, pages 26–39, 2017. (Cited in page 104.)

[Schlenoff 2015] Craig Schlenoff, E Prestes, PJ Sequeira Gonçalves, Mathieu Abel, Yacine Amirat, S Balakirsky, ME Barreto, JL Carbonera, A Chibani, S Rama Fiorini*et al. Ieee standard ontologies for robotics and automation.* 2015. (Cited in pages 18 and 223.)

[Sebastiani 2017] Eugenio Sebastiani, Raphaël Lallement, Rachid Alami and Luca Iocchi. *Dealing with on-line human-robot negotiations in hierarchical agent-based task planner.* In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS), volume 27, 2017. (Cited in page 104.)

[Shah 2011] Julie Shah, James Wiken, Brian Williams and Cynthia Breazeal. *Improved human-robot team performance using chaski, a human-inspired plan execution system.* In 6th international conference on Human-robot interaction (HRI), pages 29–36, 2011. (Cited in page 104.)

[Siegwart 2003] Roland Siegwart, Kai O Arras, Samir Bouabdallah, Daniel Burnier, Gilles Froidevaux, Xavier Greppin, Björn Jensen, Antoine Lorotte, Laetitia Mayor, Mathieu Meisser*et al. Robox at Expo. 02: A large-scale installation of personal robots.* Robotics and Autonomous Systems, vol. 42, no. 3-4, pages 203–222, 2003. (Cited in pages 52, 162, and 227.)

[Singamaneni 2020] Phani-Teja Singamaneni and Rachid Alami. *HATEB-2: Reactive Planning and Decision making in Human-Robot Co-navigation.* In EEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pages 179–186. IEEE, 2020. (Cited in page 173.)

[Singh 2020] Avinash Singh, Neha Baranwal and Kai-Florian Richter. *A Fuzzy Inference System for a Visually Grounded Robot State of Mind.* In European

Conference on Artificial Intelligence (ECAI), Including Conference on Prestigious Applications of Artificial Intelligence (PAIS), pages 2402–2409. IOS Press, 2020. (Cited in pages 16 and 222.)

[Sirin 2007] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur and Yarden Katz. *Pellet: A practical owl-dl reasoner.* Journal of Web Semantics, vol. 5, no. 2, pages 51–53, 2007. (Cited in pages 18, 44, and 224.)

[Sowa 2014] John F Sowa. Principles of semantic networks: Explorations in the representation of knowledge. Morgan Kaufmann, 2014. (Cited in page 143.)

[Squire 1982] Larry R Squire, Neal J Cohen and LS Cermak. *Remote memory, retrograde amnesia, and the neuropsychology of memory.* Human memory and amnesia, pages 275–303, 1982. (Cited in page 8.)

[Stenmark 2013] Maj Stenmark and Jacek Malec. *Knowledge-Based Industrial Robotics.* In SCAI, pages 265–274, 2013. (Cited in pages 18 and 224.)

[Studer 1998] Rudi Studer, V Richard Benjamins and Dieter Fensel. *Knowledge engineering: Principles and methods.* Data & knowledge engineering, vol. 25, no. 1-2, pages 161–197, 1998. (Cited in pages 17 and 222.)

[Sun 2019] Xiaolei Sun, Yu Zhang and Jing Chen. *RTPO: a domain knowledge base for robot task planning.* Electronics, vol. 8, no. 10, page 1105, 2019. (Cited in page 123.)

[Taylor 1992] Holly A Taylor and Barbara Tversky. *Spatial mental models derived from survey and route descriptions.* Journal of Memory and language, vol. 31, no. 2, pages 261–292, 1992. (Cited in page 54.)

[Taylor 1996] Holly A Taylor and Barbara Tversky. *Perspective in spatial descriptions.* Journal of memory and language, vol. 35, no. 3, pages 371–391, 1996. (Cited in page 55.)

[Tellex 2014] Stefanie Tellex, Ross Knepper, Adrian Li, Daniela Rus and Nicholas Roy. *Asking for help using inverse semantics.* 2014. (Cited in pages 102, 104, 105, 179, 232, and 240.)

[Tellex 2020] Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit and Cynthia Matuszek. *Robots that use language.* Annual Review of Control, Robotics, and Autonomous Systems, vol. 3, pages 25–55, 2020. (Cited in pages 102 and 231.)

[Tenorth 2013] Moritz Tenorth and Michael Beetz. *KnowRob: A knowledge processing infrastructure for cognition-enabled robots.* The International Journal of Robotics Research, vol. 32, no. 5, pages 566–590, 2013. (Cited in pages 18, 40, and 224.)

