Planning For Both Robot and Human : Anticipating and Accompanying Human Decisions

Guilhem Buisan

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *6 juillet 2021* **par :**

Guilhem BUISAN

Planning For Both Robot and Human: Anticipating and Accompanying Human Decisions

---

JURY

Stéphane CONVERSY	Professeur	Président du Jury
Olivier SIMONIN	Professeur	Rapporteur
Daniele NARDI	Professeur	Rapporteur
Julie SHAH	Professeure Associée	Examinatrice
Rachid ALAMI	Directeur de Recherche	Examinateur
Thierry SIMEON	Directeur de Recherche	Directeur de Thèse

École doctorale et spécialité :
EDSYS : Robotique

Unité de Recherche :
Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS)

Directeur de Thèse :
Thierry SIMEON

Rapporteurs :
Olivier SIMONIN et *Daniele NARDI*

À mes grands-parents, mes quatre éléments

"C'est l'expérience qui dégagera les lois, répondait-il, la connaissance des lois ne précède jamais l'expérience."

— Antoine de Saint-Exupéry, *Vol de Nuit*

"Les savants ont calculé que les chances d'exister d'un phénomène aussi manifestement absurde sont de une sur un million. Mais les magiciens, eux, ont calculé que les chances uniques sur un million se réalisent neuf fois sur dix."

— Terry Pratchett, *Mortimer*

Remerciements

À la fin de mes études d'ingénieur en aéronautique[1], alors que je me questionnais sur l'idée de faire une thèse, un des membres de l'équipe dans laquelle j'effectuais mon stage m'a dit "tu verras, une thèse, on n'en est jamais l'auteur principal, on fait office de catalyseur tout au plus. Les meilleures idées ce n'est pas seul derrière ton écran que tu les auras, mais en discutant autour de la machine à café, pendant les repas de famille ou en balade avec des amis"... et le plus beau, c'est qu'il avait raison. Ainsi, j'aimerais m'attarder pour exprimer ma gratitude à tous les auteurs de ce document, qui ont aussi écrit leurs lignes dans le livre de ma vie.

Premièrement, je souhaite remercier mon directeur de thèse, Thierry "Nic" Siméon pour la confiance qu'il m'a accordée en acceptant de me superviser, ainsi qu'à la liberté qu'il a su me donner quand mes intérêts portaient sur d'autres thématiques que les siennes.

Ensuite, je remercie Rachid Alami, directeur officieux de cette thèse, pour m'avoir guidé et permis d'explorer plusieurs facettes de l'interaction humain robot. Son plus gros défaut, qui est aussi sa plus belle qualité, est l'inspiration qu'il procure. Tous ceux ayant déjà travaillé avec lui savent que si l'on entre dans son bureau avec une question, on en sortira avec mille. Je le remercie aussi pour l'assurance dont il a fait preuve, et la responsabilité qu'il m'a laissée lorsque j'écrivais des articles ou ma thèse, ainsi que pour m'avoir poussé à toujours me dépasser. J'espère que le plaisir que j'ai eu à travailler avec lui a été partagé durant ces quatre années.

Mes remerciements vont aussi aux Professeurs Olivier Simonin et Daniele Nardi, rapporteurs de cette thèse, pour avoir pris le temps d'en lire et d'en commenter le manuscrit, me permettant de corriger les erreurs et imprécisions qui s'y étaient glissées. Je tiens à remercier Professeure Julie Shah, d'avoir accepté de faire partie du jury lors de la soutenance. Enfin, je souhaite exprimer ma plus grande gratitude au Professeur Stéphane Conversy, non seulement pour avoir présidé le jury, mais surtout pour m'avoir initié au monde de la recherche académique et pour m'avoir guidé tout au long de mon parcours à l'ENAC.

Je n'aurais sans doute jamais pu accomplir cette thèse sans mon comité de suivi composé de Patrick Danès et Michel Taix. Les présentations que j'ai dû leur faire ont à chaque fois été une grande source de stress et de questionnement, mais aussi l'occasion de lister l'avancée de mes travaux et de recevoir de précieux retours et conseils.

De même, je remercie Simon Lacroix pour le temps qu'il m'a consacré lorsque de mon avancement j'ai douté. Jamais je n'ai entendu de paroles plus libératrices que "tu as de quoi écrire et soutenir ta thèse avec ce que tu m'as présenté !".

Merci plus globalement à l'ensemble des membres de l'équipe RIS que j'ai côtoyé, anciens et actuels, permanents, doctorants, post-docs, ingénieurs et stagiaires. Travailler dans cette équipe, où la bonne humeur fréquente une connaissance scientifique sans limite, a toujours été intimidant pour moi, mais aussi infiniment valori-

[1] je tiens à le préciser.

sant. Pour toutes ces discussions, formelles ou non ; ces idées, bonnes ou mauvaises ; ces conseils, toujours bienveillants ; ces fois où mes ignorances ont ébranlé votre patience, je vous remercie.

Je souhaite maintenant présenter des remerciements plus personnels et de précision plus chirugicale[2].

Tout d'abord, j'adresse un immense merci à mes collègues, qui avouons-le, sont devenus bien plus que de simples équipiers au cours de ces quatre intenses années. En commençant par les mousquetaires, avec qui nous avons partagé l'intégralité de nos années de thèse.

Merci Kathleen, étant arrivés le même jour au laboratoire, j'imagine que ça fait de toi ma jumelle du LAAS ? Avec ton éthique à nous filer Lyme, tu as quand même résisté à nous faire interner, ce n'étaient pourtant pas les occasions qui manquaient. Merci aussi pour tous ces moments coréens, que tu as su épicer de ton caractère aussi piquant que leur nourriture. Pour ce pari gagné, je te souhaite tous les meilleurs millésimes accompagnés de plein de cacahuètes !

Merci Amandine, volleyeuse libérale, super superviseuse et organisatrice multitâches, pour tes partages d'articles brandant des débats passionnés et pour tes exigences poussant toujours plus loin nos travaux, ton esprit de compétition et ta détermination menant forcément aux réussites. Enfin, merci pour ces semaines d'intégration, en France ou en Finlande, dans lesquelles karaokés improvisés se mêlaient aux boissons à l'hibiscus.

Merci Guillaume, pour m'avoir *supporté* pendant cette thèse. Chapeauteur de l'équipe, chapeauté à sang chaud luttant contre les moulins... à parole, loin d'être un manche mais toujours prêt à lancer des piques, tu ne te sépares jamais de tes Cerv...ezas. Des terrasses de Finlande jusqu'aux tours d'Allemagne, on a partagé des fous rires et des idées, merci pour ces engueulades et ces moments de complicité.

Merci ensuite à tous les membres RIS pour votre soutien et pour ces moments toujours enrichissants. Merci Amélie, talentueuse carottophobe ludovore dont le rire éclairait l'open-space. Merci Alejandro, "je sais pas ce qu'on va faire aujourd'hui, mais on va le faire à fond !". Merci Jules, écoterroriste pacifiste. Merci Arthur, alliage raffiné de connaissance, de sourire et de bienveillance. Merci David, bushi du rire, et amateur de cacahuètes. Merci Rafa, inventeur de techniques de tarot. Merci Yannick, triathlète probable, mais ingénieur compétent certain. Merci Phani, chorégraphe de Pepper. Merci Miki, qui ne peut pas s'échapper de RIS. Merci Philippe, amateur de blagues de grands crus. Merci Jérémy, brain and body builder. Merci Ilinka, qui arrive à nous bluffer en boucle. Merci Alexandre, binôme encadrant d'INSAïens. Merci Antoine, industriel en sous-marin dans l'équipe. Merci Anthony, héritier de ce travail à ne pas trop prendre au sérieux. Merci ensuite à Yoann, Harmish, Christophe, Ellon, Jérôme, Martin, Gianluca, Dario, Eliott, Andréa, Florian, Florian, Pierre, Léa, Élise, Vivien, Jean-Hugues, Idriss, Paul, Tanguy, Sylvain et à tous les autres !

[2]Mais qui sont faits main, pas de Da Vinci ici.

Merci à tous les membres de l'AMAP Sophie et Gaya, et surtout à Sophie pour avoir, à la sueur de son front, nourri mon corps et mon esprit pendant ces années de thèse.

Merci à cette belle et grande famille aéronautique de l'aéroclub Icaria dont les membres ont toujours des histoires à raconter qui font frémir, de peur ou d'excitation, et des images à partager qui font rêver. Merci à Philippe "Phiphi" Salvato, maître de l'air à l'humilité à faire givrer un carbu et à l'anticonformisme à faire pâlir des inspecteurs de la DGAC. Tu as su de nombreuses fois me montrer que tous les problèmes de la vie paraissent beaucoup plus petits vus du ciel, et à quel point garder les pieds sur terre était nécessaire pour ne pas perdre le "bon sens paysan" qui nous fait trop souvent défaut.

Merci Michel, d'avoir été présent dans les différentes étapes de ma vie et de m'avoir initié à la sensation addictive de voler.

Merci à tous les membres du club robotique de l'ENAC pour avoir initié et développé mon amour-haine de la robotique. Merci Ludo, Quentin, Seb, Yoann, Xavier, Florian, Florian, Darian, Manu et tous les autres. Merci à toi Fab, gourou de la méca, génie de l'élec et virtuose du code de m'avoir tant appris ! J'espère que les Rusty Ducks connaîtront des roues codeuses instables, des servos qui brûlent et des bugs laissant des traces encore longtemps !

Mes amis des Los Pollos Hermanos[3], je vous remercie grandement. Merci Dzieciol, Pastre, Léopold et Fab, merci pour ces soirées, pour ces énigmes, pour ces jeux de rôles, pour ces projets, pour ces délires !

Je souhaite à présent remercier mes amis de plus longue date, pour qui se rappeler précisément le début de notre amitié n'aurait pas de sens, avec qui je ne peux plus compter les moments de rire, de peur, de discussions, de tristesse et de bonheur.

Dans cette thèse, j'écris beaucoup sur les modèles d'humains, mais ici, je veux écrire quelques mots sur l'humain modèle. Merci Guillaume, témoin clé des moments charnières de ma vie, s'échapper avec toi est à chaque fois un ravissement. Diffuseur concentré que même Pasteur ne ferait taire, toujours prêt à engager dans la zone ou à t'engager dans la vie. Merci, bien sûr, Natacha pour tes conseils avisés de docteure, je vous souhaite le meilleur dans votre belle et grande maison !

Merci Rémi, brasseur de moult malts, conteur passionné lisant nos destins sur les faces de ses dés, pour ta clairvoyance m'ayant toujours aveuglée, et pour ces moments où l'on refaisait le monde en satellisant des bières ou en jouant à des jeux aux règles trop floues. Je sais que nos débats d'idées me créeront des ampoules encore longtemps, alors dis à Claire de réparer ses chaussures, et prépare les tiennes !

Merci Chloé, pour ces huit dernières années de nos vies que l'on a choisi de partager ensemble. Merci de m'avoir soutenu pendant ces (trop ?) longues années d'études et avoir été à mes côtés pour s'en répartir les peines et les joies. Cette thèse n'aurait sans doute jamais vu le jour sans le support de la prof de maths la plus swag au monde !

[3] les initiés sauront quelles lettres remplacer.

Merci à Marie-Hélène et à Michel pour ces débats et ces mots réconfortants, autour d'un muscat, dans une rivière, lors d'une balade en forêt ou en cueillant des olives. Merci à Claire et Jeff pour leur bonne humeur et les reportages photos de leurs vacances. Peu importe d'où le vent souffle, votre nouvelle maison tiendra et vous saurez surfer sur tous les tracas !

Enfin, un gigantesque merci à ma famille, avec qui les disputes et les schismes sont si rares que mes collègues la qualifiaient de "dysfonctionnelle". Alors, pour cette famille si parfaitement dysfonctionnelle générant des discussions toujours techniques et intéressantes :

Merci maman de m'avoir élevé dans cet écrin de soleil et de verdure si magnifique et particulier. Tu as su me donner le goût de la liberté, tout en me faisant apprécier sa valeur, pour qu'à tous les cous pelés qui essaieraient de me l'acheter je puisse répondre que je "ne voudrais pas même à ce prix un trésor".

Merci papa pour ces valeurs que tu m'as transmises ; responsabilité, courage et franchise en sont juste des exemples. Merci pour ton soutien inconditionnel, ton humour, mais surtout pour me rappeler de "ne pas me prendre la tête".

Merci Alexis, Adélaïde, et la nouvelle venue Anna, première de la famille à toujours m'avoir connu avec le titre de docteur. Avec vos profils, je sais que tout ira bien dans vos maisons (tout nus) accompagnés de vos Sphynx (tous nus) !

Merci tonton Fabrice, tatie Karine et Loan, votre foyer est tellement intense qu'il fissure votre maison. Toujours sur la mer, les mains dans le cambouis, au bureau, auditant dans les plus grandes entreprises ou sur un practice, vous trouvez toujours du temps pour vos proches, et nous vous en remercions tous.

Enfin, et surtout, mes derniers remerciements vont à mes grands-parents. Merci pour l'éducation et l'instruction que vous m'avez données et que vous continuez à me transmettre, sans tout ce que vous m'avez prodigué cette thèse n'aurait jamais existé, et je ne serai certainement pas l'homme que je suis aujourd'hui. Merci du fond du cœur à vous quatre pour ces heures de conversations, téléphoniques ou présentielles, desquelles je sortais toujours avec le sourire et plein de bons conseils. Merci papi Roger, instructeur de MacGyver, pour nos discussions techniques et tes enseignements qui ont éveillé ma curiosité dès mon plus jeune âge. Merci mamie Christine, pour tes histoires abracadabrantes de l'ancien temps qui semblent toujours sortir d'un livre. Merci papi Raymond, manipulant aussi bien les casseroles que les mots, de régaler mes papilles et mes zygomatiques. Merci mamie Francine, de me transmettre ce feu de la vie, ce sourire et ces envies de voyages où l'important n'est pas ce que l'on a, mais ce que l'on vit.

PS : je ne peux quand même pas oublier les petites boules de poils à cajoler, dont l'oisiveté forçait à l'introspection en cette période de thèse. Merci à Zelda, à Logano "Monsieur Boubou" et à Olya pour leurs câlins, leurs ronrons, leurs léchouilles et leurs mignonitude !

Je souhaite aussi créditer certains artistes musicaux m'ayant aidé durant cette thèse. Pour coder, pour écrire ou pour déprimer, ils m'ont toujours accompagné, et leurs riffs resteront coincés entre mes deux oreilles encore longtemps ! Merci à :

Iron Maiden, Gojira, Jinjer, Arch Enemy, Postmodern Jukebox, Tool, Avenged Sevenfold, System of a Down, Protest the Hero, Opolopo, Heilung, Dance With The Dead, Kalisia, Steam Powered Giraffe, Igorrr, Carpenter Brut, Protokseed, Brass Against, The Police, Soilwork, Ye Banished Privateers, The Night Flight Orchestra, David Bowie, Pantera, Earth Wind and Fire, The Dreadnoughts, Epica, Hippocampe Fou, Bohren & Der Club Of Gore, Tim Minchin, Soulpersona, Cryochamber, Aerosmith, Frontback, Manowar, Corpo-Mente, Ultra Vomit, Queen, Orelsan, Metallica, Fejd, Die Antwoord, Black Sabbath, Alestorm, Audioslave, Linkin Park, McBaise, The Who, Stupeflip, Scorpions, Slaughter To Prevail, Pomplamoose, Frog Leap, Rage Against the Machine, Sleep, Rammstein, Guns N' Roses, Intervals, Limp Bizkit, In This Moment, Dire Straits, Britney, AC/DC, Eminem, Grimes, Jamiroquai, Foo Fighters, Neil Cicierega, Slipknot et tous les autres !

Contents

Acronyms

AOI Area of Interest. 34, 36, 37

CLLE Cognition, Langues, Langage, Ergonomie. 26, 27, 31

HATEB Human-Aware Timed Elastic Bands. 22, 23, 27, 28, 31, 39, 45, 46, 48, 49, 50, 154

HATP Hierarchical Agent-based Task Planner. 13, 14, 60, 86, 87, 88, 89, 92, 93, 98, 100, 107, 109, 126, 138, 151, 152, 153

HATP/EHDA Human Aware Task Planner with Emulation of Human Decisions and Actions. 3, 99, 108, 110, 115, 119, 123, 130, 133, 135, 136, 137, 141, 144, 167

HCI Human Computer Interaction. 3, 6, 7, 18, 39, 40, 41, 43, 44

HRI Human Robot Interaction. 1, 6, 18, 39, 40, 41, 42, 43, 44, 45, 57, 62, 67, 69, 85, 98, 100, 108, 119, 128, 135, 149, 151, 152, 154

HTN Hierarchical Task Network. 3, 13, 86, 100, 103, 104, 105, 107, 108, 110, 112, 119, 121, 130, 131, 132, 137, 139, 151, 152, 153, 155

KB Knowledge Base. 85, 86, 136, 137, 142, 149

MuMMER MultiModal Mall Entertainment Robot. 2, 17, 18, 47, 51

PeRDITA Pertinence of Robot Decisions In joinT Action. 29, 33, 34, 35

PyHOP Python Hierarchical Ordered Planner. 108

RE Referring Expression. 58, 59, 60, 70, 73, 79, 81, 85, 86, 91, 92, 138, 144, 145, 146

REG Referring Expression Generation. 57, 58, 59, 60, 62, 67, 68, 69, 70, 71, 72, 73, 74, 78, 79, 82, 84, 86, 88, 89, 92, 93, 114, 128, 136, 138, 144, 146, 149, 151, 154

TEB Timed Elastic Band. 23, 49

TTC Time-to-Collision. 24, 27, 28, 29, 31, 32, 35, 36, 37, 38, 39

UX User Experience. 31, 40, 41, 42

Introduction

Humans use machines for a long time. On one hand, what started as simple tools quickly gained in complexity and are now robots able to autonomously act on the world with little to no human supervision. On the other hand, humans are more and more dependent on these machines, both for everyday and more specific tasks. However, both acting on the environment and having some autonomy can lead to incidents and injuries if a robotic system is not well made or if a human has not received a specific training. This is why currently in the industry we see a complete physical separation between human and robots, or some annoying light and sound systems when robots share human environments. Indeed, while the Fitts's HABA MABA [Fitts 1951] distinction is getting blurrier, some major interaction challenges have not been tackled yet.

In this thesis we propose to explore a way to bring the human and robot closer in order to make them perform tasks in shared environments in a safer and more usable manner than existing systems. To do so, we claim that robots must be able to make their decisions based not only on their own perception of the environment but also on their estimation of the beliefs of their human partner. Moreover, the robot must be able to plan taking into account that the other agent will also plan, act and react to the actions of the robot.

Human Robot Interaction

The literature on Human Robot Interaction covers a wide range of approaches and visions. As in artificial intelligence, a distinction can be made between systems trying to act like humans and systems trying to act rationally. Moreover, some approaches also try to implement human cognitive models on robot to validate them.

In this manuscript, the approach chosen is to make a system acting rationally. Besides, we define the Human Robot Interaction (HRI) as Goodrich and Schultz as being *the field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans.* Thus, the goal is to make a robotic system allowing to, when used by or with humans, to perform a task in the most effective, efficient and satisfactory way. To put it otherwise, we aim at making the most usable (as defined in ISO 9241-11) robotic system.

It is worth mentioning that some work focusing on the goal described above still mimicry some human behaviors. Indeed, the best working interaction example we currently have access to is humans collaborating with humans. In this thesis, even if the human behavior may be taken as an inspiration, the argument that the robot should act in a certain way because the human does so will not be used. However, we present ways of using human tasks and actions modeling to improve robot decisions and planning.

Motion and Task Planning

Both humans and robots are considered as agents. An agent is an entity able to modify its environment (and its own state) by performing actions. Now if an agent is given an objective (a specific environment state) and has an estimation of what its actions will change in this environment, it can try to figure out a succession of actions leading to that objective. This process is called planning. We can identify two main types of planning.

First, task (or symbolic) planning models the world into facts and agents' actions as changes over these facts. In this thesis, we explore how human robot task planning can benefit from communication actions feasibility and cost estimation (Chapter 3). Moreover, we propose a task planning scheme aiming at emulating human planning process to generate human robot collaborative conditional plans (Chapter 4). Then, motion (or geometrical) planning represents the world in a more refined way, accounting for geometrical models of agents and the environment. Actions are represented as trajectories, moving objects or (parts of) agents across space. In this thesis, we will especially deal with navigation, a sub part of motion planning, and study in Chapter 2 how planning for both the human and robot trajectories allows to elaborate more efficient trajectories when the robot is in proximity to the human.

Summary of the Thesis

We are interested in how planning for both the human and the robot can improve the interaction while doing collaborative tasks. We start in Chapter 1 by giving context to this thesis by presenting task and navigation planning, what are the current challenges in human robot interaction and how we can model human actions and provide shared plans. Then, in Chapter 2 we explore an approach to robot navigation planning where both the robot and the human trajectories are computed at position control rate. This approach uses an optimization scheme allowing to define constraints representing the interaction between the trajectories. We use it to enhance the mutual manifestness of the robot and show through a user study how it improves the efficiency of the robot navigation in narrow crossing scenarios. We also present how we used this approach on other robots and on a complete robotic system deployed in the wild during the MultiModal Mall Entertainment Robot (MuMMER) project. Alleviating from inherent ephemeral and implicit nature of navigation interactions, we continue exploring human and robot planning in task planning.

In Chapter 3, our objective is to use the observations made in navigation and to apply them in the symbolic domain, more explicitly. More precisely, we want to consider communication as actions the robot must plan in order to have the best interaction possible. Planning communications requires planning for both agents. This lead to two contributions. First, we present an efficient algorithm running over an ontology resolving the content of communications aiming at referring to an

object of the environment to the hearer. This problem is called referring expression generation and, while it has been studied for over thirty years, we show that our approach is not only the fastest one to date but also the most suitable for human robot interaction scenarios. Then, we used this efficient method for resolving communications content to estimate the feasibility and cost of such communication actions during task planning. Using this approach allows to avoid plans that would have been unrecoverable during the execution and to find more efficient ones.

In Chapter 4 we propose a hierarchical task planning scheme that is not only able to update human and robot beliefs separately throughout the planning process, but also able to reason on distinct human and robot action models. While the robot action model is close to ones used in Hierarchical Task Network planning, the human one is thought to be made through a task modeling approach as done in Human Computer Interaction. Actions are represented as functions over the beliefs and we specify rules on which actions may update or reason on which beliefs. We implemented this scheme in a prototype planner which we named Human Aware Task Planner with Emulation of Human Decisions and Actions (HATP/EHDA) and showed that it allows to represent and to plan for intricate human robot collaborative scenarios.

Finally, in Chapter 5, we present some interesting details about how HATP/EHDA can be integrated in a robotic architecture. We end by presenting a novel human robot collaborative task: the director task. Inspired from psychology studies, it induces several challenges for human robot interaction. We introduce a full robotic architecture able to cope with the nominal cases and in which HATP/EHDA has been integrated.

List of Publications

Published

- 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.

- Buisan, G., & Alami, R. (2021, March). A Human-Aware Task Planner Explicitly Reasoning About Human and Robot Decision, Action and Reaction. In *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction* (pp. 544-548).

Accepted

- Belhassein, K.*, Buisan, G.*, Clodic, A., & Alami, R. Towards Methodological Principles for User Studies in Human-Robot Interaction. To be published in *ACM Transactions on Human-Robot Interaction Journal*.

Submitted

- Buisan, G., Compan, N., Caroux, L., Clodic, A., Carreras, O., Vrignaud, C., & Alami, R. Evaluating the Impact of Time to Collision Constraint and Head Gaze on Usability for Robot Navigation in a Corridor. Submitted to *IEEE Transactions on Human-Machine Systems Journal*.

- Buisan, G., Favier, A., Mayima, A., & Alami, R. HATP/EHDA: A Robot Task Planner Anticipating and Eliciting Human Decisions and Actions. Submitted to the *IEEE International Conference on Robotics and Automation (ICRA) 2022*.

Chapter 1

Planning for Human Robot Interaction Context and Challenges

Contents

This first chapter aims at setting the context for this thesis. While not providing a exhaustive state of the art, it presents challenges of planning for human robot interaction. More complete related work will be reviewed in the beginning of Chapter 2 and Chapter 3. In what follows, we first define robot motion and task planning, then we describe the challenges arising for the specific context of human robot interaction and finally we present general approaches trying to cope with these challenges by modeling and planning for the human.

1.1 Task and Navigation Planning

As put by Ghallab, Nau and Traverso, "the purpose of planning is to synthesize an organized set of actions to carry out some activity" [Ghallab 2016]. Planning can be *domain-specific* if the planning method (and set of actions) is precisely tailored to solve a specific type of activity. Domain-specific planning includes navigation aiming at planning a trajectory for moving the robot base from a place to another while respecting its kinodynamic constraints and avoiding obstacles. On the other hand, *domain independent* planning uses methods which can be applied to a wide varieties of problems using abstraction. Actions are then represented as functions modeling the changes they have on a symbolic world state to produce a new world

state. In any case planning requires to model the environment and the actions allowing to predict how an agent actions would impact this environment.

Besides, the robot not only needs to plan, but also to act. Acting refers to the process by which the robot decides "how to perform the chosen actions while reacting to the context in which the activity takes place". Indeed, a planning process can usually only rely on the world state estimated at the beginning of the process and the models of how it evolves (caused or not by an agent action). However, this estimation can be coarse and may lack of a lot of information. Moreover the models used are always imperfect and may not represent exactly how the world state evolves over time. For example in navigation planning, the map on which the planning is done may be incomplete as some obstacles may not be detectable at the robot starting position. Besides the robot controllers might not be able to follow exactly the planned trajectory, and thus the robot would fall outside the plan. This is why Ghallab, Nau and Traverso argue for an "interplay" between planning and acting.

In navigation this is usually done via a global/local planner approach [Choset 2005]. First the global planning process finds an obstacle-free general trajectory from the start to the end point over a known map of the environment. Then, a local planner tries to find a more precise and short term trajectory from the current estimated position of the robot to a point on the global plan while also dealing with newly detected obstacles. This local trajectory is recomputed at position control speed (around 20Hz for speeds around 2 meters per seconds) and can be as short as only a speed command sent to the controller but can also predict a trajectory for several seconds in the future. For more abstract task planning this often translates as having a "descriptive model" for planning, where tasks are represented as high level symbols and an "operational model" for acting, where tasks can be refined into low level commands depending on current world state.

Planning seldom condenses to finding a valid plan or not. It also often has to find the minimal cost plan. Costs are usually associated with actions and can represent a variety of concepts, ranging from battery consumption to money costs. In the case of navigation planning, the interest is often to find the shortest obstacle-free trajectory (*i.e.* minimizing the trajectory length or duration).

1.2 Collaborative Human and Robot Activities

When robots need to interact with humans, be it for collaborating on a task, for maintenance, for teleoperation or just because they share a common environment, new planning constraints and goals arise. The Human Robot Interaction (HRI) field studies these topics. More precisely, "HRI is a field of study dedicated to understanding, designing, and evaluating robotic systems for use by or with humans" [Goodrich 2007]. Other fields study the interaction between humans and systems, such as Human Computer Interaction (HCI) or Human-Agent Interaction (HAI) but HRI presents its unique challenges, as the system is able to take autonomous

decision and physically act on its environment. All these fields are strongly linked to psychology (more precisely cognitive psychology) as the understanding and modeling of human behavior is crucial for such intricate interactions between the systems and the human. In this section we will first define a highly desirable property of interactive systems: the usability, and how it is applied to highly autonomous systems and to robots. Then, some principles of the joint action field, studying how humans handle cooperative tasks, will be presented along with their application in human robot interaction.

1.2.1 Usability and Automation

Usability is defined by the ISO 9241 as "the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use". The interactive system used by a human must allow them to do the task, while consuming as little resources as possible (*e.g.* time, money, number of steps) and being enjoyable to use.

1.2.1.1 Human Computer Interaction

To design such systems a lot of contributions in HCI and ergonomics rely upon human cognitive models, often drawn from cognitive psychology. One of the most detailed and still being used and enriched to date is the human processor model [Card 1983], which can be used for estimating how long a human will take to achieve a certain task. Another widely used model to help with designing interactive systems is the Norman's seven stages of action model (Figure 1.1). This model represents the different cognitive processes involved when a human performs an action, and allows to design an interactive system accordingly. Moreover, Norman elicits seven design principles from this model. One of them being highly desirable for an autonomous agent system is the discoverability [Norman 2013].

Indeed, too many systems require expertise to be used. This often results from the system designer focusing on the "machine part" and not on the interface. Machines have been identified to avoid some humans flaws for a long time [Fitts 1951], but because the "interface" [1] is too often poorly designed, the human needs to make great effort to use the machine. This effort becomes more important as the system gains in complexity and automation. This has been identified by Norman as the "paradox of automation" [Norman 2013]. More precisely, when an automated system is performing a task the human situation awareness [Endsley 1988] decreases leading to misuses and errors when they need to interact with the system (*e.g.* because of a system error or incapability).

[1] We refer here to the first meaning of "interface": the common surface where two systems meet and interact.

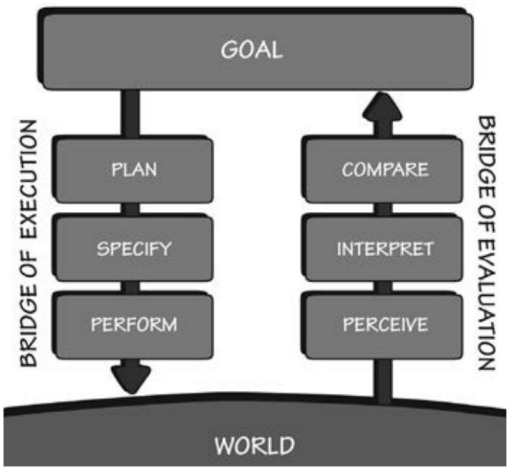

Figure 1.1: The seven stages of actions defined by Norman [Norman 2013]. By giving a precise definition of each stage, this model guides the design and engineering of interactive systems by emphasizing each step of the cognitive processes taking place during an action. The designer can estimate the scope of the system and the potential errors at each stage of action.

1.2.1.2 Human Robot Interaction

When interacting with a robot, the human is no more the only agent taking autonomous decisions and able to act on the world. Indeed, not only the robot has its own plan-act-sense cycle with its own goal, but it can also physically move and change its environment. For the decision part, Bradshaw *et al.* [Bradshaw 2011] propose eight maxims to complete and specify the recommendations of Norman. These maxims apply to human-agent interaction when they perform a joint activity:

- The agent must be *observable*: its state and intentions are clearly exposed to others.
- The agent must *appraise progress*: it should inform others about the status of its task and warn them for any foreseen potential issues.
- The agent must *know its limits*: it should be proactive or wait when performing a task, depending on its evaluation of its capabilities.
- The agent must be *predictable* and *dependable*: it should show what its capabilities are, and be trusted to use them at its best.
- The agent must be *directable*: its (sub)tasks can be preempted or modified if required and its knowledge can be updated thanks to other agents.
- The agent must be *selective*: it should expose only relevant facts in the context of the task.

- The agent must be *coordinated*: it should negotiate and deconflict plans, share resources and align beliefs between agents to create a *common ground* if it is required to perform the task.

Interestingly, in their "Architecture for autonomy", Alami *et al.* propose a robotic architecture which defines similar levels as the ones identified by Norman during human action [Alami 1998].

Robot navigation should also respect these maxims. A number of approaches are trying to cope with these specific human robot issues [Kruse 2013]. Moreover, more precise concepts have emerged from the maxims especially for the motion of a robot in a joint activity.

First of all, the motion should ensure the physical safety of the surrounding humans. This is often respected by constraining the trajectory to stay at a minimal distance from humans [Kruse 2013, Rios-Martinez 2015].

Moreover, while in a task, the motion of the robot can be used to convey information about its intentions, and thus making it more *observable* and *predictable*. Dragan *et al.* define three types of motions: functional, predictable and legible [Dragan 2015]. A *functional motion* is built only to make the robot move from one state to another one while avoiding humans and obstacles, it does not aim at conveying any intention — which does not mean it does not convey one. Then, a *predictable motion* is a motion expected by an external observer knowing the robot model and its goal. It is often the quickest or shortest trajectory to reach the goal. Finally, a *legible motion* is a motion allowing an external observer to quickly and reliably infer the robot motion goal.

However, as identified by Sisbot *et al.*, to be identified as legible or predictable, the robot motion must be seen by the human. Accounting for the visibility of the robot by the human is paramount [Sisbot 2007]. They integrate this constraint in a motion planner which is able to compute the safety of trajectories using the distance to a specified human and also prioritize visibility of the robot by estimating the field of view of the human and penalizing trajectories outside it.

A lot of work has been made to generate legible paths. In [Beetz 2010], the robot learns usual motions from humans and generates trajectories based on them. However, even if the authors highlight the improvement in legibility, one could argue that the learned motions correspond more to the Dragan's definition of *predictability*. In [Dragan 2013], a robot motion planner is presented which is dedicated to generate legible trajectory. They prove its effectiveness through a user study while reporting an interesting finding: if the legibility of the motion is stressed to much, it becomes unpredictable and confuses the observer.

1.2.2 Joint Action in Human-Robot Interaction

Besides designing behaviors from scratch, previous work takes inspiration from situations where humans already perform a task with another autonomous agent: another human. Several previous contributions have been done studying how humans perform a so-called *joint action*. *Joint action* is defined by Sebanz *et al.* as

"any form of social interaction whereby two or more individuals coordinate their actions in space and time to bring about a change in the environment" [Sebanz 2006]. Moreover, several work investigate multiple key abilities humans deploy when performing a joint action:

- humans can create and understand *joint attention*: they are able to direct, or be directed by, others' attention [Pacherie 2011, Sebanz 2006].
- humans can *predict* others' action effects and goals [Tomasello 2005, Sebanz 2006].
- humans can *represent the shared task*: they can infer others' actions without directly seeing it, by observing events in the environment [Knoblich 2011, Sebanz 2006].
- humans can *coordinate actions* by integrating others' actions into their own plan [Sebanz 2006].
- humans consider combined actions results more important than their own only [Sebanz 2006].
- yet, humans are able to *distinguish their own actions from other's*, allowing them to find any divergence in beliefs or intentions and align them (*e.g.* through communication) if necessary [Pacherie 2011].

Besides, a joint action can only occur if the involved agents know the presence of others agents but also their activity and intentions. Pacherie defines it as the mutual manifestness: "*each subject must be aware, in some sense, of the event as an event that is present to both; in other words the fact that both are attending to the same object or event should be open or mutually manifest*" [Pacherie 2011]. Thus, it is not only necessary for the agents to account for the presence of the other but they also need to know they are involved in the same task.

Finally, studies show that humans can help the others to predict and coordinate their actions by communicating or slightly modifying their own. Coordination smoothers are defined as the changes in an agent own behavior to ease the interaction with another one [Vesper 2010].

It is interesting to note that some concepts are already overlapping with the desirable features for a robot taking part in an activity with a human. For example, the legibility and predictability of motion are based on models of the human capability to understand coordination smoothers and predict future actions effects. Bradshaw *et al.* also refer to an "extra work" an agent has to perform to ensure an efficient interaction with a human partner [Bradshaw 2003], which can be linked to the mutual manifestness and the several mechanisms which can be used, and which a human expects, when interacting.

Moreover, some work already tries to incorporate these joint action concepts and abilities into robotic architectures and behaviors [Khamassi 2016, Clodic 2017].

Lemaignan et al. present a robotic architecture composed of several components addressing the cognitive skills required to perform a collaborative task [Lemaignan 2017]. These include shared plan synthesis, human-aware motion planning — both of which are discussed later in this chapter —, supervision, beliefs management and situation assessment. The situation assessment is able to generate symbolic facts from geometric data for the robot and also to estimate the beliefs of the human partner based on an estimation of their perspective of the scene [Milliez 2014]. This framework has since been improved to a more modular approach allowing more refined reasoning, especially for the estimation of human beliefs [Lemaignan 2018]. The presented supervision component is based on SHARY [Clodic 2009] allowing the robot to execute shared plans while monitoring the human activities and deciding when to communicate. Devin & Alami proposed an extended supervision system, heavily based on theory of mind, which is able to follow a shared plan but also to compute diverging agents beliefs, deciding if the divergence endangers the plan and if so, align the beliefs via verbal communications [Devin 2016].

1.3 Modeling Human Actions and Shared Plans

It has been shown previously that a robotic agent interacting with a human needs to coordinate its actions with them. Moreover, joint action theory exhibits that humans interacting together are able to represent the task as a whole, and plan not only for their actions but also for the actions of other agents. Thus, we think that for a human to perform the most efficient and satisfactory joint task with a robot, this robot must explicitly model the human actions and plan not only for its actions but also for the human ones. In this part we will first introduce some notations used throughout all this thesis, then we will briefly show some ways of modeling human tasks and link them with robot task planning. Finally, we review some systems planning for both humans and robots when performing a joint task.

1.3.1 Notations

In order to help to differentiate between the models presented in this thesis, we introduce here some notations we will use in the thesis. These notations are partially inspired by the work of Chakraborti *et al.* [Chakraborti 2018] (Figure 1.2). We will refer to the model the robot has of itself as $\mathcal{M}^R$, to the model the robot has of the human it is interacting with as $\mathcal{M}^H_r$ and to the model the human has of the robot as $\mathcal{M}^R_h$. The models can represent different parts of the agent, ranging from their beliefs to their action model.

While not being here a formal definition, the notation should help to understand to which agent we are referring to. It is important to note that all the components presented in this thesis are considered to be a part of the robot, and thus $\mathcal{M}^R$ plays a special role as we consider all the information in it as the ground truth. For example, if there is a beliefs divergence between $\mathcal{M}^R$ and $\mathcal{M}^H_r$, we always consider

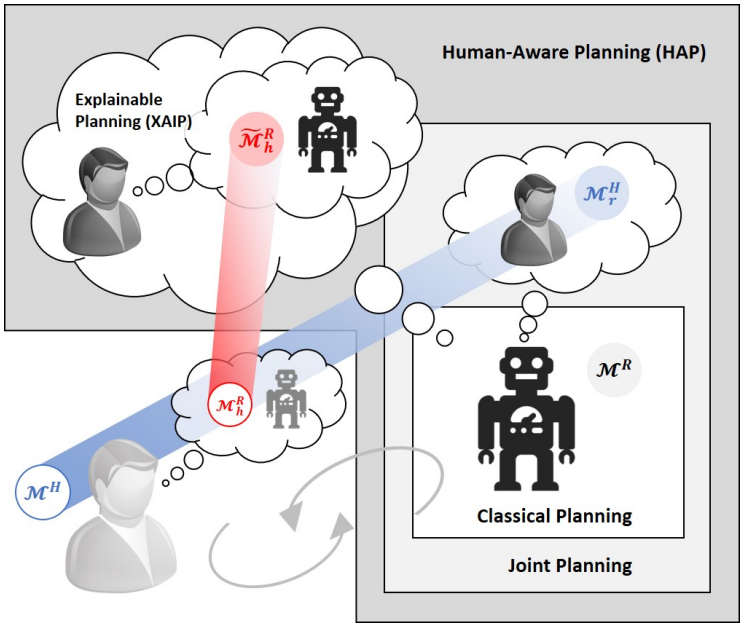

Figure 1.2: The notation coined by Chakraborti *et al.* from [Chakraborti 2018] to represent the robot model $\mathcal{M}^R$, the estimated human model $\mathcal{M}_r^H$ and the estimated robot model the human has $\tilde{\mathcal{M}}_h^R$ (denoted $\mathcal{M}_h^R$ for simplicity as the real one is not accessible to the robot).

that the $\mathcal{M}^R$ is the truth, otherwise, it would make no sense to keep this information in $\mathcal{M}^R$ while having access to the one in $\mathcal{M}_r^H$.

1.3.2 Task Modeling and Hierarchical Task Networks

Multiple approaches try to model the human activity. It is useful either for psychological studies, or to design a system that integrates gracefully in human tasks, in order to improve their performances. These tasks can be seen as hierarchical, where abstract ones decompose into smaller, more concrete ones. Such a hierarchical approach is also used in order to elaborate plans for autonomous robots. Interestingly the underlying structures share several commonalities.

1.3.2.1 Human Task Modeling

A common way of representing human activity ($\mathcal{M}_r^H$) and interaction with computer at high abstraction level is by using *task models*. The hierarchical structure of human activity was first exploited by Annett and Duncan [Annett 1967]. They state that tasks can be described at several levels of abstraction until a certain criterion is met. Each task can thus be refined into subtasks detailing the procedure followed by the human to achieve the higher level task.

Task modeling has then evolved to introduce interaction with systems, produced and needed information, potential errors and a wide variety of operators

specifying how tasks interacts with each other during their execution. Task models are now commonly used in user-centered and user interface designing processes. Most advanced notations include *ConcurTaskTrees* [Paternò 2004] and *HAMSTERS* [Martinie 2019].

These models are used to design or to evaluate interactive systems. They allow the designer to better understand the user task or to study the user workflow using their system. However, these models contain too little information for a system to be able to reason and take decision on them (either in planning or acting).

1.3.2.2 Robot Hierarchical Task Planning

On the other hand, hierarchical representations of tasks have also been used for decades in robotic planning ($\mathcal{M}^R$). In classical planning, each action of an agent is atomic and needs some conditions to hold in the environment to be executed, then it changes the environment when applied. The planning process has then to find the right sequence of actions, being applicable one after the other to change the environment to reach a certain goal state.

Hierarchical Task Networks (HTNs) allow the domain designer to help the plan search by inserting expert knowledge via a hierarchy linking these actions [Erol 1996]. Indeed, a task network is composed of one or several tasks or actions, and each task has one or several decompositions. Each decomposition is itself a task network. The goal of the planner is not to find the sequence of actions to reach a goal, but rather to select recursively for each task the right decomposition ending (if possible) with a network of actions applicable from the initial state. Such a process is named by Ghallab, Nau and Traverso as *planning with refinement methods* [Ghallab 2016].

This planning hierarchy not only allows the domain designer to guide the search by inserting some expertise into the model, but also to enhance explainability as the decompositions often offer a semantic to their subtasks (the *why* can usually be answered by going up in the hierarchy, while the *how* is answered by going down).

1.3.3 Planning for Both the Human and the Robot

While human activity modeling and autonomous planning have been studied separately for decades, there are still only few systems proposing to incorporate human activity into planning for intricate interactive tasks. Planning for both a robotic agent and a human is not trivial. Indeed, while the robotic agent is planning for itself and will surely execute the plan, the human is not directly controllable (making them follow the plan may require the robot to at least communicate and perhaps negotiate) and can also have their own plan they are trying to execute.

The Hierarchical Agent-based Task Planner (HATP) proposes a hierarchical approach to multi agents task planning [Alili 2009, Lallement 2014]. This HTN based planner is able to elaborate a multi agents plan based on a single HTN tree (unifying both $\mathcal{M}^R$ and $\mathcal{M}_r^H$). Moreover, it maintains one beliefs base per agent

allowing to write task decomposition rules and actions preconditions and effects in any agent beliefs base. Finally, HATP also computes costs for the plans found based on action costs and predefined social rules. More details about HATP will be presented in Chapter 3. This task planner has been coupled with the Geometrical Task Planner (GTP) allowing to also plan the geometrical motion of the human partner to inform HATP about the feasibility and the cost of a motion action [Gharbi 2015a]. As stated before, HATP has been used in complete robotic architecture [Devin 2016, Lemaignan 2017].

Buckingham *et al.* propose a planning scheme questioning humans mental models ($\mathcal{M}_r^H$) returning the effects of expected future humans actions [Buckingham 2020]. The planner is then able to determine a robot policy ($\mathcal{M}^R$) influencing the humans actions. In this work, they show how this framework is able to cope with interactive tasks even without assuming that the human is collaborating.

Similarly, Unhelkar *et al.* proposed a POMDP-based approach called CommPlan [Unhelkar 2020]. The POMDP is built using a user defined MDP (Markov Decision Process) representing the collaborative task and an AMM (Agent Markov Model) to represent the human decision-making process ($\mathcal{M}_r^H$). This POMDP is then solved to produce a robot policy ($\mathcal{M}^R$) which, in particular, decides when the robot has to communicate about its beliefs, to question the human about theirs and to ask the human to perform an action. Besides, the human AMM ($\mathcal{M}_r^H$) is not only specified by the programmer but also refined during the interaction via learning.

Besides, to cope with the uncertainty of the human knowledge ($\mathcal{M}_r^H$), Petrick and Foster propose to use conditional planning allowing to plan for incomplete information [Petrick 2013]. By doing so, the planner elaborates a plan for the robot ($\mathcal{M}^R$) accounting for multiple possible human choices, and depending on the knowledge received the execution component can execute the right branch of the plan.

Likewise, Sanelli *et al.* ([Sanelli 2017]) present an approach not only elaborating conditional plans for the robot ($\mathcal{M}^R$) depending on the possible human choices ($\mathcal{M}_r^H$, *e.g.* the choice of activity the human wants to perform), but they are able to transform this conditional plan into a Petri net plan to handle its execution. This contribution is inspired from a previous work by Nardi and Iocchi ([Nardi 2014]) in which they present a method for transforming (linear) joint plan (both $\mathcal{M}^R$ and $\mathcal{M}_r^H$) into a Petri net plan managing its execution. Interestingly, the human actions from the plan are changed into a part of the Petri net where the robot elicits the action (*e.g.* via a verbal communication) if the human does not perform it by themselves. However, this approach only requests the human to make single actions, instead of sharing a high level goal, which can become unpleasant if done repeatedly.

Chakraborti *et al.* uses both the robot model ($\mathcal{M}^R$) and the estimation of the model the human has of the robot ($\mathcal{M}_h^R$) to improve plan explicability [Chakraborti 2017]. Indeed, they propose a novel approach called *model reconciliation* which they present as a classical planning problem. In this problem, the

goal is to make identical both the optimal plans generated via the robot model $\mathcal{M}^R$ and the human estimation of the robot model $\mathcal{M}_h^R$. To do so, they define a list of operators on the models in order to modify them until the plans match. To our knowledge it is the only approach both reasoning on $\mathcal{M}_h^R$ and operating directly on the action models. However, it only has been applied to robot plans and not to joint plans. Indeed, the generated plans contain only robot actions, and the robot and the human do not directly collaborate in the presented tasks.

Geometrical planning in human robot activity has also been studied in several contributions. For navigation, Khambhaita and Alami propose a planner optimizing both the human and the robot trajectories ($\mathcal{M}^R$ and $\mathcal{M}_r^H$) [Khambhaita 2017]. This allows the robot not only to have a prediction of the human future trajectory, but also to model their interactions and how its own trajectory can affect the human's one. This approach will be detailed and used in Chapter 2.

Besides, representing humans as a group has also been showed as beneficial for robot navigation. For instance, for the robot-waiters problem, where multiple robots compute trajectories in order to serve multiple moving humans in a room containing obstacles, Saraydaryan *et al.* ([Saraydaryan 2015]) showed that considering clusters of humans to be served allows to decrease the human idleness (the duration for which a human has not been served by a robot). These humans clusters are called F-Formations when the humans are actively interacting with each other in the cluster (*e.g.* a group of humans discussing together). Recognizing them and integrating them ($\mathcal{M}_r^H$) in the robot trajectory computation ($\mathcal{M}^R$) in order to socially integrate them or to not disturb them also allow for more acceptable robot trajectories ([Althaus 2004]).

Another work presents a motion planner allowing to balance the effort between the robot and the human, depending on the mobility of the human, for a handover task [Mainprice 2012]. To do so, they sample an acceptable position for the human ($\mathcal{M}_r^H$) according to several parameters, including its settable mobility, then they try to plan a trajectory for both the robot ($\mathcal{M}^R$) and the human ($\mathcal{M}_r^H$) to get them into an handover configuration. This work has then be extended and generalized by Waldhart, Gharbi and Alami to handle several robots and humans [Waldhart 2015].

Finally, Waldhart, Clodic and Alami proposed a geometric planner, able to find the best robot and human positions for the robot to point at a landmark for the human [Waldhart 2019]. This planner uses the human vision ability and mobility ($\mathcal{M}_r^H$) as well as robot mobility and tries to make a pointing triangle formation between the robot, the human and the landmark to point ($\mathcal{M}^R$).

From all these contributions we see a great interest in planning for both the human and the robot. Indeed, they allow to represent multiple features highly desired in human robot interaction scenarios. First, by planning for both, shared goals and coordinated actions can be represented. The planners are able to elaborate plans (or trajectories) that do not conflict between the agents, and even that interact in a collaborative way to reach the shared goal.

Then, it allows to allocate tasks to either one agent or the other if they can

be done by both. Unlike other task allocation approaches, modeling the human helps to estimate the effort taken in the contribution of the joint plan. The planner can ensure that most of the effort is taken by the robot but can also balance other solutions depending on the context (*e.g.* urgency to perform the task, where the human may accept to contribute more to do it faster).

Finally, when integrated in a robotic architecture and in a real human robot scenario, planning for both reveals to be key. Indeed, planning before acting usually avoids deadlocks during execution or sub-optimal solutions which would have been encountered by short-term only reasoning. But for human robot interaction it also allows to estimate the effort and the contribution of each agent *before involving the human*. Moreover, the resulting joint plan can then be negotiated with the human before starting the task, allowing for a better efficiency, satisfaction, explainability and acceptability.

Not all of these challenges have been entirely tackled yet. In this thesis we propose to explore multiple ways of planning for both agents.

Chapter 2

Coplanning for Navigation

Contents

2.1 Introduction

In a lot of human robot interaction scenarios, the robot has to move in the environment to accomplish its task. It can either be that the task cannot be done in the direct vicinity of the robot or that the task itself is to move elsewhere or to transport an object. For example in the MuMMER project, a Pepper robot in a mall has to give direction instructions to guide a human to their desired location.

The robot is also able to point to visible landmarks to locate the beginning of the route (*e.g.* saying *"Take these stairs, then take the corridor on your right and the shop will be on your left"* while pointing to the stairs). However, some obstacles in the proximity of the robot and the guided human can prevent them to see the pointed landmarks, or a corridor crossing can be hidden, making the route description one step longer than it should be. Thus, to perform the task of route guiding more efficiently, the robot might decide to move.

In the Spencer project, another robot has to guide people to their gate in the Schiphol airport. Here, the robot will navigate all the way from the starting point to the final destination while ensuring the human is actually following it, but also has to avoid other pedestrians. In this example, the navigation of the robot is a main part of the task.

In both examples, the robot has to plan its motion such as the physical and psychological safety of surrounding humans are ensured. However, not taking into account the motion of these humans during the planning process may lead to sub-optimal trajectories or even deadlocks.

We propose in this chapter, after a survey of related work, to present a navigation planner algorithm taking into account both the robot and the human, then to show how this approach can be used to enhance mutual manifestness and improve efficiency in a narrow corridor crossing scenario through a user study, and finally report some extension made to the approach to include humanoid robots, flying drone and to estimate the progression of the navigation task.

This chapter presents three contributions. First, we perform a user study to evaluate the pertinence of the "planning for both agents" approach. We claim that planning a trajectory also for the human allows to express constraints on the robot trajectory that would not have been possible otherwise. One of these constraints aims at avoiding trajectories in which the robot is facing directly the human at high speed, besides, it allows for a proactive and more legible behavior. Along with robot head motions, we designed an autonomous robot behavior aiming at enhancing mutual manifestness and tested it in a user study while crossing a human in a narrow corridor. We showed that these mechanisms, only allowed by planning for both agents, increase efficiency and satisfaction of the user.

Then, from the experience gained through this user study and through a common work with a psychologist, we propose an analysis and some guidelines for user studies in HRI. In this work, we review the common issues that we identified in recent HRI contributions containing user studies. We inspire from Human Computer Interaction and psychology studies to provide several guidelines while still discussing the differences between these domains and HRI in order to adapt the proposed solutions.

Finally, we refined and extended this approach by using it in different contexts and on different robots. Especially on the humanoid (bipedal) robot HRP2 and on the Pepper robot. The latter has been deployed in a mall in Finland, thanks to the MuMMER project, using a fully autonomous architecture, including our navigation

approach, aiming at providing directions to customers.

2.2 Related Work

2.2.1 Human-Aware Robot Navigation

The aim of robot navigation is to make the robot base (the whole robot) move from one place to another while avoiding static and moving obstacles. However, when the robot has to move in an environment where humans are evolving, other constraints must be added. The robot must not only avoid the humans, as any other moving obstacle, to ensure their physical safety (not harming them), but also take into account their psychological safety (not stressing or frightening them), avoid to block them or to induce drastic changes in their motion [Sisbot 2007], [Kruse 2013]. In order to satisfy these constraints several methods have been used.

The first largely used method is based on costmap exploration. Based on the robot known humans and obstacles in the environment a grid is built, where each cell has a cost increasing in places the robot should avoid to pass through. Then, given a start and an end points, a planner can explore this grid and try to minimize the cost along the trajectory ([Sisbot 2007], [Lu 2013]). In these contributions, the costs are computed according to distance to human (the closest the more expensive the trajectory is), visibility (penalizing trajectories outside the human field of view) and "surprise" (discouraging trajectories behind an obstacle close to the human). These costs are merged usually through a weighted sum or a maximum[1]. Besides, to compute the cost of a trajectory, the cost of each cell composing it are usually summed[2]. These approaches are pretty efficient once provided with a cost grid but since a whole grid can take time to compute, they can perform poorly in dynamic environments. Moreover, they do not account for the robot and human speeds. Thus, a trajectory where the robot is rushing towards the human and another one having the same path but where the robot is gently approaching them would have the same cost.

