



Teaching Robots Behaviors Using Spoken Language in Rich and Open Scenarios

Victor Paléologue

► To cite this version:

Victor Paléologue. Teaching Robots Behaviors Using Spoken Language in Rich and Open Scenarios. Artificial Intelligence [cs.AI]. Sorbonne Université, 2019. English. NNT: 2019SORUS458 . tel-03011393v2

HAL Id: tel-03011393

<https://theses.hal.science/tel-03011393v2>

Submitted on 18 Nov 2020

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

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

Thèse de Doctorat de Sorbonne Université

Spécialité : Sciences Mécaniques, Acoustique, Électronique et
Robotique de Paris (SMAER)

Présentée par : Victor Paléologue

Pour obtenir le grade de
Docteur de Sorbonne Université



Teaching Robots Behaviors Using Spoken Language in Rich and Open Scenarios

<i>Rapporteurs</i>	Gérard BAILLY	- GIPSA-lab - Université de Grenoble-Alpes
	Tony BELPAEME	- IDLAB - Ghent University
<i>Examineurs</i>	Raja CHATILA	- ISIR - Sorbonne Université
	Peter Ford DOMINEY	- INSERM - Université de Bourgogne
<i>Invité</i>	Alexandre CONINX	- ISIR - Sorbonne Université
<i>Directeur</i>	Mohamed CHETOUANI	- ISIR - Sorbonne Université
<i>Co-encadrant</i>	Amit Kumar PANDEY	- SoftBank Robotics



Abstract

Social robots like Pepper are already found "in the wild". Their behaviors must be adapted for each use case by experts. Enabling the general public to teach new behaviors to robots may lead to better adaptation at lesser cost. In this thesis, we study a cognitive system and a set of robotic behaviors allowing home users of Pepper robots to teach new behaviors as a composition of existing behaviors, using solely the spoken language.

Homes are open worlds and are unpredictable. In open scenarios, a home social robot should learn about its environment. The purpose of such a robot is not restricted to learning new behaviors or about the environment: it should provide entertainment or utility, and therefore support rich scenarios. We demonstrate the teaching of behaviors in these unique conditions: the teaching is achieved by the spoken language on Pepper robots deployed in homes, with no extra device and using its standard system, in a rich and open scenario.

Using automatic speech transcription and natural language processing, our system recognizes unpredicted teachings of new behaviors, and a explicit requests to perform them. The new behaviors may invoke existing behaviors parametrized with objects learned in other contexts, and may be defined as parametric. Through experiments of growing complexity, we show conflicts between behaviors in rich scenarios, and propose a solution based on symbolic task planning and prioritization rules to resolve them.

The results rely on qualitative and quantitative analysis and highlight the limitations of our solution, but also the new applications it enables.

Acknowledgements

This thesis is the fruit of 3 years of intense labor with Professor Mohamed Chetouani at Sorbonne Université and Doctor Amit Kumar Pandey from SoftBank Robotics Europe (SBRE). After 6 years as an employee of SBRE, I seized the opportunity to start doctorate studies and proposed this subject, inspired by the challenges I encountered so far in the industry of social robots. I am grateful that SBRE allowed this to happen, supported by software team’s Taylor Veltrop, Laurent Lec and Nino Sapina, innovation team’s Rodolphe Gelin, Amit Kumar Pandey and Petra Koudelkova, and by the direction’s Tamara Carpentier and Bruno Maisonnier.

I thank my colleagues, especially Jocelyn Martin for his amazing software for natural language understanding, and Shin Watanabe, Jérôme Bruzard and my friends of the “Framework” team. I do not forget my doctoral studies fellows, who worked or are still working in Professor Chetouani’s team and with whom I have shared exciting discussions about robotics, machine learning, academic research, and the future.

This doctorate has been funded by SoftBank Robotics Europe, initially as part of the project ROMEO2, and is also supported by the Institut des Systèmes Intelligents et Robotiques (ISIR, UMR 7222), laboratory of the CNRS, as part of the SMART Labex (ANR-11-LABX-65).