[Tenorth 2017] Moritz Tenorth and Michael Beetz. *Representations for robot knowledge in the KnowRob framework.* Artificial Intelligence, vol. 247, pages 151–169, 2017. (Cited in pages 16, 40, 41, and 222.)

[Thrun 2007] Sebastian Thrun. *Simultaneous localization and mapping.* In Robotics and cognitive approaches to spatial mapping, pages 13–41. Springer, 2007. (Cited in pages 52 and 227.)

[Triebel 2016] Rudolph Triebel, Kai Arras, Rachid Alami, Lucas Beyer, Stefan Breuers, Raja Chatila, Mohamed Chetouani, Daniel Cremers, Vanessa Evers, Michelangelo Fiore*et al.* *Spencer: A socially aware service robot for passenger guidance and help in busy airports.* In Field and service robotics, pages 607–622. Springer, 2016. (Cited in pages 52, 162, and 227.)

[Tsarkov 2006] Dmitry Tsarkov and Ian Horrocks. *FaCT++ description logic reasoner: System description.* In International joint conference on automated reasoning, pages 292–297. Springer, 2006. (Cited in pages 19 and 224.)

[Tulving 1995] Endel Tulving. *Organization of memory: Quo vadis.* The cognitive neurosciences, 1995. (Cited in page 9.)

[Tversky 1998] Barbara Tversky and Paul U Lee. *How space structures language.* In Spatial cognition, pages 157–175. Springer, 1998. (Cited in page 53.)

[Tversky 1999] Barbara Tversky and Paul U Lee. *Pictorial and verbal tools for conveying routes.* In International Conference on Spatial Information Theory, pages 51–64. Springer, 1999. (Cited in pages 53, 54, and 66.)

[Umbrico 2020] Alessandro Umbrico, Andrea Orlandini and Amedeo Cesta. *An Ontology for Human-Robot Collaboration.* Procedia CIRP, vol. 93, pages 1097–1102, 2020. (Cited in page 124.)

[Unhelkar 2017] Vaibhav V Unhelkar, X Jessie Yang and Julie A Shah. *Challenges for communication decision-making in sequential human-robot collaborative tasks.* In Workshop on Mathematical Models, Algorithms, and Human-Robot Interaction at R: SS, 2017. (Cited in page 105.)

[Unhelkar 2020] Vaibhav V Unhelkar, Shen Li and Julie A Shah. *Decision-making for bidirectional communication in sequential human-robot collaborative tasks.* In ACM/IEEE International Conference on Human-Robot Interaction, pages 329–341, 2020. (Cited in pages 105 and 110.)

[Viethen 2013] Jette Viethen, Margaret Mitchell and Emiel Krahmer. *Graphs and spatial relations in the generation of referring expressions.* In 14th European Workshop on Natural Language Generation, pages 72–81, 2013. (Cited in pages 77, 79, 95, and 96.)

[W3C 2006] W3C. *Defining n-ary relations on the Semantic Web.* W3C Working Group Note, 2006. (Cited in page 144.)

[Waldhart 2019] Jules Waldhart, Aurélie Clodic and Rachid Alami. *Reasoning on Shared Visual Perspective to Improve Route Directions.* In IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 1–8. IEEE, 2019. (Cited in page 171.)

[Wallbridge 2019] Christopher D Wallbridge, Séverin Lemaignan, Emmanuel Senft and Tony Belpaeme. *Generating Spatial Referring Expressions in a Social Robot: Dynamic vs. Non-ambiguous.* Frontiers in Robotics and AI, vol. 6, page 67, 2019. (Cited in page 80.)

[Welty 2006] Chris Welty, Richard Fikes and Selene Makarios. *A reusable ontology for fluents in OWL.* In FOIS, volume 150, pages 226–236, 2006. (Cited in page 143.)

[Wielemaker 2003] Jan Wielemaker, A Th Schreiber and BJ Wielinga. *Prolog-based infrastructure for RDF: performance and scalability.* 2003. (Cited in pages 18, 40, and 224.)

[Wielemaker 2012] Jan Wielemaker, Tom Schrijvers, Markus Triska and Torbjörn Lager. *Swi-prolog.* Theory and Practice of Logic Programming, vol. 12, no. 1-2, pages 67–96, 2012. (Cited in page 40.)