Another approach is to use the social force model [Helbing 1995]. A robot trajectory is computed based on repulsive or attractive force fields set on humans, obstacles and goal [Ferrer 2013]. This gives good results in open environments but the trajectories can be erratic in confined ones with a lot of obstacles and humans because of the diverging "forces" applied. Besides, humans are applying the same force whether they have seen the robot or not, or whether they are moving or not. Lastly, by only considering the robot plan, these planners return no solution if the robot and the human must cross each other in a narrow corridor where the human is centered leaving no place for the robot to pass. This is why we need a planner able to *infer* that the human can move to one side of the corridor, thus contributing to the solution and allowing the robot to cross on the other side.

[1] Merging motion planning costs is still an open challenge and will not be discussed in this thesis.

[2] Similarly, aggregating costs of configurations to get the trajectory cost is an open challenge and will not be discussed in this thesis.

In their work, Kuderer et al. use social force model to both compute the robot trajectories and predict the nearby human ones [Kuderer 2012]. However, the resulting human trajectories are more reactions to robot motion than coplanning solutions.

To overcome this limitation, Khambhaita & Alami propose a navigation planner based on an optimization scheme of both the robot and the human trajectories. In this approach the trajectories of the robot and of the nearby humans are optimized together, at position control rate, to create a conavigation solution [Khambhaita 2017]. This approach will be detailed later in this chapter but more precisely, it requires a start and an end point (goal) for both the human and the robot. While for the robot the start point is given by a localization component and the goal by the supervision, for the human the start point requires a human position detection component and the goal a intention recognition component. A global planner (usually an A*) computes coarse but complete paths and estimated speeds for both the human and the robot from their respective start point to their goal. Then, the successive positions and the duration between each of them are optimized from the initial trajectories, resulting in two shot-term but precise trajectories. This ensures that at all time it exists for the humans a solution to go to their known goal, and that this solution is optimal with respect to a different set of constraints based on human models.

Although, even if the robot computes an optimal solution for the human and itself, it also needs to communicate it or to show it to the human (*e.g.* whether it plans to go to the left or the right of the corridor, so the human can either accept or decline this plan). Thus, the robot must also try to make its intention clear [Pacchierotti 2006]. This ability of a robot to exhibit its future actions is called legibility. A legible robot will have its future actions and goals inferred early [Dragan 2013], which is crucial in entangled tasks such as crossing in a narrow corridor. For navigation, legibility can be increased either by changing the robot speed along the path [Kruse 2012] or by modifying its path [Khambhaita 2017].

In a broader sense, the changes in an agent's own behavior in order to make easier the interaction with another agent are called *coordination smoothers* [Vesper 2010]. Actually, Vesper *et al.* identify two types of coordination smoothers: either slight changes in an agent behavior — called behavior modulations — to ease the coordination or the use of special objects of which affordances invite to coordination. In the case of navigation, we are interested in the former as no objects are manipulated in our examples. They define four types of behavior modulations:

- First, modulations making the behavior more predictable, by reducing the variation of a repetitive motion for instance.

- Then, they identify modulations delimiting and structuring the other agent tasks. Executing motions staying far away from the human allows to delimit both workspaces.

- Another type is coordination signal, it includes legible motions allowing to infer the goal quicker.
- Finally, synchronization is also identified as a coordination smoother. Copying and synchronizing motions between agents allow for a increased predictability.

It is clear that a robot should exhibit some coordination smoothers when interacting with a human to increase its usability. Moreover, all the coordination smoothers are not equal, as some can bring more information than others. A simple blinking light and beeping sound when the robot is moving are conveying less information than turn signals for example. In our case, since we deal with anthropomorphic robots, we can try to make even more efficient coordination smoothers by using what can be identified as the head of the robot.

2.2.2 Communicating Intents via the Robot Gaze

Some robots have a movable part which can be identified as an head, and often contains camera or similar devices that can be recognized as eyes. The resulting robot *gaze* has already been used to effectively increase the user attention and engagement [Mutlu 2006], [Zaraki 2014]. Besides, it can be used to show what the robot is attentive to and what it is monitoring [Breazeal 2005]. Moreover, the robot gaze has also been shown to be useful in navigation to indicate turning intentions [Lu 2013, May 2015], and thus increase legibility. More precisely, May *et al.* compare two navigation intention signals during a crossing with a human: the head motion and a blinking light on the side the robot planned to move to. They find that turning signals are more effective and that the human is more comfortable when the robot uses them compared to head motion. However, we argue that their experimental setup does not require precise turning indication, as the area in which the human and the robot cross is wide, and only showing roughly if the robot is going left or right is enough to ease the crossing. This may not be true in cluttered environment where showing more precisely the planned trajectory is required.

On the other hand, Lu and Smart propose a gaze behavior where the robot alternatively cycles between looking straight ahead and looking at a detected human [Lu 2013]. They performed a user study where the human and the robot cross each other in a narrow corridor. The study reveals that when the robot gaze is alternating between the human and ahead of the robot, the crossing is less efficient (*i.e.* the human goes slower) than if the robot gaze is only straight ahead. They hypothesize that the head behavior was distracting as it may have stayed on the human for too long, which may have been interpreted by the human as an intent to start an interaction. This is supported by the definition of *civil inattention* coined by Goffman specifying that eye contacts made between strangers to manifest their mutual awareness of their presence are kept under a certain duration to avoid opening a possibility of further interaction [Goffman 1966].

Finally, Khambhaita *et al.* present a framework dedicated to decide where to look when a robot is navigating [Khambhaita 2016]. They combine both looking

at the trajectory and at the human behaviors. Through a video user study they proves that looking at the trajectory improves the robot legibility as users are more able to predict the destination of the robot. Moreover, when the robot glances at the human, the satisfaction of users is enhanced as the robot acknowledges the presence of the human. However, the study has been done in video, putting the user as an observer of a robot crossing with another human. We can argue that it is largely different from being directly interacting with the robot as the cognitive load may not be the same and the physical presence of the robot may also impact the human perception. We thus propose to further explore gaze behavior through the user study reported in this chapter.

All the previous contributions support the claim that, in intricate collaborative activities, each agent must show to the other one that they are aware of their presence and actions. Pacherie defines it as the mutual manifestness: *each subject must be aware, in some sense, of the event as an event that is present to both; in other words the fact that both are attending to the same object or event should be open or mutually manifest* [Pacherie 2011]. Thus, it is interesting to know if in intricate human robot navigation tasks, making the robot show mutual manifestness increases the efficiency of the task.

To do so, we will use planning for both agents to make the trajectory more legible and also design a gaze behavior inspired from previous work.

2.3 The Human Aware Timed Elastic Band

The only work to our knowledge being able to, in real time, plan trajectories for the robot and the humans surrounding it, is the Human-Aware Timed Elastic Bands (HATEB) [Khambhaita 2017]. Thus, we used it as the backbone of our work, and made several contributions to it by implementing it on different robots and integrating it in a complete robotic architecture.

The idea of HATEB is to not only plan a trajectory for the robot while accounting for the current position of humans, but also planning a trajectory for the human. Using an elastic band optimization scheme, the planner is able in real time to generate a trajectory for the robot and a trajectory for the human. By doing so, the planner is able to find solutions where both agents must make an effort, whereas other approaches only planning for the robot would fail. Moreover, it allows to express constraints between the trajectory of the robot and the trajectory of the human, not only considering the human as static or moving linearly but also accounting for their decision capabilities (provided with an accurate enough human model). Finally, by planning for the human, the robot can compute the effort taken by the human (*e.g.* trajectory length and threat induced by the robot) and try to minimize it by balancing between multiple solutions.

2.3.1 General Scheme

The human aware timed elastic band algorithm is based on the Timed Elastic Band (TEB) approach from Rosmann et al. [Rosmann 2013]. This approach is a local optimization problem where the successive positions $(x_i, y_i) \in \mathbb{R}$ and orientations $\theta_i \in S^1$ of the robot along with the time steps $\Delta T_i \in \mathbb{R}$ between each consecutive poses are optimized to minimize a multi criteria cost function up to a fixed length horizon $n \in \mathbb{N}$. To put it otherwise, the elastic band trajectory of the robot is represented by its poses:

$$Q = \{\mathbf{s}_i\}_{i=0..n} \text{ with } \mathbf{s}_i = [x_i, y_i, \theta_i]^T$$

to which are added the time intervals between two consecutive poses:

$$\tau = \{\Delta T_i\}_{i=0..n-1}$$

Resulting in the *timed elastic band*

$$B := (Q, \tau)$$

having to be optimized to minimize the cost function f to get the optimal trajectory

$$B^* = \underset{B}{\operatorname{argmin}}\ f(B)$$

This function takes the form of a multi criteria weighted sum cost function which can be rewritten as:

$$f(B) = \sum_k \gamma_k f_k(B)$$

where $\gamma_k \in \mathbb{R}$ are weights allowing the designer to balance the importance between cost functions f_k.

This planner has been integrated has a local planner in the ROS move base architecture. Provided with a global plan (often generated with an A* algorithm) of the long trajectory, the local planner generates short term plans (up to several meters), avoiding static and dynamic obstacles (both known by the global planner and discovered with the robot sensor during the navigation) and minimizing the trajectory duration. In addition, the local planner is responsible for generating the speed command at position control rate (around 10 Hz usually). TEB does it by optimizing the local trajectory and computing the wanted robot speed from the first two poses and the time interval between them. Moreover, if the optimization process takes too long, the length horizon of the global trajectory on which the local optimization is performed is reduced, and increased if the optimization time is satisfactory.

In the Human-Aware Timed Elastic Bands (HATEB), multiple timed bands are considered. In addition to the robot band $B_{\mathcal{R}}$ representing the robot trajectory, it also considers multiple human bands $B_{\mathcal{H}_k}$ with $k \in \mathbb{N}$ the number of humans in

vicinity of the robot (simple circles on Figure 2.1). For simplicity purpose, in this thesis we will only consider one human in the vicinity of the robot, and thus one human band $B_{\mathcal{H}}$. However, the approach has been shown to be working successfully up to three humans. Moreover, the weighted-sum cost function becomes:

$$f(B_{\mathcal{R}}, B_{\mathcal{H}}) = \sum_a \gamma_a f_a(B_{\mathcal{R}}) + \sum_b \gamma_b f_b(B_{\mathcal{H}}) + \sum_c \gamma_c f_c(B_{\mathcal{R}}, B_{\mathcal{H}}) \tag{2.1}$$

where fa, fb and fc represent cost functions associated with respectively robot trajectory constraints, human trajectory constraints and human-robot social constraints (rectangles on the Figure 2.1). Then, the optimization process consists in finding the optimal robot and human trajectories $B_{\mathcal{R}}, B_{\mathcal{H}}$ such as:

$$\{B^*_{\mathcal{R}}, B^*_{\mathcal{H}}\} = \underset{\{B_{\mathcal{R}}, B_{\mathcal{H}}\}}{\operatorname{argmin}} \; f(B_{\mathcal{R}}, B_{\mathcal{H}})$$

2.3.2 Constraints

In this optimization scheme, all the constraints are represented as cost in the function. Thus, there is no *hard constraints*, but using the weight of each one, we are able to prioritize some over the others. Moreover, when a trajectory has been optimized, before being executed, the local planner checks that it respects all the defined hard constraints (kinodynamic constraints and obstacles clearance).

The new formulation of Khambhaita et al. allows to separate the constraints into three categories:

- Robot trajectory constraints: these constraints represent the robot kinodynamic constraints (non holonomic, maximum speed, maximum acceleration) as well as preventing the robot trajectory to differ too much from the global plan. Examples are c_{vel_r}, c_{acc_r}, c_{kin_r} and c_{obs} on Figure 2.1. They are presented in [Rosmann 2013].

- Human trajectory constraints: these constraints represent the human kinodynamic constraints and prevent them to differ too much from the global trajectory. They are the same as the robot ones, but their parameters (*e.g.* maximum speed threshold) must not only be set by the designer but also refined by the robot during the execution. They are represented as c_{vel_h}, c_{acc_h}, c_{kin_h} and c_{obs} on Figure 2.1.

- Human-robot social constraints: these constraints represent how the human and robot trajectory must interact with each other. Khambhaita et al. presented the *safety* constraint, ensuring a sufficient distance between the robot and the human (c_{safety} on Figure 2.1); the *directional constraint* discouraging trajectories where the robot and the human move straight to each other (c_{dir} on Figure 2.1); and the Time-to-Collision (TTC) constraint, preventing the robot and the human to adopt speeds which, if maintained, would lead to a

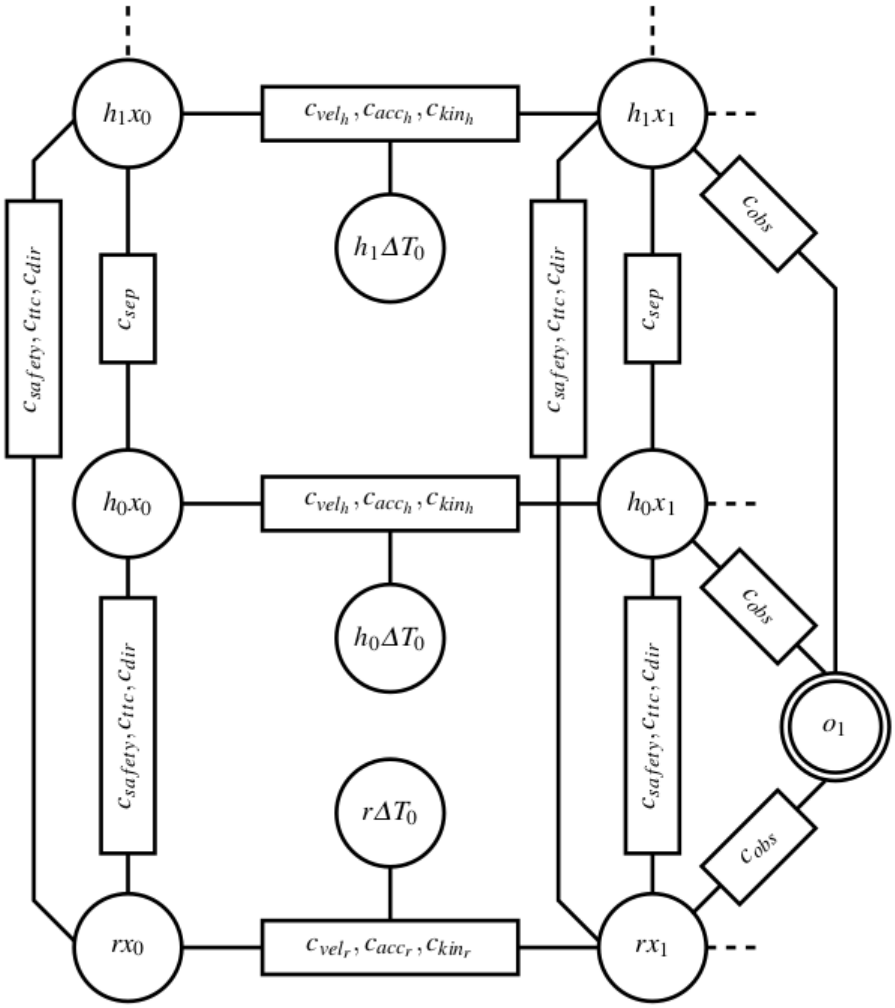

Figure 2.1: Representation of the hypergraph being optimized from [Khambhaita 2017]. Simple circles represent variables to optimize (poses of agents and duration between two consecutive poses). Double circles represent fixed poses (obstacles). Rectangles are the cost functions linked to the variables they need to be computed. The total cost of the graph is computed using a weighted sum of all the constraints. This approach allows to express constraints not only on the robot trajectory ($\mathcal{M}^R$), but also on the human ones (modeling the human behaviors, $\mathcal{M}_r^H$) and more importantly on the interaction between the multiple agents trajectory.

collision (c_{ttc} on Figure 2.1). Indeed, the idea is that trajectories containing high speeds pointing directly towards the human can increase the feeling of threat. The latter will be detailed in what follows as it was studied more in depth through a user study.

It is worth noting that different weights can be set for each constraint, and that they can be adjusted dynamically during the navigation. Moreover, by setting different weight between the robot and the human for the constraint preventing to move away from the global plan, we can adjust the *stiffness* of the trajectories, thus allowing one agent or the other to elongate their trajectory, taking more or less effort into the collaborative navigation (Figure 2.2).

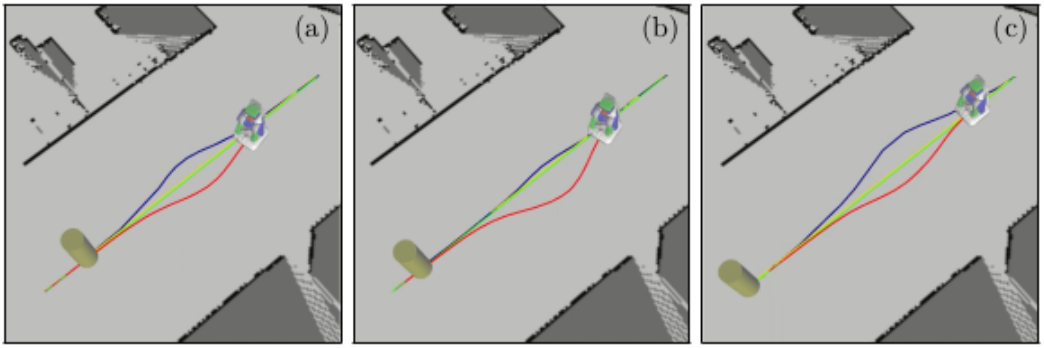

Figure 2.2: Different trajectory stiffnesses from [Khambhaita 2017]. In (a), the stiffnesses of the robot and human are set to be equal, resulting in the robot and the human altering similarly their trajectories. In (b), the robot stiffness is much lower than the human one, resulting in the robot taking most part of the effort needed to avoid the human. In rare cases, where the robot is an emergency for example, the stiffness of the human can be set lower than the robot one (c).

In what follows we want to use this approach to study if planning for both agents can allow to make more proactive and more legible robot trajectories and if in turn this would result in a more efficient and satisfactory navigation for the human. We hypothesize that such an approach is required in intricate navigation scenarios where human and robot must cooperate to navigate successfully. This is the case in confined locations.

2.4 Evaluating Enhanced Mutual Manifestness in a Crossing Scenario

The user study reported in this section has been realized with Nathan Compan, intern in psychology at the Cognition, Langues, Langage, Ergonomie (CLLE) laboratory (University of Toulouse Jean Jaurès), Loïc Caroux associate professor in cognitive ergonomics at the CLLE laboratory and Ophélie Carreras associate pro-

fessor in cognitive psychology at the CLLE laboratory. A journal article has been submitted and is currently under review for the *IEEE Transactions on Human-Machine Systems Journal*.

In this section we present a user study aiming at assessing the pertinence of using a conavigation planner in a situation where a human and a robot must cross each other in a narrow corridor. This task of crossing in narrow corridor is challenging as both agents start in the center of either end of the corridor, and there is no way for one agent to find a way if the other agent does not move to the other side. Thus, we state that not only coplanning is required to find a plan reaching the other end of the corridor (by planning that the other agent will also cooperate and move on one side), but showing intentions and awareness of the other agent is crucial for the interaction to unfold without trouble.

2.4.1 Robot Behavior Design

For this user study we were particularly interested in finding if and how navigation coplanning would lead to higher mutual manifestness and to higher efficiency in crossing. To do so, we designed a robot behavior using the HATEB navigation planner. In their work Kambhaita *et al.* showed that during a narrow crossing the robot is able to plan that the human and the robot will choose opposite sides of the corridor. But if the robot shows its plan when it faces the human, they would have little time to react, and might also move to the same side as the robot, needing negotiation and replanning, reducing the overall efficiency of the crossing. The robot must thus, indicate the plan (*i.e.* the planned trajectory, or here, the side of the corridor it plans to take) early enough in the crossing.

As defined in [Khambhaita 2017] the TTC cost for two consecutive points of simultaneous parts on the trajectory of the robot and the human is defined as:

$$f_{ttc}(ttc, \tau, \varepsilon) = \max(0, \frac{\tau + \varepsilon - ttc}{C^2}) \tag{2.2}$$

where τ is the minimum duration allowed for the time-to-collision before penalizing the configuration, ε is the tolerated gap between the computed ttc and τ and ttc is the computed duration remaining before a collision between the human and the robot if they both maintain the speed vectors they have at the considered points of the trajectories, it is infinity if no collision is computed (the cost is thus equal to 0). The complete TTC cost is the addition of the all the f_{ttc} for all the points on the trajectories.

By reducing the TTC constraint function threshold and increasing its weight, we discourage trajectories where the robot and the human are facing each other at short distance or at high speed. Thus, if the robot trajectory stiffness is lower than the human one, the robot will move to the chosen side of the corridor early in the trajectory as shown in Figure 2.3.

Moreover, as stated before several papers show that using the *head* of a robot can significantly improve legibility and mutual manifestness. Thus, we also chose to

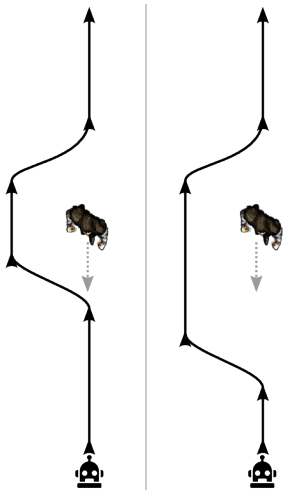

Figure 2.3: Influence of the modification of the TTC constraint cost weight on the trajectory. On the left, the weight is low, the robot will show the side and avoid the human at the last moment. On the right, the weight is high, the robot will show the chosen side and avoid the human much earlier.

make the robot look at its future planned trajectory as shown in Figure 2.4. This is possible thanks to the HATEB algorithm which, unlike many other local planner only publishing speed commands, also publishes a precise short-term trajectory. Finally, to show the robot awareness of the human presence, we made it *glance* at the human twice when they enter a large and a small radius circles both centered on the robot.

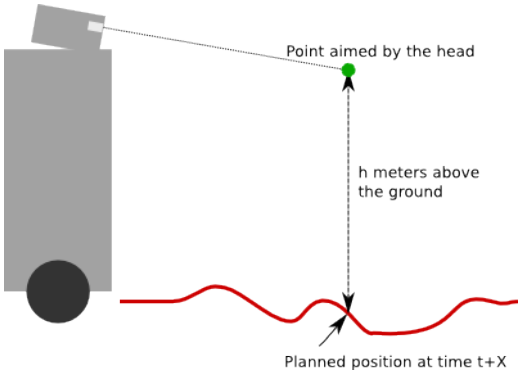

Figure 2.4: Behavior implemented for the robot head. The robot *looks* at a point placed at its planned position X seconds in the future and h meters above the ground.

2.4.2 User Study Protocol

2.4.2.1 Objective

The aim of this study was to evaluate the impact of the TTC cost constraint and head behavior on usability. We designed a user study where actual naive users have to walk through a corridor facing a fully autonomous navigating robot. The afore-explained robot behavior was used. We measured the quality of the crossing between the robot and the human with both objective (visual behavior) and subjective data. The subjective evaluation was based on three dimensions: (1) perceived efficiency of the robot navigation, (2) user satisfaction and (3) situation awareness.

2.4.2.2 Participants

We recruited a total of 28 participants (12 males and 18 females) aging from 21 to 41 (mean: 27.32, SD: 4.13). All 28 participants had never used or interacted with a PR2 for navigation tasks, and had a neutral or good vision of robotics (mean: 5.96 over a 7 points Likert scale, SD: 1.07). This research complied with the tenets of Declaration of Helsinki [Association 2013]. Informed consent was obtained from each participant.

2.4.2.3 Material

A Willow Garage PR2 robot, at its lower spine position was used in this experiment. The robot measured 1.33 meters from ground to top. The entire robot can be considered as anthropomorphic and possesses a two degrees of freedom head integrating cameras resembling eyes.

The participant position was tracked using an Optitrack motion capture system[3], tracking a worn solid headband. This system allowed the robot to track the human anywhere in the room, without looking at them.

The experiment was conducted in a L-shaped corridor (Figure 2.5). The participant and the robot started from opposite sides of the corridor. The participant had to walk 6 meters before entering the long straight corridor part and seeing the robot, then walk 13 meters.

We used a *ETG 2w* eyetracker from SMI to collect the eye movement data of the participant. It is a portable device, allowing, after a short calibration process, to track the user gaze, and measuring where the user looks at. The data were analyzed using the *BeGaze 3.6* software from SMI.

Three questionnaires and an interview were used to collect the subjective measures. They are available as submitted to the participants in French along with a proposed translation in Annex A.

- Pertinence of robot decision: The Pertinence of Robot Decisions In joinT Action (PeRDITA) questionnaire [Devin 2018], jointly developed between the

[3] https://optitrack.com/

Figure 2.5: The study environment. The participant has to go from the yellow cross marked on the ground to the green square also marked on the ground, which is the robot starting point. The crossing occurs roughly in the area where the robot and participant are on the picture. Trajectories are displayed on the picture for example only and are not marked on the ground or suggested by the experimenters at any time.

LAAS-CNRS and the CLLE in Toulouse, France, aims at evaluating the participant perceived pertinence of robot decision during a human robot collaborative task. In its complete form, it measures 5 dimensions: interaction, competence perception, verbal, acting and collaboration. However, in this study the robot is mute, and as the dimensions are independent we chose to remove the verbal dimension.

- Situation Awareness: Several techniques exist to measure the situation awareness during a task [Endsley 1988]. However, they require to freeze and hide the situation to the user, and probe their working memory by questioning them about its near future. In our setup, we cannot stop the robot and make it disappear while it is navigating. Thus, we have developed a series of 6 questions for measuring the user situation awareness. These questions are presented to the user just after the navigation, and asked them to rank on a 6 points Likert scale each 3 stages (2 questions per stage) of the Endsley's model: perception, comprehension and projection.

- User satisfaction: For measuring the user satisfaction we used the AttrakDiff questionnaire. It is a standardized User Experience (UX) questionnaire measuring both hedonic qualities and global attractiveness. We used the french translation of this questionnaire [Lallemand 2015].

- Interview: The interview was constituted of 8 semi directed questions always asked in the same order. These questions aimed at qualitatively evaluating the user experience, behavior and perception of the user during the navigation. The interviewer was only allowed to read the questions and to make the participant elaborate by asking neutral questions like "why?" or "can you tell me more?".

2.4.2.4 Experimental Design

The user study was a 2 × 2 within-participants user study to evaluate how the time-to-collision constraint and the head behavior impact the robot navigation effectiveness efficiency and satisfaction. The independent variables were the HATEB time-to-collision cost parameters (both weight and threshold) and the head behavior. The conditions for the time-to-collision variable were $\gamma_{ttc} = 0.01$ (in Equation 2.1) with $\tau = 1s$ (in Equation 2.2) for the *low TTC* condition and $\gamma_{ttc} = 15$ (in Equation 2.1) with $\tau = 4s$ (in Equation 2.2) for the *high TTC* condition. For the both *continuous* and *alternated* head behavior conditions the robot head was pointing towards the robot planned position in 1.5s in the future at 1m above the ground. In addition, in the *alternated* head behavior, the robot pointed its head towards the human when they entered the long part of the corridor during 1.5s and again during 1.2s when the robot and human were 3.5m apart. The conditions *low TTC* with *continuous* head behavior and *high TTC* with *alternated* head behavior are presented in Figure 2.6 (a) and (b) respectively.

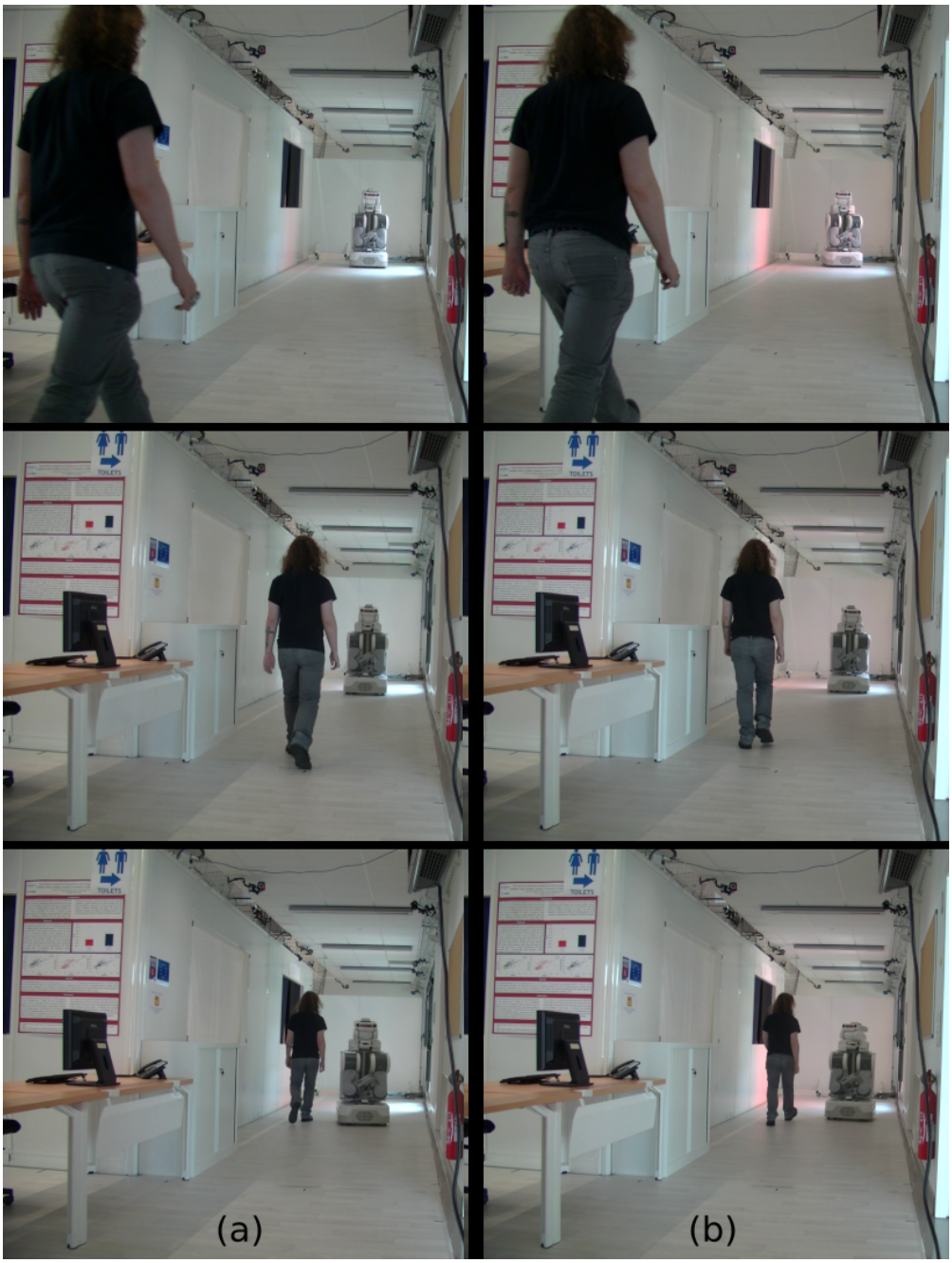

Figure 2.6: Two typical crossings. In column (a) the robot behavior is set to the condition *low TTC* for the trajectory and *continuous* for the head. In column (b) the conditions are *high TTC* for the trajectory and *alternated* for the head. In (b), thanks to the TTC cost, the robot starts to move to one side of the corridor very early, whereas in (a) the robot moves almost a the last moment. Moreover, in (b) the robot glances at the human when he enters in the corridor, and once again when he is nearby. In (a) the robot only looks at its future position.

The participant goal position was marked with a square on the ground, and was the starting point of the robot. The robot final position was 10m straight ahead of its starting position. So, the participant was able to reach their natural walking speed before turning at the corner of the L shaped corridor. The robot was only started when the participant was about to turn (2m before the turn), giving the impression that the robot was coming from further away while ensuring that the crossing happened around the same place independently of the participant walking speed.

2.4.2.5 Study Procedure

The evaluation was cut into 4 blocks. A block consisted in two same condition crossings followed by questionnaires filling. A crossing was composed by the placement of the participant and the robot on their respective starting positions, then the participant was free to go to their previously indicated goal location while crossing the robot. The three questionnaires were filled next to the participant starting location and concerned only the two crossings made in the current block. The 4 conditions order were randomized between participants and the condition change was made between two blocks but never between the two crossings inside the same block.

Before starting the experiment trials, a training trial was made with the robot starting shifted to one side of the corridor and going in a straight line with its head fixed looking straight ahead. Just after this training trial, the participant was brought close to the robot and invited to inspect it. A specific head behavior was triggered making the robot head to follow the human allowing the participant to notice without being told that the robot was able to know their position and that its head could move. Moreover, the experimenter showed that robot arms were locked in place in a tucked position, and that they kept the emergency stop remote and was able to stop the robot at any time.

After the 4 blocks have been passed by the participant, the experimenter interviewed them. The audio was recorded and the answer written down.

The whole study lasted around 45 minutes per participant.

2.4.2.6 Measures

The analysis of the data was made on 27 participants because one did not fill all the questionnaires and their data were thus removed from the study. The quantitative data (questionnaires and oculometry) were analyzed using a non parametric two-way repeated measures Friedman ANOVA test.

1. *Questionnaires*: The three questionnaires have been passed 5 times each (one trial + four blocks). The results were codified from 1 to 7 for the PeRDITA, from 0 to 6 for the AttrakDiff and from 1 to 6 for the situation awareness questionnaire while taking care of reordering inverted items.

The PeRDITA Cronbach's alphas were for each dimension: $\alpha = 0.89$ for the interaction, $\alpha = 0.87$ for the competence, $\alpha = 0.85$ for the acting and $\alpha = 0.86$ for the collaboration.

For the situation assessment questionnaire, the Cronbach's alphas were: $\alpha = 0.93$ for the perception, $\alpha = 0.88$ for the comprehension and $\alpha = 0.87$ for the projection.

2. *Oculometry*: The oculometry data were split into two parts: before the robot crossing, and after the robot crossing (when the robot is behind the human). As we were not interested in where the participant gaze after the crossing occurred we did not analyze this part. The number and duration of fixations were measured for each of the 9 defined Area of Interests (AOIs) (Figure 2.7). The semantic gaze mapping method was used, it consists in manually selecting in which AOI each automatically detected fixation lays.

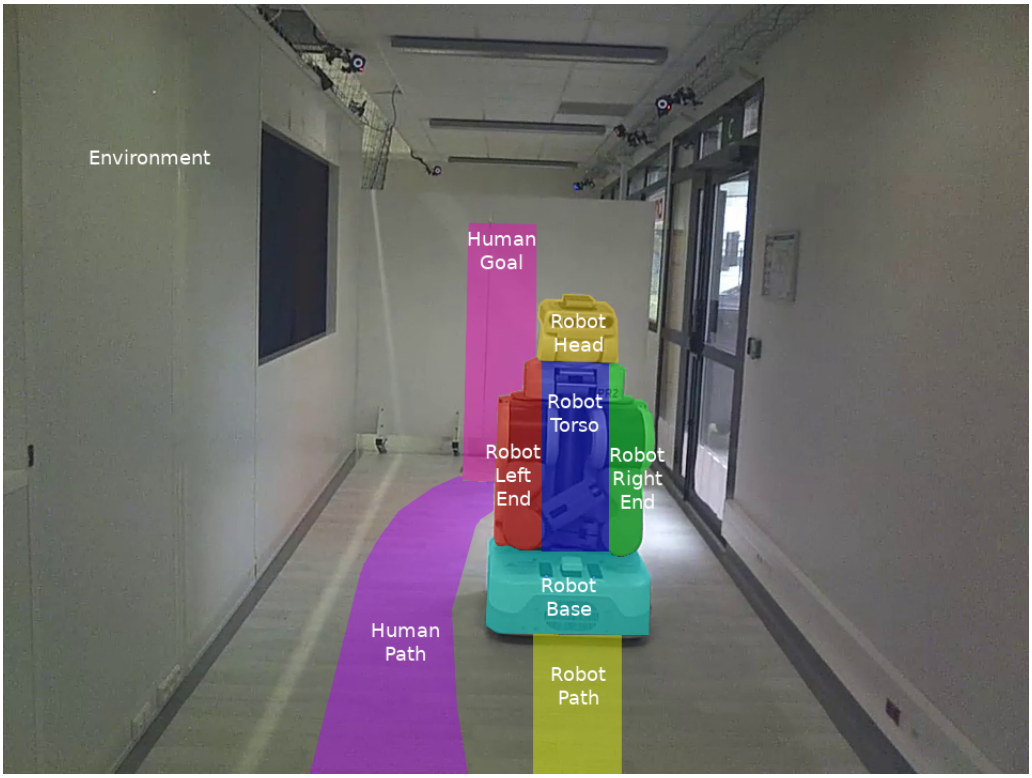

Figure 2.7: The areas of interest defined to analyze the participants oculometric data.

3. *Interview*: The first question was a control question, ensuring that the participants saw differences between each blocks. After the second question, the interviewer revealed that the robot behavior was indeed different between each block, then proceeded to the rest of the interview. The participants answers were analyzed by excerpting verbatim (keywords or general ideas) from each

question and computing their frequencies.

4. *Dependent variables*: Therefore, the dependent variables were the participant scores at the three questionnaires: PeRDITA, Situation Awareness and AttrakDiff, in addition to the number and duration of their gaze fixation during the crossing. Prior to the experiment, the participants were asked to fill a questionnaire inquiring their age, gender, education level, profession, native language, an open question about past experience with robots and a 7 points Likert scale assessing their overall opinion about robotics.

2.4.3 Results

2.4.3.1 Quantitative Results

During the crossing almost all participants (95%) went to their left (thus, letting the robot go to their right) because the experimental setup led them to do so as the interview revealed. The robot also planned that the optimal path (given the constraints described above) was to go to the right of the human. Participants who went to their right stated they were "testing" the robot, thus not respecting the given goal (which was to go to the marked goal and not to test the robot). These trials have been removed from the data.

Pertinence of Robot Decision In joinT Action The mean scores of the three dimensions of the PeRDITA questionnaire were significantly higher with an alternating head behavior than with a continuous one. The quality of interaction was higher with an alternating head behavior ($M = 5.31$, $SD = 0.97$), than with a continuous head behavior ($M = 5.03$, $SD = 1.17$), $F(1,26) = 4.41$, $p < .05$, $\eta_p^2 = .15$. The perceived robot competence was higher with an alternating head behavior ($M = 5.12$, $SD = 1.14$), than with a continuous one ($M = 4.73$, $SD = 1.14$), $F(1,26) = 9.15$, $p < .01$, $\eta_p^2 = .26$. The quality of collaboration was higher with an alternating head behavior ($M = 5.06$, $SD = 1.04$), than with a continuous one ($M = 4.73$, $SD = 1.14$), $F(1,26) = 6.07$, $p < .05$, $\eta_p^2 = .19$.

There was not any significant effect of TTC on the quality of interaction ($F(1,26) = 1.65$, $p = .21$.), on the perceived robot competence ($F(1,26) = 0.04$, $p = .84$), and on the quality of collaboration neither ($F(1,26) = 0.43$, $p = .52$). No interaction between TTC and head behaviors reached significance.

Attrakdiff The mean score of the hedonic qualities dimension was higher with an alternating head behavior ($M = 5.06$, $SD = 0.87$), than with a continuous one ($M = 4.93$, $SD = 0{,}82$), $F(1,26) = 4.13$, $p < .05$, $\eta_p^2 = .14$. There was no significant differences regarding the scores of the pragmatic qualities ($F(1,26) = 0.34$, $p = .57$) and attractiveness ($F(1,26) = 0.14$, $p = .71$) dimensions.

There was not any significant effect of TTC on hedonic qualities ($F(1,26) = 3.25$, $p = .08$), on pragmatic qualities ($F(1,26) = 2.97$, $p = .10$), and on attractiveness

neither ($F(1,26) = 0.37$, $p = .55$). No interaction between TTC and head behaviors reached significance.

Situation Awareness The mean score of global situation awareness was significantly higher with an alternating head behavior ($M = 4.64$, $SD = 0.98$), than with a continuous one ($M = 4.33$, $SD = 1.12$), $F(1,26) = 7.77$, $p < .01$, $\eta_p^2 = .23$.

The mean scores of all the three dimensions of situation awareness were higher with an alternating head behavior than with a continuous one. Perception was significantly higher with an alternating head behavior ($M = 4.60$, $SD = 1.08$), than with a continuous one ($M = 4.15$, SD = 1.29), $F(1,26) = 8.20$, $p < .01$, $\eta_p^2 = .24$. Comprehension was higher with an alternating head behavior ($M = 4.64$, $SD = 1.05$), than on a continuous one ($M = 4.36$, $SD = 1.14$), $F(1,26) = 5.99$, $p < .05$, $\eta_p^2 = .19$. Projection was higher (strong tendency) with an alternating head behavior ($M = 4.67$, $SD = 0.96$), than on a continuous one ($M = 4.47$, $SD = 1.16$), $F(1,26) = 3.52$, $p = .07$, $\eta_p^2 = .12$.

There was not any significant effect of the TTC on global situation awareness ($F(1,26) = 1.70$, $p = .20$), on the perception dimension ($F(1,26) = 0.72$, p = .41), and on the comprehension dimension ($F(1,26) = 0.60$, $p = .45$). However, the mean score of the projection dimension was higher (strong tendency) with a high TTC ($M = 4.69$, $SD = 0.98$), than with a low TTC ($M = 4.44$, $SD = 1.13$), $F(1,26) = 4.03$, $p = .06$, $\eta_p^2 = .13$. No interaction between TTC and head behaviors reached significance.

Gaze

- *Robot vs. Environment*

  The mean number of eye fixations was significantly higher on the robot ($M = 2.23$, $SD = 0.98$) than on the environment ($M = 1.17$, $SD = 0.90$), $F(1,21) = 17.82$, $p < .001$, $\eta_p^2 = .46$. The mean number of eye fixations was higher (strong tendency) with a high TTC ($M = 1.76$, $SD = 1.04$) than with a low TTC ($M = 1.65$, $SD = 1.11$), $F(1,21) = 3.86$, $p = .06$, $\eta_p^2 = .16$, regardless of the AOIs. There was not any significant effect of the robot head behavior on the mean number of eye fixations, $F(1,21) = 0.90$, $p = .35$. No interaction between the three factors reached significance.

  The mean average duration of eye fixations was significantly higher on the robot ($M = 351$, $SD = 172$) than on the environment ($M = 260$, $SD = 123$), $F(1,18) = 6.95$, $p < .05$, $\eta_p^2 = .28$. There was not any other significant effects or interactions regarding the average duration of eye fixations.

- *Robot AOIs*

  As shown in Table 2.1, the head was the part of the robot that was the most fixated by the participants ($M = 6.13$; $SD = 0.59$), $F(4,84) = 39.10$, $p < .001$, $\eta_p^2 = .65$. The number of fixations on other parts were $M = 1.28$ (SD

= 0.31) for the base, $M = 1.16$ ($SD = 0.23$) for the torso, $M = 0.95$ ($SD = 0.19$) for the right end, and $M = 1.64$ ($SD = 0.37$) for the left end.

Robot AoI	Low TTC		High TTC	
	Continuous	Alternating	Continuous	Alternating
Head	5.64 (3.42)	5.50 (2.78)	5.93 (2.90)	7.45 (3.80)
Torso	1.39 (1.39)	1.14 (1.49)	1.39 (1.65)	0.73 (0.96)
Left end	0.95 (1.20)	1.07 (1.11)	1.00 (1.15)	0.77 (1.14)
Right end	1.82 (1.92)	1.55 (1.71)	1.55 (1.94)	1.64 (2.27)
Base	1.36 (2.07)	1.23 (1.53)	1.45 (1.65)	1.07 (1.27)

Table 2.1: Mean number of eye fixations made by participants in each Area of Interest (AOI) of the robot in each experimental condition (TTC x robot head behavior) in main experiment. Standard deviations are shown in parentheses.

There was a significant interaction between the AOI type and the type of TTC, $F(4,84) = 5.72$, $p < .001$, $\eta_p^2 = .21$. The robot head was more fixated when the TTC was high ($M = 6.69$, $SD = 3.43$) than when it was low ($M = 5.57$, $SD = 3.08$). There was an interaction (strong tendency) between the AOI type and head behaviors, $F(4,84) = 2.34$, $p = .06$, $\eta_p^2 = .10$. The robot head was more fixated, when the robot head was alternating ($M = 6.48$, $SD = 3.43$) than when it was continuous ($M = 5.78$, $SD = 3.14$).

In addition, there was a double interaction (strong tendency) between the AOI type, head behaviors and the type of TTC, $F(4,84) = 2.41$, $p = .06$, $\eta_p^2 = .10$. Table 2.1 shows that the robot head was more fixated when the robot head was alternating only with a high TTC.

There was not any significant effect of the robot head behaviors on the mean number of eye fixations, $F(1,21) = 0.13$, $p = .73$. There was not any significant effect of the TTC type on the mean number of eye fixations, $F(1,21) = 1.82$, $p = .19$. The interaction between the TTC and the head behaviors, $F(1,21) = 0.74$, $p = .40$, did not reach significance.

The ANOVA for the mean duration of eye fixations was not possible due to an absence of data in some experimental conditions.

2.4.3.2 Qualitative Results

All the participants saw differences between blocks, 19 of them saw head behavior differences and 10 of them saw trajectory differences. When the participant talked about the alternated head behavior, positive adjectives were employed ("reassuring", "sympathetic", "interactive"[4]), when they talked about the continuous head behavior, negative adjectives were employed ("unsettling"[5]).

[4] "sécurisant", "sympathique", "interactif" in French
[5] "troublant" in French

After the experimenter revealed that the robot behavior was different in each block, the participants preferred when the robot was "moving its head" (12 participants). Five participants also preferred when the robot changed its direction long before they cross. Participants did not appreciate when the robot kept a "fixed head" because they thought the robot was not aware of them (8 participants), when the robot was "hesitating" (4 participants) and when it came "too close" (4 participants).

Fourteen participants found the robot was the most competent in the condition with the alternated head behavior and high TTC, because the robot became aware of them when looking at them (10 participants), because the robot changed its direction sooner (4 participants), because it was trying to avoid them (4 participants) or because it was not coming too close (3 participants).

The participants found the robot more acceptable when it made a "visual contact" (4 participants) and when it changed its trajectory early (3 participants). They found the robot less acceptable when the robot came too close (6 participants), when hesitating (3 participants), when not making visual contact (2 participants).

2.4.4 Discussion

This study aimed at exploring how taking into account the human model and planning for both human and robot during navigation allows to easily design behavior enhancing the usability of the robot. More precisely, navigation coplanning allows to easily implement coordination smoothers and increase mutual manifestness, we thus hypothesized that these changes should lead to a significant impact on robot usability in the intricate scenario of narrow corridor crossing.

Human gaze analysis showed that the head of the robot was fixed many times and even more when the robot showed its chosen corridor side early in the navigation (with a high TTC). Moreover, subjective results revealed that moving the head of the robot in an alternating pattern (pointing towards its path while sometimes pointing at the human head) improved the perception of the quality of interaction, the perceived robot competence, and the quality of collaboration. This alternative head behavior also increases the situation awareness score. **This strongly supports previous results on using the robot head during human robot navigation scenarios** [Khambhaita 2016, May 2015].

However, the satisfaction seems to have been slightly improved (on hedonic quality) when the robot presented the alternating gaze behavior rather than the continuous one. This result is only a tendency, but in the interview, the participants identified as positive when the robot moved its head to glance at them and changed its trajectory. They also identified as negative when the robot was too close or did not look at them. We think these discrepancies between the AttrakDiff results and the interview are caused by our poor questionnaire choice. Indeed, the AttrakDiff questionnaire aims at measuring the satisfaction produced by a final product (*e.g.* cellphones) and the willingness of a user to buy this product, but not the satisfaction of a user when dynamically interacting with a robot in a navigation

scenario. Yet, as no other questionnaire to our knowledge proposed to measure user satisfaction during human robot interaction, we chose the AttrakDiff.

Results show that using the HATEB navigation co-planner with a high time-to-collision weight constraint cost and low threshold, alongside an head behavior signaling future robot trajectory and the human awareness during a crossing in a constrained space allows the human to have a better situation awareness and to perceive the robot as acting more pertinently and being more competent. **This finding supports the result from Lu et al.** [Lu 2013], where the navigation efficiency is increased when the robot looks at the human and shows a "social" navigation behavior.

We speculate that both the high TTC and alternating gaze of the robot are needed to improve the interaction and that no simple effects are significant for multiple reasons. First, with only the robot choosing a corridor side early in the crossing (high TTC condition) without showing the human awareness (continuous head condition), it may not be obvious to the human that the trajectory change is due to their presence. Conversely, when the robot signals its awareness of the human (here by pointing its head towards them) but without taking any action to ease the interaction (low TTC condition), the human stays in a situation where they don't know what the robot will do next. Finally, by both making the robot show its human awareness and change its own trajectory to facilitate the interaction (coordination smoother), it becomes clear that the robot is proposing a co-navigation solution where both agents are avoiding each other.

Finally, oculometry results indicate that when the head is recognized as an information provider (in the alternating head condition), information is most likely to be sought in the robot head motion. The head is also looked longer when the robot alternately looks at the human. We can also note that people tend to think PR2 is getting data through its head, probably because of the visible cameras on it, even when it is not the case (like in our experiment).

2.5 On User Studies in HRI

After this user study, we wanted to report the experience and to discuss it with the community. Thus, we decided, with a PhD student in psychology, Kathleen Belhassein, to participate in a workshop at HRI 2019 entitled *Test Methods and Metrics for Effective HRI*. It seemed interesting to collaborate together in order to have two points of view on user studies, coming from psychology for her, and from HCI and HRI for us. We identified three main challenges concerning user studies in HRI: the users, the evaluation methods and the replication. We concluded by presenting ten recommendations for user studies in HRI [Belhassein 2019]. According to the discussion with the community during the workshop, we updated and submitted a contribution, which has been accepted, to the *Transactions on Human-Robot Interaction* journal.

2.5.1 Users in HRI studies

When conducting a user study, we obviously recruit today's users, with their past experiences and expectations with robotics. Kuhnert et al. showed the existence of a *gap* between the user's attitude towards existing robots and their expectations about the ideal everyday social robot [Kuhnert 2017]. Thus, when evaluating a human robot interaction it is important to evaluate the three aspects defined by Desmet and Hekkert: the instrumental interaction (interaction for expected purpose: always evaluated in current user study), the non-instrumental interaction (interacting for other than main purpose: often non evaluated) and the non-physical interaction (expectations: often under evaluated) [Desmet 2007]. Indeed, non-physical interaction refers to all preconceptions of the robot by the user, they could come from past experiences or imagination. Since today's users have almost no past experiences with robots, these anticipations come mainly from imagination and fantasies and can widely vary from one to another. Thus, it is really important to evaluate the mindset of the user before the study. Some tools used in psychology can be useful in this context. For example, the implicit association test [Greenwald 1998], used to measure automatic and implicit associations like prejudices, or priming paradigms could be used to control the expectations and beliefs of subjects towards robots.

Moreover, as many of the recruited users has none to very few past experiences with robots, the novelty (and *wouaw*) effect when interacting with robot during a one shot study is huge, and may not be representative of a robot long term use. To assess this *cumulative experience*, User Experience (UX) and Human Computer Interaction (HCI) designers conduct *longitudinal studies* [Lazar 2017], gathering data over a long period of time. However, in the human-robot interaction field, this method may not be applicable as is. Indeed, the used material (robots) is expensive and often in limited quantities in laboratories. In order to diminish this effect, the evaluation should be part of a larger, more cognitively intense or time pressurizing task where the user must almost *forget* about everything concerning the robot except the part to be evaluated, which should be crucial in this task. This effect could also be reduced by making an habituation phase at the beginning of the experiment, in which the user can act more freely with the robot.

In some HRI studies, a questionnaire is administered before the interaction task in order to apprehend the degree of familiarity and knowledge of the participants about the robots. Even if some recruited people are from the same professional environment and are therefore already accustomed to robots, this measure of knowledge about robots is never used to remove participants from the user study. In addition to the bias that this recruitment may pose, the sample is therefore not representative of the general population but of a particular sub-population. To ensure a better representation of the population, it would therefore be necessary to randomly sample and recruit outside of the professional circle.

Finally, HRI studies have frequently few participants. For example, about 44% of user studies published in the proceedings of the conference HRI'17 involve fewer than 30 participants. However, the size of the sample is a prerequisite for obtaining

sufficient statistical power to conclude on the results obtained, and to avoid type I errors (false positive) or type II (false negative). Beyond the statistical issues, it seems important to be modest about the conclusions drawn from studies involving too few participants, and therefore not to generalize to the population the results obtained.

The Particular Case of Web Studies

To respond to this difficulty and have a large number of participants, it is now frequent to be confronted to web studies (as in [Khambhaita 2016]), especially via the crowd-sourcing web platform Amazon Mechanical Turk. Indeed, it is then much easier to recruit participants and have them take small tasks or questionnaires directly online. It seems however important to be vigilant on the study conclusions, since we are not in a context of human-robot interaction in its strictest sense; the participant is not confronted to the robot, does not share its physical space, and therefore will not have the same reactions that he would have during an interaction in the real world. Nevertheless, it could be an interesting tool for participatory design studies, *i.e.* all the studies that do not deal with a fully implemented robot that must be evaluated but rather that serve to explore the responses of users to certain specific behaviors and to collect their opinion.