Finally, I thank my parents and my siblings for always encouraging me pursuing these studies, and Marie-José Loverini, who is also part of the family. But the person I am the most thankful to is my partner in life, Justine Lança, for accompanying me in this overwhelming challenge and for giving me the strength everyday to get through.

Contents

List of Figures	x
List of Tables	xii
Glossary	xiii
Acronyms	xix
1 Introduction	1
1.1 Motivations: Adapting Social Robots in the Wild	2
1.1.1 Real Software Applications for Social Robots	2
1.1.2 Development of Applications for Social Robots	2
1.1.3 Adapting Social Robots in Real Conditions	3
1.1.4 Challenges of Adapting Social Robots in Real Conditions . .	4
1.2 Positioning in the Research Landscape	5
1.2.1 Artificial Intelligence	5
1.2.2 Interactive Task Learning	6
1.2.3 Learning from Demonstration	6
1.2.4 Interactive Robot Learning	7
1.2.5 Human-Robot Interaction	8
1.2.6 Positioning	9
1.3 Definitions	9
1.4 Objectives: Teaching Behaviors using Spoken Language in Real Con- ditions	10
1.5 Contributions	11
1.5.1 A State of the Art	11
1.5.2 A Proof-of-Concept in Homes	11
1.5.3 Teaching behaviors in Rich Scenarios in Homes	12
1.6 Conclusion	12
2 State of the Art on Teaching Behaviors using Spoken Language	15
2.1 How Humans Teach Behaviors to Humans	16
2.1.1 Common Characteristics of Teaching Interactions	17
2.1.2 Explicit Teaching of Behaviors using Instructions	18
2.2 How Humans Teach Behaviors to Robots	19
2.2.1 Theoretical Specificities	19
2.2.2 Teaching Behaviors to Robots using Spoken Language In- structions	22
2.2.3 Conclusions on How Humans Teach Robots	25
2.3 Natural Language Interaction	26

2.3.1	Tuning Speech Recognition <i>a Priori</i>	27
2.3.2	Performing Natural Language Understanding <i>a Posteriori</i> . .	28
2.3.3	Natural Language Generation	29
2.3.4	Application in Cognitive Systems	30
2.3.5	Conclusions on Natural Language Interaction	31
2.4	Behavior Models and Learning Techniques	31
2.4.1	Review of Behavior Models and Learning Techniques	32
2.4.2	Choosing a Behavior Model	35
2.4.3	Conclusions on Behavior Models	37
2.5	Experimental Setups and Evaluation	38
2.5.1	Robotic Embodiment	38
2.5.2	Use of Microphones	39
2.5.3	Openness of Experimental Scenarios	40
2.5.4	Evaluation	41
2.6	Conclusion	42
3	Behavior Composition for a Social Robot in an Open Scenario	45
3.1	Extraction of Social Cues with Pepper	48
3.1.1	Awareness of Humans	48
3.1.2	Engagement Assessment	49
3.2	Natural Language Understanding from Speech-to-Text Output . . .	49
3.2.1	Understanding the Natural Language	49
3.2.2	Semantic Expressions	51
3.2.3	Semantic Memory: a Language-Oriented Database	53
3.2.4	Natural Language Generation from Semantic Expressions . .	54
3.3	Interaction for Teaching Behaviors	55
3.3.1	Extracting Task Teachings	55
3.3.2	Formal Model of Taught Behaviors	56
3.3.3	Interaction Scenario and Patterns for Teaching	59
3.4	Selecting Interactive Behaviors for Teaching Tasks	62
3.4.1	Using a Static Behavior Priority List	62
3.4.2	Using Independent Action Suggestions and Interaction Rules	63
3.5	Proof-of-concepts and Experiment in Homes	65
3.5.1	Preliminary Experiment	66
3.5.2	Simple Task Composition with Manual Transcription	66
3.5.3	Simple Task Composition with Automatic Speech Recognition	69
3.5.4	Task Composition in Home Conditions	74
3.6	Conclusion	77
4	Teaching Behaviors in Rich Scenarios	79
4.1	Proposed Behaviors for a Richer Interaction	80
4.1.1	Discover a Point of Interest	82
4.1.2	Label a Point of Interest and a Location	83
4.1.3	Point at a Point of Interest	83