[Williams 2016] Tom Williams and Matthias Scheutz. *A framework for resolving open-world referential expressions in distributed heterogeneous knowledge bases.* In AAAI Conference on Artificial Intelligence, volume 30, 2016. (Cited in page 78.)

[Williams 2017] Tom Williams and Matthias Scheutz. *Referring expression generation under uncertainty: Algorithm and evaluation framework.* In 10th International Conference on Natural Language Generation, pages 75–84, 2017. (Cited in pages 78 and 79.)

[Williams 2019] Tom Williams, Fereshta Yazdani, Prasanth Suresh, Matthias Scheutz and Michael Beetz. *Dempster-shafer theoretic resolution of referential ambiguity.* Autonomous Robots, vol. 43, no. 2, pages 389–414, 2019. (Cited in page 78.)

[Williams 2020] Tom Williams, Torin Johnson, Will Culpepper and Kellyn Larson. *Toward Forgetting-Sensitive Referring Expression Generationfor Integrated Robot Architectures.* arXiv preprint arXiv:2007.08672, 2020. (Cited in pages 80, 122, and 134.)

[Wiriyathammabhum 2019] Peratham Wiriyathammabhum, Abhinav Shrivastava, Vlad Morariu and Larry Davis. *Referring to Objects in Videos Using Spatio-Temporal Identifying Descriptions.* In Second Workshop on Shortcomings in Vision and Language, pages 14–25, 2019. (Cited in page 122.)

[Wooldridge 1999] Michael Wooldridge. *Intelligent agents.* Multiagent systems, vol. 6, 1999. (Cited in page 164.)

[Yamakata 2004] Yoko Yamakata, Tatsuya Kawahara, Hiroshi G Okuno and Michihiko Minoh. *Belief network based disambiguation of object reference in spoken dialogue system.* Transactions of the Japanese Society for Artificial Intelligence, vol. 19, no. 1, pages 47–56, 2004. (Cited in pages 77 and 79.)

[Zender 2008] Hendrik Zender, O Martínez Mozos, Patric Jensfelt, G-JM Kruijff and Wolfram Burgard. *Conceptual spatial representations for indoor mobile robots.* Robotics and Autonomous Systems, vol. 56, no. 6, pages 493–502, 2008. (Cited in pages 52 and 228.)

[Zuo 2006] Ming Zuo and Volker Haarslev. *High performance absorption algorithms for terminological reasoning.* In Proceedings of the International Workshop on Description Logics (DL-2006), pages 159–166, 2006. (Cited in page 31.)

Abstract: As robots begin to enter our daily lives, we need advanced knowledge representations and associated reasoning capabilities to enable them to understand and model their environments. Considering the presence of humans in such environments, and therefore the need to interact with them, this need comes with additional requirements. Indeed, knowledge is no longer used by the robot for the sole purpose of being able to act physically on the environment but also to communicate and share information with humans. Therefore knowledge should no longer be understandable only by the robot itself, but should also be able to be narrative-enabled.

In the first part of this thesis, we present our first contribution with Ontologenius. This software allows to maintain knowledge bases in the form of ontology, to reason on them and to manage them dynamically. We start by explaining how this software is suitable for Human Robot Interaction (HRI) applications. To that end, for example to implement theory of mind abilities, it is possible to represent the robot's knowledge base as well as an estimate of the knowledge bases of human partners. We continue with a presentation of its interfaces. This part ends with a performance analysis, demonstrating its online usability.

In a second part, we present our contribution to two knowledge exploration problems around the general topic of spatial referring and the use of semantic knowledge. We start with the route description task which aims to propose a set of possible routes leading to a target destination, in the framework of a guiding task. To achieve this task, we propose an ontology allowing us to describe the topology of indoor environments and two algorithms to search for routes. The second knowledge exploration problem we tackle is the Referring Expression Generation (REG) problem. It aims at selecting the optimal set of piece of information to communicate in order to allow a hearer to identify the referred entity in a given context. This contribution is then refined to use past activities coming from joint action between a robot and a human, in order to generate new kinds of Referring Expressions. It is also linked with a symbolic task planner to estimate the feasibility and cost of future communications.

We conclude this thesis by the presentation of two cognitive architectures. The first one uses the route description contribution and the second one takes advantage of our Referring Expression Generation contribution. Both of them use Ontologenius to manage the semantic Knowledge Base. Through these two architectures, we present how our contributions enable Knowledge Base to gradually take a central role, providing knowledge to all the components of the architectures.

Keywords: human-robot interaction, Knowledge representation, ontology, spatial referring