2.5.2 Evaluation Methods

The key concept in HCI is usability, which regroups effectiveness (ability to perform a task), efficiency (ability to perform the task without wasting resources) and satisfaction. More often than not, HRI user studies focus on user satisfaction (and *acceptability*) evaluated with questionnaires created or adapted specifically for this context of interaction or borrowed from HCI. Even if HRI studies join the field of Human Computer Interaction and User Experience design by the fact that they are both trying to improve the human use of interactive systems, it is a hugely different experience to interact either with a robot or a computer. We must not forget that in the case of HRI studies, there are two agents that interact and no longer an agent that interacts with a product/an interface. Therefore, the methods and tools of HCI are not always suitable to be used during a situation of interaction between a human agent and a robotic agent. First of all, people tend to attribute mental states and human traits to robots. This human tendency to anthropomorphism is not only dedicated to robots but also animals, objects or natural phenomenon. However, robots are more perceived as an agent endowed with lifelike qualities than other technologies. In addition, the perceived risks to evolve in the same environment as a robot are obviously not the same than when we use computers or other technologies, and can induce negative emotions or feeling of insecurity [Dautenhahn 2006]. Thus, these particularities in the subjective experience of interacting with a robot have to be considered in the build of experimental and evaluation tools.

2.5.2.1 Use of Self-Assessment Methods

Although the simplest and most widespread evaluation method in HRI studies is the self-assessment method with questionnaires, it is necessary to understand that there may be a significant difference between what subjects self-report of their own experience and what they really experienced and felt, for example because of the social desirability bias [Fisher 1993]. In addition, there are very few questionnaires created for HRI studies that meet the validity and standardization criteria of these methods. Indeed to be validated, a questionnaire must be reliable and consistent, *i.e.* the results must be replicated in comparable situations; it must be valid, that is to say it actually measures what it is supposed to and not another dimension; and finally, it must be sensitive to change. More often than not, HRI questionnaires are simply evaluated on their internal consistency (all the items from one dimension are correlated to each other) with Cronbach's alpha. However, to validate that the questionnaire measures the construct of interest, it is important to use on a pilot study another method of evaluation for this same construct and use correlation matrices between them [Tsang 2017].

Finally, it is common to see questionnaires used in another language. However, if a questionnaire is created in a specific language it has to be used only in that language. It is therefore absolutely necessary when using a questionnaire from another language to use the back translation method (*i.e.* translate back into the original language the questionnaire previously translated into the target language). In addition, the questionnaire once translated must imperatively be validated following the same rules as an original questionnaire, to ensure that the translated version measure the exact same construct as the original, as it was the case for example for the French translation of the UX questionnaire AttrakDiff [Lallemand 2015].

2.5.2.2 Other Evaluation Methods for Acceptability

Including heart rates, brain or skeletal muscles electrical activity, blood pressure, respiratory frequency or even galvanic skin responses, there are a number of different physiological measures that can be used to evaluate the participant's physiological response in touch with a robot. These evaluation methods have the advantage of being able to prevent participants from consciously modifying their answers, as this is the case with self-assessment measures.

In interaction situations and even more in cooperative situations, taking social signals into account may also be useful for assessing the human ability to accept to engage into a task with a robotic partner. Behavior observation techniques are for example used to measure shared gazes, in particular with video recording and eye tracking [Gharbi 2015b].

2.5.2.3 Evaluate Efficiency and Effectiveness in Human-Robot Interaction

But what is the point of making a robot behavior satisfying but useless? Once more, tools exist in HCI to measure the other components of usability but are not always adapted to HRI studies. For example, freeze-probe techniques are frequently used in the case of evaluating the situation awareness (*i.e.* the perception, comprehension and projection of elements in the environment [Endsley 1988]). They consist of freezing the task in progress and administrating a questionnaire about the situation at the exact moment of the freeze point [Salmon 2006]. Such a device, mostly developed for use in the aviation or military domain, is a real challenge and almost impossible to apply in HRI, since we are in the case of a real-world physical interaction with a robotic agent and we cannot just make the robot disappear.

Most of the existing HRI questionnaires deal more with the physical aspect of the robot and its acceptability by the user (*e.g.* the godspeed questionnaire [Bartneck 2009] and the RoSAS questionnaire [Carpinella 2017]) than its effectiveness and usefulness. In previous work, we tried to propose a preliminary version of a questionnaire specific to HRI studies and measuring the decision-making processes of the robot in a joint action context with a human [Devin 2018], but this tool remains at the draft stage and deserves to be refined, coupled with other evaluation methods, used in other studies and finally validated according to the criteria of self-assessment methods.

Moreover, adding efficiency/effectiveness measurements in a study can be pretty simple given the technical abilities inherent to robots (*e.g.* timing the task completion, the number of user errors). Thus, we could consider using the robot as a tool for measuring its impact on the human performance, and therefore evaluate the efficiency of human-robot interaction. Mayima *et al.* presented several metrics, and implemented some of them in a robotic component, to evaluate in real-time the *quality of interaction* [Mayima 2020]. These data can also be analyzed more deeply afterwards to evaluate the interaction. In addition, one of the most widely used methods of cognitive psychology is mental chronometry, which uses reaction times as a measure, and refers to the temporal study of information processing [Posner 1978].

Whether it concerns the evaluation of acceptability or efficiency and effectiveness in human-robot interactions, a more frequent use of objective measures (*e.g.* physiological or task performance measures) could improve the validity of the results of HRI studies and their methodological rigor. These different techniques must still be consistent with each other in their use in a user study, this recommendation is not always feasible but should be taken into account when establishing the methodological plan.

2.5.3 The Replication Crisis in HRI

Replication of results is an important concept for any discipline following the scientific approach; it is a question of repeating a study to determine if the results are reproducible and therefore reliable. Psychology and more generally social and medical sciences have known since the 2000s what is called the replication crisis, which is also slowly preparing in HCI and HRI fields. Indeed, if the replication crisis begins to appear in HCI [Echtler 2018] mainly because of closed source code used in the experiments, HRI presents other issues. First, the code used in robotic studies is often much more complex and heavy to use. Indeed, to evaluate a high-level component (*e.g.* a task co-planning algorithm) a whole component stack is needed (*e.g.* low-level motor controllers, path finding, trajectory following, motion planning, localization, face detection, speech synthesis, speech recognition). If some of these components are standard and widely available, others can be state of the art, unstable or even tweaked by the experimenters to match their own needs, making it more difficult to replicate the experiment if these components are not precisely described or not available. Moreover, the material used can also be changed (*e.g.* by 3D printing a gripper to fit the object manipulation task needed), and components tuned to work accordingly. These small changes on the hardware and on the software need to be reported, else the experiment replication is impossible.

Finally, many user studies in HRI use Wizard of Oz technique, because robotic systems are often not robust enough to act autonomously in an environment with humans. The use of Wizard of Oz technique makes very difficult the exact replication of the situation due to the fact that the complete scenario of the study is not available. In the case of HRI evaluation, we can also ask ourselves if humans evaluate the robot or the human controlling the robot.

2.5.4 Proposed Guidelines for Better User Studies in HRI

By taking all these issues and solutions coming from different fields we might provide some checkpoints when designing a HRI user study:

1. The *more users* the better. It improves the statistic analysis of the study and could erase some bias.

2. *Widen* the recruitment. Your colleagues know a lot about technology: randomly recruit people in your bakery, in the supermarket, using flyers...

3. *Be rigorous* with your protocol. There are so many unwanted uncontrolled variables. Don't be one!

4. Let the user *accustom* to the robot and its behavior before starting the experiment. In order to diminish the "wouaw" effect and make measures closer to a long term use.

5. Make sure your experiment is physically and psychologically *safe*. Apart from hurting a user, you risk to prevaricate your measures if the experimenter is

stressed about something going wrong.

6. Objectively *measure if your robot is useful* in what you are making it do. A robot can be really satisfying, but it will be quickly forgotten if it does nothing.

7. Use the *right tools* for your measurements. Widely used and standardized tools will give more credibility to your study. Questionnaires are not the end.

8. Make theoretically solid and valid *tools* specific to HRI and publish them if they don't exist. It will benefit the whole community.

9. *Give as much details as possible* about your study. Give the source code, hardware schematics or references and the description of the environment in order to make others able to reproduce your experiment.

10. When doing a *Wizard of Oz be rigorous.* Emulate only a small, non-evaluated, component of your robot. Write the rules you follow down, respect them and publish them along with your study.

2.6 Extending HATEB

Closing this general parenthesis about user studies in HRI, we continue on the HATEB navigation planner development. Having shown that providing co-navigation solution using HATEB can effectively be used to enhance the efficiency of the interaction, we will present in this section the different extensions made to HATEB.

2.6.1 Adapting HATEB to Other Robots

Khambhaita et al. [Khambhaita 2016] successfully used HATEB on a PR2 robot, both in simulation and on the real platform. As the approach is general, we implemented it on other robots.

2.6.1.1 Using HATEB on a Legged Humanoid Robot

Legged humanoid robots are characterized by their bipedal navigation, tremendously increasing the complexity of navigation and control with regard to wheeled robots. However, legged robots present a large advantage for social and human robot interaction as human infrastructure are often thought for being navigated with legs rather than with wheels. Thus, we tried to implement our approach on a humanoid robot.

To do so, we partnered with the Gepetto team at LAAS-CNRS specialized in humanoid systems motion. Thanks to their modular software architecture presented in [Stasse 2008], only minor changes had to be done to HATEB to integrate with their other components.

One of their component [Naveau 2017] takes as parameter a (simplified) robot kinematic model and is able, in real-time and at typical joint position control frequencies (around 200 Hz), to give position commands to all the robot joints in order for the center of mass of the robot to respect a speed command given as input. Thus, we can generate a non-holonomic trajectory for the robot with HATEB and send the returned speed commands to this component. Given a speed command, the component is able to generate footstep placements and to compute robot joint positions ensuring the equilibrium and the respect of the kinodynamic constraints (between two successive poses) of the robot. However, this component has no ROS interface whereas HATEB is heavily built for ROS use. Thus, we made a bridge transferring the speed commands output by HATEB to the Gepetto architecture and transferring back the joint position commands.

This has only be run virtually, and for simplicity, no simulator was used. Instead, we assumed the joint position command to be immediately and perfectly executed. Such an assumption has been made possible because of the effectiveness of the Gepetto architecture, having a tested precise model of the robot, ensures that the kinodynamic constraints will be respected, and consequently gives a correct robot behavior.

We tested this approach on a virtual HRP2 robot, with a virtual human (Figure 2.8). We were able to make the robot avoid the human in a face to face crossing, while accounting also for the human ability to avoid the robot. Moreover, using this approach is pertinent as HRP2 has much slower dynamics than PR2, and thus must rely a lot on the ability of the human to avoid it. Indeed, if the robot maximum speed or acceleration parameters were set too high in HATEB, the speed commands generated would make the robot fall. Planning for both the human and the robot allows to generate smoother avoiding trajectories for the robot by assuming the human can take most of the effort.

However, by using HATEB we constrained even more the motion of the legged robot. Indeed, the presented version of HATEB does not support holonomic trajectories, which would be doable by PR2 and HRP2. Using this approach, HRP2 cannot make a step on the side, but has to turn and move forward. The current version of HATEB now integrates holonomic capabilities thanks to Phani-Teja Singamaneni [Singamaneni 2020], but has not be integrated for the HRP2 robot yet.

This preliminary work not only shows the versatility of the approach, but also that planning for both can be useful for a constrained robot. Indeed, with a slow dynamic robot, a classical navigation planner may find no solution to avoid an approaching human, but by planning that the human can make a great effort to avoid the robot, it can generate a trajectory for the robot that elicit a plan for the human to follow.

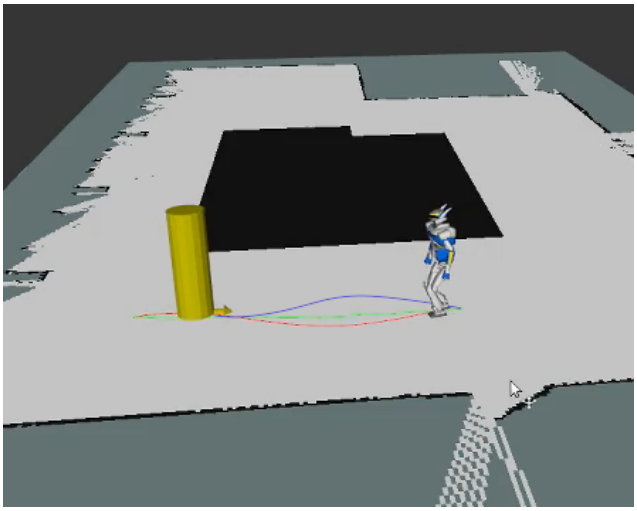

Figure 2.8: The trajectory generated for the human and HRP2 in simulation. The green trajectory is the global plan initializing the optimizer. While the red trajectory (robot) is smooth and make a wide turn, the blue trajectory (human) is planned to turn sharper and to take more effort in the crossing. Indeed, the robot model was set with slower dynamics (smaller maximum speed and acceleration) than the human model as HRP2 must move slowly not to fall.

2.6.1.2 An Experiment With the Pepper Robot for Close Human Robot Motions

The European project MuMMER[6] aimed at deploying a Pepper robot in a shopping mall in Finland to entertain and guide customers [Foster 2019]. The project regrouped seven different stakeholders bringing their knowledge to the project. The LAAS-CNRS goal was to make the guide task. However, in order to serve the most customers possible and to cope with large navigation issues of the Pepper platform, it has been chosen not to go with the customer all the way to the asked shop, but instead, as the mall employees usually do, give directions to the customer to help them find the requested shop. We call this more precise task *route description*. However, giving only verbal instructions concerning the route is not sufficient, we chose to make the robot point at the shop, if it is visible from the current position, or point its direction and the first visible element of the route otherwise.

Although, the surroundings of the robot home place (the place in the mall where the robot is located, and thus, where the interaction takes place) contains some obstacles such as barriers, structure poles and advertisement posters. Thus, the object the robot has to point to the human might not be visible from their current position. We thus endowed the robot with the ability to compute the optimal position for the human and the robot for it to point at a landmark. The computation details of this component can be found in [Waldhart 2019]. Once the

[6] `http://mummer-project.eu/`

robot has computed these optimal positions, it needs to move to its while ensuring the human can move to theirs. Moreover, the customers might be hesitant to approach the robot to start an interaction, thus, the robot is also able to approach a nearby human to ask them if they need its help. However, it has to consider that the human might also move towards the robot to encounter it.

To tackle both of these issues, we decided to use HATEB. We made a higher level component proposing three services: rotate, navigate to and approach. While the rotate service is only made for in place rotation of the robot in order to reorient itself for repositioning or to look or point at something, the other two use the HATEB planner described earlier. The navigate to service expects a robot goal pose and optionally a human identifier and a human goal (which is shown by the robot before starting the navigation). The HATEB planner is then called with empirically tuned weights to make the robot navigate to its goal position while, if provided, ensuring the human can reach theirs. The approach service expects a human identifier and a distance, and use the HATEB planner with different weights with a goal at the specified distance in front of the human. The human goal is predicted from their current pose and velocity some seconds in the future, and updated at each control loop with their new sensed pose and speed.

Moreover, by using HATEB, we were able to use the same head behavior as the one described in Figure 2.4, aiming at improving the legibility of robot motion, crucial in such close motions.

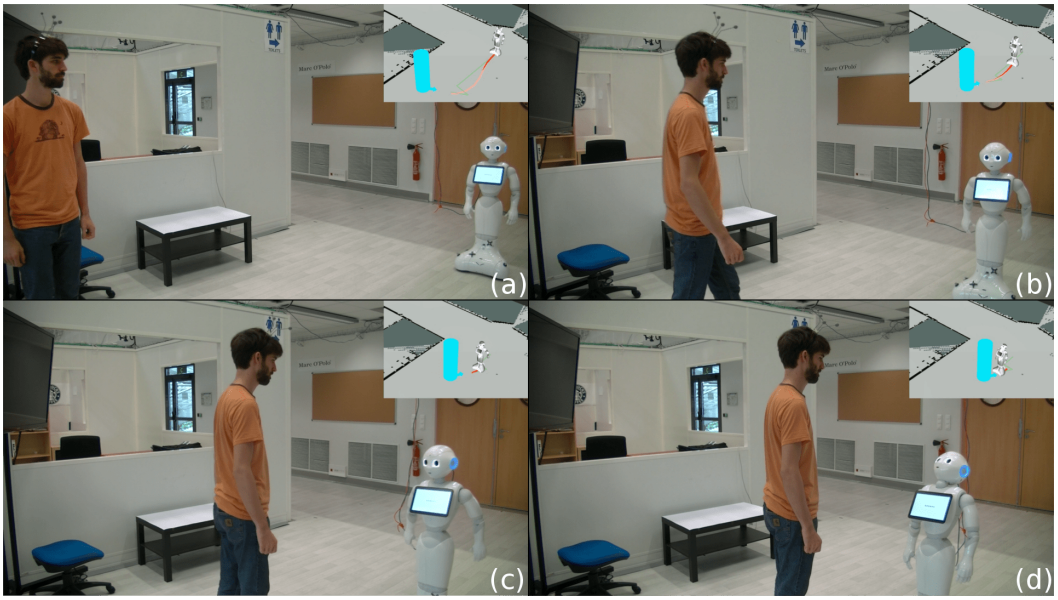

Figure 2.9: The Pepper robot approaching a human. The planner adapts to the human moves. In (a) the planned path is long but decrease in (b) and (c) as the human is actively taking part in the approach task until the robot successfully reaches a position close in front of the human (d).

While this scheme worked well for the approach, with the robot presenting an efficient approach trajectory reacting the human motion (Figure 2.9), it performed very poorly for the navigation part. Indeed, the robot navigation was erratic and made too long trajectories even between close start and goal points. We identified two causes to this over conservative behavior:

- First, the HATEB algorithm did not support holonomic motion. Although this was not a problem for long distance scenarios, it made the robot having to maneuver to achieve small displacements.
- Then, by planning these short trajectories so close to the human, the optimizer does not have a lot of latitude to play with, and the search space is very noisy and chaotic.

It has thus been decided in the project to not use HATEB, but simply an holonomic timed elastic band enriched with social constraints linked to the nearby humans (*i.e.* only optimizing the robot trajectory and not the human one).

This allowed to point out some of the limitations of HATEB, it performs poorly for small, intricate displacements. This led to the creation of HATEB2 [Singamaneni 2020], which is able to switch online between the HATEB planner, the TEB planner with social constraints or a velocity obstacle planner. Moreover, tracking the human position was really challenging. Indeed, we were provided with a human detection system based on the head mounted camera which presented two drawbacks. First, when the robot was moving, the picture was blurred, preventing any human detection. Then, as the robot and the human were moving to the planned pointing position, the human disappeared from the robot field of view. Because of these tracking problems, any navigation scheme using social constraints (based on human position) revealed hard to use.

2.6.2 Using the Estimated Time to Goal to Measure the Execution of the Planned Trajectory

Another benefit of the HATEB approach is that it does not only compute speed command to follow a global plan while avoiding obstacles like other classic local planner, but also returns a complete short term trajectory for both the robot and the human. Moreover, as the trajectories contain the expected duration between each pose, simply by summing them, we can have an estimate of the time remaining to execute the local trajectories. Besides, we also compute the estimated time remaining for the global trajectories after the local ones by dividing the computed speed at the last local trajectory pose by its length. By summing these two estimated remaining times, we can have an estimation of the remaining navigation time (the time to goals) for all the considered agents.

By monitoring the evolution of these times, we can estimate a quality of the ongoing navigation. Indeed, the local trajectories are computed at a position control rate and so are the time to goals. Intuitively, if the computed time to goals are

decreasing at the same rate as the real time duration (*e.g.* the estimation for the robot goes from 12 seconds to 11 seconds while one second has elapsed) the execution of the trajectories are going according to plan. Instead, if the computed time to goals are decreasing slower than the real time duration, staying equals or even increasing, the execution does not go as smoothly as the plan was.

We formalized it as follows:

$$s_i(t) = \frac{ttg_i(t) - ttg_i(t-x)}{x} \tag{2.3}$$

with $s_i(t)$ is the time to goal variation for the trajectory i (either the robot or one of the considered human) at time t, and x is the time window size parameter, specifying how old are the previous time to goal values we are comparing the current one with. With this definition we have:

- $s(t) \approx 1$: the execution goes according to plan,
- $s(t) > 1$: the execution outperforms the plan,
- $0 < s(t) < 1$: the execution goes slower than the plan, but progress towards the goal is still being made,
- and $s(t) < 0$: the goal is getting further.

This value can then be fed back to a supervision system which can change the navigation parameters or abort it to make a repair strategy if, combined with other task monitored values, it judges the navigation action is endangering the higher level task.

From preliminary tests, we saw that this measure is a good indicator of the nominal execution of the plan. However it still has to be refined as the measure can be coarse and does not detect the subtle plan changes. Moreover, even if the evaluation is deteriorating, it does not give insight on why. Indeed, the plan change can be caused by multiple factors. It can be from previously unknown obstacles to the robot leading to a plan change (concerns only $\mathcal{M}^R$), not necessary representing a collaboration issue. The deterioration can also be from a plan change from the human, making the robot to adapt to it. Finally, we cannot, based only on this measure, determine if the human is not following the computed plan because the model the robot has is not accurate enough ($\mathcal{M}_r^H$) or because the human is really not willing to cooperate. These questions and a similar approach are part of work continued by Amandine Mayima in order to evaluate the quality of interaction during collaborative tasks execution [Mayima 2020].

2.7 Conclusion

In this chapter, we presented a robot navigation planner HATEB, which not only computes an optimal local trajectory for the robot but also for the surrounding humans. This approach has been shown to be able to solve complex navigation tasks

where both agents must cooperate to reach their respective goals. Moreover, we proved via a user study that it allows to implement effective coordination smoothers to increase the robot mutual manifestness and facilitate the whole interaction.

Besides, the versatility of this approach has been presented via its implementation on three different robots: PR2, HRP2 and Pepper.

Finally, we used an other benefit of this scheme which is to compute precise local trajectory at position control loop rate to estimate the remaining time to reach the navigation goal. By processing this data and monitoring it over the course of interaction, quality of the ongoing navigation and interaction can be deduced and returned to a higher level supervision component. A more general approach including some principles presented here is developed by Amandine Mayima to measure the quality of interaction by a supervision system in real-time during a collaborative task execution [Mayima 2020].

The presented navigation approach that plans for both the human and the robot presents some drawbacks.

First, through the MuMMER project, we saw that it fails to generate small paths when the human and the robot are close to each other. Indeed, such intricate trajectories are complex to optimize as they are heavily constrained.

Then, the scheme breaks if the human model parameters are not accurate enough. Indeed, the optimization scheme will assume the human to follow a trajectory respecting these parameters. We plan to update these parameters on a per-human basis during the execution to get the most accurate plan for the human.

Finally, the conavigation solutions found are executed as if the human were aware of them. While it allows to check that a solution can exist it does not guarantee that it is the solution the human will choose. In most cases, if the human chooses another trajectory than the one planned, the optimizer will adapt, but in some intricate cases, not accounting for the communication needed to share the plan can lead to deadlocks. For navigation, the communication can be explicit by looking at the trajectory chosen by the robot or pointing at the human planned one, but it can also be implicit through the use of coordination smoothers.

We are particularly interested in these coordination smoothers, which can be seen as non verbal communication of the robot intents and of the shared plan. Planning for these coordination smoothers can only be done if we plan for both the robot and the human. However, due to the continuous and short duration nature inherent to navigation tasks, exploring the planning of such communications can be tedious. We thus propose to move from navigation problems to focus on human robot symbolic task planning.

Chapter 3

Evaluating Communications Feasibility and Cost During Human-Aware Task Planning

Contents

3.1 Introduction and Example

In the previous chapter, we showed interactions are more efficient and satisfactory if the robot considers the plan of the human in its own course of action. Not only it allows at least to ensure that the task is feasible for both agents (provided the

models are correct enough) but also to perform coordination smoothers or other communication actions.

In this chapter, we alleviate from the inherently ephemeral nature of interaction in geometrical navigation planning to further study at symbolic level the planning of communication actions in plans involving multiple agents. Communication between the agents can be made in two ways. Either some of the planned actions convey information but have another main effect or they are dedicated communication actions having no other (intended) effect. The former is linked to predictability and legibility of motion as formalized by Dragan [Dragan 2013]. Moreover, we explored part of it in the previous chapter, as we were using a navigation scheme planning for both the human and the robot to add coordination smoothers to the robot trajectory. The latter is what we will be discussed in this chapter. Especially, we will focus on one type of explicit communication, being verbally designating an object, a problem called referring expression generation.

This raises two questions:

- **How to determine the pertinence and the cost of a communication action in task planning?**
- **When must the robot perform a communication action?**

These questions can boil down to *what* and *when* to communicate [Mavridis 2015]. While determining *when* to insert the verbal communication actions in a human robot joint plan is the goal of a task planner, why resolving the *what* during task planning may not be obvious. First, some communications are more costly than others. We use the cost of a communication action to represent the difficulty to understand it when received by the human and the difficulty to generate it when emitted by the human. In some contexts designating an object among others can be hard. Then, in some cases the communication is not feasible not because of a physical inability (*e.g.* the robot and the human being far apart) but because the content cannot be resolved. Thus, determining the *what* will impact the *when*.

Similarly, the content of a communication will not be the same depending on when it is done. The evolving world state along the plan execution will change the way to refer to an object. Some other objects can be added or removed as the task progresses. Some communications will be crucial for the plan at a certain point while others appearing to be needed will reveal to be optional. Thus, we also need to the *when* to communicate to know *what* will be communicated.

3.1.1 Example

In order to clarify the problem and illustrate this chapter contents, let us take the situation depicted in Figure 3.1(a). In this situation, a human and his robot partner are trying to organize different car keys. The areas represent which car a key opens, and multiple keys can open the same car. The keys are distributed randomly at first. The human does not know which key opens which car, but the robot does

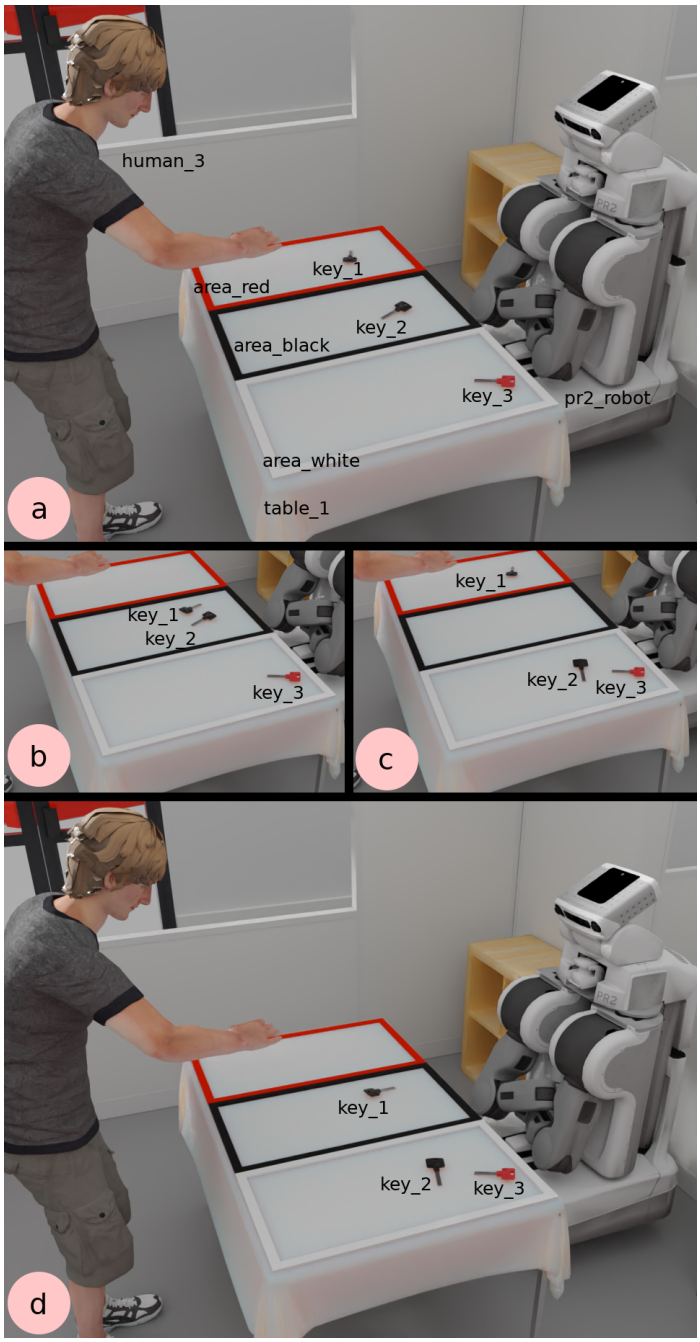

Figure 3.1: A simple example where the robot and the human must cooperate to sort keys. Only the robot knows the goal state (d). It has to communicate to the human, one at a time, which key to grab and where to put them. From the initial state (a) the robot can either communicate to move `key_1` leading to state (b) or to move `key_2` leading to state (c). In (b) the situation is locked as `key_1` and `key_2` are identical. In (c) the robot can communicate to move `key_1` to reach the goal state (d).

thanks to an integrated RFID system. The robot, for which the keys are too small to manipulate, has then to give instructions to the human for where to put the keys. The keys are only distinguishable by the human through their colors and the color of the area they sit in, and he can manipulate only one at a time. Besides, the robot cannot use "left" or "right" indications nor use past human actions. The goal, known only by the robot, is depicted in Figure 3.1(d). The robot has multiple ways to designate the keys and the areas to the human, it can either point to them or verbally designate them. While pointing can be straightforward, it becomes less and less accurate as the distance increases, and impossible if the perspective difference between the agents is too big. Thus, we propose to make the robot able to verbally designate the different entities in the environment. Whatever the communication modality, to move the keys from the initial state (Figure 3.1(a)) to the goal state (Figure 3.1(d)), the robot has two main options. First, it can ask the human to move `key_1` into the black area (Figure 3.1(b)). However, with the constraints we defined before, the robot would have no way of telling the human to take `key_2`, as the human cannot distinguish them anymore, as both `key_1` and `key_2` are the same color and in the same area. The second solution of making the human move `key_2` first, provides more options to the robot, as `key_1` is still verbally designable in the new situation (Figure 3.1(c)).

With this example, we showed that computing communication content during task planning is important as it can impact the choice of communication modalities, the plan cost, or even the feasibility of the plan (*e.g.* if the robot cannot point, the first option would not be feasible). This raises two issues we treat in this chapter: how to efficiently estimate the feasibility and cost of communication and how to use this knowledge during task planning.

To do so, we propose to use a multi-modal planning approach, where a *domain-independent* task planner delegates, for specific verbal communication actions, to a *domain-specific* planner designed to resolve the content of referring expressions. While the former will provide specific requests along with the context and the planned world state in which the communication is made, the latter will indicate not only the feasibility of the communication in the specific world state, but will also return a cost representing the estimated complexity to interpret this referring expression, allowing for balancing between multiple plans. This approach is greatly inspired by combined task and motion planning schemes [Gharbi 2015a] where a task planner uses a geometrical planner to refine motion action at the task planning level, allowing to find more precise plans.

3.1.2 References and Acknowledgments

A large part of this chapter is excerpted from our work, published for the RO-MAN 2020 conference [Buisan 2020b] and for the ICSR 2020 conference [Buisan 2020a]. However, in this manuscript, the examples are more detailed and the approaches are explained more in depth. Besides, we hope that the link between these two works appears more clearly.

The work presented in this chapter has been done in close collaboration with Guillaume Sarthou, who is working on knowledge bases for HRI and their uses.

First, we review the literature concerning referring expression generation and communication actions in task planning. Then we present a novel approach for referring expression generation, which runs on ontologies and is both efficient and suitable for human robot interaction scenarios. We then show how such a planner resolving the content of referring expressions can be included in task planning allowing for precise estimation of this communication type feasibility and cost at task planning level. This approach shows the use of hybrid planning (a domain-independent planner delegating to a domain-specific planner) for HRI, where the domain-independent planner plans for both agents and the domain-specific planner needs the estimation of the human beliefs from the first planner.

3.2 Related Work

Estimating the content of some communication at the task planning level is needed to generate feasible and optimal plans. Indeed, some communication actions are known to be necessary already while elaborating a plan, but might not be feasible. We explore this problem by focusing on a specific communication type: the referring expression. Indeed, verbally referring to an object is challenging for several reasons.

First, it heavily depends on the current situation. If the object to refer to is among multiple identical objects, the communication may be impossible. Similarly, the communication is getting easier the more heterogeneous the nearby objects are and the fewer objects there are.

Then, referring to an object to another agent requires to take their perspective or to estimate their beliefs. Indeed, to refer to a key in Figure 3.1, the robot must estimate that the human knows the color of the keys and knows in which area they are placed.

Finally, generating a referring expression must be done in a specific context that needs to be defined. In the example depicted in Figure 3.1, they may be other keys in the building that the human is aware of (and that the robot knows the human is aware of) but in the context of this task, they are not considered by the human as being potentially referred to.

However, this type of communication is explicit and can accurately be represented as an action in a task planning domain. Moreover, a referring communication action may need multiple expression generation. For instance, in Figure 3.1(a) we can represent a robot action asking the human to move `key_2` in `area_white` in one sentence. This sentence would need two referring expressions: one for `key_2` and one for `area_white`. Thus, estimating the feasibility and cost of this action would require multiple calls to the referring expression generation planner.

In this section, we will firstly review how a robot can autonomously verbally designate an object to a hearer. This problem is called the Referring Expression Generation (REG) problem. Then, we will review several task planning approaches

allowing to account for communication actions.

3.2.1 Referring Expression Generation

As defined by Reiter and Dale, Referring Expression Generation (REG) "*is concerned with how we produce a description of an entity that enables the hearer to identify that entity in a given context*" [Reiter 1997]. An intuition about what a well constructed Referring Expression (RE) is given by the Grice's maxims [Grice 1975]. These maxims aim at defining principles for smooth cooperative activities (including verbal designation communication). They fall into four categories:

- *Quantity*: The communication should be as informative as required but not more.
- *Quality*: The communication should be as true as possible. The sender should not communicate information that they consider false or unsure.
- *Relation*: The communication should be relevant in the current context. This is especially important when performing a collaborative task, where the world state is constantly changing and the relevance of a communication can quickly change.
- *Manner*: The communication should be unambiguous and brief.

The REG problem is actually composed of two parts: the content determination — aiming at deciding which attributes (and relations) to use — and the linguistic realization — refining the attributes of the content into verbalizable/writable words [Krahmer 2012]. In this thesis, we will only consider the content determination, as we assume that the linguistic realization will not have any impact on the feasibility and the cost of the referring communication action once the content of the RE has been decided.

To our knowledge, the first REG formulation and algorithm was coined by Dale and used a depth-first search over a knowledge base being a key-value tree representing the attributes of objects [Dale 1989]. However, this approach leads to over-specified referring expressions, containing redundant information and thus violating the maxim of quantity. On the example depicted in Figure 3.1(a), to refer to the `key_1`, provided with the right knowledge base, this approach would have led to a set of relations which could have been verbalized as "the black key in the red area on the table" while only "the key in the red area" would have sufficed in this context. This defect was corrected in a subsequent contribution with the *Full Brevity* algorithm [Dale 1992], always generating the shortest referring expression, but at the cost of an exhaustive search. Besides, to be as relevant as possible, the attributes of the referred object to be included in the RE should be chosen carefully. Indeed, not all the attributes are equally understandable by the hearer, the color or the shape for example will often be quicker to apprehend than spatial relation. The Incremental Algorithm is the first approach tackling this issue [Dale 1995]. By

taking as input a preference list of ordered attributes, it is able to generate the smallest RE while prioritizing the attribute used.

However, all the presented approaches are running on dedicated key-value knowledge bases representing only the attribute of the entities and are thus unable to use relations between them to generate REG. For example, an object having the same attributes (color, size, shape, ...) as another one will not have any RE generated by the previous approaches, even if one is in a blue box and the other in a green one. By introducing a new knowledge representation, being a labeled directed multigraph linking entities (as vertices) and attributes (as edges), Krahmer *et al.* were able to solve this issue. The graph is dedicated to the problem of REG and is called a *REG graph*. Moreover, a cost can be set on each edge of the graph to represent the complexity of the hearer to understand this relation. Indeed, some relations are harder to interpret than others as shown in [Belke 2002] where they show that humans infer quicker a referred object if the referring expression contains an absolute attribute (*e.g.* color) rather than a relative one (*e.g.* the size). By exploring this graph through a branch and bound approach, the Graph-Based Algorithm [Krahmer 2003] is able to generate the smallest and less costly RE for a given entity. The costs are provided as a separate function and are set for each vertex and edge of the graph. The cost of a referring expression is determined by summing the cost of each vertex and edge composing the expression. This algorithm has then been refined to integrate types of entities in the exploration [Krahmer 2012]. Indeed, previously they were not considering types of objects but only the minimal set of relations needed. Doing so can lead to suboptimal solutions or even not verbalizable or still ambiguous ones as the types (*e.g.* a key, a table) of an object is needed to verbalize the expression, and multiple objects can have the same type, requiring more attributes to be distinguished. Then, the approach has been reimplemented to be more computationally efficient [Li 2017]. Finally, this approach has also be modified to over-specify the RE [Viethen 2013]. In this version, the search algorithm can be given a minimum length parameter specifying the number of attributes the RE must contain. The algorithm explores the REG graph until this length is reached and returns the solution. The two last approaches will be further discussed and compared with ours in Section 3.3.5.3.

Other approaches also include learning for generating REs. Yamakata *et al.* use a beliefs network-based method to disambiguate entities based on multiple attributes [Yamakata 2004]. Besides, they state that their algorithm runs on the hearer estimated belief network, we think that it is an important feature to generate relevant REs. However, they indicate that a belief network should be trained for each attribute, which can be impractical in a real-world robotic application.

Every approach presented until then is relying on REG dedicated knowledge bases or data structures. Such structures can be cumbersome to maintain in a dynamic world where relations between entities can change along the task. Moreover, in complete robotic architecture knowledge bases managing relations already exist, but are not dedicated to REG. The DIST-PIA method tries to mitigate this issue by having a domain-independent Incremental Algorithm querying dedicated knowl-

edge base (called *consultants*) to elaborate the RE [Williams 2017]. By specifying four minimal features each consultant in a robotic architecture must have, the algorithm is able to query them to build a RE. To our knowledge, it is the first and only approach that does not assume a specific knowledge base format to operate. This approach has been successfully integrated into a complete robotic architecture [Williams 2019]. Another work having been integrated into a robotic architecture is made by Ros *et al.* [Ros 2010]. The knowledge base used is an ontology, which is now widely used in robotics to store symbolic knowledge. However, it does not support using the relations to generate REs (it only relies on the attributes of the entities). It has been integrated into a robotic architecture allowing the robot to play a game with the human where it has to guess, through a series of questions, which object they are thinking of in a scene. The approach successfully allows to find the right questions to ask (which would discard the most of potential objects) and to discriminate the right objects when receiving the answer [Lemaignan 2012].

To the best of our knowledge, none of these approaches have been used to determine the feasibility and the cost of a referring communication action during task planning.

3.2.2 Task Planning With Communication Actions

Recently, more and more research is dedicated to human robot verbal communication planning, mainly to answer the *what* and the *when* to communicate [Mavridis 2015]. The vast majority of contributions treats these questions during execution. Indeed, they assume a given plan (multi-agents or not) and insert verbal communication actions when needed.

Chaski is a plan execution system allowing to perform a collaborative activity with a human [Shah 2011]. The system generates verbal communication when starting or finishing a task allowing agents to coordinate their actions and to update their plans. However, it does so without any reasoning on the communication necessity and expects the human does also communicate each task they start and end.

With their *inverse semantic* algorithm, Tellex *et al.* provide the robot with a capability to ask a nearby human for help when it fails [Tellex 2014]. Indeed, when following a plan, if the robot detects an unfeasible action, it decides if and which human action would help it in the plan and is able, using REG *inter alia*, to verbally ask them to perform the selected action with the selected objects.

Sebastiani *et al.* are able, by merging multiple multi-agent HATP plans, to generate conditional plans which then can be verbally negotiated (by asking the human about task allocation) during the execution with the human [Sebastiani 2017].

Devin and Alami proposed a supervision component which is able, when given a multi-agent plan elaborated by HATP, to estimate the beliefs of the human partner [Devin 2016]. Then, they monitor divergences between the robot and the human's beliefs. If a divergence is detected as not allowing the human to perform their next actions of the plan, a verbal communication aligning the needed belief is done by

the robot. It does so on five levels. (1) The robot is monitoring if the human knows that a shared goal has been reached or aborted, and informs them if they do not. (2) The robot constantly checks if the human knows their next planned action and communicate it if the robot estimates the human does not know it; or ask them to start it if the human is estimated to know it but is lacking information about the previous finished actions, enabling theirs; or gives them the necessary beliefs about the world state if a divergence is computed to prevent the human to perform their next action. (3) The robot can also communicate that the next action the human is estimated to perform is not the right one. (4) The robot also monitors whether the next action it has to perform is the same as in the estimated human plan (*i.e.* the next action of the robot in $\mathcal{M}^R$ is the same as in $\mathcal{M}_h^R$) and corrects it if they diverge. Finally (5) if the human is expected to perform an action but is not detected to do so, the robot can asks them explicitly to do it. While some of these communications are tightly linked to the execution, such as communication about the progress of the plan or failures, others can be expected as early as during task planning, and be part of the plan.

In all the previous work, the need and the content of communication actions are solved only when executing the plan. While being unavoidable for execution related communications (*e.g.* failures), others can be known to be required during task planning and inserted into the plan, simplifying the supervision. Besides, by planning the communication needs, their costs can be balanced with other plan alternatives, leading to better plans. Finally, by not planning the communications and resolving them during the execution of the plan, the supervision can get stuck in a situation where it decides that a communication is needed but is not feasible as it can be too late in the plan (*e.g.* the human and the robot are not in the same room anymore). This is why more recent work focuses on tackling communication needs already at the task planning level.

Roncone *et al.* propose a task planner where domains are easily written and visualized thanks to a high-level task tree representation [Roncone 2017]. This domain is then translated into a POMDP which can be solved to obtain a policy. They define three types of verbal communication: (1) *command* is a robot instruction to the human, which can be accepted or declined; (2) *ask* allows the robot to question the human about the progress of their task; and (3) *inform* makes the robot speak about its next intended action. These three types of actions are coded in the POMDP and may be included in the policy depending on the situation and their cost.

A similar approach has been realized by Unhelkar *et al.* where they add one type of communication: *answer* allowing the robot to answer a human querying about its next intent [Unhelkar 2020]. These verbal communication actions are then integrated into a POMDP. This POMDP is elaborated thanks to a provided task model represented as a multi-agent MDP, a robot communication model (including communication cost model), and a human action selection model represented with an agent Markov model. This human model can be refined throughout the interaction.

The POMDP is then solved to generate a robot policy.

It is interesting to note that in the presented work, the communication costs are only based on the time of execution (the *when*) — to ensure multiple communications are not too close in time — but not on the content of said communication (the *what*, *e.g.* the length of the communication, the complexity of understanding it). Moreover, by not considering the content of the communication at planning time (communication actions are considered as a template instantiation with arguments determined at execution time) they do not ensure that it will be feasible when executing. They mitigate this issue by only considering communication about the plan and actions, and not about belief alignment or object referring. This shows the interest of our approach as it tries to tackle, at planning level, two of the five challenges identified by Unhelkar *et al.*: "estimating benefit of communication" and "quantifying cost of communication" [Unhelkar 2017]. Finally, it appears clear in the presented studies that planning for communication can only be done if the robot plans for both agents.

3.3 Ontology-Based Referring Expression Generation for Human Robot Interaction

To estimate the feasibility and the cost of communication action during task planning, we need to be able to quickly find if a communication is feasible and what is its content. To further study verbal communication planning we restrain ourselves to only one type of verbal communication: referring expressions. In this section, we present an efficient algorithm that is able to generate referring expressions for human-robot interaction based on ontologies. We first introduce the ontology representation and argue about its use in human-robot interaction scenarios. Then we propose a list of features needed for REG in human-robot interaction. Next, we formally define the problem of ontology-based REG for HRI, and present an efficient algorithm to solve it. Finally, we show the results of this approach both in terms of found solutions and time complexity.

3.3.1 Using Ontologies for Human Robot Interaction

Ontology definition An ontology is a data representation used in many domains. In robotics, it is now widely used as a knowledge base. It is richer than a "flat" key-value (or vector) of facts. Indeed, it allows to represent multiple concepts inheriting from one another and entities as the instantiation of these concepts. Moreover, the entities can be linked through properties representing relations. Reasoners can use this structure to deduce other facts through first order logic (*e.g.* if a key is in an area then the area contains the key; if the key is in an area and the area is on the table then the key is on the table) and add them to the ontology. Recently, ontologies are even standardized for robotic applications such as the IEEE-SA P1872.2 Standard for Autonomous Robotics Ontology.

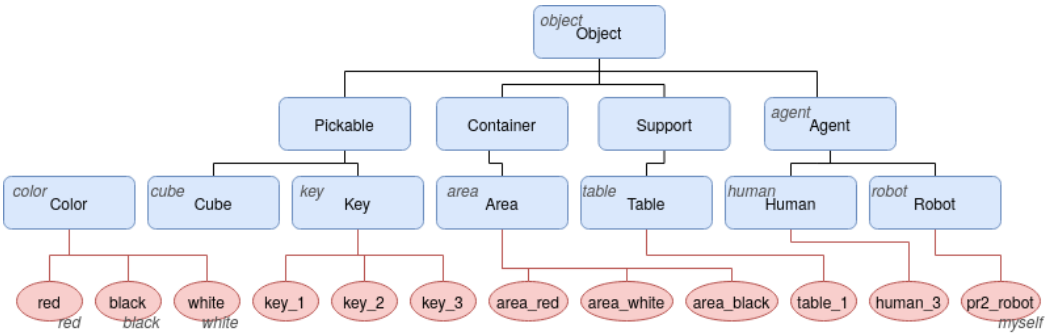

Figure 3.2: The TBox: the types and their hierarchy (in blue), and a part of the ABox (A and C_0): the entities and their types (in red), of an example of an ontology representing the situation depicted in Figure 3.1.

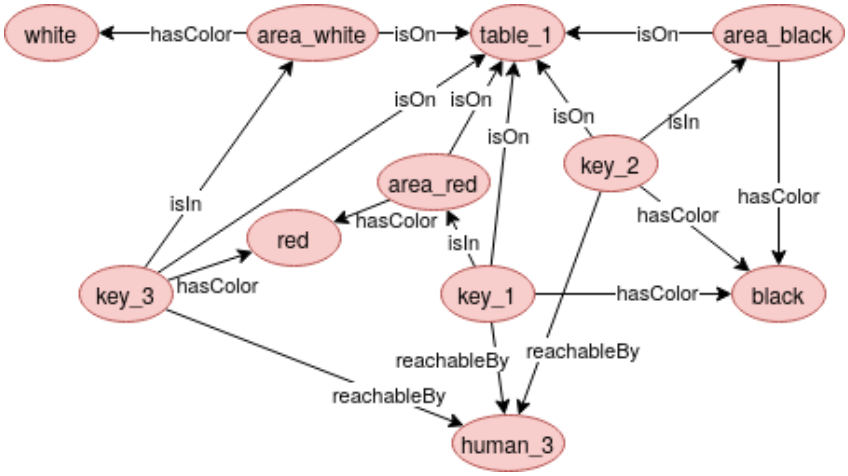

Figure 3.3: A part of the ABox (A and R) of an example of ontology representing the situation depicted in Figure 3.1(a).

Formally, as coined by Fokoue *et al.* [Fokoue 2006] and Krötzsch *et al.*, a knowledge base ontology is defined by the tuple $K = \langle \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$. The *TBox* $\mathcal{T}$ contains the concepts, called *classes* representing the possible types of entities known by the agent. More specifically, it is a finite directed acyclic graph (DAG) $\mathcal{T} = \langle T, H \rangle$ with T the set of classes/types and H the directed edges representing the inheritance/inclusion links between them. For simplicity purposes, we will refer to them as "*isA*" links. For instance, in an ontology representing the example depicted in Figure 3.1(a), we have:

$$\{Key, Table, Area, Object, Agent, Pickable, Robot, Human\} \subset T,$$
$$\{(Key, Pickable), (Pickable, Object), (Table, Object),$$
$$(Robot, Agent), (Human, Agent)\} \subset H$$

(*i.e.* (*Key*, *isA*, *Pickable*), (*Pickable*, *isA*, *Object*), (*Table*, *isA*, *Object*), (*Robot*, *isA*, *Agent*), (*Human*, *isA*, *Agent*)) as represented by the blue graph in Figure 3.2.

The RBox $\mathcal{R} = \langle P, Incl, Inv \rangle$ contains the properties, their inheritances and inverses known by the agent. P is the set of properties, $Incl$ the finite DAG representing inheritances/inclusions between the properties and $Inv = \{(p_i, p_j) \in P^2\}$ representing the inverse properties. The properties can denote both the attributes of objects (*e.g.* the color) and the relations between the objects (*e.g.* which object is on which other one) In an ontology representing the example depicted in Figure 3.1 the RBox may include:

$$\{isIn, hasIn, isOn, hasOn, geometricProperty\} \subset P,$$
$$\{(isIn, geometricProperty), (hasIn, geometricProperty),$$
$$(isOn, geometricProperty), (hasOn, geometricProperty)\} \subset Incl,$$
$$\{(isIn, hasIn), (hasIn, isIn), (isOn, hasOn), (hasOn, isOn)\} \subset Inv$$

Note that to fully match the definition of Fokoue *et al.* [Fokoue 2006] it would require to declare the disjunctive, transitive, reflexive and chain relations in $\mathcal{R}$ and the disjunctive classes in $\mathcal{T}$. As they will be reasoned upon in this thesis, we chose to omit them.

Finally, the ABox $\mathcal{A} = \langle A, C_0, R \rangle$ contains the entities, their types and relations. A is the set of entities. $C_0 = \{(a, t) | a \in A, t \in T\}$ contains the direct types of each entities (an entity must have at least one direct type, but can have multiple ones). Finally $R = \{(s, p, o) | (s, o) \in A^2, p \in P\}$ is the set of relations between entities. s is called the *subject* of the relation, p the *property* and o the *object*. The relations set R actually contains both attributes of objects (*e.g.* $(key\_1, hasColor, red)$) and relations between objects (*e.g.* $(key\_1, isOn, table\_1)$) in our case. For example, in an ontology representing the example of Figure 3.1(a) we would have as part of

the ABox:

$$\{key\_1, key\_2, area\_red, table\_1, human\_3, pr2\_robot\} \subset A,$$
$$\{(key\_1, Key), (key\_2, Key), (area\_red, Area), (table\_1, Table),$$
$$(human\_3, Human), (pr2\_robot, Robot)\} \subset C_0 \text{ (red part of Figure 3.2)},$$
$$(key\_1, isOn, table\_1) \in R \text{ (Figure 3.3)}$$

By using the hierarchy of types we also define C representing the graph of direct and inherited types of entities. C is constructed by adding all the types that can be reached from a direct type of an entity by following a path in H. For example $(key\_2, Key) \in C_0 \implies (key\_2, Key) \in C \wedge (key\_2, Pickable) \in C \wedge (key\_2, Object) \in C$ if we reuse the example H presented before. We define the "isA" property for simplicity purpose. The "isA" property allows to represent hierarchy of types and entities types (as defined in C) while only representing triplet, as typical relation (*e.g.* (key_2, isA, Key), $(key\_2, isA, Pickable)$, $(key\_2, isA, Object)$. This definition is only intended to help with the notation. It is important to note that the *id* of an entity must remain internal to the robot and is not intended to be communicated. It is a unique identifier that can be shared across all the robotic architecture. In our examples (and in practice) we use meaningful ids in order for the ontology to be understandable when analyzing it.

In what follows we will consider the TBox and RBox as static. They will be defined before any experiment and will not be modified at runtime. In our architecture, they contain the semantic knowledge of the robot and are not meant to change during a scenario (we do not consider cases where the robot would learn about new categories of objects or about unknown attribute types of objects). The ABox, on the other hand, will contain both predefined entities and relations but also sensed entities and computed facts. It will contain usual symbolic facts, computed by the situation assessment, found in the knowledge bases of typical robotics architecture. However, thanks to their typing and the hierarchy of both types and properties deduction and reasoning can be done on them.

In this thesis we will not present the different reasoners of the ontology, but rather assume that the ontologies used are all been preprocessed and are consistent (*e.g.* if a relation is in R, all the inverse properties of this relation have been added to R). More in depth descriptions and uses of ontology in robotic architecture can be found in [Sarthou 2019] and in Guillaume Sarthou's thesis.

SparQL queries In addition, ontologies often come with a way of requesting data upon them. A common way of doing so is using SparQL queries. SparQL queries allow to bind variables with classes or entities respecting the relations specified in the request. The syntax which we will follow in this thesis is for the variable names

to begin with a question mark. Let define a SparQL query:

$$SELECT\ ?key$$
$$WHERE\{?key\ isA\ Key.\ ?key\ isOn\ table\_1.\ ?key\ hasColor\ black\}$$

This query, issued on the ontology representing the scene depicted in Figure 3.1(a), would return *{(key_1), (key_2)}*. Indeed, the first clause[1] *?key isA Key* would lookup for all the entities which are inhering from the *Key* class (Figure 3.2). At this stage, *?key* can thus be bound to *key_1*, *key_2* and *key_3*. The second clause, *?key isOn table_1*, make the SparQL engine lookup for entities from the previous set being subject of a relation having as property *isOn* and as object *table_1* in R (Figure 3.3). All the entities previously returned does have that relation, so for now *?key* can still be bound to *key_1*, *key_2* and *key_3*. The last clause *?key hasColor black* would again make the engine lookup in R (Figure 3.3). *key_3* does not have the relation *hasColor black* so it is removed from the possible bindings of $?key$. The final result is thus *{(key_1), (key_2)}*.