4.1.4	Locate a Point of Interest	83
4.1.5	Visit a Location	84
4.2	Proposed Ontology for Interoperability	84
4.2.1	Social Agents	85
4.2.2	Physical Objects	85
4.2.3	Locations, Areas and Maps	87
4.2.4	Events	87
4.2.5	Communicative Acts	89
4.2.6	Actions	89
4.2.7	Semantic Templates	90
4.2.8	Conclusion on Sharing Knowledge and Behaviors	91
4.3	Teaching Parametrized Behaviors	91
4.3.1	Extracting Parametrized Task Teachings	91
4.3.2	Formal Model for Parametric Behaviors	93
4.3.3	Interaction Patterns for Parametric Behaviors	94
4.4	Rule-Based Task Selection	95
4.4.1	Goal-Oriented Approach and Planning	96
4.4.2	Building Problems from Task Suggestions	97
4.4.3	Rules for HRI Tasks	100
4.4.4	Conclusion on Rule-Based Task Selection	102
4.5	Experiment	102
4.5.1	Experimental Protocol	102
4.5.2	Deployment in Pepper@Home	105
4.5.3	Results and Analysis	106
4.5.4	Conclusion on the Experiment	115
4.6	Conclusion	116
5	Conclusions and Perspectives	119
5.1	Conclusions and Take-Aways	120
5.2	Generalization on Behaviors	123
5.2.1	Generalization of Implementation	123
5.2.2	Behaviors Everywhere	123
5.2.3	Reasoning on Behaviors	124
5.3	Interaction Repair and Meta-Interaction	125
5.3.1	Interaction Repair	126
5.3.2	Interaction for Defining Goals and Rules	127
5.3.3	Meta-Interaction for Learning to Achieve Goals	127
5.4	Detecting Mistakes and Learning from Them	128
5.4.1	Detecting, Repairing and Learning from User's Mistakes	128
5.4.2	Adapting from Robot's Mistakes and User Corrections	129
5.4.3	Autonomously Making Mistakes for Learning	129
5.5	Next Behaviors and Potential Challenges	130
5.5.1	Correcting Behavior Compositions	130
5.5.2	Teaching Animation Actions	130

5.5.3	Autonomous Recharge	130
5.5.4	Human Greetings and Identification	131
5.5.5	Hush Rule	131
5.5.6	Behavior Modification Evaluation	131
Annex A: NAOqi		133
Annex B: Review of Cognitive Systems		137
Bibliography		155

List of Figures

1.1	Adaptation of “A simplified illustration of the Learning from Demonstration pipeline.”	7
3.1	Overall information flow within the cognitive system. NAOqi is the interface with the environment, see annex 5.5.6. STT, NLU, NLG and TTS are introduced in section 2.3. Behaviors implementations produce the reasoning of the system: they react processed inputs (events and semantic expressions) may exchange data with databases (semantic memory and symbolic knowledge) and produce NAOqi actions in response, that may be run if selected.	47
3.2	Intermediate and final results of the NLU. Final result is a semantic expression. Adapted from [Paléologue et al., 2017], figure 2.	52
3.3	Semantic expression for a sample task teaching: “to make a diabolo is to put syrup and then to put sparkling water“.	57
3.4	Overview of the software architecture with behaviors ordered by priority.	63
3.5	Overview of the software architecture with behaviors suggesting tasks in reaction.	64
3.6	Instruction sheets given to participants before starting the experiment. Red: home; green: health care; blue: business.	67
4.1	Extract of the taxonomy of the DOLCE basic categories presented in [Gangemi et al., 2002]. Entity is the root of the taxonomy. We borrowed the categories: <i>Social Agent</i> , <i>Physical Object</i> , <i>Space Region</i> and <i>Event</i>	86
4.2	Template of knowledge graph produced by the exploration behavior when discovering an object. White nodes are resources, green nodes are localized strings, red nodes are datetime data, grey nodes are resources private to that application. Figure by Shin Watanabe, employee of SBRE.	88
4.3	Template of knowledge graph produced by dialogues. Communication act <code>event:x</code> was addressed to the robot (<code>agent:self</code>). The robot performs the communication act <code>event:y</code> , as a response to <code>event:x</code>	89
4.4	PDDL description of the “visit” action.	98
4.5	C++ code for sorting all possible combinations of suggested goals, with respect to an ordered list of priorities.	99
4.6	Screenshot of the phase 2.a). It runs a regular chat, and displays recommendations about what the users can say to the robot.	103
4.7	Instructions presented to participants in the phase 2.b), to explain how to teach behaviors to the robot.	104