A more complete example would be the query:

$$SELECT\ ?key\ ?area$$
$$WHERE\ \{?key\ isA\ Key.\ ?area\ isA\ Area.\ ?key\ isIn\ ?area.\}$$

Again, thanks to the first clause ,*?key* can thus be bound to *key_1*, *key_2* and *key_3*. For the second clause, we need to find all the entities inheriting from *Area* in the ABox (Figure 3.2). This results in the variable *?area* being possibly bound to *area_red*, *area_white* and *area_black*. For the last clause, possible bindings combinations of both variables are processed, and if R (Figure 3.3) contains a relation linking them with the *isIn* property, the couple is added to the set of results. The full query thus returns *{(key_1, area_red),(key_2, area_black), (key_3, area_white)}*

The multi-agent case Finally, we want to be able to estimate and reason on the human beliefs. To do so, we will use one knowledge base (*i.e.* ontology) per agent considered by the robot in addition to its own. To follow the notation of Chakraborti [Chakraborti 2018] presented earlier in this thesis, we will note $K^R = \langle \mathcal{A}^R, \mathcal{T}^R, \mathcal{R}^R \rangle$ the knowledge base of the robot and $K_r^H = \langle \mathcal{A}_r^H, \mathcal{T}_r^H, \mathcal{R}_r^H \rangle$ the robot estimated knowledge base of the human it is interacting with. In practice, we will have $\mathcal{T}^R = \mathcal{T}_r^H$ and $\mathcal{R}^R = \mathcal{R}_r^H$, and only have differences in the ABoxes. Indeed, we assume that the robot estimates that the human knows the same concepts (classes and properties along with their inheritances) as itself. Only the relations between the entities can differ. While this requirement is not needed in our approach, we assume it for clarity purposes.

[1] Actually, SparQL engines do not necessary process the clauses in the order they were submitted. Requests are often analyzed and their processing optimized to get the best performance.

The content of the ontology in HRI The content of the ontology should be carefully chosen for HRI. In our architecture, the content of the robot ontology can be decomposed into three parts (usually not exclusive).

First, there are the classes and properties used for consistency and to reason on the represented world state. They correspond for example to the inheritance of properties, disjoint classes, inverse properties.

Then, some classes and properties are dedicated to the "programming" of the robot. They allow to abstract some facts in the knowledge base in such a manner that other components can easily access the data they need. For example, the *Container* and *Support* classes can be used by a task planner to easily retrieve objects in the environment that can contain other objects or on which other objects can be placed. Another example can be the *hasMesh* property, allowing the components of the architecture (*e.g.* motion planner, situation assessment) to share the same geometrical model of an entity.

Finally, some classes and properties are more HRI oriented as they allow to make the interface between the robot knowledge representation and the human. They allow to verbally communicate about entities, to understand situated dialog or to draw the scene for example. They include the classes such as *Cube*, *Key*, *Color* and properties such as the label of entities or *hasColor*.

It is clear that some of the knowledge does not intend to be communicated to the human, however, it can still be in their estimated ontology. For example, the *hasMesh* property cannot be verbally communicated but is in the human estimated ontology as it allows to represent how the human may perceive an object, and allow for perspective taking.

3.3.2 REG Features for Communication Action Estimation During Task Planning

We saw previously that REG is an important and interesting problem for human-robot interaction scenarios. However, as its application will be on an environment perceived in real-time, along with a collaborative task and with respect to a specific human, additional constraints have to been considered.

1. We want to be able to **use the relations between entities**. Indeed, we saw that some state of the art approaches were able to only use the attributes of entities in their referring expression and not the relations between them. For instance in Figure 3.4(e) and (f) the pen can only be referred to by using its relations to the pencil boxes.

2. Then, we want the algorithm to **run on existing knowledge bases**. Many presented approaches rely on a dedicated knowledge representation. Such representation can be cumbersome to maintain during an interaction in an evolving environment. Moreover, as stated before, the ontologies used in HRI may already contain the knowledge needed to perform the REG (but often much more).

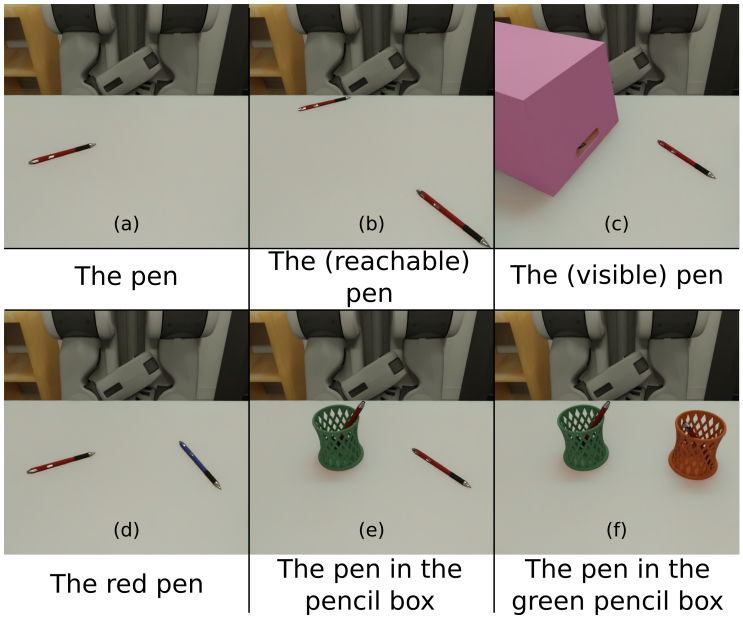

Figure 3.4: Six scenes as viewed by a human interacting with a robot at the other side of a table. In each scene, the configuration of objects leads to different mechanisms to refer to a pen without ambiguities.

3. We also want to support the **preference ordering** per agent. Indeed, some relations are understood better and quicker than others, and this preference can change depending on the agent we are interacting with.

4. In addition, we want the algorithm to consider the verbalization through **the use of types**. All the approaches presented before only focus on the content determination of the REG and consider that the linguistic realization (the verbalization) will be perfect. They consider that all the contents of the used graph (often dedicated to REG) can be verbalized (*e.g.* it exists a word for every edge of the REG graph and each word correspond to only one edge). While this can be assumed when dealing with a dedicated knowledge base, in an ontology we need the type of an entity as the minimal information needed to refer to an entity (*e.g.* the *pen_6* in Figure 3.4 (a) cannot be verbalized directly as "pen 6", only its type can be verbalized as "the pen").

5. Likewise, in large robotic ontologies, every type or relation cannot be verbalized. Indeed, we saw that they also can contain data to be used by other components (*e.g.* planners) or to be shared across the architecture. For instance we do not want the robot to say *Pickable* type, the *geometricProperty* or the *hasMesh* property. Thus, our algorithm should be able to **select only verbalizable types and properties**.

6. Finally, in an interaction, it is clear for the hearer that some entities will not be referred, and should not be taken into account as distractors by the

algorithm (in the example depicted in Figure 3.4(a), it should be clear to the human that, unless specified otherwise, if the robot asks about a pen, it is one on the table and not one in another room). Also, some relations will be implicit in the communication (*e.g.* if the robot asks the human to *give* it a pen, it is implied that the pen is not reachable by the robot and reachable by the human, as in the Figure 3.4(b) and (c)). Thus, the algorithm must **use the context of the ongoing task**.

Determining the right context (what are the implicit relations, which do not need to be said) is an open challenge. It appears to be linked to the location of the agents and to the action intended to perform with the referred object. We think *affordances* of objects can be a way of determining this context [Gibson 2014, Norman 2013]. Indeed, by emulating the perceived affordances of the different objects perceived by the human, we can deduce which one would be filtered out when asking to perform an action with a referred object. For instance, if the robot asks the human to place the object they are holding on top of a referred object, the human will probably not consider all the object not providing a support as an affordance, the implicit communication would be that the referred object is a support. For now, some affordances are represented as classes in the ontology (*e.g. Pickable*, *Support*). While some effort has still to be put in clearly defining and determining the context for each referring expression, in this thesis the contexts will be deduced by hand and provided to the REG algorithm.

3.3.3 Ontology Based REG Problem Definition

To formally define the REG problem for HRI, we need to enhance our knowledge base with three functions. First, we define a class labeling function $\mathcal{L}_t : T \mapsto str \cup \bot$ where str denotes a set of character strings used as words in the common human robot vocabulary. We define that a class $t \in T$ is labeled iif $\mathcal{L}_t(t) \neq \bot$ and call $\mathcal{L}_t(t) \in str$ the label of t. Besides, we require this label to be unique, *i.e.* for any pair of labeled classes $t, t' \in T^2, t \neq t' \Leftrightarrow \mathcal{L}_t(t) \neq \mathcal{L}_t(t')$. We define similarly a an entity labeling function $\mathcal{L}_a : A \mapsto str \cup \bot$ associating some entities to their speakable/writable unique names (gray names in Figure 3.2). These functions can be defined in the ontology by using the commonly used property *rdf:label* to the labeled classes and entities. Adding them that way, allows to make these function agent dependent. In the example depicted in Figure 3.1(a) we would have among others $\mathcal{L}_t(Table) =$ "*table*", $\mathcal{L}_t(Pickable) = \bot$ and $\mathcal{L}_t(Key) =$ "*key*". Indeed, the types and entities are defined by their identifier in the ontology, however these identifiers usually make no sense to the human (*e.g. key_1*), or are not intended to be verbalized (*e.g. Pickable*). The labels serve to distinguish the individuals and types that are verbalizable from the other and to give a unique verbalizable name to them. In practice, entities are rarely named except for very specific objects that would have been given a unique name.

Moreover, to support the preference ordering we introduce a *comprehension*

cost function depending on the agent $\mathcal{C}^H : P \mapsto \mathbb{R}^{+*}$ assigning a positive cost to properties. It allows to represent that some relations are harder to interpret for the hearer than others. We will not present in this thesis how to compute these costs. However, some approaches manage to estimate this cost and the preference order [Belke 2002, Koolen 2012]. Belke and Meyer evaluated the reaction times to find if two patterns were different, moreover, they analyzed the referring expression generated by participants when presented with two different patterns to refer to one. They found that differences in object colors (absolute difference) appear to make the distinction easier than differences in the size (relative difference). Besides, people tend to over-specify the Referring Expression (RE) by specifying the color even when it is not needed. This shows that color attributes should be preferred over size when generating a RE [Belke 2002]. Koolen *et al.* used a learning approach based on human generated RE to assign costs to the relations used by the Incremental Algorithm and Graph-Based Algorithm [Koolen 2012]. In what follows, all the properties have a unit cost.

We are aiming to unambiguously designate, through its relations to other entities, an entity $a_t \in A$ in a knowledge base K. We will call the entity we are trying to refer to the *target entity* a_t. However, the RE is meant to be used in the context of a task. As stated previously, the RE needs to account for certain implicit relations. This is why the problem must be given a **context** $Ctx = (R_{ctx}, C_{ctx})$, a set of relations and direct types that are implicit in the current situation, which will be used to reference a_t, but not included in the generated RE. For the interactions of Figure 3.1, the context could be defined as:

$$Ctx = (\{\langle a_t, isOn, table\_1\rangle, \langle a_t, isVisibleBy, human\_3\rangle, \\ \langle a_t, isReachableBy, human\_3\rangle\}, \emptyset)$$

With this context, we restrict the disambiguation to the entities present on the table $table_1$ and visible and reachable by *human_3*, the human partner. Indeed, when engaged in a tabletop scenario, objects on the table are prioritized when designating one. Besides, as the robot asks to move the keys, we model that the human infer that the keys referred to are reachable and visible by him.

Finally, to be able to run on our ontologies and select only verbalizable properties, we provide the problem with a set of **usable properties** $U \subseteq P$. Because of properties inheritance $Incl$ all the properties inheriting from the ones in U are usable in the problem.

We thus define the REG problem as follows:

Definition 1 (The referring expression generation problem)**.** The Referring Expression Generation (REG) problem is a tuple $REG = \langle a_t, K, Ctx, U\rangle$ with $a_t \in A$ the target entity, K the hearer's estimated knowledge base as an ontology containing the facts the robot estimates the hearer knows, Ctx the context and $U \subset P$ the set of usable properties.

It is important to note that the knowledge base we use is the hearer's one. Indeed, for the referring expression to make sense, we need to ensure that all the types and attributes used are known to the hearer. We see here that such a problem requires the robot to perform perspective taking.

A solution to the REG problem is a set of relations (the attribute of object, their relations between them and their type) which could be verbalized afterward. We define more precisely what must be a solution to the REG problem. Because some entities (actually the vast majority) are *not* labeled (anonymous,) and thus cannot be referred to directly (their id *e.g. key_1* is only an internal identifier, and does not make any sense for the human), some of the relations might be under-specified. For instance in Figure 3.1, the sentence "the black key" is under-specified in that "the key" does not identify a unique entity but any entity with the class *Key*. In addition, it might be the case that a unique, anonymous, entity participates in more than one relation, *e.g.* "the black key on the table". To keep track of anonymous entities in under-specified relations, we introduce a variable set X, representing the anonymous entities, just as in SparQL queries. Again, we choose a syntax where variables will be prefixed with a question mark (*e.g.* $?y \in X$). An under-specified relation is thus a triplet $(s, p, o) \in (X \cup A) \times U \times (X \cup A)$, *e.g. (?y, hasColor, black)* where $?y \in X$ is a variable and $black \in A$ is a labeled entity in the knowledge base.

When speaking about anonymous entities, one must know its type to serve as a placeholder in sentences (*e.g.* "the pen"). Thus, the solution should associate each variable and a type (previously denoted as "isA" relations). For simplicity, we chose to represent them also as triplets: $X \times "isA" \times T$ (*e.g. (?y, isA, Cube)*).

Definition 2 (Reference)**.** Thus, a **reference** E is a set of triplets, each triplet in E being either an under-specified relation in $(X \cup A) \times U \times (X \cup A)$ or a type ascription in $(X \times "isA" \times T)$.

For example for the ontology depicted in Figure 3.2 and Figure 3.3, $\{(area\_white,\ hasColor,\ white)\}$, $\{(?0,\ isOn,\ table\_1), (?0,\ isA,\ Key)\}$ and $\{(key\_2,\ hasColor,\ ?0), (?1,\ hasColor,\ ?0), (?0,\ isA,\ Color)\}$ are three references. However, a reference may not be verbalizable as is, nor represent a valid situation of the knowledge base. We thus introduce three constraints:

Constraint 3.3.1 (Nameability of entities)**.** *Each entity $a \in A$ present in any tuple of a reference E (as first or third component) must have a label: $\mathcal{L}_a(a) \neq \bot$.*

For instance for Figure 3.1(a) with the ontology depicted in Figure 3.2 and Figure 3.3 a reference being: $\{(key\_3, hasColor, red)\}$ violates the Constraint 3.3.1. Indeed, even if red does have a label ($\mathcal{L}_a(red) = "red"$), key_3 does not ($\mathcal{L}_a(key\_3) = \bot$, the individual key_3 does not have a gray name in Figure 3.2).

Constraint 3.3.2 (Nameability of variables)**.** *For each variable $x \in X$ present in any tuple of a reference E (as first or third component) there must also be a unique tuple in E specifying one of its labeled type ($(x, "isA", t) \in E$ with $t \in T$ and $\mathcal{L}_t(t) \neq \bot$).*

Again, in the situation of Figure 3.1(a) with the ontology in Figure 3.2 and Figure 3.3, the references $\{(?0, hasColor, red)\}$ and $\{(?0, hasColor, red), (?0, isA, Pickable)\}$ respect the Constraint 3.3.1 (*red* has a label) but violate the Constraint 3.3.2. For the first one, ?0 is a variable but the reference does not contain an *isA* relation linking it to a type. For the second one, it does contain an *isA* relation linking the variable ?0 to a type, but *Pickable* does not have a label ($\mathcal{L}_t(Pickable) = \perp$; it does not have a gray name in Figure 3.2).

Constraint 3.3.3 (Correct instantiation of variables)**.** *For a reference E there must exist at least one mapping function $f : X \mapsto A$ of the variables in E into entities in A such that the types and relations linking entities in E are still present in T and R once f has been applied. In practice, f transforms the under-specified relations of E into fully specified ones that must appear in the knowledge base.*

As an example, in the context of the Figure 3.1(a) represented by the ontology depicted on Figure 3.2 and Figure 3.3, the reference $\{(?0, isA, Key), (?0, hasColor, white)\}$ respects the Constraints 3.3.1 and 3.3.2, but does not respect the Constraint 3.3.3. Indeed, the variable ?0 cannot be replaced by an entity having the specified relation in the given ontology (no entity of type *Key* has the relation *hasColor* with the entity *white*).

We can now define a **valid reference**:

Definition 3 (Valid reference)**.** A reference E is valid with respect to an ontology K if and only if it respects the constraints 3.3.1, 3.3.2 and 3.3.3.

Besides, we define a solution and a complete solution to a REG problem $REG = \langle a_t, K, Ctx, U \rangle$:

Definition 4 (Referring expression)**.** A solution to a REG problem $REG = \langle a_t, K, Ctx, U \rangle$ is called a referring expression and is a tuple $S = \langle E, x_g \rangle$. E is a valid reference and $x_g \in X$ is a variable, such as for each mapping function f respecting the constraint 3.3.3, $f(x_g) = a_t$.

Definition 5 (Complete referring expression)**.** A complete solution to a REG problem $REG = \langle a_t, K, Ctx, U \rangle$ is a solution where the mapping function f respecting the constraint 3.3.3 is unique.

In the situation of Figure 3.1(b), a referring expression for the entity *area_black* could be $\{(?0, isA, Area), (?1, isA, Key), (?1, hasColor, black), (?1, isIn, ?0)\}$ with $x_g = ?0$. In this situation, ?1 can be bound to both *key_1* or *key_2*, (two possible mapping functions), but in either case ?0 can only be bound to *area_black*. Thus, this referring expression is not complete.

The aim of the hearer is then to instantiate all the variables from their knowledge to find the entity referred to.

Finally, we define an optimal solution (referring expression) $S^* = \langle E^*, x_g \rangle$ as being the a solution minimizing $\sum_{(s,p,o) \in E^*} \mathcal{C}(p)$ over the set of all possible solutions for a REG problem, where $\mathcal{C} : P \mapsto \mathbb{R}^{+*}$ is the properties cost function defined previously. In this thesis, all the properties will have a unit cost, the optimal referring expression will then be the shortest one.

3.3.4 Efficient REG Algorithm Presentation

We aim at building an algorithm to find a solution to the REG problem. The algorithm must build a set of relations, from the content of a provided ontology, of which all possible substitution of variables would result, for a specific variable x_g to a unique entity of the ontology, the target entity. We choose to inspire from the previous contributions of REG and to formalize the problem as a graph search. The nodes of this graph are references (as defined in Definition 2), and the transitions (edges) are relations from the ontology being added to this reference.

3.3.4.1 Formalization as a Graph Search Problem

Let **node** $\mathfrak{n} = \langle \mathcal{T}_\mathfrak{n}, X_\mathfrak{n}, A_\mathfrak{n}, \mathcal{S}_\mathfrak{n} \rangle$. $\mathcal{T} \subseteq R \cup C$ is a set of triplet relations representing some relations in the knowledge base K. $X_\mathfrak{n} \subseteq X$ is the variable set used in this node, $A_\mathfrak{n} \subseteq A$ is the set of anonymous entities of $\mathcal{T}_\mathfrak{n}$ and $\mathcal{S}_\mathfrak{n} : X_\mathfrak{n} \mapsto A_\mathfrak{n}$ is the bijective mapping function linking variables to the anonymous entities they represent. We will note $\mathcal{S}^{-1}(T)$ the resulting *reference* (as defined in Definition 2) after the application of $\mathcal{S}^{-1}$ on all the entities in each triplet of T which is also in $A_\mathfrak{n}$. The **initial node** is specified by the user's query through the *context* of the problem. The idea is then to explore these nodes until the reference generated from the node $\mathcal{S}^{-1}(T)$ is valid and solution of the REG problem.

To find all substitution functions defined in the Constraint 3.3.3, and thus, all the entities which can be bound to the variables in the reference (what the hearer will do to find the entity referred to), we use the SparQL queries presented previously. From any node $\mathfrak{n}$ we can construct a SparQL query from $\mathcal{S}^{-1}(T)$, and submit it on the knowledge base to know how many entities can bound to the variables of the request. In some sense, the SparQL queries can be seen as an emulation of the cognitive process the hearer will do when receiving the RE. A node $\mathfrak{n}$ is a **goal node** if a_t is the only solution to the variable x_g of the SparQL query created from the node (Definition 4), and possibly all the variables in the SparQL query have only one assignation (Definition 5).

A **transition** $\mathfrak{t}$ in the unambiguous reference generation problem consists in the insertion of a new triplet (s, p, o) to the set $\mathcal{T}_\mathfrak{n}$ of a node $\mathfrak{n}$ resulting in the creation of a new node $\mathfrak{n}'$. The inserted relation in a node $\mathfrak{n}$ can be a typing relation ($p \equiv isA$) or a relation which differs between ambiguous entities in $\mathfrak{n}$. We define two kinds of difference between ambiguous entities.

Definition 6 (Hard difference)**.** A **hard difference** $a_i \,\Delta\, a_j$ exists when two entities have the same property towards a different entity (i.e $(a_i, p, b_i) \in R \wedge (a_j, p, b_j) \in R | b_i \neq b_j$).

Definition 7 (Soft difference)**.** A **soft difference** $a_i \,\delta\, a_j$ exists when an entity has a property that is not present for any other ambiguous entity (i.e $(a_i, p, b_i) \in R \wedge (a_j, p, \cdot) \notin R$).

For example, in Figure 3.1(b) we would have for a hard difference:

$$(area\_black, hasColor, black) \in area\_black \,\Delta\, area\_red$$

as both *area_black* and *area_red* have a relation containing the property *hasColor* but *area_black* has this property with the entity *black* while *area_red* has it with the entity *red*. Then, a soft difference would be:

$$(key\_1, isIn, area\_black) \in area\_black \,\delta\, area\_red$$

as the property *isIn* is present in this relation concerning *area_black* but not present in any relation concerning *area_red*.

As the hard differences respect the **open-world assumption** but the soft differences do not, we propose to encourage the use of hard differences when possible by adding an extra cost to transitions coming from soft differences.

Finally, the **cost** of a node is the sum of the costs of each transition leading to this node. If we assume that each transition $\mathfrak{t}_j$ corresponds to the addition of a triplet (s_j, p_j, o_j) to the set $\mathcal{T}_\mathfrak{n}$ of a node $\mathfrak{n}$ with a cost $\mathcal{C}(p_j)$, the cost to $\mathfrak{n}$ is $\mathcal{C}_\mathfrak{n} = \sum_{(s,p,o)\in\mathcal{T}_\mathfrak{n}} \mathcal{C}(p)$.

3.3.4.2 Algorithm Presentation

We chose to perform this search and solve the REG problem to use a uniform cost search algorithm on the graph presented before. From an initial node built from the context of the query, the algorithm generates new nodes by adding possibly disambiguating relations to the current node. We use a uniform-cost search which is **optimal** and **complete** with positive transition costs and a finite number of entities and properties in K. Just like Dijkstra's algorithm, it expands the nodes in increasing cost order until a solution is discovered or the search space is exhausted.

The graph search algorithm is presented in Algorithm 1. The different transitions (edges) (GetTransitions function) exploring through the nodes can be of two types. Either the reference $\mathcal{S}^{-1}(T)$ of the node contains some variable that are not typed (violating the Constraint 3.3.2) and the function returns a typing (*isA*) relation to be added to the set of relations (TypingTransitions function); or it returns the set of soft and hard differences between the anonymous individuals of the node and the other individuals having matched in the SparQL query (DifferenceTransitions function) for their substituting variable. By doing so, we ensure that if a node produces a reference that is not valid (violating the Constraint 3.3.2), the next transitions are only dedicated to make it valid through typing (*isA*) relations. Moreover, by only adding relations that are in the ontology K we ensure that all the relations of a node are in the ontology, and thus respecting the Constraint 3.3.3). Then, the transitions returned are explored and used to create new nodes (ApplyTransition function). A new node is created by copying the relations of its parent and adding the relation of the transition. Moreover, if the transition adds a new anonymous entity, it creates a new variable and adds the

Algorithm 1 Uniform cost search algorithm for referring expression generation.

```
1: function REG(a_t, K, Ctx, U)
2:     node ← Ctx
3:     frontier ← a priority queue of nodes ordered by their cost, initialized with
   node having a cost 0 as only element
4:     explored ← an empty set of nodes
5:     loop
6:         if IsEmpty(frontier) then
7:             return failure
8:         node ← Pop(frontier)
9:         if GoalTest(node) then
10:            return S^{-1}(T_node)
11:        explored ← explored ∪ node
12:        for each transition in GetTransitions(node) do
13:            child ← ApplyTransition(node, transition)
14:            if child ∉ explored and child ∉ frontier then
15:                Insert(child, frontier)
```

binding to the mapping function, ensuring the respect of the Constraint 3.3.1). Finally, when a node is explored, its reference is created using its mapping function, then the validity of the reference is checked and if it is valid, a SparQL query is constructed and submitted to the ontology to check if the valid reference is solution; if not, the exploration continues.

The different called functions are presented hereafter and the pseudo-codes are given for the most interesting ones:

ToQuery: Performs a direct translation of a *reference* into a SparQL query.

SparqlResult: The function that takes a SparQL query as input and returns a match table $\mathcal{M} : X \mapsto \mathcal{P}(A)$[2] in the way that $\mathcal{M}(x)$ is the set of entities matching the variable $x \in X$ in the given query.

GoalTest: First, this function checks if the reference produced with $\mathcal{S}^{-1}(T)$ is valid. Then, it constructs a SparQL query from a node using the ToQuery function. Then, it submit it to the ontology K with the function SparqlResult. Finally, using the resulting match table, it returns $\top$ (true) if the variable denoting the target entity x_g has only one match, being the target entity a_t ($\mathcal{M}(x_g) = \{a_t\}$) and $\bot$ (false) otherwise.

GetTransitions: (Algorithm 2) At each step, we consider two kinds of possible transitions. The TypingTransitions function (Algorithm 4) consisting in the addition of an inheritance (*isA*) relation if at least one entity has no label and no inheritance relation in $\mathcal{T}_\mathfrak{n}$. Otherwise, the DifferenceTransitions concatenates the transitions from the **hard difference transitions** (Algorithm 3) and the **soft differences transitions** (Algorithm 3 with the δ operator at line 7). These

[2]This is a simplification of the result returned by a SparQL query, as we do not use the relation between the tuples really returned.

transitions add relations that differ as hard and soft differences between ambiguous entities for each variable in $\mathcal{M}$.

Algorithm 2 The pseudo-code of the function returning the different transitions (edges) to explore.

```
1: function GetTransitions(node)
2:     transitions ← TypingTransitions(node)
3:     if transitions ≠ ∅ then
4:         return transitions
5:     transitions ← DifferenceTransitions(node)
6:     return additions
```

Algorithm 3 Hard difference transitions pseudo-code.

```
1: function HardDifferenceTransitions(node)
2:     transitions ← an empty set of transitions
3:     M ← SparqlResult(ToQuery(S_node^-1(T_node)))
4:     for each x in X_node do
5:         for each a in M(x) do
6:             if a ≠ S_node(x) then
7:                 for each r = (S_node(x), p, o) in S_node(x)Δa do
8:                     r_inv ← (o, Inv(p), S_node(x))
9:                     if r ∉ T_node ∧ r_inv ∉ T_node ∧ p ∈ U then
10:                        transitions ← transitions ∪ {r}
11:    return transitions
```

The Δ (resp. δ) operator returns all the relations that are hard differences (resp. soft) between two entities as defined in 3.3.4.1. In the difference actions algorithm, an action can be added only once and must not be present in the current state to avoid redundancy. The inverse relation to the one added by the action is also retrieved from the Inv set defined in the knowledge base and checked if not present in the current state and in the current actions set, again to avoid redundancy.

TypingTransitions: The TypingTransitions function (Algorithm 4) stops at the first entity which has no label nor type. This specificity reduces the branching factor while ensuring that each entity has a label or at least a type. Since typing actions are the first tested in the GetTransitions function, all entities not typed during a first execution will be during the next ones. In the implementation, this function has been optimized by observing that once all the entities from the context are typed, the only entities in $\mathcal{T}_{\mathfrak{n}}$ which may not be typed are added as the *object* of a DifferenceTransitions. Thus, by storing the object entity of a transition and only checking if it is labeled or has already been typed (and is thus present in $A_{\mathfrak{n}}$) we reduce the complexity of the TypingTransitions function.

UsableClass: The function UsableClasses returns the most specific *labeled* classes of an entity a, *i.e.* the set of classes $t \in T$ such that $(a, t) \in C$, t is labeled and there are no labeled sub-classes of t.

Algorithm 4 Typing transitions pseudo-code.

```
1: function TypingTransitions(node)
2:     for each (s, p, o) in T_node do
3:         if ∄x s.t. (s,"isA", x) ∈ T_node ∧ L_a(s) = ⊥ then
4:             return { (s,"isA", t) | t ∈ UsableClasses(s) } and the creation of a new variable
5:     return ∅
```

This strategy differs from the one of [Dale 1995] that prefers the least specific types (so-called basic-level classes). However, in domain-independent knowledge bases such as ours, their scheme could often result in "Object" or "Thing" which can lead to confusion. Furthermore, by being conservative in our estimation of the receiver's knowledge base, we can guarantee that the labels of the considered classes are known to the human partner. Finally, using the most specific classes might reduce the ambiguities, and thus the branching factor early in the search, without impacting completeness. Note that the restriction to the most specific classes is not necessary but might reduce the branching factor of the algorithm without impacting completeness.

ApplyTransition: The ApplyTransition function (Algorithm 5) creates a new node $\mathfrak{n}'$ by applying a transition to an existing node $\mathfrak{n}$. It always add the triplet of the transition to $\mathcal{T}_{\mathfrak{n}'}$ but, in case of a transition coming from the TypingTransitions function, a new variable is created and added to the mapping function $\mathcal{S}_{\mathfrak{n}}$ (line 8 of Algorithm 5). Indeed, if an entity needs to be typed, it means that it is unlabeled and thus need to be represented through a variable in a valid *reference*.

Algorithm 5 Transition application pseudo-code.

```
1: function ApplyTransition(node, transition)
2:     newnode ← a copy of node
3:     (s, p, o) ← transition
4:     if p ≡ "isA" then
5:         x ← a new variable such that x ∈ X ∧ x ∉ X_newnode
6:         X_newnode ← X_newnode ∪ x
7:         A_newnode ← A_newnode ∪ s
8:         Update S_newnode such that S_newnode(x) = s
9:     T_newnode ← T_newnode ∪ (s, p, o)
10:    return newnode
```

3.3.4.3 Implementation

The algorithm has been implemented in C++. This choice was motivated by the performance we want our algorithm to have since we aim at using it during task planning. The software must be able to solve several requests per second.

The interface between the algorithm and the ontology (to compute the hard

and soft differences and to make the SPARQL requests) has been done with the Ontologenius API, ultimately using ROS for low level communications. This allows to have the ontology and the REG algorithm to run on different computers. However, it is important to note that a low-level interface has also been used, linking Ontologenius as a dynamic library and running on the same process to avoid communications latency. This approach, while preventing Ontologenius to be used by other software, allows for the best performance. This is the version used to compare the computation times of our approach with the others in what follows.

Besides, the software has been integrated as a ROS node allowing other components, such as the supervision and the task planner presented later in this thesis, to perform REG requests in a distributed architecture.

3.3.5 Results

We present hereafter the solutions given by our algorithm to the illustrative examples. Then we provide results involving a large scale knowledge base describing a full apartment in terms of execution time, solution length and composition. Finally, we provide comparative performance measures with two state-of-the-art methods on their own domains.

3.3.5.1 Solutions Analysis

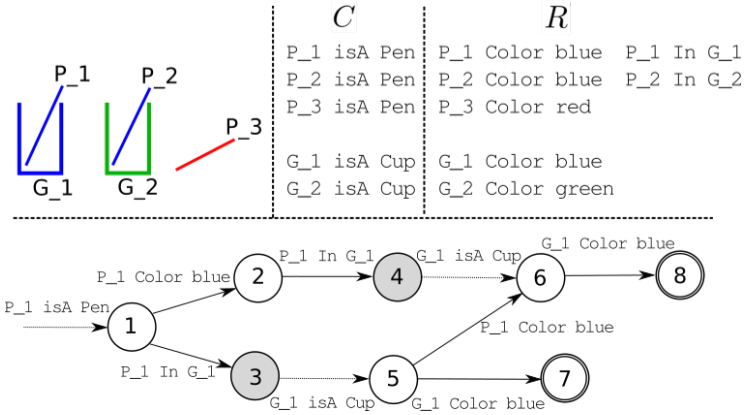

Figure 3.5: A simple example showing how our ontology-based referring expression generation algorithm explores the search space. The scene is depicted in the top left corner, and C and R represent respectively the graph of direct and inherited types of the entities and the relations between them. The graph exploration is presented to generate a referring expression for the entity *P_1*. Dotted arrows represent typing transitions and grayed nodes do not respect 3.3.2.

In order to have a better grasp of the solutions, we propose to present some of them. For every presented solution, the variable denoting the entity to refer to will be x_g =?0. The first setup is for illustration purpose, and operates on the static knowledge base K_{pens} illustrated in Figure 3.5. In this setup, three pens

of two colors are represented, two of them are in two different cups of different colors. Since this setup is really small, the context is always empty, all the relations are usable and no entity is labeled. Moreover, the knowledge base being small, we specify that all the properties P_{pens} are usable. Thus, based on this setup we propose two REG problems. The first one REG_1 aims at finding a RE for the pen *P_1*: $REG_1 = \langle P\_1, K_{pens}, \varnothing, P_{pens} \rangle$. The second one REG_2 aims at finding a RE for the cup *G_1*: $REG_2 = \langle G\_1, K_{pens}, \varnothing, P_{pens} \rangle$. We only tested with two interesting entities since the others present similar characteristics. The solution for REG_1 is *{(?0, isA, Pen), (?0, isIn, ?1), (?1, isA, Cup), (?1, Color, blue)}*. For REG_2 it is *{(?0, isA, Cup), (?0, Color, blue)}*. They can be read respectively as "the pen in the blue cup" and "the blue cup". These two solutions are complete (as in Definition 5, allowing to read "the" and not "a" in the verbalization), as ?0 and ?1 bind to only one entity. Here, we see how referring to another entity lead to interesting solutions.

In order to give the reader a sense of how the context is useful as defined in the problem, we propose to come back to Figure 3.4. In a knowledge base describing Figure 3.4(b), with a labeled entity *Bob*, representing the human, giving a empty context to the problem would lead to the solution *{(?0, isA, Pen), (?0, isReachableBy, Bob)}*, which would read as "The pen reachable by Bob". Whereas, if the robot wants the human to give it the pen, the reachability of the pen is obvious. So the context would become: *{(pen0, isReachableBy, Bob)}*, the ensuing solution would be *{(?0, isA, Pen)}*, simply verbalizable as "the pen", as taking into account the given context resolve the ambiguity.

3.3.5.2 Scaling Up

To assess the relevance of our approach, we created a larger, realistically-sized, knowledge base (101 entities, 36 classes, 40 properties and 497 relations), describing an apartment with three rooms including several furniture (tables, shelves) and objects (cups, boxes) linked through geometrical relations (*atLeftOf*, *onTopOf*) and attributes (color, weight). We ran our algorithm over all the 77 entities inheriting from the "Object" class, representing physical entities.

As this algorithm must be used in a human robot interaction application, we want it not to spoil the interaction when the robot is computing an explanation. In this setup, 100% of the entities have been referred in under 4.33ms that 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 357μs and 75% under 772μs (Figure 3.6). On average, 10.6 nodes are explored to refer to an object with an average of 67.35μs/node explored[3]. These execution times are promising from a combined use with a task planner, as many requests can be performed while planning without slowing too much the task planner.

Over the 77 entities, 32 (41.56%) are referred to with 2 or less relation meaning that only the type of the entity and one relation is needed to refer to them. We can

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

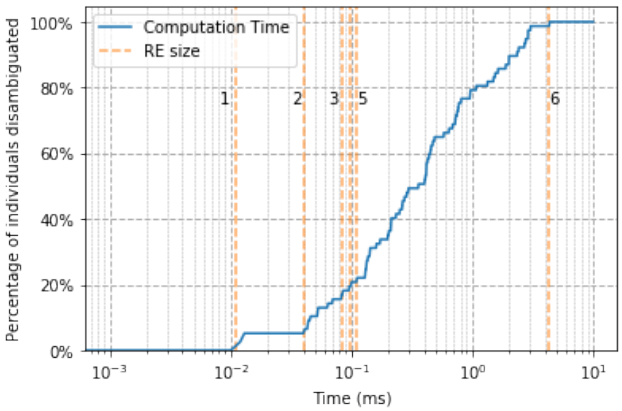

Figure 3.6: Computation times for generating a referring expression on all the 77 objects of the knowledge base representing a three rooms apartment. Orange lines correspond to the minimal size of the RE found at that duration.

also note that 25 entities (32.46%) are referred to using 4 or more relations with a maximum of 6 for one of them. Finally, 49.4% need to be referred by referring to another entity and two of them need to be referred by referring to two other entities. This means that 49.4% of the entities can not be referred using approaches like [Ros 2010] or [Dale 1995].

These results over a large scale knowledge base highlight the need to be able to refer to an entity through the use of relation linking it with other entities. They also show that the use of the type of an entity is often sufficient with the use of only one attribute. With this experiment, we also demonstrate that our algorithm is suitable for use with a realistic large scale knowledge base.

3.3.5.3 Comparisons With Other State-of-the-Art Algorithms

Longest First The Longest First (LF)[4] algorithm [Viethen 2013] has been tested on the GRE3D3 Corpus composed of 20 scenes with three objects with different spatial relations relative to one another (*onTopOf*, *atLeftOf*). Each object can be referenced by its color, its size (large or small) and its type (cube or ball). The target referent is marked by an arrow and is always in a direct adjacency relation (*onTopOf* or *inFrontOf*). Among the 20 scenes, 8 target objects can be referenced without any ambiguity using only their type, 7 can be referenced using only their type in addition to an attribute (color or size) and the other five can be referenced using their types and both color and size attribute. This means that spatial relations are never necessary to reference the target object. We perform the comparison on the 19^{th} case which consists of a small green cube on a large green cube and a small blue cube to the right of the green cubes (Figure 3.7). We chose this case with only cubes because the LF algorithm does not consider the types when generating the

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

RE and adds them only as a post-process. The other cases requiring only the type are resolved in less than 100μs and those requiring the type and an attribute are resolved in less than 250μs with our algorithm.

Since their objective is to obtain an over-specification of the RE, their results are strongly impacted by the maximum length parameter. By setting it to 4 as recommended, we get the result which we can read as "*The small green cube on top of a cube*" in 311ms. By setting the maximum length to 3 we obtain the shortest admissible result which can be read as "*The small green cube*" in 109ms. This last result is the one given by our algorithm in just 0.87ms.

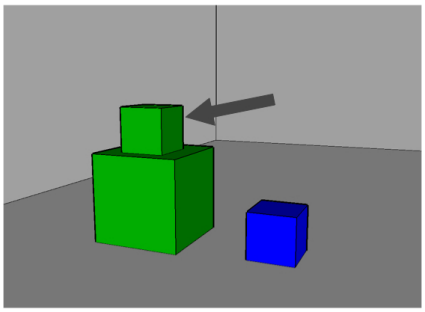

Figure 3.7: The scene 19 of the corpus GRE3D3 [Viethen 2013].

We see here that the results given by the LF algorithm largely depend on the maximum length parameter. This parameter also has a significant impact on the execution time. Besides, in the realistic scenario presented previously, 13% of the entity need a reference expression length greater than 4. Thus, even if the over-specification is the goal of the LF approach, it can hardly scale-up. Moreover, for a maximum length fixed at the optimal length, both their approach and ours give identical results.

Graph Based Algorithm A computationally improved version of the original Graph-Based Algorithm [Viethen 2013] is presented in [Li 2017]. It aims at extracting, from a dedicated entities relations graph G, the lowest cost subgraph which is graph isomorphic to one and only one subgraph in G containing the entity to refer to. Their approach is evaluated on a corpus containing multiple tabletop scenes [Scalise 2018], presenting numerous cubes of different colors.

We generated the graph (relations and costs) used for scene 1 (Figure 3.8), converted it into an ontology, and ran our algorithm on it. This scene contains 15 cubes, GBA algorithm and ours are able to find a solution for the same 10 of them. In all the 10 cases, as we used the same costs, both algorithms returned the same solution (with the types of used entities added in our approach). For the other 5 cases, the two algorithms detect the absence of a solution in a few milliseconds.

On all the 10 cases with a solution, our approach performs faster than the GBA algorithm (29.4 times faster on average). We can note that the speed increase is more important in cases where there are many solutions (under 4 times faster on 50% of the cases, but more than 50 times faster for 25% of the cases, up to 130

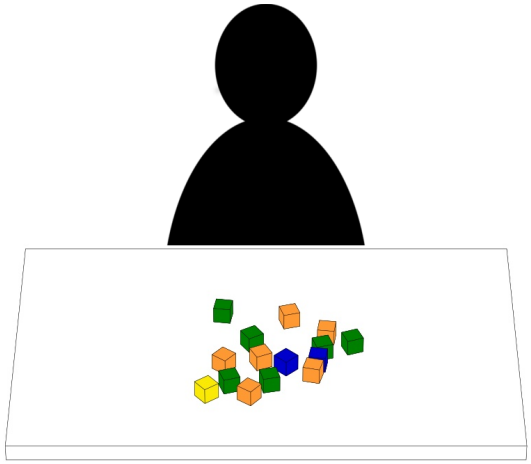

Figure 3.8: The scene 1 of the Li & Scalise corpus.

times faster). Indeed, the GBA approach uses a branch and bound algorithm where the search graph is bounded if the branch exceeds the cost of the current best found solution. Thus, it can explore a large part of the graph if the optimal solution is not found early in the search. Whereas our approach uses a uniform cost search algorithm, ensuring the first found solution is optimal. Moreover, we think that in cases where the knowledge base contains entities with different types, our approach should work faster, since we prioritize the use of the type. We were not able to test this, as we could not manage to run the GBA algorithm on other data than their own corpus.

3.3.6 Integration

Our ontology-based REG method has been integrated on a PR2 robotic platform and used in a tabletop scenario. The used architecture presented in this section is represented in Figure 3.9.

The objects on the tables are detected with the ROBOSHERLOCK[5] perception system [Beetz 2015]. This software does not require any a priori on the environment such as CAD model, mesh or training. It provides the position (not used to extract relations), the shape ("circular" or "rectangular"), represented through a *hasShape* property in the ontology; the color, represented with a *hasColor* property; and the size of the objects ("large", "medium" or "small"), represented with a *hasSize* property. Since the types of objects is not determined by the system, all the objects were set in the ontology with the labeled type "Object". This allows us to challenge our method with situations where the robot is not able to use high-level concepts and where various ambiguities will be raised.

As presented before, the knowledge base is managed using the Ontologenius[6] system [Sarthou 2019]. It uses a custom internal structure to store and manipu-

[5] `http://robosherlock.org/`
[6] `https://sarthou.github.io/ontologenius/`

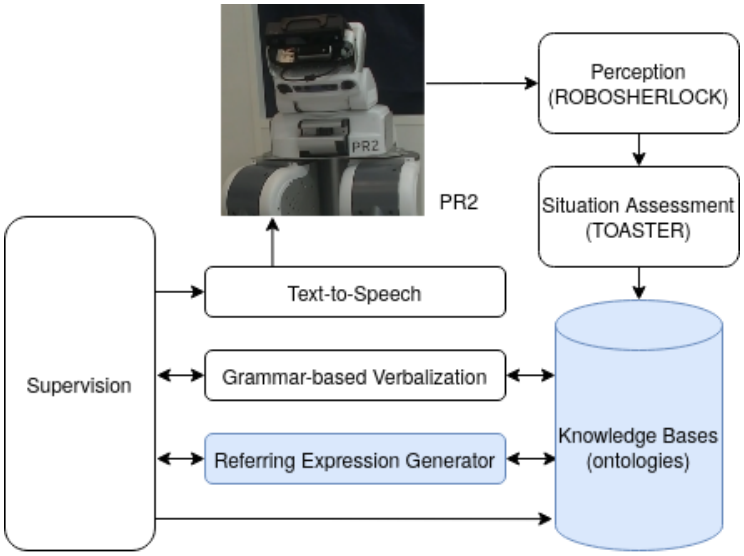

Figure 3.9: The robotic architecture built to test the presented REG approach in real conditions. The blue components are the ones presented in this thesis.

late assertions as triplets, and offers reasoning capabilities in the form of plugins. Ontologenius provides a low level API allowing to manipulate the knowledge base as a classical data structure in addition to a SparQL interface. The ontology is dynamically fed to keep it up to date on the basis of a simple situation assessment consisting only of filtering and object tracking. The software used is TOASTER, which is an improved version of SPARK [Milliez 2014]. However, only basic capabilities are used, consisting solely of filtering and object tracking. Based on the confidence on the properties extracted, it dynamically feeds the knowledge base to keep it up to date.

A simple linguistic realization has been made, taking as input a SparQL query and generating an English sentence as output. For example, it transforms the query "?0 isA Cup, ?0 isOn ?1, ?1 isA Table, ?1 hasColor black" into "*the cup on the black table*". It is an ad-hoc implementation based on a simple grammar and on the labels contained in the ontology.

Finally, a supervision component was built by another PhD. student Amandine Mayima. This supervision component orchestrates all the other ones. It receives as input an object being clicked on the RViz visualization, finds its identifier from the ontology, requests the REG component for a referring expression to this object, then sends it to the verbalization component and finally sends the produced sentence to the text-to-speech component making the robot play it.

The task involves six objects on a table (Figure 3.10). A commented video is available at `https://youtu.be/mKDLvDbHfvk`. The entity to reference is *obj_4* (a white mug). The robot generates the solution "*The white circular object*" since there are other non-white circular objects and other white non-circular objects (Figure 3.10(a)). Then, a human adds a new object which is a white and circular

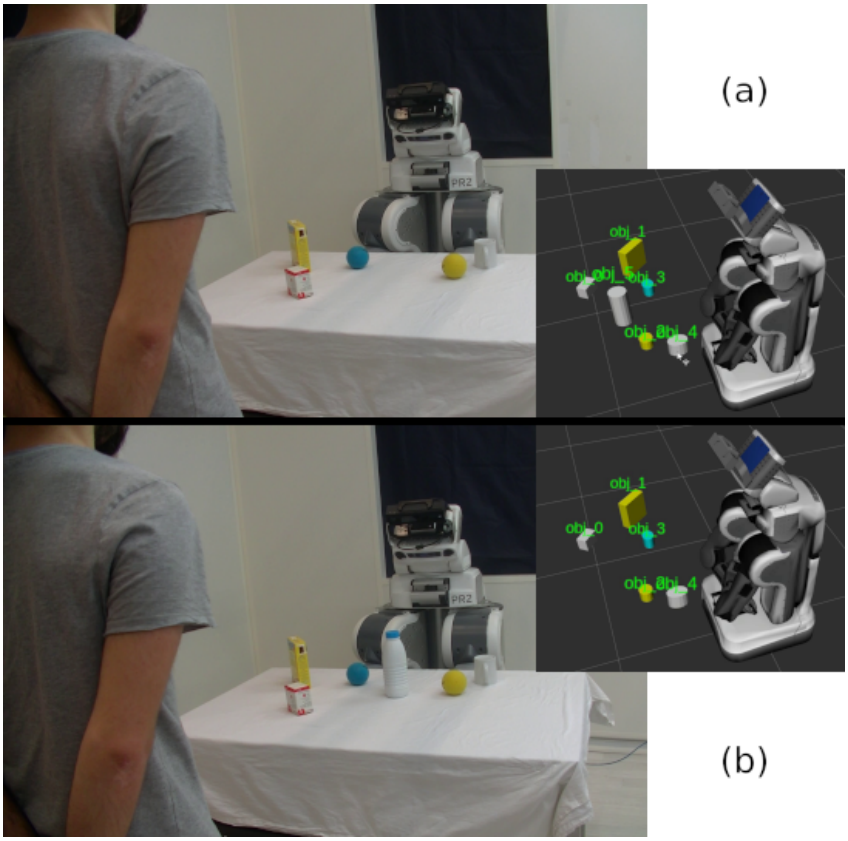

Figure 3.10: The experimental setup for the integration of the REG algorithm. The robot only perceives the size, color and shape of the objects on the table. It estimates that the human is also aware of these attributes. To refer to the white mug (*obj_4*) the sentence generated in condition (a) is *"The white circular object"*, while in condition (b) it is *"The white small circular object"*. Indeed, in (a) only the color and the shape of the mug are enough to discriminate it, but in (b) a milk bottle is added, as it is also small and white, the algorithm adds the size of the mug to referring expression.

milk bottle (*obj_5*) (Figure 3.10(b)). When the robot is asked to describe the white cup, it generates the sentence "*The white small circular object*". With this simple task, we show that our REG algorithm can be used within a robotic architecture, can deal with a dynamic environment and can adapt its explanation to the current situation.

3.4 Planning Communication Actions Using Referring Expression Generation

We are now able to determine efficiently the content needed for the robot to refer to an object to a human based on an ontology as knowledge base. Moreover, this algorithm is able to determine if such a communication is feasible (whether it fails or not to find a solution) and gives an estimation of the cost of interpreting the referring expression. In what follows we integrate this approach in a task planner, allowing to evaluate the feasibility and the cost of verbal referring communication actions during task planning.

3.4.1 Method

In this section, we first provide an overview of our approach and briefly describe the used Hierarchical Agent-based Task Planner [Lallement 2014]. Then, we present the integration of the REG in it and how it allows the planner to be informed of the feasibility and the cost of the communication actions.

3.4.2 Approach

The communication actions that we consider in this paper are instructions issued by the robot to its human partner based on Referring Expressions (REs). Typical instructions are "*Take X*" and "*Put it in Y*". We thus have a static part and the rest depends on the situation when the communication is performed and must be solved by a REG. It is this variable part that could make a communication costly or infeasible. As stated before, REG must be performed on the human's partner Knowledge Base (KB) to only use facts and concepts that the robot estimates to be known by the human. Thus, we target a planner that is already suitable for HRI to integrate the estimation of communications. This means that we need a planner able to distinguish between the different agents involved in the task and to maintain a representation of the environment for each of them.

Because the task the planner has to solve does not necessarily imply all the elements present in the current environment, the planner does not need a full representation of the environment. In the same way, it does not necessarily need to have all the characteristics of the entities such that their colors or their types. In the example of Figure 3.1, the two keys to move can only be represented as movable objects in the planner and not as keys to make the planning domain more generic.

However, the REG needs all the semantic information of each entity of the environment to generate accurate RE. Furthermore, if another key which is not part of the task, thus not part of the planner internal representation, is present on the table (such as *key_3*), it will also impact the REG and thus the complexity and feasibility of the communication action. Hence, the REG can not be performed on the planner internal representation of the world only. To solve this issue, we endow the planner with the ability to update a semantic KB that is used by the REG. Since maintaining this external representation can be a heavy process, it is updated only when a communication action has to be evaluated.

The general workflow executed for each communication action encountered during the planning process consists of: 1) updating the external semantic KB of the human partner with the expected world state 2) identifying the objects to which to refer to in the communication 3) execute the REG for each of these objects 4) calculate 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 the examples used in this paper only involve one RE but the same method can be used for communications of type "*give me X and Y*". In this case, the external semantic KB is only updated once and both REG are executed on this KB.

3.4.2.1 Hierarchical Task Planner

In order to implement our approach, we need a task planner able to maintain an estimated knowledge base of each agent at each planning step. We chose the Hierarchical Agent-based Task Planner (HATP) [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 totally compatible with them.

Moreover, during the exploration of the task tree, HATP will assign actions to available agents, robot or human (when an action can be done by both). By doing so, HATP is able to 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. The value of each variable depends on the agent it is requested for. 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 previous integration of HATP with a Geometrical Task Planning (GTP) [Gharbi 2015a]. This work aimed at refining geometric and motion planning requests during the task planning process. The geometric planner would then compute, in context, the feasibility, the cost and the side effects of the action. In a similar way, we propose here to integrate and run REG, in context, to determine communication action feasibility and pertinence with respect to other courses of actions.

The HATP Data Structures As presented in [De Silva 2015], the HATP data structures are organized around the so-called *HATP entities*. These entities are a collection of any number of attributes being manipulated during the planning process. The *type of the attributes* can either be a basic type (*i.e.* integer, floating point number, boolean or string) or another entity type. Besides, an attribute can either be *a set*, holding multiple values of the specified type or *an atom*, being a unique element of the specified type. Finally, an attribute can either be set as *static* or *dynamic*. Static attributes cannot be modified once they have been set in the world state initialization, they will only be read during the planning process, whereas dynamic attributes can also be changed by the effects of the primitive tasks (actions). The Listing 3.1 gives an example of entity definitions in order to represent the situation in Figure 3.1.

```
// Agent entity type is implicit
define entityType Cube, Area;
define entityAttributes Cube{
  dynamic atom Area isIn;
  dynamic atom Agent isHeldBy;
}
define entityAttributes Area{
  dynamic set Cube hasIn;
}
define entityAttributes Agent{
  static atom string type;
  dynamic atom Cube isHolding;
}
```

Listing 3.1: Example of a part of the HATP domain describing the situation of Figure 3.1.

3.4.2.2 Integration of REG Within Action Planning

The representation of the communication actions: For clarity purposes, we only place ourselves in scenarios 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. Thus, while planning, if a task is allocated to the human,

as she has no way of guessing it, a preceding communication is required. In the HATP domain, this translates as a method being decomposed into a sequence of a communication action (the instruction to the human) and an action made by the human when the task is attributed to the human. The communication action feasibility is 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 depending on the human receiver and the entities to refer to in the communication.

We have chosen here for illustration purposes a simple planning problem where a communication needing a REG is involved in each plan step concerning the human, but the method is general and compatible with problems which need to estimate and ensure the pertinent context and plan step (the *when*) of a communication action during plan elaboration (*e.g.* [Devin 2016], [Unhelkar 2020]).

Updating the right knowledge base, at the right time: On one hand, we have large, complete semantic knowledge bases on which a REG algorithm is able to run and to return the feasibility, the cost and the content of a verbal entity referring communication for a specified agent (top part of Figure 3.11). On the other hand, we have reduced knowledge bases dedicated to task planning (bottom part of Figure 3.11). In order to know the feasibility and cost of a verbal communication action during the planning, we have to reconcile both sides. Indeed, the estimated ontology of the communication receiver must be updated to reflect her planned estimated beliefs at the time of the communication. All the knowledge representation used here are from the robot point-of-view and managed internally by the robot decisional and knowledge management processes.

First, the attributes of all the entities present in the planning knowledge base are initialized for each agent (left part of Figure 3.11). To do so, the name of every HATP entity types declared in the planning domain are listed. Then, for each entity type, the entities in the ontology inheriting from these types are created in the planning knowledge base. Then, each attribute (both static and dynamic) declared in the domain of every entity 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 ontology. If so, all the pointed entities are added to the set. If the attribute is an *atom*, only one value is retrieved from the ontology and is set to the planner entity attribute. For instance, if we initialize the planning knowledge base defined in Listing 3.1 with the content of the ontology depicted in Figure 3.2 and Figure 3.3 representing the situation of Figure 3.1, the entity types retrieved from the domain would be *Agent*, *Key*, *Area*, the planning knowledge base would then be initialized with the entities *human_3*, *pr2_robot* as Agent; *key_1*, *key_2*, *key_3* as Key; and *area_red*, *area_white*, *area_black* as Area. Then, the attributes of *key_1* would be retrieved from the ontology and set as $isIn = area\_red$, the same process is repeated for all the other attributes. Attributes that are not in the ontology can be set in the planning domain manually. Finally,

an ontology dedicated to the task planning process (called planning ontology) is created by copy of the present one for every agent other than the robot present in the planning domain. These copies are made to avoid modifying the original ontologies during the planning process as other components may rely on them.

When a communication action is encountered during the task tree exploration (right part of Figure 3.11), the ontology of the communication receiver needs to be updated to be able to run the REG on it. The planning ontology copy of the receiver human is retrieved by her identifier. Then, for each of the entities of her planned beliefs at the time of the communication, an update is made. The update is only made on *dynamic* attributes as *static* ones do not change during the planning process. All the relations having the same name as the attribute of the entity in the planning domain are deleted from the ontology, and replaced with new planned values. If the attribute is a *set*, a new relation with the same name is created for every value in the set.

A REG request is then issued on the updated ontology with the goal individual being the entity to refer. The REG returns a solution with a cost or a failure which is taken into account by the planner as classical cost or a non fulfillment of the action preconditions respectively. Alternatively, a communication action may need to refer to multiple entities. In that case, multiple REG requests are issued on the same updated ontology and their costs are summed.

With this approach, we are able to change the *context* and *usable relations* provided to the REG algorithm depending on the task and world state the planner is in. For example, if the human and the robot are in front of the same table, the context would be to consider only entities on that table. Moreover, if the currently explored action is to ask the human for pick, the context provided would be to consider only the *Pickable* objects that are estimated in the human beliefs to be *reachableBy* themselves.

By doing so, we are linking two knowledge bases: the HATP on one side and the ontology on the other. While the former can hardly represent inheritances between entity types and deduce facts as we do not want to put more load on the task planner, using the latter at specific chosen steps of the task planning process allows to use its rich semantic to make the resulting plan more precise.

3.4.3 Case Studies

In this section, we present three case studies. The two first ones are run in simulation on a minimalist setup and show respectively that the estimation of the communication content during the planning can prevent execution dead-end and can reduce the global communication complexity during the task. The third case study is run on a PR2 robot with a perception of its environment and presents a more complex task with twelve objects to organize. With this last case, we show that our method makes it possible to compare different means of communication and to choose the most appropriate.

To realize tests in laboratory conditions, we chose to replace the keys of Fig-

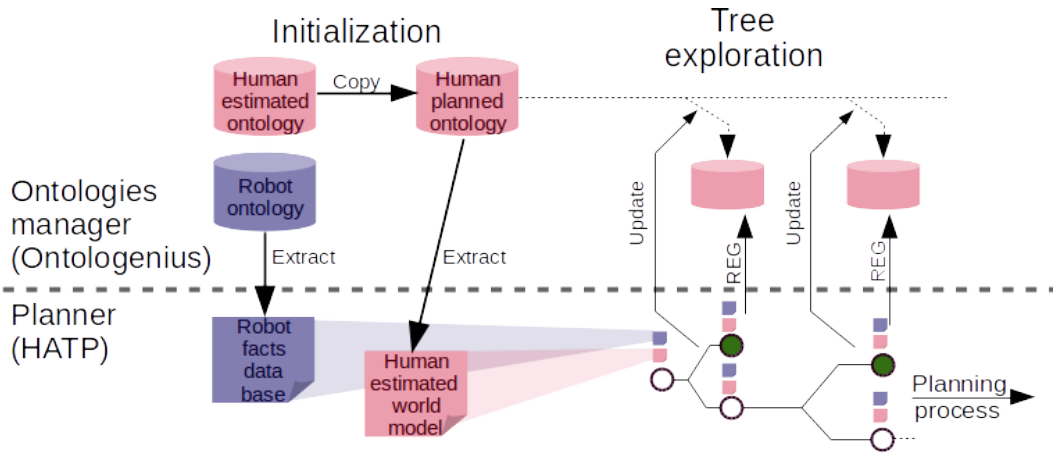

Figure 3.11: A representation of the exploration of potential mental states and ontologies conducted by the planner. The ontology representing estimated human knowledge is first copied in order to plan it without altering the original one. The human and robot planning information is extracted from the ontologies. During the tree exploration, for each verbal communication action, the planned human ontology is updated with the current explored state and the REG is executed on it.

ure 3.1 with cubes (Figure 3.12). Thus, all three case studies are based on a cube arrangement task. The human can distinguish the cubes by their color and the digit written on them (one or two) if there is one. As in Figure 3.1, the table surface is composed of three storage areas of different colors and cubes can be placed only in one of them. This symbolic position information can also be used by the human to distinguish the cubes. In this way, the robot can refer to a cube with a REG of the type: "*the black cube with the number 2 which is in the black area*". In all the cases, only the robot knows the goal position of the cubes but can not manipulate them. It thus has to guide the human in the arrangement task. The robot can only point at the cubes in the third case study. In the first two, it can only use verbal communication.

3.4.3.1 Preventing Execution Deadlocks

In this case study, we consider the initial state presented in Figure 3.12(a). The cube *C1* is in the red area and the cube *C2* in the black one. The goal state is to have the cube *C1* in the black area and the cube *C2* in the white one (Figure 3.12(b)). Taking into account the cost and the feasibility of the communication, the planner elaborated the plan presented in Listing 3.2. Cube *C2* is moved first because otherwise 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 require another communication mean.

```
HR - TellHumanToTake(C2) // (?0, isA, Cube), (?0, isIn, ?1),
                         // (?1, isA, Area), (?1, hasColor, black)
```

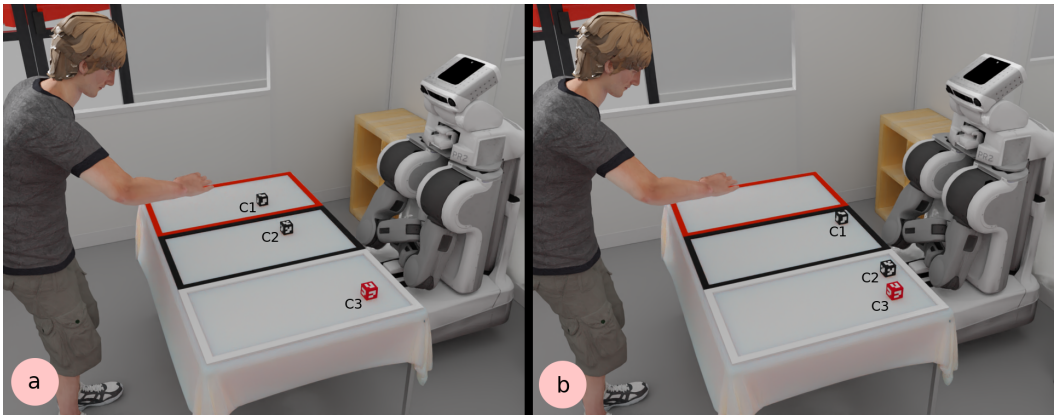

Figure 3.12: The situation depicted in Figure 3.1 with keys replaced with cubes to realize the setup in laboratory conditions. The human and the robot are in the situation depicted in (a). Only the robot knows the goal configuration (b) but cannot move the cube. It has to elaborate a plan in which it asks the human to move the cubes.

```
H  - Take(C2)
HR - TellHumanToPlace(C2, AW) // (?0, isA, Area),  (?0, hasColor, white)
H  - Place(C2, AW)
HR - TellHumanToTake(C1) // (?0, isA, Cube), (?0, isIn, ?1),
                         // (?1, isA, Area), (?1, hasColor, red)
H  - Take(C1)
HR - TellHumanToPlace(C1, AB)  // (?0, isA, Area), (?0, hasColor, black)
H  - Place(C1, AB)
```

Listing 3.2: 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.

We consider once again the initial state presented in Figure 3.12(a). This time the goal is to invert the positions of the two cubes. In this situation, 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 deadlock at plan execution. Indeed, regardless of the first cube moved, if it is moved directly to its goal position, it will end up in the same area as another cube of the same color. In such a situation, no RE can be found for the other cube, leading to a deadlock in the execution. The solution found with our method is to add a supplementary action. It consists of putting the cube *C1* away (in the white area). This additional action avoids a deadlock by making the referring to the cube *C2* feasible.

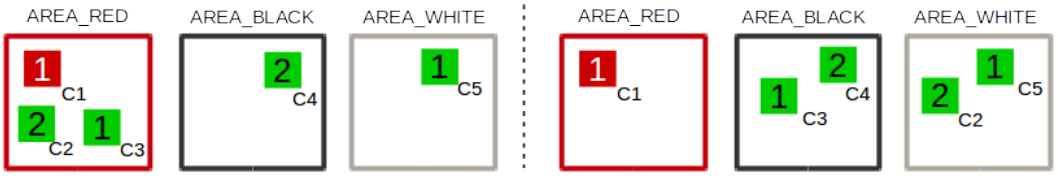

Figure 3.13: The initial state (left) and the goal state (right) of a task where the robot has to explain to the human partner how to move the cubes to complete the task.

3.4.3.2 Reduction of the Overall Communication Complexity

In this second case study, we show how the estimation of communication by verbal designation can be used to reduce the complexity of global communication. This time we consider the initial state and the target state represented in Figure 3.13. Only cubes *C2* and *C3* should be moved. The extended HATP with REG capabilities finds the solution consisting in 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, the cube *C3* can also be referred to only by three relationships 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 relationships: its type, its color, the number on it and the colored area in which it is located. The solution chosen by HATP extended with REG capabilities communicates a sum of six relations rather than seven with the reverse order.

3.4.3.3 Balancing Between 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 3.14. Such a number of similar objects leads to long explanations to refer to certain cubes. Therefore, we would like to end with the task planner choosing another means of communication to refer to these cubes (*e.g.* a pointing action). We model the pointing action as having a constant cost that is higher than a simple verbal referring expression 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.

This setup has been implemented on a real PR2 robot. The architecture is similar to the one presented in Figure 3.9, but HATP has been added, and the perception has been replaced by an AR tags reader. The verbalization has been made through an ad-hoc grammar-based component, and a simple supervision has been written by Amandine Mayima to follow the plan generated by HATP. Finally, a situation assessment component named TOASTER, inspired from SPARK [Milliez 2014] has

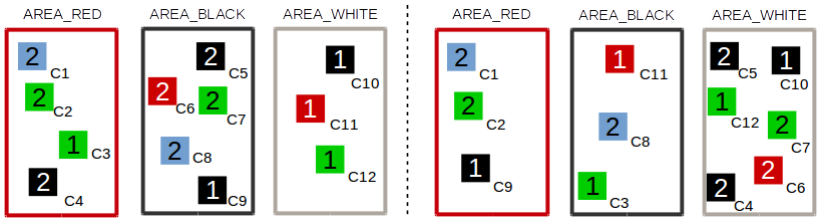

Figure 3.14: The initial state (left) and the goal state (right) of a task where the robot has to explain to the human partner how to move the cubes to complete the task.

been used in its simplest form, only to read AR tags from the cubes, finding if their position is inside predefined zones on the table (the area) and feeding the ontology with relevant facts. The execution of the computed plan can be found in the video available at `https://youtu.be/3YnGh_t-UpY`.

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 pointing motion is preferred. For example, the cube *C7* in the initial situation needs a long and complex explanation that is: *"Can you take the green cube, with the number two, which is in the black storage area"* (Figure 3.15). Even in the case where the pointing action takes more execution time, it could be faster for the human partner to interpret 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, verbal communication is balanced with other communication means, but it can also be balanced with other actions or assignments requiring less or no communication.

3.5 Conclusion

In this chapter, we have presented an approach allowing to accurately estimate the feasibility and the cost of verbal communication actions containing referring expressions during task planning.

First, we introduced the referring expression generation problem and proposed a formalization of it dedicated to human robot interaction scenarios. Then, we presented an efficient algorithm generating referring expressions by using an ontology as a knowledge base.

Finally, we integrated this algorithm into a task planner HATP, allowing it to resolve the content of such verbal communication when needed. With this extended version of HATP enhanced with REG capabilities, the planner can check if communication can be done in the planned world state, and it can plan different order or sequence of action depending on the communication estimation.

However, even if HATP is able to maintain one belief base per agent during the

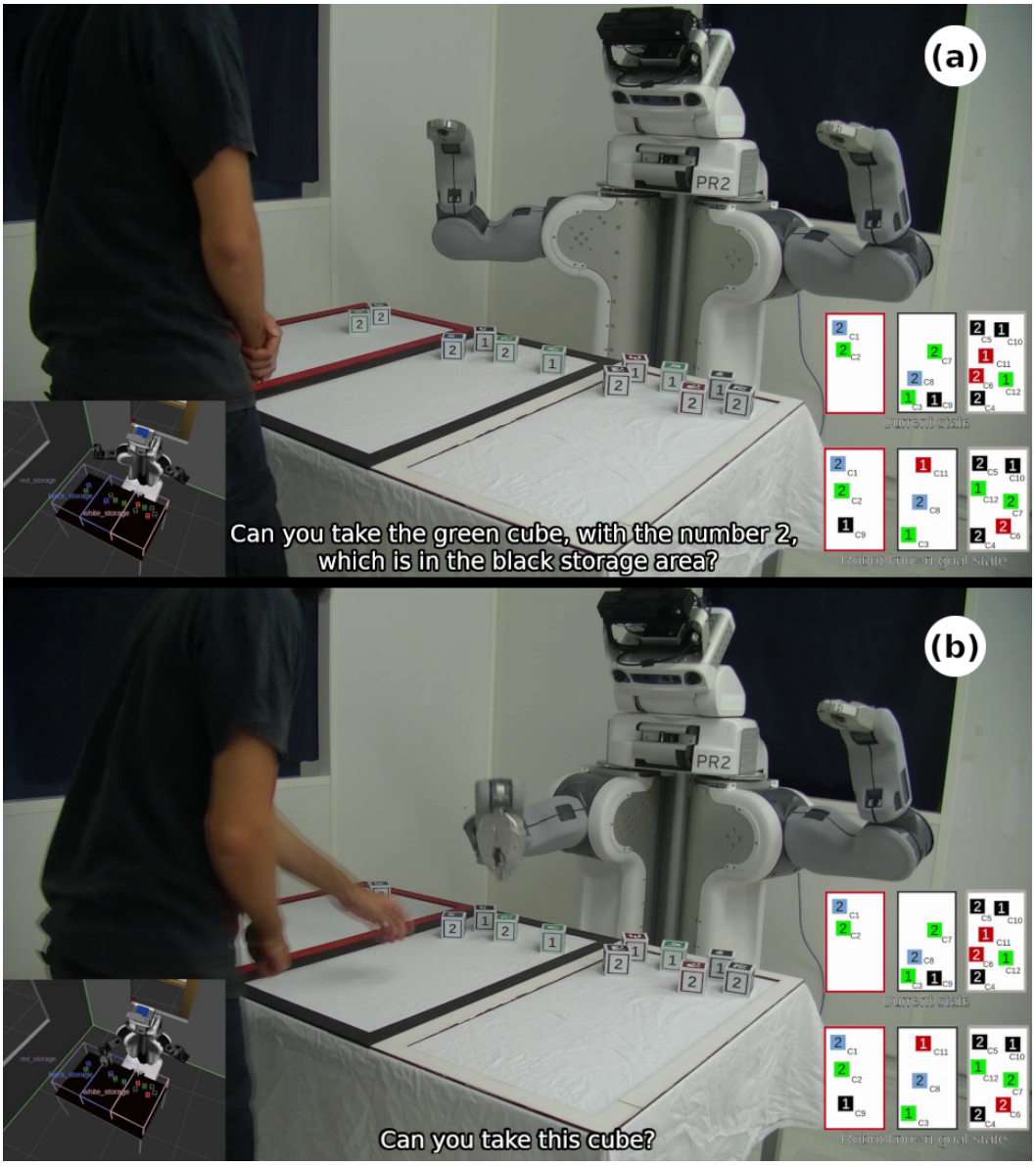

Figure 3.15: The PR2 robot and a human sorting cubes. To designate the cube C7, the robot can either use the verbal referring (a) or point at it (b). However, the planner has found the pointing less costly for the human to understand and thus, has returned a pointing action in the executed plan (b).

planning process, it only allocates task between agents (considering the capability of each one) but does not ensure that the generated plan is known to the human nor that they have every piece of information needed to accomplish the task. Moreover, we do not take into account that the human might also plan for their own goal (which may or may not be shared with the robot). We propose to extend this

approach by not only planning for the robot and the human, but instead planning for the robot and emulating the action, reaction and planning processes of the human the robot is interacting with.

Chapter 4

Emulating the Human Decision and Action Processes During Task Planning

Contents

4.1 Introduction

In the previous chapter, we successfully integrated a verbal communication planner for referring expressions into a multi-agents human-aware task planner. This allows for a communication action containing reference to an object of the environment, to find its feasibility by not relying only on symbolic facts the task planner is manipulating but by precisely determining its content. Moreover, by determining the content of the referring expression, the complexity for the human to understand it can be estimated and is returned as a cost to the task planner. Thus, during the

execution, we can avoid many plan failures, repair actions or inefficiency. Although we saw that this approach needs a task planner able to maintain one set of beliefs per agent during the planning process, HATP, the planner used was only allocating tasks to the human without considering if the human was aware or not of the generated plan.

HATP is provided with a human model ($\mathcal{M}_r^H$) in several ways. The planning domain contains the actions the humans can perform and their beliefs are updated all along the planning process. Moreover, human preferences can be set on the plan through the *social cost*, adding extra-cost to the plan if some criteria are met such as bad chaining of actions (*e.g.* the robot mopping the floor just before making a sandwich). However, HATP assumes that a shared goal has been established between the robot and the human. This shared goal is supposed to have been acquired previously in the interaction, for example via a human request. Besides, as stated before, it generates a plan which is unknown if it needs to be communicated to the human or if it can be easily *guessed* (*i.e.* which is predictable) by the human. Finally, the plan is "fixed" and does not account for whether the human chooses to perform a different path of actions or not. Any deviation of the human from their generated stream of action either needs the supervision to perform repair actions or to request for a replanning. This approach has been shown to be suitable and pertinent in some applications (*e.g.* when communication can easily be done at any point of the plan).

In this chapter, we present a novel task planning approach dedicated to HRI which, by planning for both the human and the robot, tries to satisfy multiple objectives:

1. **Plan without assuming a prior shared goal**. In HRI scenarios, the robot and the human are not always sharing a goal. The robot can for example plan to perform a task around humans that are not involved at first, or it may be requested by a human to do a task without wanting to take a part in it. By doing so, our planner can balance between integrating the sharing of a goal with a human (assumed to be collaborative) in the plan and making the robot do the task alone, or integrate the eventuality to ask for punctual human help.

2. **Model the human decision processes**. When taking part in a task, a human (assumed willing to collaborate with the robot) will also plan to reach their (potentially shared) goal. Our planner must be able to account for this to provide plans that are expected and explainable by the human partner.

3. **Help the human decision, but not compel it**. Unlike HATP, our planner should account for the human flexibility in their decision. While by modeling the human decision processes it is possible to narrow down the possible human actions, the generated plans must be able to help the supervision to avoid to replan or to repair during the execution by considering several human actions.

4. **Model the potential human reactions**. It is possible to predict that the

human may react to some situations, interrupting or helping their current task. We identify two causes of these reactions. First, they can ensue from some specific world states, that have been perceived and interpreted by the human. Then, they can also originate from explicit communications issued by the robot. These communications can either be a belief alignment, updating the human knowledge and impacting their decisions; a request to perform a specific action or a request to help the robot along with a shared goal, needing the human to plan for it.

5. **Act and decide on the different agents' beliefs**. It is important to be able to represent actions as having different effects on the beliefs of the robot or the human. Indeed, some robot actions are partially or not observable by the human, when performing them, the human has no way of knowing the complete new world state. Besides, these effects and their observability often depend on the current world state, which representation must be supported in the planner. Then, decisions made while planning may require to reason on both the robot and the human beliefs. This is especially the case with communication actions aiming at aligning knowledge or ask questions for example. Finally, some actions of pure decision have no direct effect on the world, but only on the internal beliefs of the agents. For example, observation actions will only update the beliefs of the agent doing it.

6. **Decide not only on the world state but also on the decision processes of the agents**. Some decisions made during the planning process require access not only to the beliefs of the agents representing the world state, but also to the estimation of their planning processes. For example, the decomposition of a task by the robot may be impossible if some other task is already performed in its partial plan. Other decisions may also need the estimation of the human current planning process. For example, if it has been estimated earlier in the plan that the human will perform a certain decomposition of a task, the planner would assign a complementary task to the robot.

7. **Adapt to the human experience, trust and preferences**. We also want the planning process to be adjusted depending on the actual human it is planning with. It must perform its plan search differently whether the human has the habit to perform this particular task with the robot or not. Moreover, the human model can be adjusted to the trust the human has in the robot and to their preferences.

First, we describe our approach and introduce the notations used in this chapter before detailing the planning process. Then, we present the implementation of this approach into a prototype planner in Python that we named Human Aware Task Planner with Emulation of Human Decisions and Actions (HATP/EHDA). Finally, we demonstrate the planner capabilities in two illustrative situations.

4.2 Description

We separate the agents who may take part in a given task into two categories: the controllable agent (*i.e.* the robot) for which the planner needs to select the best course of actions to generate a plan; and the uncontrollable agent (*i.e.* the human) on whom the planner has no direct control but, still, has a representation of their decision, action and reaction models.

The two agent types are fundamentally different:

1. the robot is controllable since the process is run by the robot,
2. the human agent is not controllable since the process can only "speculate" on their decisions and actions, but can model that the robot actions can still influence them and that some of them are observable by the robot,
3. the two agents are not equivalent, the robot agent role is to help, assist and facilitate human and to synthesize pertinent, legible and acceptable behavior.

We want to devise a planner allowing the controllable agent to plan for its actions while anticipating the decisions, actions and reactions of the uncontrollable agent. Moreover, we want the planner to be able to generate plans where the robot actions elicit situations calling for human decision, action and reaction, thus creating and anticipating collaboration and interaction.

This problem may be seen as a classical non deterministic planning problem, but enriched with the ability of the robot to model the actions, beliefs and decision process of the human. Thus, we have to consider distinct action models, beliefs and execution streams for each of the agents involved. Doing so with classical STRIPS-style planning approaches would potentially lead to an intractable search space. Moreover, HTN approaches have already been shown to be suitable for HRI as they allow to communicate about the plan more easily [Lallement 2014]. Therefore, we chose to use HTN planning for both the controllable and uncontrollable agents. HTN planning aims at decomposing abstract tasks into atomic, primitive tasks by choosing from a list of available context-dependent refinements for each abstract task, ensuring that preconditions and effects of refined primitives tasks are satisfied throughout the created plan. Similarly to HATP [Sebastiani 2017], our planner elaborates a plan with several streams of actions each assigned to an agent involved in the task. But while in HATP, all the streams are built starting from on initial root node corresponding to a shared goal between all agents, our planner starts from multiple initial root nodes corresponding to the decision process of the different agents.

The main structure manipulated by our planner is the **agent**, more precisely two will be represented, the *human* and the *robot*. Each agent has their own **beliefs**, **action model**, **agenda**, **plan** and **triggers**. The planner has to use their action models and beliefs to decompose the tasks in their agenda into primitive tasks (actions) that are inserted in their plan. By doing so, it also has to update the beliefs of each agent and to model their reaction by executing the triggers.

Agents: First, we define an agent state as a tuple $\sigma_\alpha = \langle d_\alpha, \pi_\alpha, s_\alpha \rangle$, with d_α the agenda, π_α the partial plan and s_α the beliefs of the agent α (more details are presented in what follows). Then, we define an agent as being $\alpha = \langle \text{name}_\alpha, \sigma_\alpha, \Lambda_\alpha, Tr_\alpha \rangle$, with name_α the agent name, σ_α the agent state, Λ_α the action model and Tr_α the triggers of the agent α (detailed in what follows). Then we define two agents: the controllable one — the *robot* —; and the uncontrollable one — the *human* —. We have $\sigma = \langle \sigma_{robot}, \sigma_{human} \rangle$ representing an agent**s** state, being the state of all the agents at a certain plan step. Let Σ be the set of all the possible agents states.

Beliefs: Let S be the set of all possible world states, we call beliefs of an agent α the state $s_\alpha \in S$ in which this agent thinks the world is in. It is important to note that the state of the controllable agent (robot) is assumed to be the real world state estimation for the planner, as we consider the planner as being part of the controllable agent.

Action models: We represent the action model of an agent α as $\Lambda_\alpha = \langle Op_\alpha, Ab_\alpha, Me_\alpha \rangle$ where Op_α are the primitive tasks (*i.e.* operators, actions) that the agent α can perform, Ab_α the set of abstract tasks and Me_α are the methods (*i.e.* decompositions) describing how an agent α can perform an abstract task through a refinement process. It is important to note that while this representation makes a clear distinction between the robot and the human tasks, it does not prevent representing joint abstract tasks or tasks that can be either done by one or the other agent. Indeed, as we show later, complementary abstract tasks can be represented and some tasks can have the same operational model even if they are not in the same agent action model.

More precisely, the primitive tasks (operators) are defined as functions: $Op \ni o : \Sigma \to \Sigma \cup \bot$ which produce new agents state, being the effect of the application of the primitive task, or *false* if the task is not applicable. We represent operators as being instantaneous (or all having the same duration) in their realization. In the future, to represent more accurately intricate coordination, we want to include the expected duration of an operator.

Then, methods are defined as tuple, containing an abstract task and a decomposition function: $Me \ni m = \langle \alpha, \delta \rangle$ with $\alpha \in Ab$ and $\delta : \Sigma \to (Op \cup Ab)^n \cup () \cup \bot$ with $n \in \mathbb{N}^*$, which, depending on agents states, decompose the abstract task returning a sequence of tasks (primitive or abstract), an empty sequence if the abstract task does not need to be decomposed, or *false* if the task cannot be decomposed in the current state. Multiple methods can address the same abstract task, the goal of the HTN planner is then to choose the right one to create a plan.

Agents agendas and plans: An agenda d_α and a plan π_α (this agent only stream of actions) are defined for each agent α. The agenda d_α is a sequence of tasks (abstract or primitive) having to be performed by the agent. The plan π_α is a

sequence of primitive tasks, built from the agenda, which the agent has to perform. The links of actions order between the two streams of actions (plans) are kept in each plan, allowing for coordination.

Agent triggers: Finally, we define for each agent α a set of so-called *trigger functions* Tr_α. These trigger functions aim at representing reactions of agents to certain situations (subsets of world states). They are useful to model event-driven behavior, as in PRS [Ingrand 1996], when a specific world state *triggers* a reaction from an agent. Besides, these triggers can be used to represent social norms as defined in [Carlucci 2015], where the user can specify literals which, if true in the world state during the planning process, add some specific robot actions to the plan.

Trigger functions are defined as: $Tr \ni t : \Sigma \rightarrow (Op \cup Ab)^n \cup ()$ with $n \in \mathbb{N}^*$, returning a sequence of tasks to be inserted in an agent agenda as a reaction to specific agent states. For now, the tasks returned by a trigger function are added on top of the agenda, thus preempting any task that may have started to be decomposed. A considered solution is to support the flagging of some abstract tasks in the domain as being *atomic*. We can then prevent the tasks returned by a trigger to be inserted between any tasks resulting from the decomposition of an atomic task.

4.3 The Proposed Planning Process

The cooperative agents planning problem consists of two agents Ag_{start} with their respective agenda filled with tasks to achieve and their beliefs about the current world. For the controllable agent (*i.e.* the robot), the beliefs correspond to the planner ground truth, for the uncontrollable agent, their beliefs need to be estimated, through, for example, situation assessment component [Milliez 2014, Lemaignan 2018]. Both beliefs are then updated separately during the planning process, allowing to detect and correct belief divergences for example.

The result is a robot conditional plan Π being a tree of alternating robot and human primitive tasks. Any path from the root to the leaves is a feasible sequence of primitive tasks (*i.e.* each primitive task application leads to a state where the following one is applicable) leading to a state where the robot agenda is empty[1]

To solve such a problem we need to augment the search space from world states S only to all the agents states considered by the planner σ, with their agenda, plan and beliefs. The exploration starts with σ^{start} and consecutively applies operators associated to the robot and to the human, leading to new agents states σ^i until the controllable agent has an empty agenda: $d_{robot} = ()$.

The search, which will be detailed after, is done in two parts:

[1]We also intend to make the planner stops at a certain depth. Thus, it could be used during the execution to plan a few steps ahead but not until the task is over.

1. First, the robot and human HTNs are explored to find all the feasible plans (and in future works, also all the failing plans). This first exploration results in a tree of alternating robot and human primitive tasks but where robot ones are not selected yet (in the tree, several robot actions can be issued after a human one). Moreover, this tree has an additional dimension corresponding to the task hierarchy originating from the HTNs decompositions.

2. Then, a conditional plan is selected from this tree. The conditional plan is also a tree of alternating human and robot primitive tasks, but the robot tasks have been selected (according to cost functions detailed later in this chapter) so only one robot primitive task can follow a human one (*i.e.* the human primitive tasks now have only one child).

Again, the generated conditional plan keeps the task hierarchy of the HTNs decomposition, allowing the supervision component to more easily communicate about it, to be reasoned on, or to be stored with a semantic meaning for it to be reused later in other components. Selecting robot actions while still accounting for the possible human ones allows for simpler plan post-processing at the execution. For example, Levine and Williams propose to let the robot action choices and causality analysis to the execution [Levine 2014]. While with this approach repairing plan is easier, the execution component may not have all the information to make decisions.

4.3.1 Action Models Restriction

Some constraints on the operator, method and trigger functions must be respected. Indeed, depending on whether the agent is controllable or not, their planning process will not take decisions based on the same information, and their action will not impact the world state in the same manner. We thus impose restrictions on what a function can read and write (writing means here having effects on agents states and is only in the case of primitive task functions) in the agents state. Then, the function constraints also depend on which agent is performing the action or making the decision (in method and trigger functions). The rules for read and write accesses are given in Table 4.1.

Agent type	Readable	Writable
Controllable (robot)	$s_{robot}, s_{human},$ π_{robot}, π_{human} (a)	s_{human}, s_{robot},(b) d_{human}, d_{robot} (c)
Uncontrollable (human)	s_{self}, π_{self} (d)	$s_{self}, s_{other}, d_{self}$ (e)

Table 4.1: Readable and writable elements (belief states, agenda, plan) of the agents state by method, primitive task and trigger functions.

Table 4.1(a): During robot planning, the decision and the action can depend on the beliefs of the robot and on the planned estimated beliefs of the human.

Moreover, the current partial plan of the robot and the anticipated plan of human one can also be used to make decisions.

Table 4.1(b): The effects of robot actions obviously impact its own belief state (considered as the real world state by the planner), but also the beliefs of the human, for example, through their observation process and first order logic reasoning. More elaborate schemes to compute the effects can also be devised such as those described in [Gharbi 2015a].

Table 4.1(c): Besides, a robot action can add a new task to the agenda of the human. This is to account for communication actions requesting the human to do something.

Table 4.1(d): The human decisions and actions can only be done according to her own beliefs and partial plan. Indeed, we cannot add the robot ones as it is, or we would consider that the human estimation of the robot knowledge and past actions is always perfect. This would require a third type of agent, being the robot model as estimated by our estimation of the human. For now, we chose not to represent the estimation of the robot partial plan in the human beliefs. Indeed, it would require to reason on the observability of the robot actions to represent them in the human beliefs. However, the human beliefs can be updated through robot actions.

Table 4.1(e): The effects of the human actions obviously impact their beliefs and the robot (planner) ones. Moreover, the human agenda could also be updated through, for example, a positive answer to a task request.

4.3.2 Exploration Algorithm

Our planner operates in a turn-taking scheme, based on the update of the agents' beliefs states, the HTNs of the robot and the human are explored successively.

4.3.2.1 Controllable Agent HTN Exploration

The robot HTN exploration is a pretty standard depth-first algorithm presented in Algorithm 6. The first task λ from its agenda d_{robot} is popped, then if it is an abstract task $\lambda \in Ab$, all the applicable methods are applied, and their results are prepended to the agenda, thus giving new agents states (with the same beliefs as the previous ones but with the robot agenda updated) and branching our search space. We iterate with the new task popped from the new robot agenda. Eventually, the popped task will be a primitive one $\lambda \in Op$, its function will then be applied to the currently explored agent states. If it returns $false(\perp)$, the action is not applicable, and the exploration backtracks to another decomposition of an abstract task. However, if the action is applicable (returns a new agents state), the triggers

are run for each agent, updating their agenda if necessary. Then, we question the human HTN to get their possible next actions from this new agents state, and, for each possible new agents state, we apply the triggers of each agent then we continue the robot HTN exploration. This exploration continues until the robot agenda is empty, or all the branches return *false*.

4.3.2.2 Uncontrollable Agent HTN Exploration

The human HTN exploration differs from classical HTN planners as the goal is not to produce a complete plan, but rather to list all the actions the human is likely to perform in a given agents state. As presented in Algorithm 7, we recursively decompose the first task of the human agenda d_{human} with every applicable method, until we reach an applicable operator. All the operators from all the applicable decompositions are returned to the robot HTN exploration and applied.

4.3.2.3 Default Actions

Two special cases are handled during the exploration. If the human agenda is empty whereas the robot one is not, the exploration returns a default action *IDLE* — which does not modify agents beliefs nor agendas — for the human. This action represents the non-involvement of the human in a task. Besides, if for the human no applicable action is found a default action *WAIT* — which does not modify agents nor agendas — is returned. This action represents the impossibility of the human to act in the current situation, making them wait for the robot to proceed. This default action can also be used in a domain to represent the human decision to wait for the robot to act.

Once the robot agenda is emptied, the agents state is set as a success, the plan is added to the valid plans tree and the search can be continued until no decomposition is left for any task.

4.3.3 Conditional Plan Selection

Once this exhaustive search has been done, the result is a valid plans tree of alternating robot and human feasible actions along with their current beliefs leading to a task completion. For simplicity we will represent the function returning the children of each operator in this tree as $\textsc{NextActions} : Op \mapsto \mathcal{P}(Op)$. More precisely, as the agents of the action alternate between the robot and the human we have $\textsc{NextActions} \in \mathcal{P}(Op_{human})^{Op_{robot}} \cup \mathcal{P}(Op_{robot})^{Op_{human}}$ (a robot action can only have human actions as children and a human action can only have robot actions as children).

The goal of this second planning step is to select robot actions such as each human action in the plan has only one robot action as a child. To do so, we define a cost function $cost : \sigma \times Op \mapsto \mathbb{R}^+$ representing the cost of an action in a specific state. The data structure is now similar to a two players game tree. However, *MinMax* approaches are not suitable here, as we are not in an adversarial setup

Algorithm 6 Double HTN main exploration algorithm.

```
1: function SeekPlans(robot, human)
2:     solutions ← an empty list of plans
3:     result ← ExploreTree(robot, human, solutions)
4:     if result = failure then return failure
5:     return solutions

6: function ExploreTree(r, h, solutions)
7:     if isEmpty(d_r) then
8:         add the plan π_r ∪ π_h in solutions
9:         return success
10:    λ ← Pop(d_r)
11:    if λ ∈ Ab_r then
12:        isOneDecompositionValid ← false
13:        for each ⟨α, δ⟩ in Me_r s.t. α = λ do
14:            decomposition ← δ(s_r, π_r, d_r, s_h, π_h, d_h)
15:            if decomposition ≠ ⊥ then
16:                r', h' ← Copy(r, h)          ▷ The exploration is branching on the robot decompositions, so we deep copy the agents to avoid interactions between branches
17:                d_r' ← decomposition.d_r'
18:                result ← ExploreTree(r', h', solutions)
19:                if result = success then isOneDecompositionValid ← true
20:        if isOneDecompositionValid then return success
21:    if λ ∈ Op_r then
22:        result ← λ(s_r, π_r, d_r, s_h, π_h, d_h)
23:        if result = ⊥ then return failure
24:        r', h' ← Copy(r, h)          ▷ The exploration is branching on the human operators, so we deep copy the agents to avoid interactions between branches
25:        s_r', d_r', s_h', d_h' ← Apply(result)
26:        π_r' ← π_r'.λ
27:        d_r', d_h' ← ApplyTriggers(s_r', π_r', d_r', s_h', π_h', d_h')
28:        humanApplicableOperators ← GetHApplicableOperators(h')
29:        isOneOperatorValid ← false
30:        for each o in humanApplicableOperators do
31:            r'', h'' ← Copy(r', h')
32:            s_r'', s_h'', d_h'' ← Apply(o)
33:            d_r'', d_h'' ← ApplyTriggers(s_r'', π_r'', d_r'', s_h'', π_h'', d_h'')
34:            result ← ExploreTree(r'', h'', solutions)
35:            if result = success then isOneOperatorValid ← true
36:        if isOneOperatorValid then return success
```

but more into a collaborative one. Indeed, trying to minimize the maximal possible cost is assuming that the human will always do the actions leading to the worst

Algorithm 7 Human HTN exploration algorithm, returning the feasible human actions.

```
1: function GetHApplicableOperators(h)
2:     solution ← ExploreApplicableOps(h)
3:     if solution = () then return (WAIT)

4: function ExploreApplicableOps(h)
5:     if isEmpty(d_h) then return (IDLE)
6:     λ ← Pop(d_h)
7:     if λ ∈ Ab_h then
8:         applicableOps ← an empty set of operators
9:         for each ⟨α, δ⟩ in Me_h s.t. α = λ do
10:            decomposition ← δ(s_h, π_h)
11:            if decomposition ≠ ⊥ then
12:                h' ← Copy(h)
13:                d_h' ← decomposition.d_h'
14:                applicableOps ← applicableOps ∪ ExploreApplicableOps(h')
15:        return applicableOps
16:    if λ ∈ Op_h then
17:        if λ(s_h, π_h, d_h) ≠ ⊥ then
18:            return {λ}
19:        else
20:            return an empty set
```

plan. This defensive behavior could lead to non optimal plans. We assume that given the right indications, the human will do their best to achieve the task with minimal cost. We thus propose to explore this tree differently.

Moreover, like in HATP we allow to define *social costs* functions. These functions take a complete human and robot sequence of actions (π_r and π_h) and return a cost ($\mathbb{R}^+$) which is added to the cost of the plan previously determined. By doing so, we can penalize non acceptable sequence of robot actions (*e.g.* serving a meal just after taking out the trash) or non satisfactory human required contribution (*e.g.* a plan requiring the human to act after a long series of *IDLE* actions, meaning the human could have disengaged from the task; or requesting the human to perform small tasks multiple times instead of giving the big picture of the real task to perform).

Minimizing the Average Cost The approach we propose for plan selection is to minimize the average cost (Algorithm 8). It represents the human potentially selecting any course of actions in their stream (while still respecting the action model defined in their HTN).

The algorithm is given the root action of the task network previously generated and returns the cost of the conditional plan selected while having selected the robot

actions in the task network. The functions used in Algorithm 8 are described hereafter:

- **PathEndingWith**: This function recursively backtracks from an action to get all its parents. This results in a path in the conditional plan tree, *i.e.* a plan as a sequence of alternating human and robot actions on which a social cost can be computed.
- **PerformingAgent**: This function returns the agent performing the action given as argument (either the robot or the human).
- **ApplicationState**: Return the state σ on which the action given as argument is applied.

While this is a first step and additional work is required for conditional plan selection, it has proven to work on several examples shown later in this chapter.

This method for selecting a conditional plan is quite simple but is enough to show the principles and how the approach can be useful for planning in HRI. We propose other methods of plan selection, accounting even more on the human modeling, which have yet to be refined in Section 4.6.1.

4.4 Implementation

The previous section presented the general ideas and concepts behind this new planning paradigm. This short section shows some interesting details about the actual implementation of a prototype of the planner, along with some explanations preparing the following examples presentation. We named this prototype planner Human Aware Task Planner with Emulation of Human Decisions and Actions (HATP/EHDA). In the next chapter, we will also show how it has been integrated with other components to extend its capabilities and be used in a real robotic architecture.

4.4.1 A Python Planner

We chose to implement HATP/EHDA in Python[2]. It is originally based upon the Python Hierarchical Ordered Planner (PyHOP) from Nau but has been largely modified, and only remain the general data structures. As in PyHOP, it allows to represent world states as Python objects having dictionary as attributes. For example `s.isReachableBy["cube_23"] = ["human_3", "pr2_robot"]` specifies that the *cube_23* is reachable by both the *human_3* and the *pr2_robot*. Moreover, like in PyHOP the planning domains (HTNs for both the robot and the human) are written using plain Python functions. While using a separate domain-specific

[2]This prototype is available, as work in progress, at `https://github.com/guilhembn/HATPEHDA` along with the example presented in this chapter, and the interfaces with the other components presented in the following chapter.

Algorithm 8 Conditional plan selection algorithm. Explores a search space (a bipartite tree of alternating robot and human actions) to choose the robot actions minimizing the average of the total plan cost over all the possible human actions.

```
1: function SelectRobotActions(action, actualCost)
2:     if NextActions(action) = ⊥ then  ▷ If it is the last action, we compute the social cost of the plan and return its addition with the per-action cost.
3:         actualCost ← actualCost + Cost(ApplicationState(action), action)
4:         return actualCost + SocialCost(PathEndingWith(action))
5:     if PerformingAgent(action) = robot then  ▷ If this action is done by the robot, its direct children are performed by the human
6:         totalCost ← 0
7:         actionCost ← Cost(ApplicationState(action), action)
8:         for each child in NextActions(action) do
9:             totalCost ← SelectRobotActions(child,
10:                                            actualCost + actionCost)
11:        return totalCost/Card(NextActions(action))
12:    else  ▷ This action is done by the human, its direct children are done by the robot and need to be selected
13:        minCost ← +∞
14:        chosenChild ← null
15:        actionCost ← Cost(ApplicationState(action), action)
16:        for each child in NextActions(action) do
17:            childCost ← SelectRobotActions(child,
18:                                           actualCost + actionCost)
19:            if childCost < minCost then
20:                minCost ← childCost
21:                chosenChild ← child
22:        Set the result of NextActions(action) to chosenChild
23:        return minCost
```

language (DSL) for writing planning domains enables some optimizations and advanced search algorithms, it tremendously reduces the expressiveness and makes representing real worlds scenarios more complex. Besides, Python being an interpreted language, iterating over these domains is quicker as, unlike HATP, they do not require any compiling before being planned upon. The decomposition functions take the world states (beliefs), partial plans of the agents (depending on the type of the agent Table 4.1) and any other optional parameters (*e.g.* goals, entities) as arguments, and return a list of tasks with optional parameters to be put in the agent's agenda. Likewise, the actions receive as arguments the world states (beliefs), the partial plans and the agendas of the agents (also depending on the type of the agent Table 4.1) and return new world states and agendas.

The search algorithms have also been implemented in Python. A lot of optimization can be done, and it is planned to entirely rewrite the software in C++. The current prototype, while not being efficient compared to other approaches, still

allows to find plans in a reasonable time for realistic human robot scenarios.

4.4.2 Drawing the Plans

Thinking about the robot programmer is almost as important as considering the human partner the robot will interact with. Thus, in order to debug the designed HTNs, we provide a way of visualizing the valid plans tree and the generated conditional plan.

An example of visualization of valid plans tree (before selecting a conditional plan) is presented in Figure 4.3. The ellipses correspond to primitive tasks while rectangles represent abstract ones. The octagons represent the default actions (*IDLE* and *WAIT*). The blue shapes are robot tasks and the orange ones are the human ones. Finally, red arrows indicate the sequence of actions in the plans and gray ones represent hierarchical links, *i.e.* the links between the tasks as defined in the HTNs, along with the decomposition numbers next to them.

It is interesting to note that in Figure 4.3 some human actions are followed by two potential robot actions (orange shapes of which originate red arrows leading to several blues shapes). Indeed, these actions are feasible by the robot, but no plan has been selected yet. Whereas in Figure 4.4, representing a conditional plan selected from the previous figure, the robot actions are set, and the non determinism is only on the human actions.

4.5 Examples

In this section, we will present some case studies in which are presented small task examples. Each example will present some features of the planner and comparisons between multiple plans depending on some initial conditions (beliefs or action costs). Besides, they will give insights into the rationale used when creating the domains for both the robot and the human. The Python domains for both examples are available in Annex B.

The two following cases are set in the same context presented in Figure 4.1. We envision a super-scenario in a company office where a robot assistant is verbally requested by a human worker to bring her a coffee. The robot must take her mug, go to the coffee machine, manage to fill the mug with coffee and bring back the filled mug to the worker. We instantiate this scenario into two precise subtasks highlighting multiple features of HATP/EHDA.

4.5.1 Plan for Robot Unknown Human Knowledge

First of all, after the robot has been commanded by the worker to bring her a coffee, it must pick her mug. We want to illustrate how we can represent human knowledge that is unknown to the robot (but with a small number of possibilities), and how the planner can elaborate different plans depending on action costs and the number of possibilities. To do so, we place the robot in a scenario where the worker and it

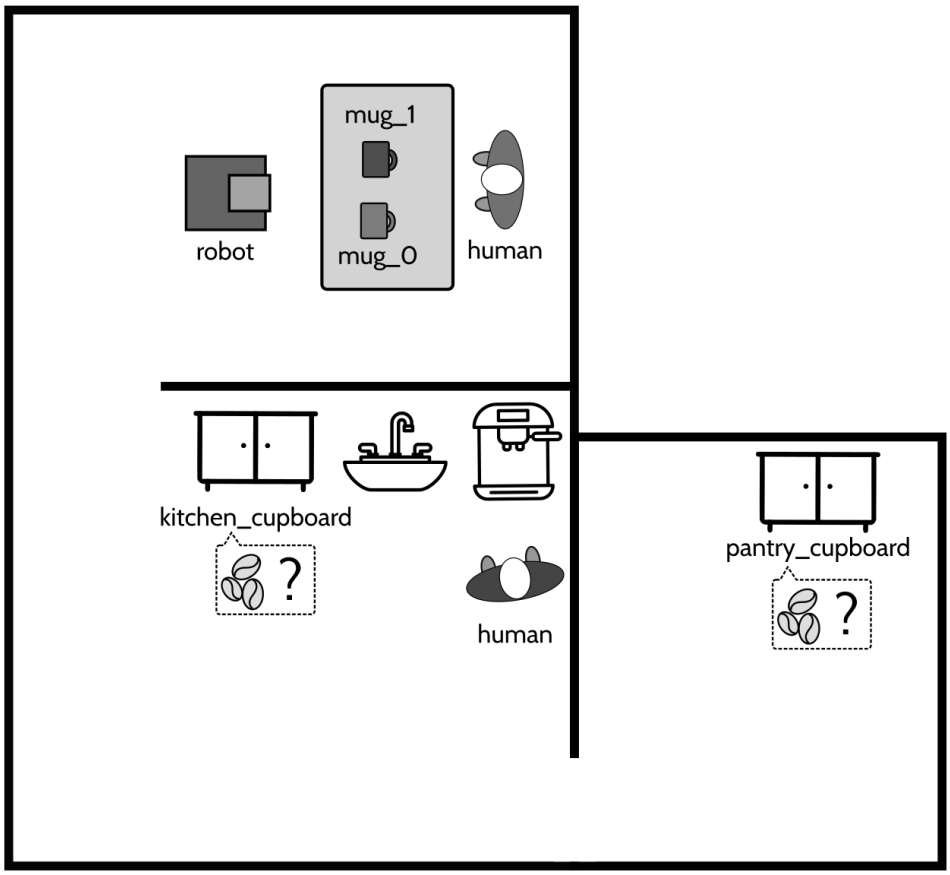

Figure 4.1: An example scenario where the robot is requested by a worker to bring her a coffee. It must first take her mug, then go to the coffee machine, fill it with water and coffee grounds and finally fill the mug with coffee.

are face to face when she asks it for a coffee. There is a table between them and $n \in \mathbb{N}$ mugs are placed on it. Only one mug belongs to the worker, the robot knows she knows which one it is ($\mathcal{M}_r^H$) but the robot does not know it ($\mathcal{M}^R$). The goal of the robot is thus to take the right mug, and to go to the coffee machine. The general idea is that we want the robot to have two ways of grabbing the right mug. Either it can take a random one, and if the human is not protesting, the robot can proceed with the rest of the task, else it tries again while knowing this mug was not the right one; or the robot can ask the human for the right mug and take it. However, as specifying the right mug through verbal communication can be costly for the human (*e.g.* highly resembling mugs, a noisy environment), asking her might not always be feasible not be the optimal solution.

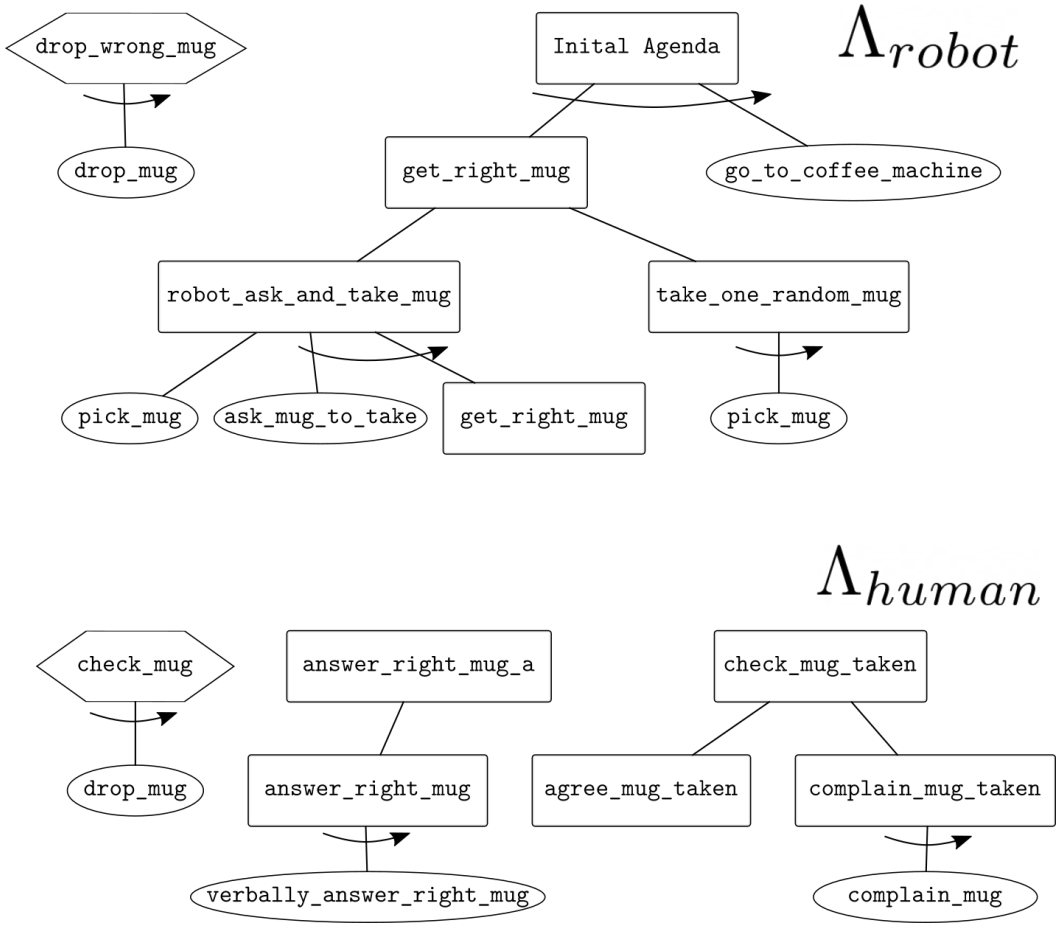

Figure 4.2: The action models (HTNs) of the robot and the human for the mug selection task. Rectangles represent abstract tasks, ellipses represent primitive tasks and hexagons represent triggers. The links with an arrow are "and" links, the others are "or" links. Please note that some decompositions have been merged for clarity.

First, we go through the design of the robot HTN (Figure 4.2). The com-

plete domain in Python code is presented as supplementary material in Annex B.1. The robot agenda is initialized with two tasks `get_right_mug` and `go_to_coffee_machine`. The robot abstract tasks and their decompositions are presented here:

- `get_right_mug` aims at making the robot pick the mug belonging to the human. It has two decompositions:
  - `robot_ask_and_take_mug` representing the robot asking the human which one is her mug. It returns either `pick_mug` if the human's mug is known or `ask_mug_to_take` and `get_right_mug`
  - `take_one_random_mug` representing the robot go through trial and errors. It returns either `pick_mug` if the human's mug is known or `pick_mug` with all the mugs potentially belonging to the human.

The primitive tasks (operators) of the robot and their effects are presented here:

- `pick_mug` updating the beliefs of all the agents in the room by removing the mug given as parameter from the table and adding it to the robot gripper.
- `ask_mug_to_take` adding the task `answer_right_mug_a` to the human agenda.
- `drop_mug` updating the beliefs of all the agents in the room by removing the mug they believe the robot is currently holding and adding it on top of the table.
- `go_to_coffee_machine` updating the beliefs of all agents in the room that the robot has left the room and is now in the coffee room. Moreover it updates all agents in the coffee room beliefs such as the robot is in the coffee room and updates their beliefs of what it is carrying.

Moreover, if the human is complaining about the mug the robot is currently holding, we want the robot to drop it, and to know that this mug is not the right one. To do so, we use the triggers mechanisms and we define a trigger function for the robot:

- `drop_wrong_mug` checking if the human just complained about the mug we taken (`complain_mug` $\in \pi_h$) and if so, adds `drop_mug` and `get_right_mug` to the robot agenda.

Then, we go through the human task model. We assume the order of getting a coffee has already been given to the robot, and thus assume the human has no task to decompose initially in her agenda. When asked for which mug belongs to her, we model that she can answer whatever mug she has not ruled out. Moreover, we model that when the robot picks a mug, she will complain if it is not the right one. Here are the two abstract tasks and their decompositions we used to model this human behavior:

- `answer_right_mug_a` representing the task of answering the robot for the right mug. For this example, it has only one decomposition:
  - `answer_right_mug` if the right mug is present in the human beliefs, it returns only the `verbally_answer_right_mug` primitive task, else, it returns $m \in \mathbb{N}$ alternatives of `verbally_answer_right_mug` with m being the number of mug not having been ruled out in the plan at this state.
- `check_mug_taken` models the human expectation of the robot taking the right mug. It has two decompositions:
  - `agree_mug_taken` when the robot has the right mug in its gripper. It always returns an empty task list.
  - `complain_mug_taken` when the robot has a wrong mug in its gripper. It return the primitive task `complain_mug`.

The following presents the modeled human primitive tasks:

- `verbally_answer_right_mug` updating the beliefs of all the agents in the room with the human being the owner of the mug passed as a parameter. To estimate the feasibility and the cost of this action we run our REG component as detailed in the previous chapter.
- `complain_mug` updating the beliefs of all the agents in the room with the human not being the owner of the mug passed as parameter.

Finally, to model the human reaction when the robot grabs the wrong mug, we use the triggers system for the human:

- `check_mug` adding the task `check_mug_taken` to the human agenda each time the robot takes a mug that has not been specially designated by the human (*i.e.* the robot has a mug in its gripper and `verbally_answer_right_mug` $\notin \pi_h$).

Another way of representing the human reaction is to represent the shared goal of the human and the robot (as the human just asked the robot to perform a task) by adding a monitoring task to the human agenda in the initial state. This monitoring task could then be refined into a *WAIT* or into a `complain_mug` or an empty task list depending on the mug taken by the robot. Such a representation allows to model the joint activity between the human and the robot. However, it also models that the human can do nothing else except monitor the robot actions until it has picked the right mug. Using the triggers as presented here, allows to represent that the human is maybe performing another task (modeled by the $IDLE$) action and is not committed to interact with the robot. The triggers still present a drawback in this case, as it would be triggered whenever the robot is picking a mug (provided that the human is estimated to see the action), even if it is not for serving the human. It is not an issue in the presented example as the task is quite short, but

we still have to find ways to provide more context to the triggers. An easy solution would be to add a fact to the human beliefs enabling or not the trigger. We are currently investigating how to represent complex shared goals, and think they can be an answer to this issue.

The search space for $n = 2$ mugs is presented in Figure 4.3. In this example, both mugs are distinct and RE can be computed. On the right hand side of the figure is the decomposition where the robot explicitly asks the human to designate her mug. The answer can either be `mug_0` or `mug_1`. The robot then picks the right one and goes to the coffee machine, leaving no task to decompose. On the left hand side of the figure is the decomposition where the robot proceeds via trials and errors. The robot can either pick `mug_0` or `mug_1` and the human will either react by doing nothing (if the robot took the right mug) or by complaining, in which case the robot will drop the mug, take the other one, and leave. Interestingly, we model the human reaction such as not expecting her to complain when taking the second mug after a first failure (in Figure 4.3, the only modeled returned human action after the `pick_mug` actions denoted (a) and (b) is `IDLE` and there are no `complain_mug_taken` action as the other mug as been excluded before in this branch, so we assume that the last one is the right one).

Next, we will compare different action costs and conditional plan selection criteria based on the same search space. To select a plan we used the Algorithm 8. First, we set the cost of `complain_mug` action much lower than the `verbally_answer_right_mug` action. Here, the costs might have been set by the supervision component, estimating we are interacting in a noisy environment, where verbal communications are difficult to make, or that the human does not bother to correct the robot. The conditional plan returned is presented in Figure 4.4. The chosen plan is the one containing trials and errors. Indeed, as it can lead to much shorter and thus less costly plans, it is the minimum average.

However, imagine that the human (who has still not had her coffee) is in a hurry, or that the mugs are really easy to distinguish from one another (*e.g.* different color) and thus, we decrease the cost of the `verbally_answer_right_mug` action and increase the cost of the `complain_mug` action. The conditional plan selected is presented in Figure 4.5. The robot now prefers to ask for the right mug rather than trying to pick one at random.

As we increase the number of mugs n, the cost of `verbally_answer_right_mug` has to also increase to make the robot choose the trials and errors decomposition, as the average of this decomposition increases since the number of potential errors increases.

In this example, we show one really interesting feature of HATP/EHDA: representing human knowledge that is not known by the robot. While not being tractable when there are a lot of possibilities, which is expected when exploring all the possibilities in non-deterministic planners, it allows to select more or less conservative conditional plans depending on the cost of each action. Computational times are given in Table 4.2. We see that this task modeling begins to fail at 4 mugs, as a large computation time will spoil the interaction. Although, we think it is still

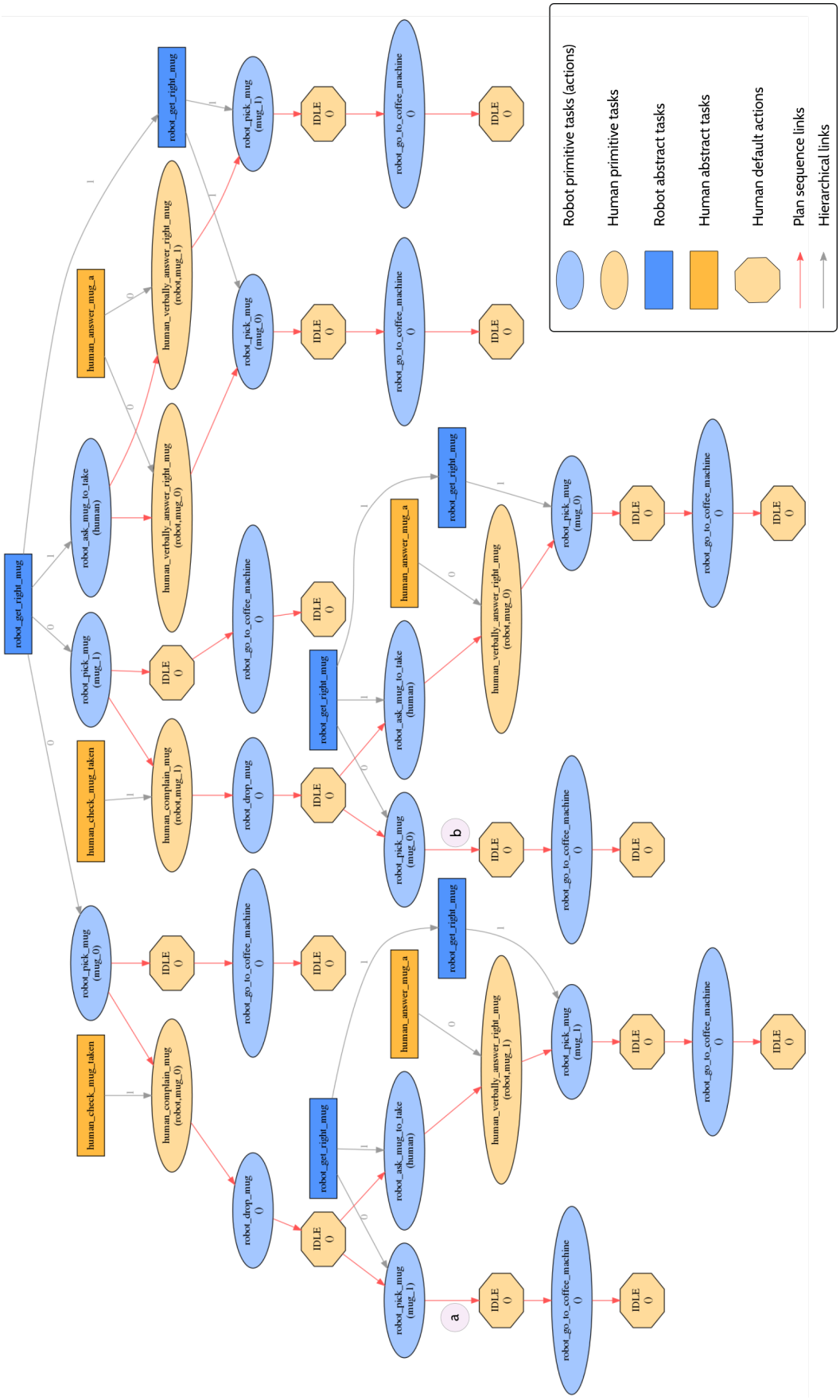

Figure 4.3: The valid plans tree produced after the exploration of the HTNs.

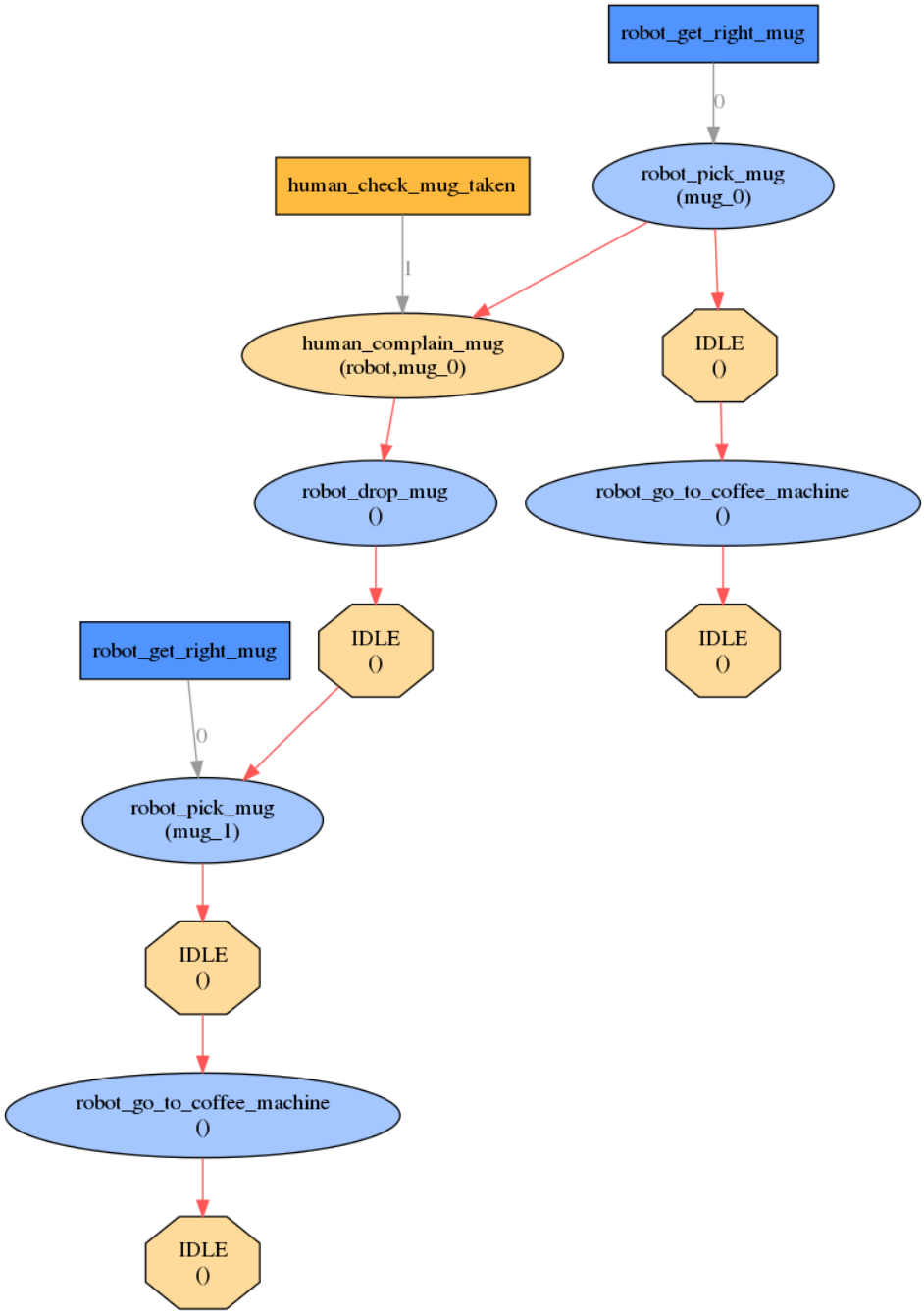

Figure 4.4: A conditional plan selected from the tree depicted in Figure 4.3. In this plan, the robot starts by picking the `mug_0` and expects the human to either complain that it is not her mug or do nothing allowing the robot to leave. If it is not the right mug, the robot would take the other one before leaving the room.

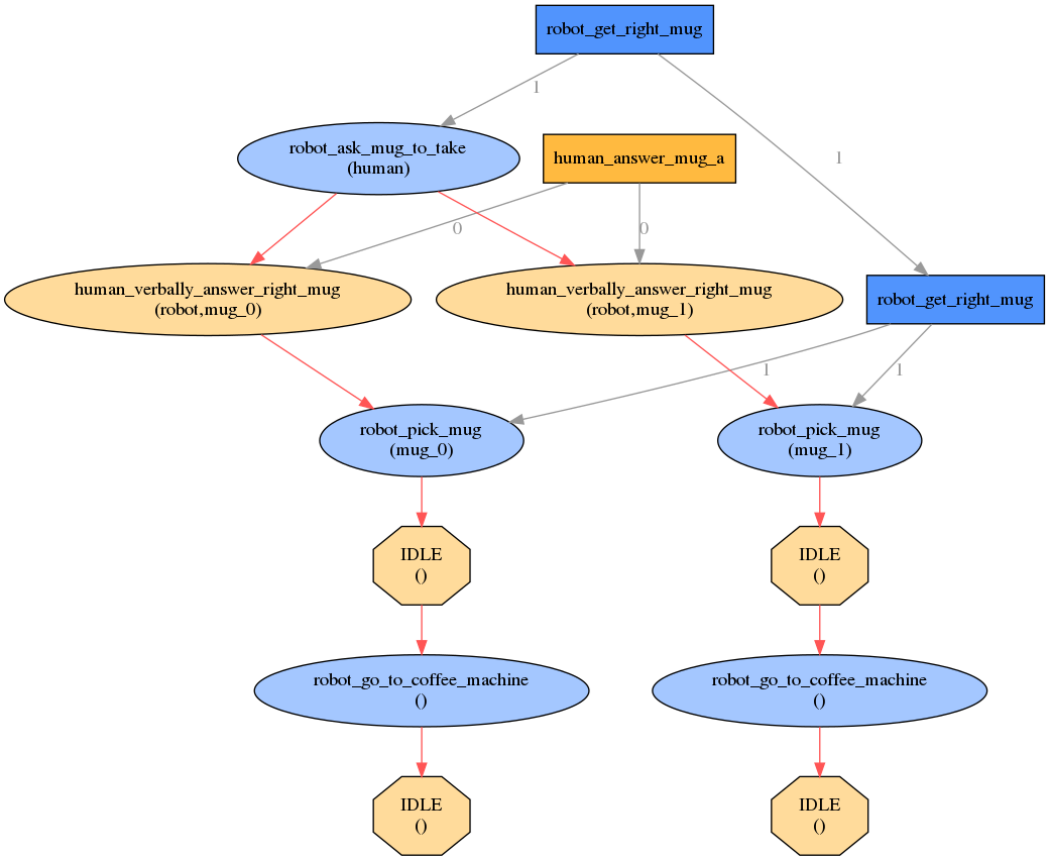

Figure 4.5: Another conditional plan selected from the tree depicted in Figure 4.3. In this plan, the robot starts by asking which mug belongs to the human. Depending on the answer, the robot would then pick one or the other and leave the room.

Number of mugs	Number of explored branches	Duration of the exploration step (s)	Duration of the plan selection step (s)
1	2	0.010	<0.001
2	8	0.115	0.003
3	30	1.162	0.006
4	128	26.425	0.035
5	650	987.057	0.149

Table 4.2: Planning computation durations depending on the specified number of mugs on the table.

pertinent in HRI as we seldom deal with a large number of objects and long plans. Besides, thanks to the task decomposition being Python functions, we can make a decomposition fail if it would lead to a combinatorial explosion. Here we could have made the "trial and error" decomposition fails if the number of mugs was greater than 4. Moreover, this example allowed to see how updating the human agenda and how triggers can be used to model the agents interaction in the HTNs planning. However, the human model was pretty simple, and we propose to challenge HATP/EHDA in the next example with a task where the human is more involved. The complete domain in Python code is presented as supplementary material in Annex B.2.

4.5.2 Balance Difficult Communications, Decomposition Cost and Task Attribution

The robot is now heading to the coffee machine with the right mug in its gripper. On its way it detects another human taking a break near the coffee machine. The coffee has to be made. To brew coffee, ground coffee and water must be put in the coffee machine, and then the coffee can be served. While water is considered as always available, ground coffee is not. There are two places where ground coffee can be retrieved: either in the kitchen cupboard (close to the coffee machine) or in the pantry cupboard.

4.5.2.1 Handling the Robot Only Case

First, we want the robot to be able to make coffee by itself, without requiring human help. To do so, we implement the following abstract tasks tree depicted in Figure 4.6 in the robot model (here we prepend the task names with `r` where task are different in the robot and the human model):

- `r_make_coffee` only having one decomposition (for now), represented by Figure 4.6(a):
  - `r_make_coffee_alone` returning, in both orders (to represent partially ordered task tree), the tasks `get_water`, `pour_water_in_machine` and `r_get_coffee`, `put_coffee_in_machine`. Only `r_get_coffee` is an abstract task.

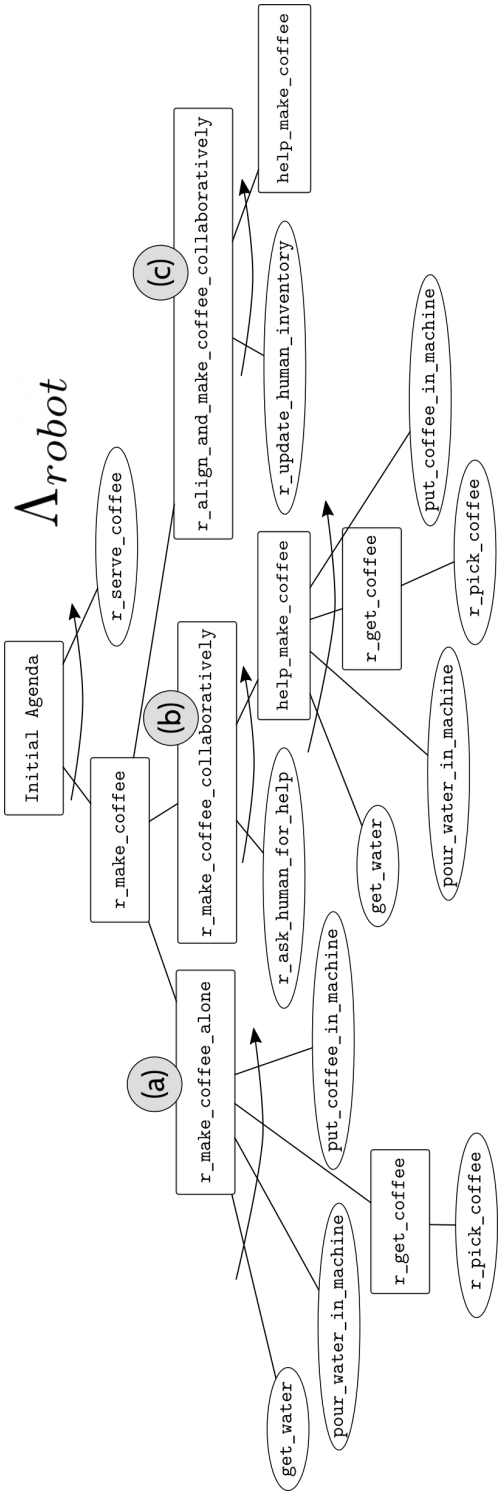

Figure 4.6: The action models (HTN) of the robot for the coffee preparation task. Rectangles represent abstract tasks and ellipses represent primitive tasks. The links with an arrow are "and" links, the others are "or" links. Please note that some decompositions have been merged for clarity.

- `r_get_coffee` representing the ways for the robot to obtain coffee. It has only one decomposition:
  - the decomposition returns () if the robot has already coffee in its gripper. Else, it selects the closest cupboard and returns `r_pick_coffee` with it as parameter.

The robot primitive tasks as are follow:

- `get_water` returning $\perp$ if the robot is already holding something; updating the beliefs of all the agents in the room with the fact that the robot holds water otherwise.
- `pour_water_in_machine` updating all the agents in the room beliefs with the machine being filled with water.
- `r_pick_coffee` returning $\perp$ if the robot is already holding something or if the cupboard passed as parameter does not contains coffee (in the robot beliefs); updating the beliefs of all agents in the room with the fact the the robot holds coffee otherwise.
- `put_coffee_in_machine` updating all the agents in the room beliefs with the machine being filled with coffee.
- `r_serve_coffee` updating all the agents in the room with the mug being filled with coffee.

Now, for the initial conditions we set that the robot knows there is coffee in the kitchen cupboard (the closest) and we add two tasks in its agenda: `r_make_coffee` and `r_serve_coffee`. The human has nothing in its agenda. The two possible plans for this really simple case are presented in Figure 4.7(a) and (b). The plan selection would then choose one of the plans based on robot action costs.

This simple example shows that our approach can still be used as a classical HTN planner and that it can plan for partially ordered task networks. By doing so, we show the planner does not require to always involve the human and can balance joint plans with plans where it does all the task when applicable.

4.5.2.2 Incorporating the Human Planning Process

As we also want the robot to be able to ask the idle human to help it, we add, to its action model, a decomposition to the abstract task `r_make_coffee` and a new abstract task `help_make_coffee`:

- `r_make_coffee` containing the previous decomposition and the new one, represented in Figure 4.6(b):
  - `r_make_coffee_collaboratively` returning a sequence containing the primitive task `r_ask_human_for_help` and the abstract task `help_make_coffee`

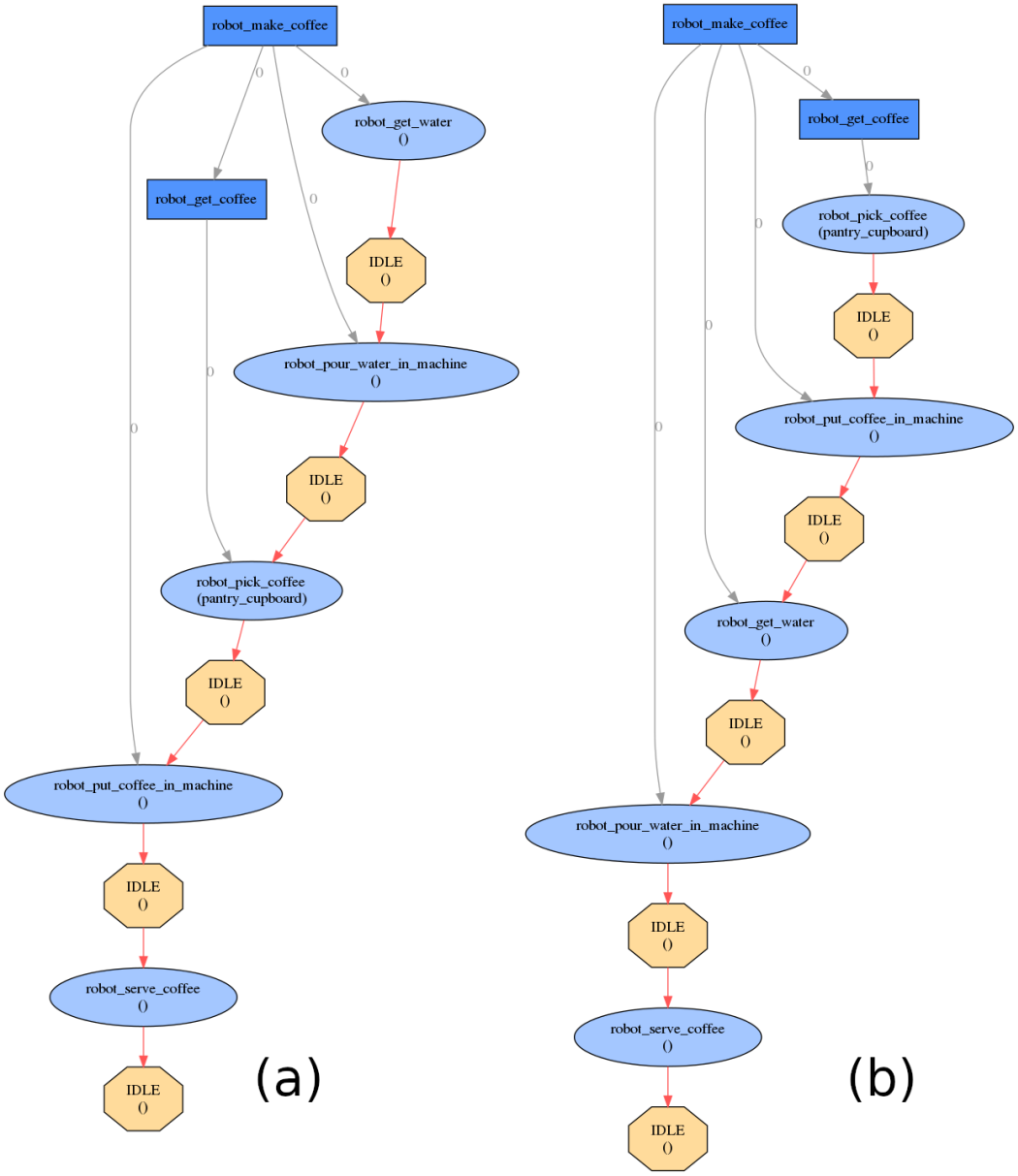

Figure 4.7: The conditional plans generated by our approach. In both (a) and (b) the robot chooses not to involve the human and do the coffee on its own, only the order of action changes.

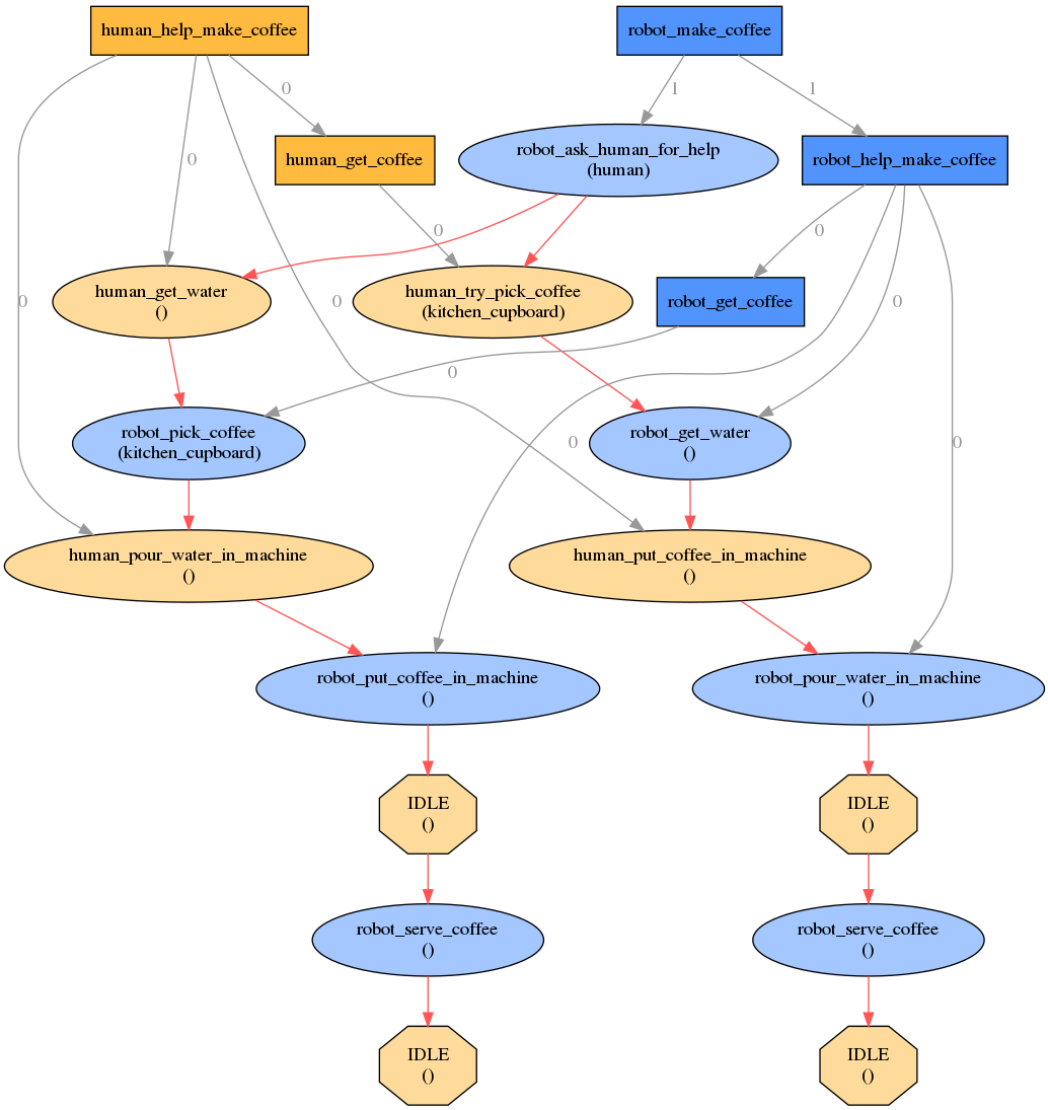

Figure 4.8: The conditional plan involving the human selected by HATP/EHDA. The robot chooses to ask for human help. We plan that the human will either get the coffee or fill the water and adapt accordingly, the choice of human actions is not made, but thanks to the conditional plan, both possible solutions are planned and it is up to the supervision component to follow the right one depending on the human action detected during execution.

- `help_make_coffee` representing the ways for the robot to help another agent to make coffee. It has only one decomposition:
  - It returns `get_water` and `pour_water_in_machine` if there is no water in the machine (in s_{robot}) and the human is doing a task related to bringing coffee (in d_{human}). Likewise, it returns `r_get_coffee` and `put_coffee_in_machine` if there is no coffee in the machine (in s_{robot}) and the human is doing a task related to fill the machine with water (in d_{human}). Then, if the human is not doing any task (in d_{human}), we add to the exploration `r_get_coffee`, `put_coffee_in_machine` and `help_make_coffee` if there is no coffee in the machine (in s_{robot}) and `get_water`, `pour_water_in_machine` and `help_make_coffee` if there is no water in the coffee machine (in s_{robot}). The idea here is to complete the human actions if they take the initiative of a task, but to be proactive by exploring both possible alternatives if they are not. The recursion allows to reevaluate the need for this task later in the planning process.

The primitive action added to the robot model is:

- `r_ask_human_for_help` adding the task `help_make_coffee` to the human agenda. Here we could have represented the possible refusal of the human by adding an abstract task leading to two possible decompositions for the human, accepting or declining, leading in similar schemes as in 4.5.2.1. However, to keep this example as simple as possible, we assume the human will always help the robot if asked to do so.

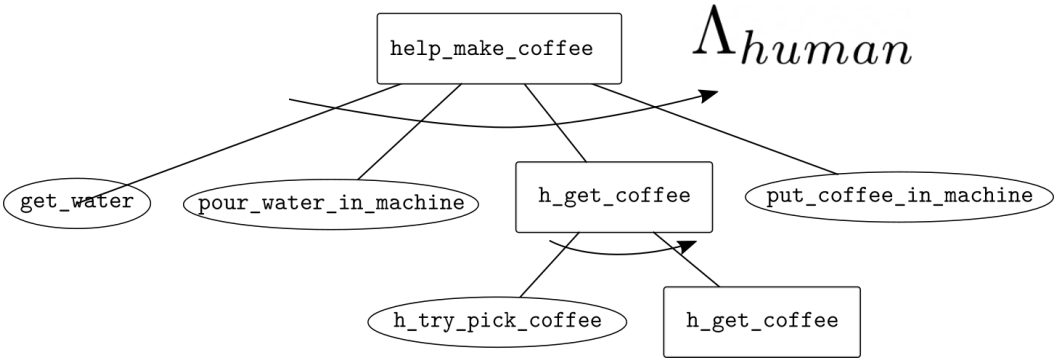

Figure 4.9: The action models (HTN) of the human for the coffee preparation task. Rectangles represent abstract tasks and ellipses represent primitive tasks. The links with an arrow are "and" links, the others are "or" links. Please note that some decompositions have been merged for clarity.

We model the human actions similarly to the robot ones. Their abstract tasks are represented in Figure 4.9 and defined as:

- `help_make_coffee` representing the ways for the human to help another agent to make coffee. It has only one decomposition, which is the same as the robot one.
- `h_get_coffee` representing the ways for the human to obtain coffee. It has only one decomposition:
  - the decomposition returns () if the human is already holding coffee (in s_{human}). Else, it selects the closest cupboard and returns `h_try_pick_coffee` with it as parameter and `h_get_coffee`. It differs from the robot one, indeed, whereas the knowledge of the robot is assumed to be the world state, the human's one can be false. Thus, the human might try to perform `h_try_pick_coffee` on a cupboard not containing coffee. We take this into account with the recursion of this abstract task, and with the primitive task `h_try_pick_coffee` described hereafter.

The model of the human primitive tasks are:

- `get_water` as defined for the robot
- `pour_water_in_machine` as defined for the robot
- `h_try_pick_coffee` differs from the one defined for the robot as it checks if the cupboard passed as parameter really contains coffee (*i.e.* in the robot beliefs s_{robot}). If it does not, the human's beliefs (s_{human}) about this cupboard are updated to match the robot ones (s_{robot}, modeling the human going in front of the cupboard, opening it and seeing the absence of coffee). If the cupboard does contain coffee in the robot beliefs, all the agents in the room beliefs are updated with the human having coffee in their hand.
- `put_coffee_in_machine` as defined for the robot.

The initial conditions are the same as presented before, but we also add that the kitchen cupboard contains coffee in the human beliefs. In addition to the two plans where the robot does not seek human help (Figure 4.7(a) and (b)), another valid plan is found. This plan is presented in Figure 4.8. This conditional plan has two alternatives, depending on the initiative taken by the human.

Moreover, we see how recursive abstract tasks are used to reevaluate the agents state later in the plan to adapt for the other agent planned actions. However, recursions in abstract tasks usually break the semantic of the returned plan hierarchy and can prevent (or alter) their communication or storage. We envision in the future to allow to flag some abstract tasks as iterative to decouple their exploration from the returned plan hierarchy.

The planner is then able to balance between asking the human to take part in the task (Figure 4.8) or doing it all on its own (Figure 4.7(a) and (b)). Creating a shared goal or engaging in a joint action now depends on the costs set and is

decided by the planner. Moreover, we chose to not impose a specific task to the human (either fetching the coffee or filling the water) but to give a high level shared goal of making coffee. We thus rely on the human planning capabilities to perform the actions. Besides, the human has the initiative of the part of the shared goal he wants to perform, but thanks to conditional plans, both choices are covered and planned for. It allows to make more accurate plans and to choose between them in a more informed way as they account for multiple alternatives. Once a plan is selected, it is up to the supervision component to follow the right branch depending on the detected human action during the execution (*i.e.* for the plan represented in Figure 4.8 if the human is getting water from the tap, or if he is picking coffee from the cupboard).

4.5.2.3 Updating Human Beliefs

We can also change the initial conditions to elicit new behaviors. We keep the same action models for both the robot and the human, but we change the estimation of the human beliefs given as initial conditions to the planner. In a real robotic architecture, the human knowledge base would be updated with an estimation provided by situation assessment components. We specify that the human believes that both the kitchen and the pantry cupboard contain coffee (s_{human}). However, the robot knows (*e.g.* using specific sensors or having been told about) that there is coffee only in the pantry cupboard (s_{robot}). With these conditions, the search space extends from the three plans presented in Figure 4.7(a), (b) being when the robot prepares the coffee by itself, to include the one presented in Figure 4.10. In this last plan, we indeed model that the human will tend to first go to the nearest cupboard he thinks contains coffee. If this cupboard does not contain coffee, he will go to the next one. We can also note that only the left branch of the plan in Figure 4.10 is impacted by this beliefs divergence. However, this branch choice is not up to the robot as we modeled the human as having the initiative of selecting a task. This subtlety cannot be represented in HATP. Indeed, in HATP, only one task would have been assigned to the human corresponding to the optimal plan being the one where the human fills the water. However, without the communication of the plan, the human would have had no way of knowing which task is preferred.

Depending on the cost of the human being disappointed and of the actions, the plan selected can either be that the robot does all the task (Figure 4.7(a) and (b)), as the human first searching in the wrong cupboard can increase too much the average cost; or the new plan (Figure 4.10) asking the human for help but taking the risk that he may do the coffee part and, given his beliefs, go to the wrong cupboard.

To improve our robot, we want to make it able to realign the beliefs of the human, so, whatever the task he chooses, he will not go to the wrong cupboard.

To do so, we add a third decomposition to the `r_make_coffee` abstract task and one new primitive task to the robot models.

- `r_make_coffee` containing the previous two decompositions and the new one,

Figure 4.10: A conditional plan returned by the planner to make coffee with human help in case of (known) beliefs divergence. The robot chooses to keep this divergence, potentially leading to the human erroneously opening the wrong cupboard to fetch coffee before trying the right one.

represented in Figure 4.6(c):

  - `r_align_and_make_coffee_collaboratively` returning $\perp$ if no beliefs divergence is detected between the robot and the human. Otherwise, the decomposition returns the new primitive task `r_update_human_inventory` along with `r_ask_human_for_help` and `r_help_make_coffee`. An alternative for `r_update_human_inventory` is returned with as parameter each cupboard in diverging beliefs between the robot and the human.

- `r_update_human_inventory` being a primitive task. It updates the human beliefs (s_{human}) concerning the cupboard passed as parameter with the beliefs of the robot (s_{robot}).

In this example, we see how representing separately the agents beliefs allows to plan for communication actions helping human decision and hopefully avoiding their disappointment (increasing their satisfaction). Our planning scheme allows to plan for communication actions aligning the beliefs, thanks to the rich operator model (as Python functions) presented before. Not discovering the need of aligning the beliefs during the execution allows to include the cost (or even the feasibility) of such communication during the planning process and thus to balance with other plans not needed them.

With this new decomposition, the new plan presented in Figure 4.11 is added to the search space. In this plan, the human beliefs are updated before asking him to help the robot to make coffee. The human does not make the first failure of going first to the kitchen cupboard to take the coffee.

Depending on the communication cost (estimated using the REG approach presented in the previous chapter to designate the cupboards), the human disappointment cost (which can be added as a social cost) and the other actions cost, any one of the four possible plans can be selected. For example, to minimize the human involvement and if the communication has a high cost, the selected plan would be Figure 4.7(a) or (b). If the communication is costly but the pantry and kitchen cupboard are not too far away, the selected plan is Figure 4.10, finally, if we represent that the human would be upset if he makes an erroneous action or if the communication for aligning beliefs is not expensive, the plan Figure 4.11 would be returned. Indeed, the planner may choose to leave the human with their false beliefs as it finds that it does not prevent to successfully perform the task.

Through all these examples we show that this task planning approach, which separates human and robot beliefs and action models, can be suitable for multiple problems. We are able to plan for robot unknown human beliefs, to rely on the human planning process while keeping inherent uncertainties (*i.e.* not making choices for them, without communicating them) and also to plan diverging beliefs and balance the actions of realigning them with plans containing mistakes. In the next chapter, we present a HRI task, inspired from psychology, that has never

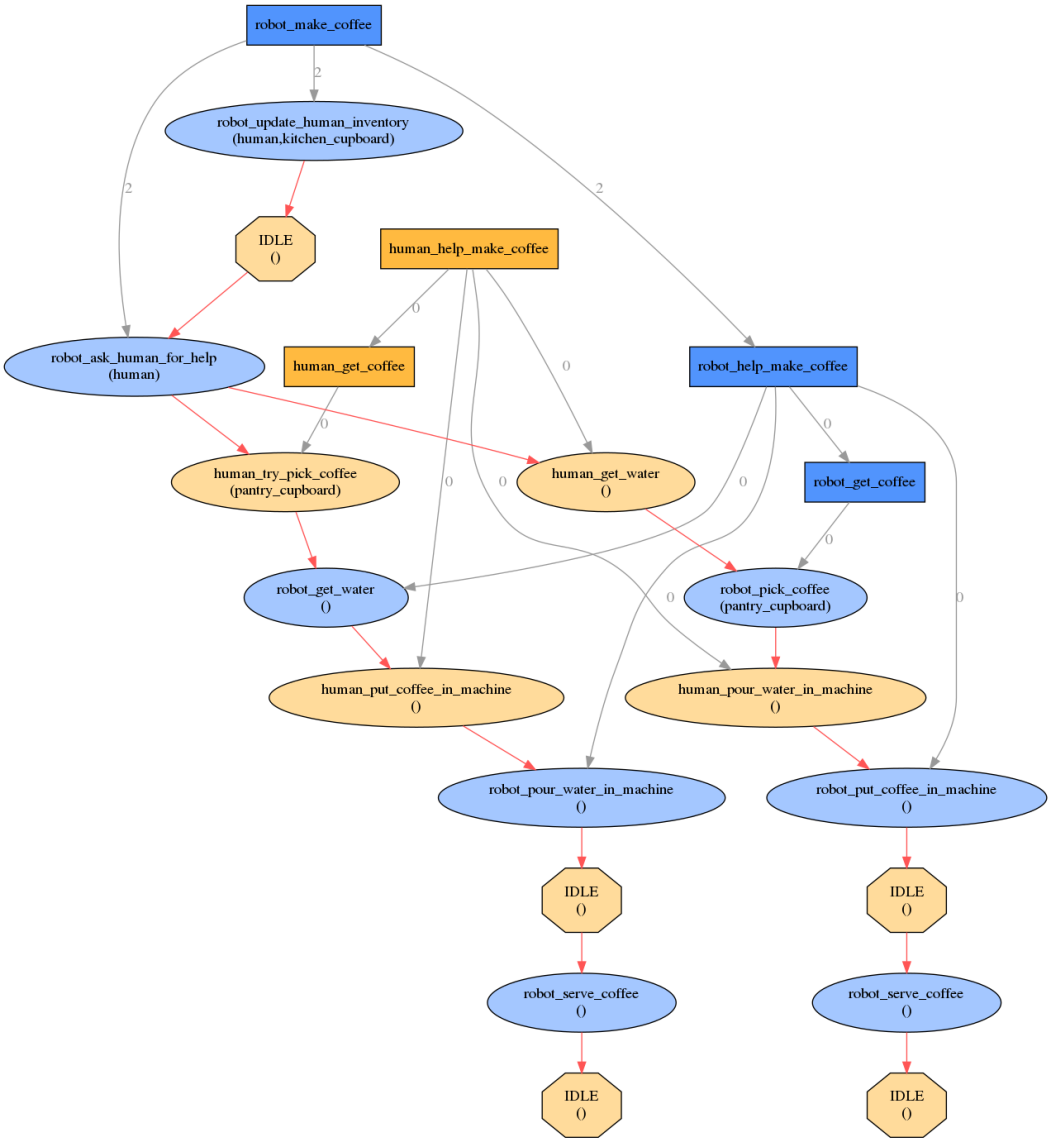

Figure 4.11: A conditional plan returned by the planner to make coffee with human help in case of (known) beliefs divergence. Here, the robot chooses to communicate to realign the beliefs with the human, then asking him for help, preventing any human misleading.

been tackled in robotics and we show how HATP/EHDA is integrated into a fully functional robotic architecture dedicated for this task.

4.6 Conclusion and Future Work

In this chapter, we have proposed a new task planning approach for human robot interaction. This approach not only explicitly represents and plan upon both the human and the robot beliefs but also use two separate action models as HTNs for the human and the robot.

We proposed a formalism along with an algorithm allowing to plan for robot actions, while considering the possible human actions according to their task model. By doing so, we are able to represent and to account for the human planning and reaction processes.

Then, we presented a successful implementation of this approach in Python called Human Aware Task Planner with Emulation of Human Decisions and Actions (HATP/EHDA) and showed the first results through two examples. These examples highlighted both the features of HATP/EHDA and the rationale behind the action models crafting. The planner is able to represent and to plan for robot unknown human knowledge, human reactions to robot actions, multiple human possible plans and intricate human robot tasks. It is also capable of balancing between communicating, letting the human perform a mistake and attributing different roles to the robot or the human.

We look forward to continuing exploring this approach, find its benefits and its limits. More importantly, we aspire at improving it, especially on the following topics.

4.6.1 Selecting Conditional Plans Using the Human Model

We presented and implemented a method for selecting a conditional plan among the multiple possible ones, by selecting robot actions that minimize the average cost of the plan. This method intends to model the human as being cooperative but maybe not having all the information to be able to project themselves throughout the whole task, and maybe not performing the action leading to the optimal plan. However, this method is quite simple and does not reflect the full complexity of human decision process. We propose two other methods for conditional plan selection, which use even more the human model, that have yet to be implemented and tested.

4.6.1.1 The Human Planning Process Modeled as a Limited Depth Planner

First, we propose to consider the human as being able to plan a few steps ahead. Thus, instead of considering the average cost of all the possible human actions, we could set less importance to the ones failing a few steps ahead (as the human is

likely to have foreseen the failure, and will probably not perform this action) and more importance to the ones leading to a smaller cost in the near future (as the human is likely to perform a greedy approach on the short horizon we model they can plan to).

To do so would require to also store the failing operators in the valid plans tree during the first exploration step. Moreover, it would require a new parameter being the depth at which the human is modeled to foresee in the plan.

Then, when exploring the valid plans tree, if a failing operator is encountered at a depth smaller than the depth parameter, the cost of the corresponding plan would not be added to the average as we assume the human would not make an action leading to a failure that they can foresee.

Moreover, the average can be weighted with smaller number for plans being evaluated as more costly than others at the depth specified by the parameter. It models that the human will not perform actions that they estimate to lead in costly plans.

This approach still needs refinement but is seen as a great improvement over considering only the cost average.

4.6.1.2 Guiding Human Choices Towards the Least Costly Solution

Then, we also want to add another step between the HTNs exploration and the conditional plan selection being the addition of communication actions related to the plan. The general idea is to analyze the valid plans tree resulting from the HTNs exploration and to insert robot communication action before some potential human actions that are leading to a plan failure or a costly plan. These communications would aim at encouraging some human decisions toward less costly plans, or to inform the human about the difference in the plan their actions will induce. To avoid unnecessary communications, the insertion can be conditional on the human prediction depth defined before. If the human can foresee the failure the communication is not needed.

Such communication can hardly be inserted in the HTNs or during the HTNs exploration step, as it requires having the full plan to decide whether they are needed or not. However, analyzing and inserting them during planning rather than after a plan has been selected or at the execution is interesting as it would allow to balance between keeping some uncertainty for non crucial human decisions or accepting the additional cost of these communication actions but reducing the uncertainty in human actions for plans easily risking a failure.

Finally, the decision to add or not such a communication can also depend on the trust the robot has in this human and on the estimated experience the human has in doing this specific task or in working with the robot.

4.6.2 Representing Explicitly Observation Processes

A lot of code is common between the primitive tasks not only in a planning domain but also between multiple ones. Indeed, the part where we update the beliefs of the agent performing the action, but also the ones of the other agents in the room is present in almost all the primitive actions. This commonality raises the question of the observability of actions. Indeed, even if we can assume that when an agent is planned to do an action they will be aware of its effects, and that if the human is planned to perform an action, even if it is not observable from the planned robot position, we update the beliefs of the robot (as it is emulating the human actions), it is not the case for a robot action and the update of human beliefs. We can represent it coarsely in the primitive task effects by relying on heuristics such as the presence of the agents in the same room. Still, observability of actions may need a specific representation for belief update in the planner core.

Besides, even if we plan that the human will do one action or the other at one point in the plan, recognizing and distinguishing between them may not be possible during the execution. This can endanger the interaction as the wrong branch of the plan may be executed. Thus, the supervision component must be informed, along with the plan, of the observability of the human actions, which can, in turn, impact the plan selection process.

4.6.3 Pruning During the Search Space Exploration

For now, all the search space is explored to find valid plans, and only then the cost of actions are evaluated to select the optimal conditional plan. However, even if the search is guided by the HTNs, the branching factor can become large, especially if the planning process is executed in a robotic architecture while interacting with a human. Thus, it is possible to evaluate the plan cost during exploration and to prune some branches in the search space according to the best plan found so far. Yet, doing so would prevent from applying plan wide (social) costs, as they could change the optimality of the plan used during the search.

Another approach would be to learn which actions a human is more likely to perform in a certain state. We could then prune all the least probable human actions (still returned by their modeled task tree). Although this approach can be easy to implement, learning must be done per human (as two humans may react differently in the same situation) and on a few learning samples as the same world state seldom appears during specific tasks on typical short term laboratory scenarios and generalization could be difficult.

Finally, we can also imagine that the robot might ask the human which tasks they are likely to perform at a specific step in the plan, while elaborating the plan. This would lead to negotiations with the human making the final plan more acceptable while also reducing the branching factor as only the tasks answered by the human would be explored. However, for long and complex domains, this solution may confuse the human because it would require them to project on the

long-term among multiple conditional eventualities.

In the next chapter, we will show how HATP/EHDA can be integrated into a robotic architecture and introduce a simple to reproduce yet challenging collaborative task inspired by psychology studies: the director task. To be performed by a robot, this task includes several prerequisites such as being able to take the perspective of the human partner and to refer objects in a dynamically evolving environment. Moreover, it involves some challenges by the approach presented in this chapter. The robot architecture which will be presented is built to handle the task allowing us to show how HATP/EHDA can be used on a real task in a complete robotic architecture.

Chapter 5

Task Planner Integration Within a Robotic Architecture for Human Robot Interaction

Contents

5.1 Introduction

In the last chapter, we have presented a task planning approach dedicated to HRI where the robot not only plans for itself but also models the human decisions, actions and reactions in order to predict and adapt to their actions and elicit or even guide their decisions. This approach has been implemented on a prototype planner named Human Aware Task Planner with Emulation of Human Decisions and Actions (HATP/EHDA) and used on several examples demonstrating its strengths and drawbacks.

In this short chapter, we present how this prototype planner has been integrated into a complete robotic architecture, once again showing the versatility and usefulness of this new task planning approach. First, we provide details on the integration of HATP/EHDA with different components of the architecture. We conclude this chapter by presenting a new simple yet challenging task for Human Robot Interaction which, to our knowledge, has not been tackled by the community and for which

we have developed a robotic architecture, integrating HATP/EHDA, to handle it in the nominal case.

Integrating into a complete robotic architecture is obviously not only personal work, so we want to thank two other PhD. students Guillaume Sarthou, for designing and developing the ontology engine used by the planner in this chapter and Amandine Mayima for designing and developing the supervision component in charge of requesting plans and using them to perform the task while managing the interaction and handling any contingencies.

5.2 Integrating With Other Components

Such a task planning component as the one presented earlier only takes on its full meaning when integrated into a robotic architecture. Indeed, its use is quite limited if initial world states have to be set by hand in the planning domain and if the elaborated plans are not executed. In this section we describe how we have integrated HATP/EHDA with other components allowing to retrieve the initial states (including human beliefs) from an ontology, to match the hybrid approach presented in Chapter 3 by using the REG component and to communicate with the supervision through ROS.

5.2.1 Retrieving the Current State and Beliefs From the Knowledge Base

Before any planning process could start, the planner must be initialized with the current world state (robot beliefs) and the estimation of the human's beliefs. To do so, we use the Knowledge Base (KB) presented in Chapter 3: Ontologenius, a component managing the acquired symbolic knowledge and representing it as ontologies. In our architecture, each human the robot is interacting with has a dedicated ontology, fed by a perspective taking component estimating the human beliefs; in addition to the robot own ontology, representing the world state. Thus, each agent beliefs σ are initialized with the facts in its respective ontology. However, ontologies can contain a lot of information that is not needed for planning, and worse, that can hamper keeping world state consistency if the planning domains do not consider some of these facts (*e.g.* an action deleting a fact but not deleting the inverse relation). To cope with this issue, we define two special attributes in the world state objects: the *types* and the *individuals* attributes.

The *types* attribute must be set by the user before retrieving the current world state. It is a dictionary linking a type name to a list of property names. A world state must be initialized solely with these attributes before being passed to a function retrieving the world state from the ontology. This function takes the agent name as the argument, and will fill the world state with the ontology matching to this name. This function fills the *individuals* attribute of the world state with a dictionary linking the types specified as keys of the *types* dictionary to a list of entities/individuals inheriting from these types. Then, a world state attribute is

created for each property defined in the *type* attribute, and filled with a map linking a subject entity to existing entities list according to the relations in the ontology.

Communication with the ontology is done using the Ontologenius [Sarthou 2019] python API. This API allows to make low level requests to the KB. The underlying communications are done between HATP/EHDA through the ROS framework. The entities inheriting from a certain type are for example retrieved through the function `getTypes`, querying the ontology with the name specified as parameter and returning the list of such entities.

For example, in the ontology from Chapter 3 describing the situation of Figure 3.1(a) represented in Figure 3.2 and 3.3, in a domain aiming at sorting the keys collaboratively does not need to represent the color of the keys nor of the areas. In this example, the *type* attribute would be set to:

```
[...]
state.types = {'Key': ['isIn', 'reachableBy'], 'Agent': [], 'Area': ['isOn']}
[...]
```

In which case, the resulting state would have its *individuals* attribute filled and have new attributes created such as:

```
[...]
print(state.individuals)
# {'Key': ['key_1', 'key_2', 'key_3'], 'Agent': ['human_3', 'pr2_robot'], 'Area': ['area_black', 'area_white', 'area_red'], 'Table': ['table_1']}

print(state.isIn)
# {'key_1': ['area_red'], 'key_2': ['area_black'], 'key_3': ['area_white']}

print(state.reachableBy)
# {'key_1': ['human_3', 'pr2_robot'], 'key_2': ['human_3', 'pr2_robot'], 'key_3': ['human_3', 'pr2_robot']}

print(state.isOn)
# {'area_black': ['table_1'], 'area_white': ['table_1'], 'area_red': ['table_1']}
[...]
```

Besides reading an initial state from the KB, the prototype planner is also able to write in it. Indeed, HTNs can not only be used as operational models for planning, but also can be used as a semantic source making verbal communications able to use past plans. To do so, however, the KB must be informed with the decompositions of task and with the parameters and their types they require. Our prototype is able to parse a planning domain in order to extract the required information and write them in an ontology friendly format.

5.2.2 Using REG at Planning Time

In Chapter 3, we presented how we integrated referring expression generation algorithm during task planning to precisely evaluate communication feasibility and cost. We showed how we successfully integrated this approach in HATP. However, because of the HATP architecture, its use was restrained to compute only action feasibility and cost when an action was explored by the planner. With this new planning scheme, we are able to make a REG request and fully use the returned RE at any time in action or decomposition functions (as they are Python functions). For example, in a decomposition, we can request a REG for multiple entities and return sub-tasks concerning only the least costly one.

Besides, we can also use the REG failure information to know which entities prevent another from being referred and select decomposition accordingly. For example, if an entity is not distinguishable from another through the REG, we can try to rely on another communication mean, such as pointing, or to move the distractor entity to remove it from the context entities.

The REG component is available to be requested via a ROS service. The requests must contain:

- The name of the ontology on which to perform the REG: in our case we pass the name of the ontology of the hearer agent that we copied and updated for the planning, as described in Chapter 3.

- The context: here, the context can be defined in the planning domain. By doing so, we can define it depending on the task in which the communication is planned to occur.

- The target entity: the entity for which we want a referring expression.

5.2.3 Communicating Through ROS

Finally, even if the planning process can be initialized and started through a Python script, it is not enough to make it useful in a real robotic architecture. Thus, we integrated planning requests support via the ROS framework. This allows the supervision component to use the planner with a specified domain by requesting a sequence of tasks to decompose for both agents. The resulting conditional plan is then used to manage the execution.

Once the planning service is launched, the specified domains are loaded for controllable and uncontrollable agents, then a service server is started, waiting for planning requests.

The **requests** need to be filled with two parameters for each of both controllable and uncontrollable agents:

- The **name** of each agent is used to retrieve their beliefs and to run the REG on their ontologies (retrieved by name) if needed.

- The sequence of tasks to decompose (which will be inserted in the initial agenda) along with their parameters for both the controllable and the uncontrollable agents.

  While the tasks for the controllable agent are simply the tasks a classical HTN planner would have to decompose, the uncontrollable agent's ones represent the tasks that the robot estimates the human is performing. Such information can be given via actions and intentions recognition components.

  The parameters of the task can be given as strings if they are simple enough (*e.g.* entity names, agent names) but they also can be formatted through JSON, allowing for more complex parameters (*e.g.* goals — which are represented as partial world state) allowing more subtle reasoning in the decomposition functions. The complex parameters are then deserialized into corresponding Python objects.

Once a request is received the planning process can start, and, once a conditional plan has been elaborated is sent back as a response to the request.

The **response** is constituted of a list of tasks each containing multiple fields:

- a unique id,
- a list of their parameters,
- the name of the agent to whom it has been assigned,
- its type (either abstract or primitive/action),
- the id of the previous task (if any),
- the id of the task it decomposes from (if any)
- a list of ids of the following tasks (if any),
- and a list of ids of the task it decomposes into (for an abstract task only)

Using this information a component can easily reconstruct the conditional plan computed.

In the future, we envision changing this ROS interface to allow for replanning accounting for past failed plans; and to perform anytime planning, by being able to return a plan early in the search while keeping refining plans to decrease the solution cost.

5.3 The Director Task

This work has been made in close collaboration with three other PhD. students Amandine Mayima, Kathleen Belhassein and Guillaume Sarthou.

5.3.1 A Task Used in Psychology

The director task is an experiment setup largely used and derived in psychology. It places two agents, the director and the receiver, in front of each other with a shelf in between. Usually, only the receiver is the participant, and the director is either an accomplice or a remote controlled-agent (for computer-based experiment). The shelf consists in several compartments possibly containing objects. Each compartment can either be open on both opposite faces or open on only the receiver's side (hiding any contained object from the director, and being obvious for the receiver that the director cannot see inside it).

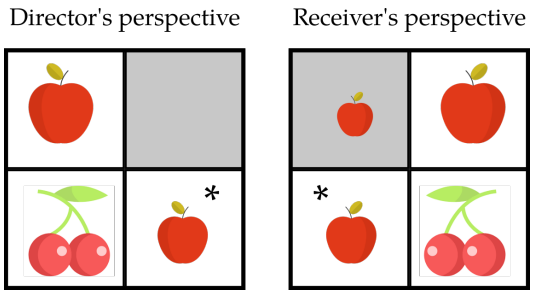

Figure 5.1: An example of the perspective difference between the director and the receiver.

The director asks the receiver to move some objects by (verbally) giving information allowing to discriminate them. However, some descriptions of the director will also match other objects —called competitors— that are only visible by the receiver. Hence, the receiver must think that the object matching the given director's utterance cannot be the one referred by the director as they are not aware of it, and find the object matching the description in the director's beliefs. This process must then be maintained all along the interaction as the situation evolves. For example, to designate the starred apple in Figure 5.1, the director may say "the small apple", however in the receiver's perspective, directly interpreting this sentence does not result in the starred apple. A decisional process is required, called perspective taking, to decide that the smallest apple in the receiver's perspective cannot be the one designated by the director as they cannot see it, so they are referring to the second smallest (being the smallest in their perspective)

This task is used in psychology to study how perspective-taking is used for communication understanding while performing a task with another agent [Keysar 2000]. Results show that, even if the receiver considered or took a competitor for the first trial, they are able to take the one designated by the director during subsequent trials. This shows that even if participants understand language in an egocentric way, they are able to do perspective-taking to successfully perform the task [Keysar 2003].

Even if this task is well known by psychologists, to our knowledge, no robot has ever performed it. A robot that does would illustrate its ability to maintain the

self-other distinction of beliefs but also to perform perspective taking on its human partner. We propose a robotic architecture integrating HATP/EHDA, that is able to handle both sides of the director task: both the director role giving instructions to its human partner and the receiver role interpreting instructions and selecting the right object. Besides, we propose some changes for both the director and receiver roles to be interesting along with planning challenges some modifications of the task can arise.

5.3.2 Setup

The setup we propose is a slight variation of the original director task used in psychology. First, instead of moving objects between compartments, the high level known goal is to remove a subset of the objects in the compartments and to place them in a receiver accessible only area (on top of a table).

Then, objects are replaced with blocks having four special attributes: their color, the color of their border, the shape drawn on them and the color of this shape. The colors can either be blue or green, and the shapes either a triangle or a circle. This allows for a maximum of ambiguity between the blocks. The material used is presented in Figure 5.2.

Figure 5.2: The material used for the director task. It aims at being simple and cheap to build to encourage the replication of the task.

Besides, to make the task interesting for both the director and receiver roles, compartments are not only either fully opened or hiding content from the director, they can also be hiding content from the receiver while being opened towards the director. Thus, not only the receiver must take the perspective of the director to understand the right block, but the director has also to take the perspective of the receiver to make the smallest instruction possible to respect the maxim of

quantity [Grice 1975]. In the example depicted in Figure 5.3, the minimal sentence to designate the circled block is "the blue block", but it requires both agents to perform perspective taking. Indeed, to rule out the distractors (designated with an arrow on the figure), it requires to deduce that they cannot have been referred by the other as they are not visible (and to an other extent, not known) by them.

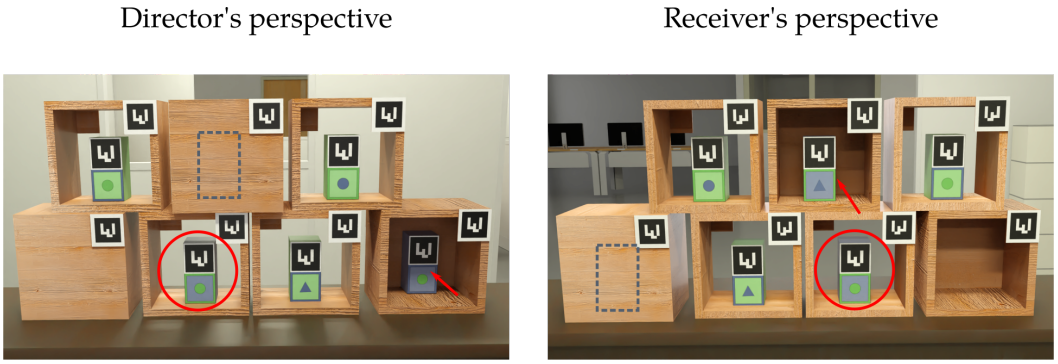

Figure 5.3: An example of both the director and receiver perspective in the director task. Here, to designate the circled block the sentence "the blue block" is enough as the distractors (designated with red arrows) can be ruled out through perspective taking.

In addition, to increase the number of ambiguous situations, we prohibit the use of geometrical relations during the communications (*e.g.* "the leftmost block", "the block above the green one") to only allow the use of the block attributes. Likewise, pointing at a block is forbidden.

Finally, all blocks and compartments are equipped with AR-tags (different on each face[1]) allowing the robot to easily detect them and to make an accurate representation of the environment (Figure 5.2).

5.3.3 The Robotic Architecture

The robotic architecture (Figure 5.4) is inspired by the one described by Lemaignan *et al.* [Lemaignan 2017] and is composed of several elements.

First, the Knowledge Base chosen is managed by Ontologenius and is structured as multiple ontologies. As we have shown before, ontologies are more and more common in robotics as they allow for rich and complete reasoning mechanisms and efficient requests of symbolic facts. The software used is *Ontologenius* [Sarthou 2019]. An ontology is created for the robot ($\mathcal{M}^R$, which we use as the true state of the environment) and another one is created for the human ($\mathcal{M}^H_r$) representing the robot estimation of the human knowledge. As stated in the previous chapter, these ontologies can be queried or updated through an efficient low-level API or higher-level SPARQL queries. The interfaces between the ontologies and our planner are

[1]Even if we assume that the humans cannot read and use the tags, we still stipulate to the participant that they are not the same on both sides of the cubes and compartments.

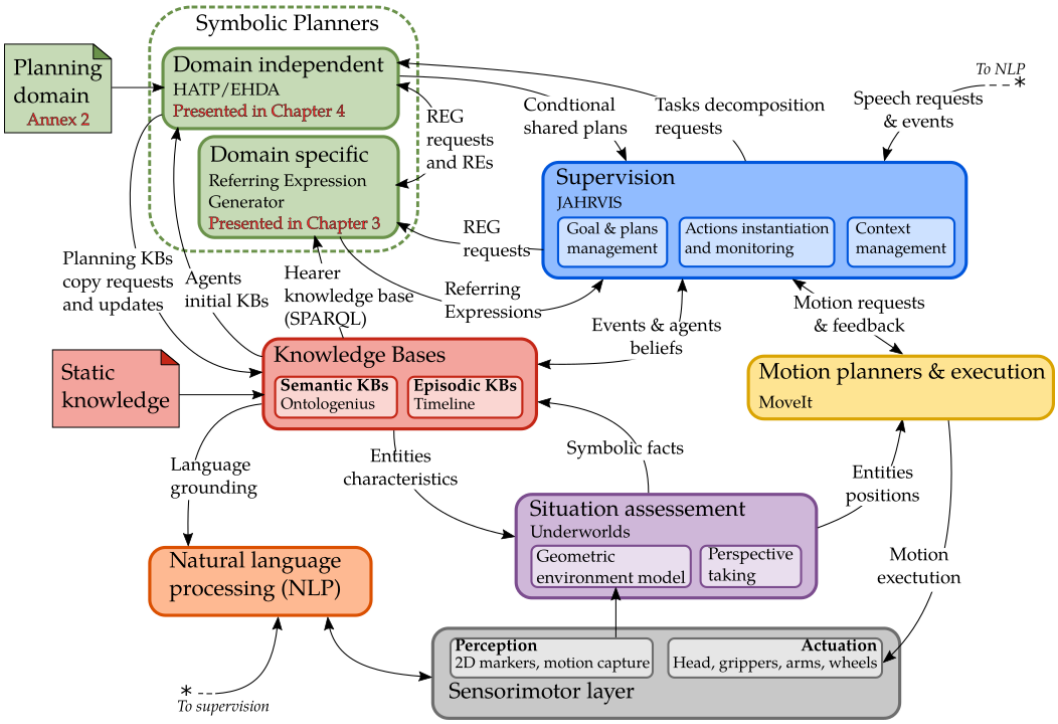

Figure 5.4: The robotic architecture implemented to handle both perspectives of the director task. The components circled with a dashed green line are presented in this thesis.

presented in Section 5.2. The ontology is initialized with static facts being (1) the link between each AR-tag and their matching block and compartment, (2) the attributes of each block, (3) the 3D models of each block and compartment.

To gather all the robot sensing data, create a geometric representation of the environment, compute symbolic facts from it for both the robot and the human and update the ontologies with them, a situation assessment component is added to the architecture. This component is based on *Underworlds* [Lemaignan 2018] allowing for modular and reusable reasoners. It first gathers the data from the perception algorithm (AR-tags positions for the objects and motion capture for the human), then creates a geometrical scene of the environment. Based on it, the component is able to compute symbolic facts on the objects present in the scene (and whose tags are detected): *isIn* when a block is inside a compartment, *isVisibleBy* when an agent is seeing an object, *isReachableBy* when an agent can reach (pick) an object and *isOnTopOf* when an object is on top of another. These symbolic facts are then fed in real time to the robot ontology. This component also estimates the geometrical scene viewed and known by the human, computes the symbolic facts defined before on this estimated scene and feeds the human ontology.

Then, to be able to give and understand instructions, the REG component presented in the previous chapter is also included. A grammar-based verbalization component allows to transform the generated RE into natural language. Similarly, a natural language request can be interpreted as a SparQL query and matched against an ontology.

To orchestrate all the components, the supervision system called JAHRVIS (Joint Action-based Human-aware supeRVISor) is dedicated to managing the interaction. It not only handles the robot actions but also estimates the human mental state, monitors the human actions and manages the communication with them. It manages five facets of the interaction: (1) interaction sessions, (2) communication, (3) human, (4) task and (5) quality of interaction. It is responsible for task planning requests and plans execution and monitoring.

Finally, the task planning component used in this architecture is HATP/EHDA, presented in the previous chapter. Task planning is only required when the robot is the director, since when the robot is receiver it only has to execute the director instructions one at a time after a request. When the robot is the director, the supervision system is given a list of blocks (via their ids) to remove from the shelves. This list is then passed as the goal parameter of a task to decompose to the planner.

The task to decompose is called `clear_blocks` and is filled with a parameter representing the goal and another one being the human id. The goal is simply an under specified world state, composed of triplets containing for example `(block_23, isIn, disposalArea)`. The robot task planning domain ($\mathcal{M}^R$) is presented hereafter.

- The `clear_blocks` abstract task has only one (recursive) decomposition. The decomposition returns () if the blocks specified in the goal matches the relations also specified in the goal. Otherwise, a REG request is performed for all

the misplaced blocks, and the tasks `clear_one_block` and `clear_blocks` are returned, with the easiest block to refer to (*i.e.* lowest RE cost) as parameter of the `clear_one_block` task.

- The `clear_one_block` task has also only one decomposition returning the primitive task `tell_human_to_clear_block` and the abstract one `wait_for_human_to_clear_block`.
- The `wait_for_human_to_clear_block` abstract task has only one decomposition. It aims at recursively planning to wait until we planned the human has removed and put the right block away. It recursively decomposes into () if the block passed as argument is in the right place; and returns `wait` and `wait_for_human_to_clear_block` otherwise.

The robot primitive tasks are defined as follow:

- The `tell_human_to_clear_block` primitive task returns $\perp$ if the block specified as parameter is not reachable by the human, or if it is not present in the human beliefs $\mathcal{M}_r^H$. Else, it adds the abstract task `clear_told_block` to the human agenda passing the specified block as parameter and does not update any beliefs.
- The `wait` primitive task does not update any beliefs.

On the human tasks side, we modeled a cooperative human, non declining asked tasks. Thus, the task model is as follows:

- The abstract task `clear_told_block` has only one decomposition, returning the primitive tasks `pick_block` and `place_block`.
- The primitive task `pick_block` returns $\perp$ if the block passed as parameter is not reachable by the human in their beliefs; and updates the beliefs of all the agents in the room with the human holding the block otherwise.
- The primitive task `place_block` returns $\perp$ if the human is not carrying anything; else it updates the beliefs of all the agents in the room with the block being placed in the area specified as parameter, and the human not holding anything.

In the domain described previously, we compute in the abstract task the least costly block to refer to decompose the task. This greedy approach can be used because we remove the block during the task. The referring cost of blocks thus can only decrease at each iteration of the task. However, it would be completely different if some blocks were to be added to the shelf along with others to remove. There, our greedy approach would not work as a not costly action would make future actions more costly leading to a sub-optimal plan. Another approach could be to have the decomposition of `clear_blocks` to return all the orders of the block removing instructions to explore.

In the proposed version of the director task, only the cube order can be planned. Thus, in the following subsection, we propose slight variations of the task increasing the planning complexity and interest, along with some possible modeling directions to overcome these challenges.

A video presenting the architecture used in a real interaction to both tackle the director and the receiver perspective is available at `https://www.youtube.com/watch?v=jtSyZeqBkp0`. When the robot is the director, thanks to the planning scheme presented in this thesis, it is able to compute the minimal referring expression to designate the blocks while accounting for the human perspective. Moreover, it gives the instructions in the order minimizing the complexity of referring expressions at each step (Figure 5.5). Finally, the robot is also able to handle the receiver role by taking the human perspective when it processes their instructions. Moreover, using the REG algorithm presented in Chapter 3, if the human instruction is detected as being ambiguous (multiple entities matching the SPARQL built from the sentence they said), the robot is able to ask the right questions to find the right entity (Figure 5.6). For each matching entity, a REG is performed with the relations found in the human utterance (and used in the SPARQL query) are given as context. This allows to find the missing relations disambiguating the entities. The supervision then uses the REs returned to ask "Do you mean *RE1* or *RE2*?", and keeps the previous context to add it to the SPARQL query built with the human answer.

5.3.4 Challenges for Planning

To spice up the planning challenge in the director task, we propose some extensions to the presented task. First, by adding blocks with identical visual features to the shelf, and asking to remove a precise one of them, we introduce situations where verbal communication (and REG) is not enough to refer to a block. To solve this problem, we could for example add a decomposition to `clear_one_block`. In this decomposition, we could have the robot moving one distractor of the REG in a non visible compartment to be able to refer to the intended block. The planner would then have to balance between making the referring easier by moving a block before giving the instruction and giving a long and complex instruction (when feasible).

With the same scenario, using the REG extension presented in the previous chapter, allowing the RE to contain relations to past common actions, we could introduce another way for the robot to refer to a block. The planner would need to balance previous communication means with creating a unique past experience with a block to easily refer to it (*e.g.* "the blue block that I just moved").

Besides, we can add multiple distinct disposal areas for blocks. The director would then have to instruct the receiver not only the block to pick but also which area they have to place it in. Furthermore, we could also add a decomposition where the robot asks to pick an under specified block, ensuring that all the matching ones need to be removed, and plan for all the possible blocks picked by the human matching this description. Depending on the block picked, it would be planned to

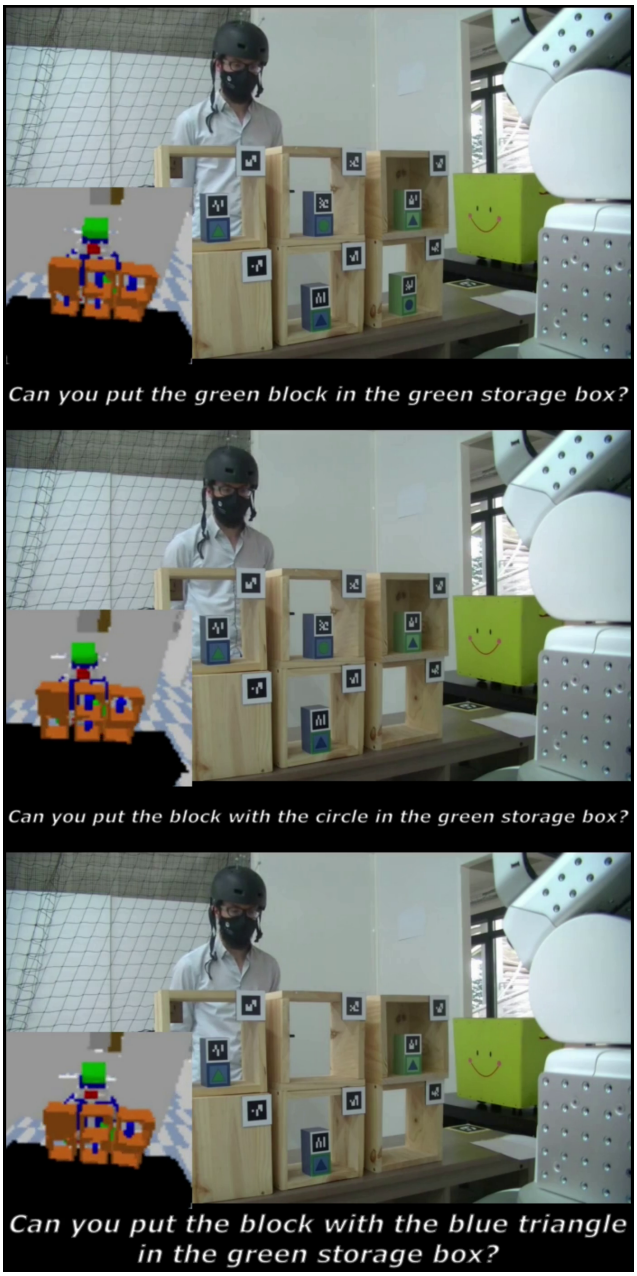

Figure 5.5: The director task handled by an autonomous PR2 robot in the director role. The computed human perspective is displayed in the bottom left hand corner of the picture. The robot said sentence is printed under each picture.

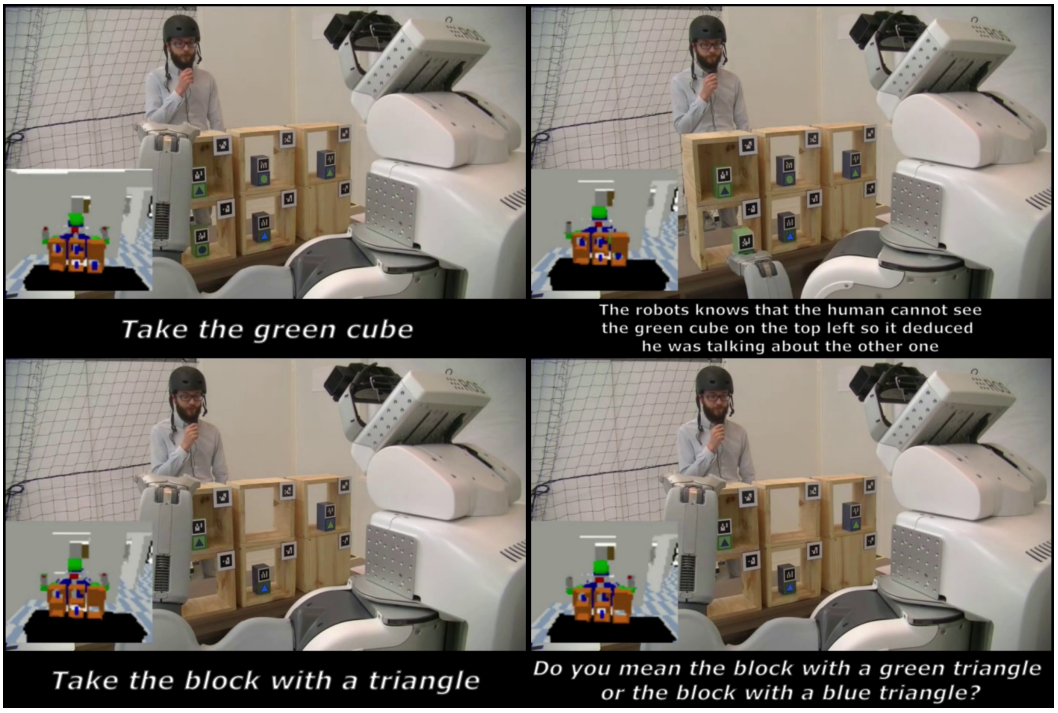

Figure 5.6: The director task handled by an autonomous PR2 robot in the receiver role. The robot is able to take the perspective of the human to understand his instructions, and can ask pertinent questions if the human instruction is ambiguous. Please note that some shapes on the blocks have been redrawn on the picture to enhance the contrast and improve its readability.

ask to place the cube in its corresponding area. This may result in better efficiency as it would lead to less complex referring expressions.

Finally, performing this task will necessarily bring errors, either from the human (*e.g.* the wrong block may be picked) or the robot (*e.g.* a block may fail to be picked). Even if some errors can be planned for (*e.g.* modeling all the blocks the human can pick and planning accordingly) doing so for all the possible contingencies is not possible. Therefore, a strong link with supervision must be envisioned as it will ask for replanning. Replanning is not as simple as passing the current state and the same goal and starting the process all over again, since some task decomposition may have been executed partially and cannot be changed in the replanning process.

5.4 Conclusion

In this chapter, we have shown how the task planning scheme presented earlier can be integrated into a robotic architecture. It can use larger Knowledge Bases to initialize the robot and the human beliefs, interface with domain-specific planners such as the REG approach presented in Chapter 3 and provide a service mechanism of ROS for the supervision to make planning requests.

Finally, an easy to reproduce yet challenging task for HRI was presented, along with a robotic architecture to handle this task. The planning scheme presented in Chapter 4 was used when the robot is the director to plan for the sequence of actions minimizing the communication complexity, and the REG component presented in Chapter 3 was used for both the director and receiver role to either find referring expressions for objects to be included in the instructions or to refine a human instruction when it has been detected as ambiguous.

Conclusion

In this thesis, we presented several contributions focusing on endowing the robot with the ability to plan explicitly for itself and for the human. Indeed, several approaches to human-aware navigation planning (and motion planning in general) account for human presence by including social costs influencing the trajectory, but only plan a course of actions for the robot. While being successful when humans are static or when evolving in large environments, these approaches are challenged when the interaction becomes intricate, where the robot and the human must collaborate to solve a problem. In robotic task planning, where human robot intricate interactions are easier to explore, some approaches do include planning for both the human and the robot. However, these approaches seldom maintain different beliefs for both agents during the planning process, whereas it is what a human is expecting according to joint action theory. Moreover, these approaches consider the human as a totally controllable agent, close to what is done in multi robot planning. They do not account for communications needed to align beliefs or to share the plan.

In Chapter 2, we showed why planning for both the human and the robot is important for navigation planning in intricate interaction scenarios. Not only it allows finding valid solutions where other approaches would have not, but, by anticipating the possible human trajectory we can make the robot more efficient and its behavior more satisfactory for the human. Indeed, by estimating the human future positions and speed, we are able to make the robot trajectory less threatening, more legible and to enhance the mutual manifestness of the robot. These results were validated through a user study, involving a totally autonomous PR2 robot crossing a human in a narrow corridor. We also presented how this navigation scheme has been implemented on other robots, including an HRP2 humanoid robot and a Pepper robot which has been deployed autonomously for several weeks providing route descriptions to customers in a mall.

Alleviating from the ephemeral nature of human robot navigation interaction, we presented a new way of planning for communication during task planning in Chapter 3. We chose to make a hybrid planning approach where a domain-independent HTN planner (HATP) delegates the resolution of feasibility and cost of communication actions to a domain-specific communication planner. We focused on communication actions needing to designate objects to the other agent. Thus, we needed a domain-specific planner able to determine the content of such communications. This problem is called referring expression generation and, albeit studied for a long time, we did not find any suitable existing work to be integrated into a human robot task planning scheme. Indeed, to be used in task planning, such a planner must be efficient and some specific constraints imposed by human robot interaction have to be satisfied. We formalized the REG problem for HRI using an ontology as a knowledge base and we proposed an efficient algorithm to solve

it. This algorithm has been shown to be the most efficient to our knowledge while being designed for HRI scenarios. It then has been integrated into HATP, an HTN planner able to maintain beliefs of multiple agents during task planning. Resolving the content of communication actions at this stage is only doable if the planner plans for both human and robot, it allows preventing to reach situations where the execution is blocked because a communication action cannot be performed and also improve the quality of plans.

However, HATP relies on exploring only one Hierarchical Task Network (HTN) and allocate tasks to either the robot or the human depending on the task constraints to find an optimal plan. Representing interactive tasks this way leads to execution where the human is considered as knowing the plan before it begins. Indeed, several works use similar approaches and solve contingencies (*e.g.* beliefs divergence, plan communication) during the plan execution.

In Chapter 4 we propose a new task planning scheme enriching current human-aware task planners. The general idea is to not only keep distinct beliefs between the robot and the human but also have two separate action models. Both models are HTNs, but do not represent the same concepts. The robot HTN, as in classical HTN planning, is designed to give expert knowledge to the planner on the different ways for the robot to perform a task. The human HTN on the other hand, is closer to a human task model as used in interactive systems engineering in human computer interaction. It aims at representing how a human may achieve a task (*i.e.* emulating parts of their planning process) and how they may react to a particular world state or to a robot request. The presented planner uses both HTNs to elaborate valid conditional plans then selects the optimal one. These conditional plans contain the possible human actions deduced via their task model. We presented our approach in scenarios involving intricate human robot interactions and showed how it is suitable and results in interesting plans.

Finally, in Chapter 5, we presented how this new task planning scheme can be integrated into a complete robotic architecture dedicated to HRI. Moreover, we introduced a new task for HRI inspired by psychology experiments along with an implemented robotic architecture including our planning scheme, allowing to tackle some of the challenges of this task.

On the Human Agent Interaction Guidelines

We presented in Chapter 1 maxims coined by Bradshaw *et al.* [Bradshaw 2011] for human-agent interaction. We propose here to sum up which maxims have guided the work presented in this thesis and to what extent we were able to implement them.

By completely exploring the search space with the proposed planning method of Chapter 4, we can increase the *progress appraisal* of the robot. Indeed, by returning the complete conditional plan to the supervision it can compute how much the task is progressing, and communicate it to the human if needed. Besides, we proposed

a plan post-processing step where robot actions are added to guide human actions away from potential errors and to the optimal plan.

Moreover, by explicitly representing the action effects in the beliefs of both agents, we explicitly represent the *observability* of the robot. Moreover, as we consider some effects (or even some actions) to be not observable, and thus, not updating the human beliefs, the robot observability will impact the elaborated plan.

Then, we showed our approach can be used to balance plans where the robot is proactive and ones where it let the human choose the tasks attributions. This matches the agent *knowing its limits* maxim. Moreover, unlike HATP, the human decisions are not set, the planner provides a robot course of action for multiple possible human actions, thus adapting to the human choices. We also envision using the HTNs exploration to guide the human choices or inform them about potential outcomes.

Moreover, by representing robot unknown human beliefs, the robot is able to plan to ask questions and predict possible human answers to them, making the robot more *directable*. A strong link between the task planning process and the supervision is required to explore further this maxim.

Similarly, a first step has been made towards the negotiation and deconfliction of plans and beliefs alignment, increasing the robot *coordination*. Some possible approaches were presented in Chapter 4 to increase it even more by interacting with the human during the plan elaboration process, in addition to the human planning process emulation. This also requires a stronger link with supervision, currently being explored.

Besides, we allowed making the robot more *selective* as we showed in Chapter 4, by allowing to plan belief alignment actions only when they are needed, and only with the beliefs required by the human to perform better.

Finally, we showed in Chapter 2 that planning navigation for both the human and the robot allows to make the robot deconflict the trajectories earlier increasing the robot *predictability*. Moreover, we implemented a head behavior for the navigation aiming at showing its future trajectory while acknowledging the human presence.

Limitations and Future Work

We think there is plenty of potential for the task planning approach presented in Chapter 4 and it must be refined and enriched. While it seems promising, as it allows to represent and find plans for intricate interaction scenarios, has been integrated with a domain-specific communication planner and used in a robotic architecture, limitations can be identified. First, the turn-taking approach used does not translate the duration of actions. Indeed, some actions will be longer than others, and not accounting for it may lead to suboptimal plans and wrong prediction of human actions. Then, by not pruning some part of the exploration graph during the search, the approach is not efficient for large HTN domains. Efficiency can also

be gained by a better implementation in C++ rather than in Python. However, pruning while searching would prevent applying plan-wide costs.

Bringing Planning and Supervision Closer

Some future work has already been identified to extend the approaches presented in this thesis. The first one, already mentioned before, is to build a stronger link with supervision. In common robotic architectures, the link between the supervision and geometric and task planning comes down to planning request and plan response. However, this link may not be enough for long term interaction or intricate and dynamic situations usually found in HRI.

For HATEB, the navigation scheme presented in Chapter 2, the solution proposed assumes the human will respect the model provided, and adjust the robot trajectory accordingly. For example, if the robot is following the human in a narrow corridor, if the human goes slower than what is set in the navigation planner parameters, the robot will never overtake them. Indeed, the planned trajectory for the human is for them to accelerate making the robot following the human permanently. The human model provided to the planner ($\mathcal{M}_r^H$) must be as accurate as possible. Providing a perfect model for each human encountered is obviously impossible, that is why the supervision must not only provide an initial model to the planner with the planning request but must also update this human model during the execution, especially if contingencies are detected to happen while following the plan. Here, for example, a supervision system might decrease the human speed parameter if they are repeatedly detected to move slower than expected.

For the REG algorithm presented in Chapter 3 this human model update is also crucial to have a good estimate of the communication capabilities of the human and the associated difficulties to understand. As presented, some relations to describe an object are more difficult to understand than others. In our approach, we represented it by a cost associated with each property. This difficulty depends on the person the robot is interacting with. For example, using color to refer to an object is less efficient or even impossible when speaking to a color-blind person. Our costs are indeed defined per human we are interacting with. Besides, the difficulty to understand is also context-dependent. For instance, colors relation can be hard or even impossible to perceive if the scene is lit with colored lights. Again, to cope with these issues, the supervision must update on the human model used for planning during the execution. Moreover, it can request and iterate with the planner during the planning process to allow for more or less risky communications, leading to more or less efficient plans.

The same is true for the planning approach depicted in Chapter 4. The human action model along with associated costs must be updated on a per-human basis all along the interaction. By refining the human model the best prediction would be made for them, leading to more efficient plans. Besides, some tasks or actions can be enabled or disabled depending on the human and their level of expertise in the task and for robot collaboration. Heuristics can also be learned as to which

decomposition a human may use for a task in a specific context. Highly probable decompositions can then be explored first, resulting in a more efficient planning process.

To reduce the branching factor in human HTN exploration, we can also try to negotiate the plan while elaborating it. For example, if too many human actions are returned during the search, the planning process can request the supervision to propose the different task alternatives to the human and ask which one they would perform in the specific state the planner is in. Only the answered alternatives can then be explored by the planner. Not only it would help to reduce the branching factor, but the robot may appear more *predictable* and the plan more *explainable* as some actions would have been chosen by the human. Thanks to the HTN structure, communication about the tasks made easier. This negotiation may need multiple iterations as the alternatives proposed by the human may lead to unfeasible plans. However, if the choice is proposed for a point too far in the future, the human may have a hard time projecting themselves in that situation.

In addition to considering the possible human actions in the conditional plan, the planner could also expose the effects of the actions and more precisely their observable effects. By doing so, the supervision would know what to expect from the human and what to monitor to determine which action the human did, influencing the branch of the plan executed. Going further, the supervision may use the entire human action model to be able to also predict human behavior, in case of plan repair for example.

Theory of Mind Level Up

To make a better prediction of the human decisions and actions, some advanced scenarios require the robot to represent the model the human has made of it ($\mathcal{M}_h^R$). Indeed, as shown by Chakraborti *et al.* [Chakraborti 2017], using it can lead to more legible and predictable plans. As described in Chapter 1, joint action theory informs about the capabilities a human is using when interacting with another agent. Especially, humans can *predict* other's actions and *integrate them into their own plan*. Thus, the model the human is making about the robot will influence their decision process and how they will perform a task. This is why it is important to build this model. One approach to do so is to analyze the actions performed by the robot by taking the perspective of the human and emulate an inferring process to build a robot model.

For example, in the navigation scheme from Chapter 2, integrating this model would lead to a better prediction of human trajectory. The robot trajectory costs would also be more accurate, as more constraints could be added. A surprise cost for instance can be estimated by comparing the planned robot trajectory ($\mathcal{M}^R$), with the one expected by the human ($\mathcal{M}_h^R$). This could, in turn, be used to better respect and evaluate the respect of the maxim of *predictability* and *dependability*.

Likewise, in the planning approach presented in Chapter 4, using the estimation of the robot model that the human has would result in more accurate predictions

of human actions. Indeed, we know that the human will integrate actions of other agents in their own plan, as shown by joint action theory. Until now, we assumed that it would not be the case as we envisioned interaction with novice users who have never interacted with a robot before, or not enough to be confident to integrate the robot action in their planning process. But, as they would gain experience, trust and habits, human partners will expect the robot to perform in a certain way. Plans quality would increase by integrating these expectations in the emulation of the human planning process. Moreover, we could improve the *directability* and *observability* of the robot by adding possible actions explicitly updating the human's robot model.

Appendix A

Navigation User Study Questionnaires

A.1 Original PeRDITA Questionnaire Without Verbal Dimension (French)

Numéro de participant :

Afin d'étudier votre évaluation personnelle du robot, vous allez répondre à un questionnaire. Vous allez devoir vous situer entre deux adjectifs en plaçant une croix dans la case qui se rapporte le plus à votre impression.

- Lisez attentivement les énoncés en gras avant de répondre aux échelles. Si besoin, n'hésitez pas à vous y reporter.
- Plus vous vous rapprochez d'un adjectif, plus il correspond à votre impression.
- Veillez à ne mettre qu'une seule croix par échelle.
- Ne vous reportez pas à une réponse donnée précédemment.
- Veuillez répondre à toutes les échelles.
- Il n'y a pas de bonne ou de mauvaise réponse. Répondez le plus sincèrement possible.

Selon vous, le robot est plutôt :

Non réactif	☐	☐	☐	☐	☐	☐	☐	Réactif
Performant	☐	☐	☐	☐	☐	☐	☐	Non performant
Inintelligent	☐	☐	☐	☐	☐	☐	☐	Intelligent
Incompétent	☐	☐	☐	☐	☐	☐	☐	Compétent

Selon vous, l'interaction avec le robot a été :

Négative	☐	☐	☐	☐	☐	☐	☐	Positive
Simple	☐	☐	☐	☐	☐	☐	☐	Compliquée
Pas pratique	☐	☐	☐	☐	☐	☐	☐	Pratique
Claire	☐	☐	☐	☐	☐	☐	☐	Ambiguë
Déplaisante	☐	☐	☐	☐	☐	☐	☐	Plaisante

Selon vous, la collaboration avec le robot a été :

Contraignante	☐	☐	☐	☐	☐	☐	☐	Adaptative
Utile	☐	☐	☐	☐	☐	☐	☐	Inutile
Acceptable	☐	☐	☐	☐	☐	☐	☐	Perturbante
Insatisfaisante	☐	☐	☐	☐	☐	☐	☐	Satisfaisante
Inefficace	☐	☐	☐	☐	☐	☐	☐	Efficace

Selon vous, le robot a choisi d'agir de manière :

Adéquate	☐	☐	☐	☐	☐	☐	☐	Inadéquate
Gênante	☐	☐	☐	☐	☐	☐	☐	Accommodante
Imprévisible	☐	☐	☐	☐	☐	☐	☐	Prévisible
Sécurisante	☐	☐	☐	☐	☐	☐	☐	Insécurisante

A.2 Unofficial Translation of the PeRDITA Questionnaire

In order to study your personal evaluation of the robot, you are going to fill this questionnaire. You need to place yourself between two adjectives by checking the box that matches the most to your impression.

- Carefully read the bold statements before checking the boxes. If needed, do not hesitate to read them again.
- The closest you place yourself to an adjective, the more it matches your impression.
- Only check one box per scale.
- Do not refer to a previous answer.
- Please fill all scales.
- There are no wrong or right answers. Answer as genuinely as possible.

From your point of view, the robot is rather:

Unresponsive	☐	☐	☐	☐	☐	☐	☐	Responsive
Efficient	☐	☐	☐	☐	☐	☐	☐	Not efficient
Not intelligent	☐	☐	☐	☐	☐	☐	☐	Intelligent
Incompetent	☐	☐	☐	☐	☐	☐	☐	Competent

From your point of view, the interaction with the robot has been:

Negative	☐	☐	☐	☐	☐	☐	☐	Positive
Simple	☐	☐	☐	☐	☐	☐	☐	Complicated
Not convenient	☐	☐	☐	☐	☐	☐	☐	Convenient
Clear	☐	☐	☐	☐	☐	☐	☐	Ambiguous
Unpleasant	☐	☐	☐	☐	☐	☐	☐	Pleasant

From your point of view, the collaboration with the robot has been:

Restrictive	☐	☐	☐	☐	☐	☐	☐	Adaptive
Useful	☐	☐	☐	☐	☐	☐	☐	Useless
Acceptable	☐	☐	☐	☐	☐	☐	☐	Confusing
Not satisfying	☐	☐	☐	☐	☐	☐	☐	Satisfying
Not effective	☐	☐	☐	☐	☐	☐	☐	Effective

From your point of view, the robot chose to act in a way:

Appropriate	☐	☐	☐	☐	☐	☐	☐	Inappropriate
Inflexible	☐	☐	☐	☐	☐	☐	☐	Compliant
Not expectable	☐	☐	☐	☐	☐	☐	☐	Expectable
Reassuring	☐	☐	☐	☐	☐	☐	☐	Unsafe

A.3 Situation Assessment Questionnaire (French)

Consigne : Pour chacune des affirmations suivantes, indiquez à quel point vous êtes en accord ou en désaccord en entourant la bonne réponse.

J'étais en capacité d'anticiper quel déplacement le robot allait effectuer. (*entourez la bonne réponse*)

1	2	3	4	5	6
Pas du tout d'accord	Pas d'accord	Plutôt pas d'accord	Plutôt d'accord	D'accord	Tout à fait d'accord

J'ai remarqué les signaux envoyés par le robot lors de son déplacement. (*entourez la bonne réponse*)

1	2	3	4	5	6
Pas du tout d'accord	Pas d'accord	Plutôt pas d'accord	Plutôt d'accord	D'accord	Tout à fait d'accord

Les indications du robot concernant son déplacement avaient du sens pour moi. (*entourez la bonne réponse*)

1	2	3	4	5	6
Pas du tout d'accord	Pas d'accord	Plutôt pas d'accord	Plutôt d'accord	D'accord	Tout à fait d'accord

J'ai trouvé les déplacements du robot prévisibles. (*entourez la bonne réponse*)

1	2	3	4	5	6
Pas du tout d'accord	Pas d'accord	Plutôt pas d'accord	Plutôt d'accord	D'accord	Tout à fait d'accord

J'ai compris le comportement de déplacement du robot. (*entourez la bonne réponse*)

1	2	3	4	5	6
Pas du tout d'accord	Pas d'accord	Plutôt pas d'accord	Plutôt d'accord	D'accord	Tout à fait d'accord

J'ai perçu les élements pertinents indiquant les déplacements du robot. (*entourez la bonne réponse*)

1	2	3	4	5	6
Pas du tout d'accord	Pas d'accord	Plutôt pas d'accord	Plutôt d'accord	D'accord	Tout à fait d'accord

A.4 Translation of Situation Assessment Questionnaire Items

- J'ai perçu les éléments pertinents indiquant les déplacements du robot.

  I perceived the relevant elements indicating the robot motion.

- J'ai remarqué les signaux envoyés par le robot lors de son déplacement.

  I noticed the signals sent by the robot during its motion.

- J'ai compris le comportement de déplacement du robot.

  I understood the robot motion behavior.

- Les indications du robot concernant son déplacement avaient du sens pour moi.

  The robot signs about its motion made sense to me.

- J'étais en capacité d'anticiper quel déplacement le robot allait effectuer.

  I was able to anticipate which motion the robot was going to do.

- J'ai trouvé les déplacements du robot prévisibles.

  It seemed to me that the robot motions were predicatables.

A.5 AttrakDiff Questionnaire (French)

This questionnaire is the official translation of the AttrakDiff questionnaire [Lallemand 2015].

Ce questionnaire se présente sous forme de paires de mots pour vous assister dans l'évaluation du système. Chaque paire de mots représente des contrastes. Les échelons entre les deux extrémités vous permettent de décrire l'intensité de la qualité choisie. Ne pensez pas aux paires de mots et essayez simplement de donner une réponse spontanée. Vous pourrez avoir l'impression que certains termes ne décrivent pas correctement le système. Dans ce cas, assurez-vous de donner tout de même une réponse. Gardez à l'esprit qu'il n'y a pas de bonne ou de mauvaise réponse.

Consigne: Notez le **ROBOT** sur les dimensions suivantes en cochant les cases qui correspondent à vos choix.

Simple	☐	☐	☐	☐	☐	☐	☐	Compliqué
Laid	☐	☐	☐	☐	☐	☐	☐	Beau
Pratique	☐	☐	☐	☐	☐	☐	☐	Pas pratique
De bon goût	☐	☐	☐	☐	☐	☐	☐	De mauvais goût
Prévisible	☐	☐	☐	☐	☐	☐	☐	Imprévisible
Bas de gamme	☐	☐	☐	☐	☐	☐	☐	Haut de gamme
Sans imagination	☐	☐	☐	☐	☐	☐	☐	Créatif
Bon	☐	☐	☐	☐	☐	☐	☐	Mauvais
Confus	☐	☐	☐	☐	☐	☐	☐	Clair
Ennuyeux	☐	☐	☐	☐	☐	☐	☐	Captivant

Appendix B

HATP/EHDA Domains for the Coffee Bringer Examples

The Human Aware Task Planner with Emulation of Human Decisions and Actions (HATP/EHDA) is presented in Chapter 4 along with illustrative examples allowing to present its features. We propose to report here the interesting parts of the domains in Python, as they were used for running the examples.

B.1 Plan for Robot Unknown Human Knowledge

In this example, the robot has been asked by a human to bring her a coffee. However, multiple mugs are on the table in front of the robot and the human, and the robot does not know which one belongs to the human.

```
import hatpehda
from copy import deepcopy
from hatpehda import gui
from hatpehda.reg import REGHandler

regHandler = None

### Helpers

def same_last_tasks(plan, n, task=None):
    """
    Given a partial 'plan', returns True if the 'n' last tasks of the
        partial plan are the same (and optionally equal to 'task')
    """
    if len(plan) < n:
        return False
    last_tasks = [plan[-i].name for i in range(1, n + 1)]
    if task is not None and last_tasks[0] != task:
        return False
    return last_tasks.count(last_tasks[0]) == len(last_tasks)

### Primitive tasks

def robot_pick_mug(agents, self_state, self_name, mug):
    """
    Checks if the 'mug' is reachable by the agent (robot) and if the agent
        does not carry anything.
```

```
    Then updates the beliefs of all the agents in the same room as the
        robot as for the robot having picked the 'mug'
    """
    if self_name in self_state.isReachableBy[mug] and self_state.isHolding[
        self_name] == []:
        robot_room = self_state.isInRoom[self_name][0]
        for agent_name in self_state.agentsInRoom[robot_room]:
            a = agents[agent_name]
            a.state.isReachableBy[mug] = []
            a.state.isHolding[self_name] = [mug]
            a.state.isHeldBy[mug] = [self_name]
        return agents
    return False


def robot_ask_mug_to_take(agents, self_state, self_name, human):
    """
    Asks the 'human' which one is their mug by adding to their agenda that
        they will answer the question.
    Note that we could have checked if the communication was feasible
    """
    hatpehda.add_tasks(human, [("human_answer_mug_a", self_name)], agents)
    return agents

def robot_drop_mug(agents, self_state, self_name):
    """
    Checks if the robot is holding something.
    If so, update all the agents in the room beliefs with the fact that the
         robot as dropped what they thought it was holding.
    """
    if self_state.isHolding[self_name] != []:
        robot_room = self_state.isInRoom[self_name][0]
        for agent_name in self_state.agentsInRoom[robot_room]:
            a = agents[agent_name]
            mug = a.state.isHolding[self_name]
            if mug != []:
                mug, = mug
                a.state.isReachableBy[mug] = [self_name]
                a.state.isHolding[self_name] = []
                a.state.isHeldBy[mug] = []
        return agents
    return False

def robot_go_to_coffee_machine(agents, self_state, self_name):
    """
    This action has no effect nor preconditions. It only represents the
        robot leaving the room (for the example)
    """
    return agents
```

```
def human_verbally_answer_right_mug(agents, self_state, self_name, robot,
    mug):
    """

    """
    # We make a REG request to check if we estimate that the human will be
        able to refer to their mug
    ctx = [("?0", "isAbove", "table_1")]
    symbols = {"?0": mug}
    # This function copy the ontology of 'self_name' (human), update it
        with beliefs from 'self_state' and runs
    # a REG request with 'ctx' and 'symbols' as context with the target
        entity 'mug'
    reg = regHandler.get_re(self_name, self_state, ctx, symbols, mug)
    if not reg.success:
        return False

    human_room = self_state.isInRoom[self_name][0]
    for agent_name in self_state.agentsInRoom[human_room]:
        a = agents[agent_name]
        a.state.isOwnedBy[mug] = self_name
    return agents

def human_complain_mug(agents, self_state, self_name, robot):
    """
    This primitive task models the human complaining about the mug that the
         robot is 'holding' not being theirs.
    We update all the agents in the room beliefs, with the mug held by the
        'robot' not being the one 'self_name' (human)
    """
    human_room = self_state.isInRoom[self_name][0]
    for agent_name in self_state.agentsInRoom[human_room]:
        a = agents[agent_name]
        mug = a.state.isHolding[robot]
        if mug is not None and mug != []:
            mug, = mug
            a.state.isNotOwnedBy[mug] = self_name
    return agents

# As we don't know the agents name in advance, we store the operators here,
     until a plan request
ctrl_operators = [robot_pick_mug, robot_drop_mug, robot_ask_mug_to_take,
    robot_go_to_coffee_machine]
unctrl_operators = [human_verbally_answer_right_mug, human_complain_mug]


### Abstract tasks decompositions
```

```
def robot_ask_take_mug(agents, self_state, self_name, human):
    if same_last_tasks(agents[human].plan, 3, "WAIT"):
        return False
    for mug, h in self_state.isOwnedBy.items():
        if h == human:
            return [("robot_pick_mug", mug)]
    for mug in self_state.individuals["Mug"]:
        if mug not in self_state.isNotOwnedBy or human not in self_state.
            isNotOwnedBy[mug]:
            return [("robot_ask_mug_to_take", human), ("robot_get_right_mug"
                , human)]
    return False

# This decorator informs the planner that this decomposition return several
     disjunctive sequences of tasks
# (typically for multiple possible parameter instantiation
@hatpehda.multi_decomposition
def robot_take_one_random_mug(agents, self_state, self_name, human):
    if same_last_tasks(agents[human].plan, 3, "WAIT"):
        return False
    for mug, h in self_state.isOwnedBy.items():
        if h == human:
            return False
    task_list = []
    for mug in self_state.individuals["Mug"]:
        if mug not in self_state.isNotOwnedBy or human not in self_state.
            isNotOwnedBy[mug]:
            task_list.append([("robot_pick_mug", mug)])
    if task_list == []:
        return False
    return task_list


def human_agree_mug_taken(agents, self_state, self_name, robot, mug):
    return []

def human_disagree_mug_taken(agents, self_state, self_name, robot, mug):
    return [("human_complain_mug", robot)]

@hatpehda.multi_decomposition
def human_answer_mug(agents, self_state, self_name, robot):
    for mug, h in self_state.isOwnedBy.items():
        if h == self_name:
            return [[("human_verbally_answer_right_mug", robot, mug)]]
    task_list = []
    for mug in self_state.individuals["Mug"]:
        if mug not in self_state.isNotOwnedBy or self_name not in self_state
            .isNotOwnedBy[mug]:
            task_list.append([("human_verbally_answer_right_mug", robot, mug
```

```
            )])
    if task_list == []:
        return False
    return task_list


# We don't know the agents name beforehand so we store them here, until we
    can add the proper agents
# Syntax: ('abstract_task_name', decompo1, decompo2, ...)
ctrl_methods = [("robot_get_right_mug", robot_take_one_random_mug,
    robot_ask_take_mug)]
unctrl_methods = [("human_answer_mug_a", human_answer_mug), ("
    human_check_mug_taken", human_agree_mug_taken, human_disagree_mug_taken
    )]


# Triggers
def human_check_mug(agents, self_state, self_name):
    if agents["robot"].plan[-1].name == "robot_pick_mug":
        mug = self_state.isHolding["robot"][0]
        if mug in self_state.isNotOwnedBy and self_state.isNotOwnedBy[mug]
            == self_name:
            return False
        for action in agents[self_name].plan:
            if action.name == "human_verbally_answer_right_mug" and action.
                parameters[1] == mug:
                return False
        return [("human_check_mug_taken", "robot", mug)]
    return False

def robot_check_wrong_mug(agents, self_state, self_name):
    if self_state.isHolding[self_name] is not None and self_state.isHolding
        [self_name] != []:
        heldMug = self_state.isHolding[self_name][0]
        if heldMug in self_state.isNotOwnedBy and "human" == self_state.
            isNotOwnedBy[heldMug]:
            return [("robot_drop_mug", ), ("robot_get_right_mug", "human")]
    return False


if __name__ == "__main__":
    regHandler = REGHandler()

    n_mug = 2

    state = hatpehda.State("robot_init")
    state.types = {"Agent": ["isHolding"], "Mug": ["isHeldBy", "
        isReachableBy"]}
    state.individuals = {'Mug': ["mug_{}".format(i) for i in range(n_mug)]}
```

```
state.isHeldBy = {m: [] for m in state.individuals["Mug"]}
state.isHolding = {"human": [], "robot": []}
state.isOwnedBy = {}
state.isNotOwnedBy = {}
state.isReachableBy = {m: ["human", "robot"] for m in state.individuals
    ["Mug"]}
state.agentsInRoom = {"office": ["human", "robot"]}
state.isInRoom = {"human": ["office"], "robot": ["office"]}

hatpehda.declare_operators("robot", *ctrl_operators)
for me in ctrl_methods:
    hatpehda.declare_methods("robot", *me)
hatpehda.declare_triggers("robot", *ctrl_triggers)
hatpehda.declare_operators("human", *unctrl_operators)
for me in unctrl_methods:
    hatpehda.declare_methods("human", *me)
hatpehda.declare_triggers("human", *unctrl_triggers)

hatpehda.set_state("robot", state)
hatpehda.add_tasks("robot", [("robot_get_right_mug", "human"), ("
    robot_go_to_coffee_machine", )]) # Agenda initialization

human_state = deepcopy(state) # No beliefs divergence (detected) in
    this example
human_state.__name__ = "human_init"
hatpehda.set_state("human", human_state)


sols = []
fails = []
hatpehda.seek_plan_robot(hatpehda.agents, "robot", sols, "human", fails
    )

cost, cplan = select_plan(sols)
gui.show_plan(cplan, "robot", "human")
```

B.2 Balance Difficult Communications, Decomposition Cost and Task Attribution

In this example, the robot has to prepare coffee. It detects a human nearby, who does not appear to be performing a task. To prepare coffee, coffee needs to be retrieved from one of the two cupboards (one is closer than the other) and put in the machine; and water must be poured in the machine. We want the robot to balance between doing all on its own or asking the human for help. Moreover, we estimate that the human thinks there are some coffee in the nearest cupboard, while the robot knows there is not. Thus, we also want the robot to balance between

aligning these beliefs or letting the human potentially make the mistake.

```
import hatpehda
from copy import deepcopy
from hatpehda import gui

### Helpers

def agent_plan_contains(plan, task_name):
    for p in plan:
        if p.name == task_name:
            return True
    return False

### Primitive tasks

def robot_get_water(agents, self_state, self_name):
    if self_state.isHolding[self_name] is not None and self_state.isHolding
        [self_name] != []:
        return False
    for ag in agents.values():
       ag.state.isHolding[self_name] = ["water"]
    return agents

def robot_pour_water_in_machine(agents, self_state, self_name):
    if self_state.isHolding[self_name] is None or self_state.isHolding[
        self_name] == []:
        return False
    for ag in agents.values():
        ag.state.contains["coffee_machine"].append(ag.state.isHolding[
            self_name][0])
        ag.state.isHolding[self_name] = []
    return agents

def robot_pick_coffee(agents, self_state, self_name, closet):
    if self_state.isHolding[self_name] is not None and self_state.isHolding
        [self_name] != []:
        return False
    if self_state.contains[closet] is None or self_state.contains[closet]
        == []:
        return False
    for ag in agents.values():
       ag.state.isHolding[self_name] = ["coffee"]
    return agents

def robot_put_coffee_in_machine(agents, self_state, self_name):
    if self_state.isHolding[self_name] is None or self_state.isHolding[
        self_name] == []:
        return False
    for ag in agents.values():
```

```
            ag.state.contains["coffee_machine"].append(ag.state.isHolding[
                self_name][0])
            ag.state.isHolding[self_name] = []
        return agents

def robot_ask_human_for_help(agents, self_state, self_name, human):
    hatpehda.add_tasks(human, [("human_help_make_coffee", self_name)],
        agents)
    return agents

def robot_serve_coffee(agents, _, __):
    return agents

def robot_update_human_inventory(agents, self_state, self_name, human,
    closet):
    agents[human].state.contains[closet] = self_state.contains[closet]
    return agents

def human_get_water(agents, self_state, self_name):
    if self_state.isHolding[self_name] is not None and self_state.isHolding
        [self_name] != []:
        return False
    for ag in agents.values():
        ag.state.isHolding[self_name] = ["water"]
    return agents

def human_pour_water_in_machine(agents, self_state, self_name):
    if self_state.isHolding[self_name] is None or self_state.isHolding[
        self_name] == []:
        return False
    for ag in agents.values():
        ag.state.contains["coffee_machine"].append(ag.state.isHolding[
            self_name][0])
        ag.state.isHolding[self_name] = []
    return agents

def human_try_pick_coffee(agents, self_state, self_name, closet):
    if self_state.isHolding[self_name] is not None and self_state.isHolding
        [self_name] != []:
        return False
    if agents["robot"].state.contains[closet] is None or agents["robot"].
        state.contains[closet] == []:
        self_state.contains[closet] = agents["robot"].state.contains[closet]
        return agents
    for ag in agents.values():
        ag.state.isHolding[self_name] = ["coffee"]
    return agents

def human_put_coffee_in_machine(agents, self_state, self_name):
```

```
    if self_state.isHolding[self_name] is None or self_state.isHolding[
        self_name] == []:
        return False
    for ag in agents.values():
        ag.state.contains["coffee_machine"].append(ag.state.isHolding[
            self_name][0])
        ag.state.isHolding[self_name] = []
    return agents


# As we don't know the agents name in advance, we store the operators here,
     until a ros plan call
ctrl_operators = [robot_get_water, robot_pour_water_in_machine,
    robot_pick_coffee, robot_put_coffee_in_machine,
                robot_update_human_inventory, robot_ask_human_for_help,
                    robot_serve_coffee]
unctrl_operators = [human_get_water, human_pour_water_in_machine,
    human_try_pick_coffee, human_put_coffee_in_machine]

### Abstract Tasks

@hatpehda.multi_decomposition
def robot_make_coffee_alone(agents, self_state, self_name):
    return [[("robot_get_water",), ('robot_pour_water_in_machine',), ("
        robot_get_coffee", ), ("robot_put_coffee_in_machine", )],
            [("robot_get_coffee",), ("robot_put_coffee_in_machine",), ("
                robot_get_water",), ('robot_pour_water_in_machine',)]]

def robot_collaborate_make_coffee_no_comm(agents, self_state, self_name):
    return [("robot_ask_human_for_help", "human"), ("robot_help_make_coffee
        ", "human")]

@hatpehda.multi_decomposition
def robot_collaborate_make_coffee_with_belief_update(agents, self_state,
    self_name):
    tasks = []
    for cupboard in self_state.individuals["Cupboard"]:
        if agents['human'].state.contains[cupboard] != self_state.contains[
            cupboard]:
            tasks.append([("robot_update_human_inventory", "human", cupboard
                ), ("robot_ask_human_for_help", "human"), ("
                robot_help_make_coffee", "human")])
    if tasks == []:
        return False # Another decomposition handles it
    return tasks

@hatpehda.multi_decomposition
def robot_help_make_coffee(agents, self_state, self_name, human):
    if "water" in self_state.isHolding[human] and "coffee" not in
```

```
        self_state.contains["coffee_machine"]:
        return [[("robot_get_coffee",), ("robot_put_coffee_in_machine",)]]
    if agent_plan_contains(agents[human].plan, "human_try_pick_coffee") and
         "water" not in self_state.contains["coffee_machine"]:
        return [[("robot_get_water",), ("robot_pour_water_in_machine",)]]
    tasks = []
    if "coffee" not in self_state.contains["coffee_machine"]:
        tasks.append([("robot_get_coffee",), ("robot_put_coffee_in_machine"
            ,), ("robot_help_make_coffee", human)])
    if "water" not in self_state.contains["coffee_machine"]:
        tasks.append([("robot_get_water",), ("robot_pour_water_in_machine",)
            , ("robot_help_make_coffee", human)])
    return tasks

def robot_get_coffee(agents, self_state, self_name):
    if "coffee" in self_state.isHolding[self_name]:
        return []
    min_cupboard = None
    min_dist = 999999.0
    for cupboard in self_state.individuals["Cupboard"]:
        if "coffee" in self_state.contains[cupboard] and self_state.
            distances[cupboard][0] < min_dist:
            min_dist = self_state.distances[cupboard][0]
            min_cupboard = cupboard
    if min_cupboard is None:
        return False
    return [("robot_pick_coffee", min_cupboard), ("robot_get_coffee", )]

@hatpehda.multi_decomposition
def human_help_make_coffee(agents, self_state, self_name, robot):
    if "water" in self_state.isHolding[robot] and "coffee" not in
        self_state.contains["coffee_machine"]:
        return [[("human_get_coffee",), ("human_put_coffee_in_machine",)]]
    if "coffee" in self_state.isHolding[robot] and "water" not in
        self_state.contains["coffee_machine"]:
        return [[("human_get_water",), ("human_pour_water_in_machine",)]]
    tasks = []
    if "coffee" not in self_state.contains["coffee_machine"]:
        tasks.append( [("human_get_coffee",), ("human_put_coffee_in_machine"
            ,), ("human_help_make_coffee", robot)])
    if "water" not in self_state.contains["coffee_machine"]:
        tasks.append([("human_get_water",), ("human_pour_water_in_machine",)
            , ("human_help_make_coffee", robot)])
    return tasks


def human_get_coffee(agents, self_state, self_name):
    if "coffee" in self_state.isHolding[self_name]:
        return []
```

```
    min_cupboard = None
    min_dist = 999999.0
    for cupboard in self_state.individuals["Cupboard"]:
        if "coffee" in self_state.contains[cupboard] and self_state.
            distances[cupboard][0] < min_dist:
            min_dist = self_state.distances[cupboard][0]
            min_cupboard = cupboard
    if min_cupboard is None:
        return False
    return [("human_try_pick_coffee", min_cupboard), ("human_get_coffee",)]


# We don't know the agents name in advance so we store them here, until we
    can add the proper agents
ctrl_methods = [("robot_make_coffee", robot_make_coffee_alone,
    robot_collaborate_make_coffee_no_comm,
    robot_collaborate_make_coffee_with_belief_update),
               ("robot_help_make_coffee", robot_help_make_coffee),
               ("robot_get_coffee", robot_get_coffee)]
unctrl_methods = [("human_help_make_coffee", human_help_make_coffee), ("
    human_get_coffee", human_get_coffee)]


if __name__ == "__main__":
    state = hatpehda.State("robot_init")
    state.types = {"Agent": ["isHolding"]}
    state.individuals = { "Cupboard": ["kitchen_cupboard", "pantry_cupboard
        "]}
    state.isHolding = {"human": [], "robot": []}
    state.contains = {"coffee_machine": [], "kitchen_cupboard": [], "
        pantry_cupboard": ["coffee"]}
    state.distances = {"kitchen_cupboard": [2.0], "pantry_cupboard": [4.0]}

    hatpehda.declare_operators("robot", *ctrl_operators)
    for me in ctrl_methods:
        hatpehda.declare_methods("robot", *me)
    hatpehda.declare_operators("human", *unctrl_operators)
    for me in unctrl_methods:
        hatpehda.declare_methods("human", *me)
    hatpehda.set_state("robot", state)
    hatpehda.add_tasks("robot", [("robot_make_coffee", ), ("
        robot_serve_coffee", )]) # Agenda initialization

    human_state = deepcopy(state)
    human_state.__name__ = "human_init"
    human_state.contains = {"coffee_machine": [], "kitchen_cupboard": ["
        coffee"], "pantry_cupboard": ["coffee"]}
    # Belief divergence: while the robot knows that kitchen_cupboard does
        not contain anything, the human thinks it is
```

```
# containing coffee
hatpehda.set_state("human", human_state)


sols = []
fails = []
hatpehda.seek_plan_robot(hatpehda.agents, "robot", sols, "human", fails
    )

cost, cplan = select_policies(sols)
gui.show_plan(cplan, "robot", "human", with_abstract=True)
```

Appendix C

Résumé en Français

Nous fournissons ici un résumé en langue française des travaux présentés dans ce manuscrit de thèse.

Introduction

Les outils utilisés par les humains, autrefois simples, ont rapidement gagné en complexité. Ces outils aujourd'hui devenus robots sont capables d'agir de manière autonome et nécessitent de moins en moins de supervision humaine. Dans le même temps, les humains sont devenus de plus en plus dépendants de ces machines à la fois pour la vie quotidienne et pour des tâches plus spécifiques. Cependant, dans l'industrie, les robots et les humains ont très souvent leurs espaces de travail séparés, lorsque ce n'est pas le cas, les machines voient leurs capacités limitées et les humains requièrent une formation poussée pour les utiliser.

Dans cette thèse, nous explorons des méthodes pour rapprocher les humains et les robots afin de leur faire effectuer des tâches collaboratives au sein d'environnements partagés. Pour ce faire, nous soutenons que les robots doivent être capables de prendre des décisions non seulement sur leurs propres connaissances et perception de l'environnement, mais aussi sur leurs estimations des croyances de leur partenaire humain. De plus, le robot doit pouvoir planifier en prenant en compte que l'agent humain va aussi planifier, agir et réagir aux actions du robot.

Résumé de la Thèse

Nous nous intéressons aux moyens de planifier à la fois pour le robot, mais aussi pour son partenaire humain afin de permettre d'améliorer l'interaction lors de tâches collaboratives.

Chapitre 1 : Contexte et défis de l'interaction humain robot Dans le Chapitre 1 nous présentons le contexte de cette thèse en donnant une définition de la planification de tâches et de navigation, quels sont les défis de l'interaction humain robot et comment peut-on modéliser les actions de l'humain et élaborer des plans partagés.

Nous définissons tout d'abord les problèmes de planification de tâches et de navigation en robotique. La planification de tâche consiste à construire une séquence avec des actions données, permettant au robot (à un agent) d'atteindre un but

donné. La navigation, quant à elle, a pour but de générer (et de suivre) une trajectoire menant le robot (l'agent) d'un point A à un point B tout en évitant les obstacles qu'il rencontrera éventuellement sur son chemin.

Nous déclinons ensuite ces problèmes au contexte particulier de l'interaction humain robot en nous concentrant sur les concepts d'utilisabilité (efficacité, efficience et satisfaction dans la réalisation d'une tâche) et d'automatisation. De plus, nous présentons une liste de fonctionnalités requises par un agent autonome (*e.g.* robot) pour collaborer avec un humain. Ces prérequis sont ensuite mis en contexte par rapport aux capacités déployées par un humain lorsqu'il interagit avec un autre humain telles que décrites dans le domaine de la psychologie cognitive et plus particulièrement de l'étude de l'action jointe.

Enfin, plusieurs approches permettant la modélisation de l'activité humaine ainsi que la génération de plans incluant à la fois l'humain et le robot sont présentées.

Chapitre 2 : Coplanification pour la navigation Dans le Chapitre 2, nous explorons une approche à la planification de navigation pour le robot dans laquelle les trajectoires du robot, mais aussi de l'humain sont calculées à une fréquence de contrôle en position. Cette approche utilise un schéma d'optimisation permettant de définir des contraintes entre les trajectoires, représentant l'interaction entre elles. Nous nous servons de ces contraintes afin d'améliorer la *manifesteté* mutuelle du robot, notamment en décourageant les mouvements du robot pouvant être perçus comme menaçants par un humain. Ce faisant, le robot expose ses décisions plus tôt dans l'interaction, améliorant ainsi la *lisibilité* de ses mouvements. De plus, nous avons conçu et implémenté un mouvement de la tête du robot, permettant d'encore augmenter la lisibilité de la trajectoire tout en montrant à l'humain qu'il a bien été perçu.

Nous montrons grâce à une étude utilisateur, effectuée en partenariat avec des chercheurs en psychologie cognitive et ergonomie du CLLE, que ce comportement permet d'améliorer l'efficience de la navigation du robot dans des scénarios de croisement dans des passages étroits. Cette étude utilisateur a été soumise au journal *IEEE Transactions on Human-Machine Systems* et est en cours de revue. Suite à cette étude utilisateur, nous menons, avec une doctorante en psychologie, une réflexion sur les défis, problèmes et conduites à tenir concernant les études utilisateurs dans le domaine particulier de l'interaction humain robot. Cette discussion a été acceptée en tant qu'article dans le journal *Transactions on Human-Robot Interaction.*

De plus, nous présentons comment nous avons utilisé cette approche sur d'autres robots ainsi que dans un système robotique complet déployé "dans la nature" en Finlande au cours du projet MuMMER (MultiModal Mall Entertainment Robot[1]). Puis, nous continuons d'explorer la planification pour le robot et l'humain dans la planification de tâches.

[1]robot de divertissement multi-modal pour centres commerciaux

Chapitre 3 : Évaluation des communications pendant la planification de tâches Dans le Chapitre 3, notre objectif est d'utiliser les observations effectuées au cours du chapitre précédent pour les appliquer dans le domaine symbolique, de manière plus explicite. Le but est de considérer la communication comme actions à part entière que le robot doit planifier pour obtenir la meilleure interaction possible. Planifier de telles communications nécessite bien sûr de planifier pour les deux agents. Ce chapitre contient deux contributions.

Premièrement, nous présentons un algorithme efficient, utilisant une ontologie, permettant de calculer le contenu d'une communication visant à référencer (désigner) un objet de l'environnement à un autre agent. Ce problème est appelé le problème de la génération d'expressions de référence (*referring expression generation*) et, bien qu'il est étudié depuis plus de trente ans, nous montrons que notre approche n'est pas seulement la plus rapide à ce jour, mais qu'elle est aussi la plus adaptée pour des scénarios d'interaction humain robot. Notre algorithme se base sur l'exploration de l'ontologie (utilisée en tant que base de connaissances) de l'agent à qui est destiné la communication. Il est capable de prendre en compte le contexte dans lequel s'effectue la tâche ainsi qu'un coût représentant la difficulté de certaines propriétés à être interprétées par rapport à d'autres (*e.g.* la couleur est plus rapide à interpréter que la taille d'un objet).

Enfin, nous utilisons cette approche rapide pour déterminer le contenu d'une communication afin d'estimer la faisabilité et le coût de telles actions de communication durant la phase de planification de tâche. Pour ce faire, nous intégrons HATP, un planificateur pour réseaux hiérarchisés de tâches multi-agents, conçu pour l'interaction humain robot avec notre algorithme de génération d'expression de référence. Cette approche permet d'éviter la génération de plans qui auraient pu mener dans des situations irrécupérables durant leur exécution ainsi que de trouver les plans les plus efficients.

Chapitre 4 : Planification de tâches avec émulation des décisions et des actions de l'humain Dans le Chapitre 4 nous présentons un schéma de planification hiérarchique de tâches qui n'est pas seulement capable de maintenir les croyances du robot et de l'humain séparément durant le processus de planification, mais qui raisonne aussi sur deux modèles de l'action distincts de l'humain et du robot. Tandis que le modèle de l'action du robot est proche de ceux utilisés dans la planification avec réseaux hiérarchisés de tâches, celui de l'humain vise à s'inspirer d'approches de modélisation de tâches telles qu'utilisées dans le domaine de l'interaction humain machine. Les actions sont représentées en tant que fonctions qui opèrent sur les croyances des agents. Nous spécifions aussi des règles concernant les croyances pouvant être mises à jour ou accédées suivant l'agent effectuant l'action. Ainsi, les plans conditionnels générés ne fixent pas d'actions à l'humain, mais comportent des actions possibles issues de l'émulation de ses processus décisionnels. Ce schéma a été implémenté dans un planificateur prototype en Python dont nous donnons quelques détails importants d'implémentation.

De plus, au travers de deux exemples que nous construisons de manière incrémentale, nous montrons qu'il permet de représenter et de planifier pour des scénarios d'étroite collaboration entre le robot et l'humain.

Chapitre 5 : Intégration du planificateur de tâches dans une architecture robotique complète Enfin, dans le Chapitre 5, nous présentons des détails intéressants concernant l'intégration de ce planificateur prototype dans une architecture robotique. Nous finissons en proposant une nouvelle tâche pour la collaboration humain robot : la tâche du directeur. Cette tâche inspirée d'expériences de psychologie induit plusieurs défis pour l'interaction humain robot. De plus, à notre connaissance, elle n'a jamais été étudiée dans le cadre de l'interaction humain robot et aucun robot n'a jamais été conçu pour y répondre. Nous présentons donc ensuite une architecture robotique complète capable d'effectuer cette tâche dans son cas nominal et dans laquelle notre planificateur prototype a été intégré.

Conclusion Nous concluons cette thèse en revenant sur les contributions principales. De plus, nous les résumons en les mettant en correspondance avec les prérequis de l'interaction humain agent présentés dans le Chapitre 1.

Enfin, nous décrivons les limitations de nos approches ainsi que certaines pistes d'amélioration. Deux améliorations nous semblent particulièrement intéressantes : celle de rapprocher l'exécution et la planification, ce qui est crucial dans des environnements réactifs tels que rencontrés dans l'interaction humain robot ; ainsi que celle de non pas raisonner seulement sur le modèle du robot et celui de l'humain, mais aussi sur celui que l'humain se fait du robot durant son utilisation.

Bibliography

[Alami 1998] Rachid Alami, Raja Chatila, Sara Fleury, Malik Ghallab and Félix Ingrand. *An architecture for autonomy.* The International Journal of Robotics Research, vol. 17, no. 4, pages 315–337, 1998. (Cited in page 9.)

[Alili 2009] Samir Alili, Rachid Alami and Vincent Montreuil. *A task planner for an autonomous social robot.* In Distributed autonomous robotic systems 8, pages 335–344. Springer, 2009. (Cited in page 13.)

[Althaus 2004] Philipp Althaus, Hiroshi Ishiguro, Takayuki Kanda, Takahiro Miyashita and Henrik I Christensen. *Navigation for human-robot interaction tasks.* In IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004, volume 2, pages 1894–1900. IEEE, 2004. (Cited in page 15.)

[Annett 1967] John Annett and Keith D Duncan. *Task analysis and training design.* Journal of Occupational Psychology, 1967. (Cited in page 12.)

[Association 2013] World Medical Association*et al. World Medical Association Declaration of Helsinki: ethical principles for medical research involving human subjects.* Jama, vol. 310, no. 20, pages 2191–2194, 2013. (Cited in page 29.)

[Bartneck 2009] Christoph Bartneck, Dana Kulić, Elizabeth Croft and Susana Zoghbi. *Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots.* International journal of social robotics, vol. 1, no. 1, pages 71–81, 2009. (Cited in page 43.)

[Beetz 2010] Michael Beetz, Freek Stulp, Piotr Esden-Tempski, Andreas Fedrizzi, Ulrich Klank, Ingo Kresse, Alexis Maldonado and Federico Ruiz. *Generality and legibility in mobile manipulation.* Autonomous Robots, vol. 28, no. 1, page 21, 2010. (Cited in page 9.)

[Beetz 2015] Michael Beetz, Ferenc Bálint-Benczédi, Nico Blodow, Daniel Nyga, Thiemo Wiedemeyer and Zoltán-Csaba Marton. *Robosherlock: Unstructured information processing for robot perception.* In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 1549–1556. Citeseer, 2015. (Cited in page 82.)

[Belhassein 2019] Kathleen Belhassein, Guilhem Buisan, Aurélie Clodic and Rachid Alami. *Towards methodological principles for user studies in Human-Robot Interaction.* In Test Methods and Metrics for Effective HRI in Collaborative Human-Robot Teams Workshop, ACM/IEEE International Conference on Human-Robot Interaction, 2019. (Cited in page 39.)

[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 59 and 70.)

[Bradshaw 2003] Jeffrey M Bradshaw, Maarten Sierhuis, Alessandro Acquisti, Paul Feltovich, Robert Hoffman, Renia Jeffers, Debbie Prescott, Niranjan Suri, Andrzej Uszok and Ron Van Hoof. *Adjustable autonomy and human-agent teamwork in practice: An interim report on space applications.* In Agent autonomy, pages 243–280. Springer, 2003. (Cited in page 10.)

[Bradshaw 2011] Jeffrey M Bradshaw, Paul Feltovich and Matthew Johnson. *Human-agent interaction.* Handbook of Human-Machine Interaction, pages 283–302, 2011. (Cited in pages 8 and 152.)

[Breazeal 2005] Cynthia Breazeal, Cory D Kidd, Andrea Lockerd Thomaz, Guy Hoffman and Matt Berlin. *Effects of nonverbal communication on efficiency and robustness in human-robot teamwork.* In 2005 IEEE/RSJ international conference on intelligent robots and systems, pages 708–713. IEEE, 2005. (Cited in page 21.)

[Buckingham 2020] David Buckingham, Meia Chita-Tegmark and Matthias Scheutz. *Robot Planning with Mental Models of Co-present Humans.* In International Conference on Social Robotics, pages 566–577. Springer, 2020. (Cited in page 14.)

[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, pages 554–565. Springer, 2020. (Cited in page 56.)

[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 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 349–356. IEEE, 2020. (Cited in page 56.)

[Card 1983] Stuart K Card, Allen Newell and Thomas P Moran. *The Psychology of Human-Computer Interaction.* 1983. (Cited in page 7.)

[Carlucci 2015] Fabio Maria Carlucci, Lorenzo Nardi, Luca Iocchi and Daniele Nardi. *Explicit representation of social norms for social robots.* In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4191–4196. IEEE, 2015. (Cited in page 102.)

[Carpinella 2017] Colleen M Carpinella, Alisa B Wyman, Michael A Perez and Steven J Stroessner. *The robotic social attributes scale (rosas) development*

and validation. In Proceedings of the 2017 ACM/IEEE International Conference on human-robot interaction, pages 254–262, 2017. (Cited in page 43.)

[Chakraborti 2017] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang and Subbarao Kambhampati. *Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy.* In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pages 156–163, 2017. (Cited in pages 14 and 155.)

[Chakraborti 2018] Tathagata Chakraborti, Sarath Sreedharan and Subbarao Kambhampati. *Human-aware planning revisited: A tale of three models.* In Proc. of the IJCAI/ECAI 2018 Workshop on EXplainable Artificial Intelligence (XAI). That paper was also published in the Proc. of the ICAPS 2018 Workshop on EXplainable AI Planning (XAIP), pages 18–25, 2018. (Cited in pages 11, 12, and 66.)

[Choset 2005] Howie M Choset, Seth Hutchinson, Kevin M Lynch, George Kantor, Wolfram Burgard, Lydia E Kavraki, Sebastian Thrun and Ronald C Arkin. Principles of robot motion: theory, algorithms, and implementation. MIT press, 2005. (Cited in page 6.)

[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 11.)

[Clodic 2017] Aurélie Clodic, Elisabeth Pacherie, Rachid Alami and Raja Chatila. *Key elements for human-robot joint action.* In Sociality and Normativity for Robots, pages 159–177. Springer, 2017. (Cited in page 10.)

[Dale 1989] Robert Dale. *Cooking up referring expressions.* In 27th Annual Meeting of the association for Computational Linguistics, pages 68–75, 1989. (Cited in page 58.)

[Dale 1992] Robert Dale. Generating referring expressions: Constructing descriptions in a domain of objects and processes. The MIT Press, 1992. (Cited in page 58.)

[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 58, 77, and 80.)

[Dautenhahn 2006] Kerstin Dautenhahn, Michael Walters, Sarah Woods, Kheng Lee Koay, Chrystopher L Nehaniv, A Sisbot, Rachid Alami and Thierry Siméon. *How may I serve you? A robot companion approaching a seated person in a helping context.* In Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, pages 172–179, 2006. (Cited in page 41.)

[De Silva 2015] Lavindra De Silva, Raphaël Lallement and Rachid Alami. *The HATP hierarchical planner: Formalisation and an initial study of its usability and practicality.* In 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), pages 6465–6472. IEEE, 2015. (Cited in page 87.)

[Desmet 2007] Pieter MA Desmet and Paul Hekkert. *Framework of product experience.* International journal of design, vol. 1, no. 1, pages 57–66, 2007. (Cited in page 40.)

[Devin 2016] Sandra Devin and Rachid Alami. *An implemented theory of mind to improve human-robot shared plans execution.* In 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 319–326. IEEE, 2016. (Cited in pages 11, 14, 60, and 88.)

[Devin 2018] Sandra Devin, Camille Vrignaud, Kathleen Belhassein, Aurelie Clodic, Ophelie Carreras and Rachid Alami. *Evaluating the Pertinence of Robot Decisions in a Human-Robot Joint Action Context: The PeRDITA Questionnaire.* In 2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pages 144–151, Nanjing, August 2018. IEEE. (Cited in pages 29 and 43.)

[Dragan 2013] Anca D. Dragan, Kenton C.T. Lee and Siddhartha S. Srinivasa. *Legibility and predictability of robot motion.* In 2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 301–308, March 2013. ISSN: 2167-2148. (Cited in pages 9, 20, and 54.)

[Dragan 2015] Anca D Dragan, Shira Bauman, Jodi Forlizzi and Siddhartha S Srinivasa. *Effects of robot motion on human-robot collaboration.* In 2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 51–58. IEEE, 2015. (Cited in page 9.)

[Echtler 2018] Florian Echtler and Maximilian Häußler. *Open source, open science, and the replication crisis in HCI.* In Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, pages 1–8, 2018. (Cited in page 44.)

[Endsley 1988] Mica R. Endsley. *Design and Evaluation for Situation Awareness Enhancement.* Proceedings of the Human Factors Society Annual Meeting, vol. 32, no. 2, pages 97–101, October 1988. (Cited in pages 7, 31, and 43.)

[Erol 1996] Kutluhan Erol, James Hendler and Dana S Nau. *Complexity results for HTN planning.* Annals of Mathematics and Artificial Intelligence, vol. 18, no. 1, pages 69–93, 1996. (Cited in page 13.)

[Ferrer 2013] G. Ferrer, A. Garrell and A. Sanfeliu. *Robot companion: A social-force based approach with human awareness-navigation in crowded environments.*

In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1688–1694, November 2013. (Cited in page 19.)

[Fisher 1993] Robert J Fisher. *Social desirability bias and the validity of indirect questioning.* Journal of consumer research, vol. 20, no. 2, pages 303–315, 1993. (Cited in page 42.)

[Fitts 1951] Paul M. Fitts. Human engineering for an effective air-navigation and traffic-control system. Human engineering for an effective air-navigation and traffic-control system. National Research Council, Div. of, Oxford, England, 1951. Pages: xxii, 84. (Cited in pages 1 and 7.)

[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 64.)

[Foster 2019] Mary Ellen Foster, Bart Craenen, Amol Deshmukh, Oliver Lemon, Emanuele Bastianelli, Christian Dondrup, Ioannis Papaioannou, Andrea Vanzo, Jean-Marc Odobez, Olivier Canévet*et al. Mummer: Socially intelligent human-robot interaction in public spaces.* arXiv preprint arXiv:1909.06749, 2019. (Cited in page 47.)

[Ghallab 2016] Malik Ghallab, Dana Nau and Paolo Traverso. Automated planning and acting. Cambridge University Press, 2016. (Cited in pages 5 and 13.)

[Gharbi 2015a] Mamoun Gharbi, Raphaël Lallement and Rachid Alami. *Combining symbolic and geometric planning to synthesize human-aware plans: toward more efficient combined search.* In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6360–6365. IEEE, 2015. (Cited in pages 14, 56, 87, and 104.)

[Gharbi 2015b] Mamoun Gharbi, Pierre-Vincent Paubel, Aurélie Clodic, Ophélie Carreras, Rachid Alami and Jean-Marie Cellier. *Toward a better understanding of the communication cues involved in a human-robot object transfer.* In 2015 24th IEEE international symposium on robot and human interactive communication (RO-MAN), pages 319–324. IEEE, 2015. (Cited in page 42.)

[Gibson 2014] James J Gibson. The ecological approach to visual perception: classic edition. Psychology Press, 2014. (Cited in page 69.)

[Goffman 1966] Erving Goffman. Behavior in Public Places: Notes on the Social Organization of Gatherings. Free Press, New York, NY, reissue edition edition, September 1966. (Cited in page 21.)

[Goodrich 2007] Michael A. Goodrich and Alan C. Schultz. *Human-Robot Interaction: A Survey.* Foundations and Trends® in Human-Computer Interaction, vol. 1, no. 3, pages 203–275, 2007. (Cited in page 6.)

[Greenwald 1998] Anthony G Greenwald, Debbie E McGhee and Jordan LK Schwartz. *Measuring individual differences in implicit cognition: the implicit association test.* Journal of personality and social psychology, vol. 74, no. 6, page 1464, 1998. (Cited in page 40.)

[Grice 1975] Herbert P Grice. *Logic and conversation.* In Speech acts, pages 41–58. Brill, 1975. (Cited in pages 58 and 142.)

[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 BNC@ ECAI 2012, 2012. (Cited in page 86.)

[Helbing 1995] D. Helbing and P. Molnár. *Social force model for pedestrian dynamics.* Physical Review E, pages 4282–4286, 1995. (Cited in page 19.)

[Ingrand 1996] François Félix Ingrand, Raja Chatila, Rachid Alami and Frédéric Robert. *PRS: A high level supervision and control language for autonomous mobile robots.* In Proceedings of IEEE International Conference on Robotics and Automation, volume 1, pages 43–49. IEEE, 1996. (Cited in page 102.)

[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 140.)

[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 140.)

[Khamassi 2016] Mehdi Khamassi, Benoît Girard, Aurélie Clodic, Sandra Devin, Erwan Renaudo, Elisabeth Pacherie, Rachid Alami and Raja Chatila. *Integration of action, joint action and learning in robot cognitive architectures.* Intellectica-La revue de l'Association pour la Recherche sur les sciences de la Cognition (ARCo), vol. 2016, no. 65, pages 169–203, 2016. (Cited in page 10.)

[Khambhaita 2016] Harmish Khambhaita, Jorge Rios-Martinez and Rachid Alami. *Head-Body Motion Coordination for Human Aware Robot Navigation.* In 9th International workshop on Human-Friendlly Robotics (HFR 2016), page 8p, 2016. (Cited in pages 21, 38, 41, and 45.)

[Khambhaita 2017] Harmish Khambhaita and Rachid Alami. *Viewing Robot Navigation in Human Environment as a Cooperative Activity.* In International Symposium on Robotics Research (ISSR 2017), page 18p., Puerto Varas, Chile, December 2017. (Cited in pages 15, 20, 22, 25, 26, and 27.)

[Knoblich 2011] Günther Knoblich, Stephen Butterfill and Natalie Sebanz. *Psychological research on joint action: theory and data.* Psychology of learning and motivation, vol. 54, pages 59–101, 2011. (Cited in page 10.)

[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 70.)

[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 page 59.)

[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 58 and 59.)

[Kruse 2012] T. Kruse, P. Basili, S. Glasauer and A. Kirsch. *Legible robot navigation in the proximity of moving humans.* In IEEE Workshop on Advanced Robotics and its Social Impacts, pages 83–88, May 2012. (Cited in page 20.)

[Kruse 2013] Thibault Kruse, Amit Kumar Pandey, Rachid Alami and Alexandra Kirsch. *Human-aware robot navigation: A survey.* Robotics and Autonomous Systems, vol. 61, no. 12, pages 1726–1743, December 2013. (Cited in pages 9 and 19.)

[Kuderer 2012] Markus Kuderer, Henrik Kretzschmar, Christoph Sprunk and Wolfram Burgard. *Feature-Based Prediction of Trajectories for Socially Compliant Navigation.* In Proceedings of Robotics: Science and Systems, Sydney, Australia, July 2012. (Cited in page 20.)

[Kuhnert 2017] Barbara Kuhnert, Marco Ragni and Felix Lindner. *The gap between human's attitude towards robots in general and human's expectation of an ideal everyday life robot.* In 2017 26th IEEE international symposium on robot and human interactive communication (RO-MAN), pages 1102–1107. IEEE, 2017. (Cited in page 40.)

[Lallemand 2015] C. Lallemand, V. Koenig, G. Gronier and R. Martin. *Création et validation d'une version française du questionnaire AttrakDiff pour l'évaluation de l'expérience utilisateur des systèmes interactifs.* Revue Européenne de Psychologie Appliquée/European Review of Applied Psychology, vol. 65, no. 5, pages 239–252, September 2015. (Cited in pages 31, 42, and 164.)

[Lallement 2014] Raphaël Lallement, Lavindra De Silva and Rachid Alami. *HATP: An HTN planner for robotics.* In 2nd ICAPS Workshop on Planning and Robotics, 2014. (Cited in pages 13, 85, 86, and 100.)

[Lazar 2017] Jonathan Lazar, Jinjuan Heidi Feng and Harry Hochheiser. Research methods in human-computer interaction. Morgan Kaufmann, 2017. (Cited in page 40.)

[Lemaignan 2012] Séverin Lemaignan, Raquel Ros, E Akin Sisbot, Rachid Alami and Michael Beetz. *Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction.* International Journal of Social Robotics, vol. 4, no. 2, pages 181–199, 2012. (Cited in page 60.)

[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 pages 11, 14, and 142.)

[Lemaignan 2018] Séverin Lemaignan, Yoan Sallami, Christopher Wallhridge, Aurélic Clodic, Tony Belpaeme and Rachid Alami. *Underworlds: cascading situation assessment for robots.* In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 7750–7757. IEEE, 2018. (Cited in pages 11, 102, and 144.)

[Levine 2014] Steven Levine and Brian Williams. *Concurrent plan recognition and execution for human-robot teams.* In Proceedings of the international conference on automated planning and scheduling, volume 24, 2014. (Cited in page 103.)

[Li 2017] Shen Li. *Automatically evaluating and generating clear robot explanations.* Master's thesis, 2017. (Cited in pages 59 and 81.)

[Lu 2013] David V. Lu and William D. Smart. *Towards more efficient navigation for robots and humans.* In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1707–1713, Tokyo, November 2013. IEEE. (Cited in pages 19, 21, and 39.)

[Mainprice 2012] Jim Mainprice, Mamoun Gharbi, Thierry Siméon and Rachid Alami. *Sharing effort in planning human-robot handover tasks.* In 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, pages 764–770. IEEE, 2012. (Cited in page 15.)

[Martinie 2019] Célia Martinie, Philippe Palanque, Elodie Bouzekri, Andy Cockburn, Alexandre Canny and Eric Barboni. *Analysing and demonstrating tool-supported customizable task notations.* Proceedings of the ACM on Human-Computer Interaction, vol. 3, no. EICS, pages 1–26, 2019. (Cited in page 13.)

[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 pages 54 and 60.)

[May 2015] A. D. May, C. Dondrup and M. Hanheide. *Show me your moves! Conveying navigation intention of a mobile robot to humans.* In 2015 European Conference on Mobile Robots (ECMR), pages 1–6, September 2015. (Cited in pages 21 and 38.)

[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 2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 291–298. IEEE, 2020. (Cited in pages 43, 50, and 51.)

[Miller 1968] Robert B Miller. *Response time in man-computer conversational transactions.* In Proceedings of the December 9-11, 1968, fall joint computer conference, part I, pages 267–277, 1968. (Cited in page 79.)

[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 The 23rd IEEE international symposium on robot and human interactive communication, pages 1103–1109. IEEE, 2014. (Cited in pages 11, 83, 92, and 102.)

[Mutlu 2006] B. Mutlu, J. Forlizzi and J. Hodgins. *A Storytelling Robot: Modeling and Evaluation of Human-like Gaze Behavior.* In 2006 6th IEEE-RAS International Conference on Humanoid Robots, pages 518–523, December 2006. (Cited in page 21.)

[Nardi 2014] Lorenzo Nardi and Luca Iocchi. *Representation and execution of social plans through human-robot collaboration.* In International Conference on Social Robotics, pages 266–275. Springer, 2014. (Cited in page 14.)

[Naveau 2017] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur and P. Souères. *A Reactive Walking Pattern Generator Based on Nonlinear Model Predictive Control.* IEEE Robotics and Automation Letters, vol. 2, no. 1, pages 10–17, 2017. (Cited in page 46.)

[Norman 2013] Don Norman. The design of everyday things: Revised and expanded edition. Basic books, 2013. (Cited in pages 7, 8, and 69.)

[Pacchierotti 2006] Elena Pacchierotti, Henrik I. Christensen and Patric Jensfelt. *Evaluation of Passing Distance for Social Robots.* In ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication, pages 315–320, September 2006. ISSN: 1944-9437. (Cited in page 20.)

[Pacherie 2011] Elisabeth Pacherie. *14 The Phenomenology of Joint Action: Self-Agency versus Joint Agency.* Joint attention: New developments in psychology, philosophy of mind, and social neuroscience, page 343, 2011. (Cited in pages 10 and 22.)

[Paternò 2004] Fabio Paternò. *ConcurTaskTrees: an engineered notation for task models.* The handbook of task analysis for human-computer interaction, pages 483–503, 2004. (Cited in page 13.)

[Petrick 2013] Ronald Petrick and Mary Ellen Foster. *Planning for social interaction in a robot bartender domain.* In Proceedings of the International Conference on Automated Planning and Scheduling, volume 23, 2013. (Cited in page 14.)

[Posner 1978] Michael I Posner. Chronometric explorations of mind. Lawrence Erlbaum, 1978. (Cited in page 43.)

[Reiter 1997] Ehud Reiter and Robert Dale. *Building applied natural language generation systems.* Natural Language Engineering, vol. 3, no. 1, pages 57–87, 1997. (Cited in page 58.)

[Rios-Martinez 2015] Jorge Rios-Martinez, Anne Spalanzani and Christian Laugier. *From proxemics theory to socially-aware navigation: A survey.* International Journal of Social Robotics, vol. 7, no. 2, pages 137–153, 2015. (Cited in page 9.)

[Roncone 2017] Alessandro Roncone, Olivier Mangin and Brian Scassellati. *Transparent role assignment and task allocation in human robot collaboration.* In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 1014–1021. IEEE, 2017. (Cited in page 61.)

[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, pages 570–575. IEEE, 2010. (Cited in pages 60 and 80.)

[Rosmann 2013] Christoph Rosmann, Wendelin Feiten, Thomas Wosch, Frank Hoffmann and Torsten Bertram. *Efficient trajectory optimization using a sparse model.* In 2013 European Conference on Mobile Robots, pages 138–143, Barcelona, Catalonia, Spain, September 2013. IEEE. (Cited in pages 23 and 24.)

[Salmon 2006] Paul Salmon, Neville Stanton, Guy Walker and Damian Green. *Situation awareness measurement: A review of applicability for C4i environments.* Applied ergonomics, vol. 37, no. 2, pages 225–238, 2006. (Cited in page 43.)

[Sanelli 2017] Valerio Sanelli, Michael Cashmore, Daniele Magazzeni and Luca Iocchi. *Short-term human-robot interaction through conditional planning and execution.* In Proceedings of the International Conference on Automated Planning and Scheduling, volume 27, 2017. (Cited in page 14.)

[Saraydaryan 2015] Jacques Saraydaryan, Fabrice Jumel and Olivier Simonin. *Robots delivering services to moving people: Individual vs. group patrolling strategies.* In 2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO), pages 1–8. IEEE, 2015. (Cited in page 15.)

[Sarthou 2019] Guillaume Sarthou, Aurélie Clodic and Rachid Alami. *Ontologenius: A long-term semantic memory for robotic agents.* In 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 1–8. IEEE, 2019. (Cited in pages 65, 82, 137, and 142.)

[Scalise 2018] Rosario Scalise, Shen Li, Henny Admoni, Stephanie Rosenthal and Siddhartha S Srinivasa. *Natural language instructions for human–robot collaborative manipulation.* The International Journal of Robotics Research, vol. 37, no. 6, pages 558–565, 2018. (Cited in page 81.)

[Sebanz 2006] Natalie Sebanz, Harold Bekkering and Günther Knoblich. *Joint action: bodies and minds moving together.* Trends in cognitive sciences, vol. 10, no. 2, pages 70–76, 2006. (Cited in page 10.)

[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, volume 27, 2017. (Cited in pages 60 and 100.)

[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 Proceedings of the 6th international conference on Human-robot interaction, pages 29–36, 2011. (Cited in page 60.)

[Singamaneni 2020] Phani-Teja Singamaneni and Rachid Alami. *HATEB-2: Reactive Planning and Decision making in Human-Robot Co-navigation.* In International Conference on Robot & Human Interactive Communication, 2020, 2020. (Cited in pages 46 and 49.)

[Sisbot 2007] Emrah Akin Sisbot, Luis F. Marin-Urias, Rachid Alami and Thierry Siméon. *A Human Aware Mobile Robot Motion Planner.* Transactions on Robotics, vol. 23, no. 5, pages 874–883, 2007. (Cited in pages 9 and 19.)

[Stasse 2008] Olivier Stasse, Bjorn Verrelst, Pierre-Brice Wieber, Bram Vanderborght, Paul Evrard, Abderrahmane Kheddar and Kazuhito Yokoi. *Modular Architecture for Humanoid Walking Pattern Prototyping and Experiments.* Advanced Robotics, vol. 22, 06 2008. (Cited in page 45.)

[Tellex 2014] Stefanie Tellex, Ross Knepper, Adrian Li, Daniela Rus and Nicholas Roy. *Asking for help using inverse semantics.* 2014. (Cited in page 60.)

[Tomasello 2005] Michael Tomasello, Malinda Carpenter, Josep Call, Tanya Behne and Henrike Moll. *Understanding and sharing intentions: The origins of cultural cognition.* Behavioral and brain sciences, vol. 28, no. 5, pages 675–691, 2005. (Cited in page 10.)

[Tsang 2017] Siny Tsang, Colin F Royse and Abdullah Sulieman Terkawi. *Guidelines for developing, translating, and validating a questionnaire in perioperative and pain medicine.* Saudi journal of anaesthesia, vol. 11, no. Suppl 1, page S80, 2017. (Cited in page 42.)

[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 62.)

[Unhelkar 2020] Vaibhav V Unhelkar, Shen Li and Julie A Shah. *Decision-making for bidirectional communication in sequential human-robot collaborative tasks.* In Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction, pages 329–341, 2020. (Cited in pages 14, 61, and 88.)

[Vesper 2010] Cordula Vesper, Stephen Butterfill, Günther Knoblich and Natalie Sebanz. *A minimal architecture for joint action.* Neural Networks, vol. 23, no. 8-9, pages 998–1003, October 2010. (Cited in pages 10 and 20.)

[Viethen 2013] Jette Viethen, Margaret Mitchell and Emiel Krahmer. *Graphs and spatial relations in the generation of referring expressions.* In Proceedings of the 14th European Workshop on Natural Language Generation, pages 72–81, 2013. (Cited in pages 59, 80, and 81.)

[Waldhart 2015] Jules Waldhart, Mamoun Gharbi and Rachid Alami. *Planning handovers involving humans and robots in constrained environment.* In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6473–6478. IEEE, 2015. (Cited in page 15.)

[Waldhart 2019] J. Waldhart, A. Clodic and R. Alami. *Reasoning on Shared Visual Perspective to Improve Route Directions.* In 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pages 1–8, 2019. (Cited in pages 15 and 47.)

[Williams 2017] Tom Williams and Matthias Scheutz. *Referring expression generation under uncertainty: Algorithm and evaluation framework.* In Proceedings of the 10th International Conference on Natural Language Generation, pages 75–84, 2017. (Cited in page 60.)

[Williams 2019] Tom Williams, Fereshta Yazdani, Prasanth Suresh, Matthias Scheutz and Michael Beetz. *Dempster-shafer theoretic resolution of refer-*

ential ambiguity. Autonomous Robots, vol. 43, no. 2, pages 389–414, 2019. (Cited in page 60.)

[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 page 59.)

[Zaraki 2014] A. Zaraki, D. Mazzei, M. Giuliani and D. De Rossi. *Designing and Evaluating a Social Gaze-Control System for a Humanoid Robot.* IEEE Transactions on Human-Machine Systems, vol. 44, no. 2, pages 157–168, April 2014. (Cited in page 21.)

Abstract: Robots are capable to autonomously handle more and more complex tasks. However, today's robots either have their workspaces physically separated from the humans' ones or their abilities severely restricted when acting near humans. In this thesis, we investigate several approaches allowing to plan not only for the robot but also for the human, enabling to predict and elicit their decision process and actions, leading to better human-robot interactions (HRI).

First, we show through a user study, why such a scheme applied to robot navigation is crucial for efficient and satisfactory interaction. Planning for the robot and the human allows to find solutions in intricate situations where collaboration is necessary but also for the robot to be proactive and legible while navigating. Secondly, we alleviate from inherent ephemeral nature of interaction in collaborative navigation to explore how co-planning can be applied for task planning. We introduce a new referring expression generation algorithm, using an ontology as a knowledge base, and show that it is the fastest one to date while being designed for HRI application. We use it in a human-aware task planner to estimate the feasibility and cost of communication during task planning, preventing deadlock or suboptimal plans. Finally, a novel approach to human-aware task planning is proposed where action models and decision streams of the robot and the human are distinct and used to produce conditional plans.

Résumé : Les robots sont aujourd'hui capables d'effectuer de manière autonome des tâches toujours plus complexes. Cependant, ils sont encore soit utilisés avec une séparation physique des humains, soit limités lorsqu'ils évoluent au voisinage immédiat d'humains. Dans cette thèse, nous proposons plusieurs approches visant à planifier pour le robot mais aussi pour l'humain. Les plans alors produits prennent en compte l'humain en prédisant et encourageant leurs actions, menant à de meilleures interactions humains-robots (HRI).

Premièrement, nous montrons à travers une étude utilisateur, dans quelle mesure une telle approche appliquée à la navigation est cruciale pour une interaction efficiente et satisfaisante. Planifier pour le robot et l'humain permet en effet de trouver des solutions dans des situations d'interaction complexes mais aussi au robot d'avoir un comportement plus lisible et proactif. Deuxièmement, nous présentons un algorithme de génération d'expression de référence, rapide et pensé pour la HRI. Nous utilisons ensuite cet algorithme pour estimer la faisabilité et le coût des actions de communication pendant la planification de tâches, permettant d'éviter des impasses et des plans sous optimaux. Enfin, nous proposons une approche novatrice à la planification de tâches humain robot, dans laquelle les modèles de l'action et décision des deux agents sont distincts et utilisés pour produire des plans conditionnels.

Keywords: human-robot interaction, navigation, task planning

Mots clés : interaction humain-robot, navigation, planification de tâches