4.8	Instructions presented to participants in the phase 3, to explain how to teach behaviors to the robot, while letting baseline chat suggestions visible.	105
5.1	Unified model omitting the distinction between behaviors and actions. Active behaviors (blue boxes) may react by creating new behaviors, and goal suggestions. Goal suggestions are prioritized using the rule system. The action selection produces the tasks to perform: a selection of behaviors to activate for next iteration.	124
2	Communication of a RPC from a remote client to a NAOqi service. Black boxes right under the NAOqi box are daemons. Black boxes emerging from daemons are services. Blue lines represent the registration links. Green arrows represent the RPC calls and their response propagated on the network. Green dotted boxes represent a proxy to a remote object. Here a client gets the service "ALTextToSpeech" provided by the daemon "audio" using a "service" RPC. Then calls "say" on the service.	134
3	Communication of a RPC from a remote client to a Qi Object provided by NAOqi service. Black boxes right under the NAOqi box are daemons. Black boxes emerging from daemons are services. Blue lines represent the registration links. Green arrows represent the RPC calls and their response propagated on the network. Green boxes represent a non-service Qi Object. Green dotted boxes represent a proxy to a remote object. Here a client gets the service "Conversation" provided by the daemon "dialog" using a "service" RPC. Then calls "makeSay" on the service to get a "Say" object. Then calls "run" on the "Say" object.	136
4	The canonical architecture of a cognitive system.	138

List of Tables

2.1	“Basic common pragmatic frames for the category: passive learning.“	21
2.2	Some properties exhibited by the behavior models and learning techniques used in the teaching of behaviors using spoken language. . .	36
2.3	Publications exhibiting teaching behaviors to robots using natural language. They may not necessarily exhibit HRI or spoken language. [Paléologue et al., 2018] is the first to meet all the studied characteristics.	44
3.1	Usual pragmatic frame involved in our teaching of tasks.	61
3.2	Pragmatic frame for asking whether the task description is complete.	61
3.3	Measurements on the transcripts per theme, on the experiment presented in [Paléologue et al., 2017].	68
3.4	Contingency of NLU results by theme, for teaching behaviors with perfect STT. G stands for successful understanding, S for semantic analysis error, U for not understood by the behavior. Independence test produces $\chi^2 = 13.4$ and $p = 0.146$, which is inconclusive.	69
3.5	Measurements on the transcripts per theme (<i>a.k.a.</i> domain) on the experiment presented in [Paléologue et al., 2018]. The measure <i>Mis%</i> includes the speech recognition errors, when the text produced differed from the utterance. At the time it was considered different from <i>Err</i> , defined in [Gemignani et al., 2015] as the absence of recognition.	70
3.6	Comparison of the error measurements between the experiments presented in [Paléologue et al., 2018]. N-1 is [Paléologue et al., 2017], N is [Paléologue et al., 2018]. <i>MisA%</i> is the rate of misrecognition error due to the speech recognition. <i>MisS%</i> is the rate of misrecognition error due to the semantic analysis. <i>MisU%</i> is the rate of misrecognition error due to the proper exploitation of the semantic structure.	71
3.7	Contingency of misrecognized (Mis) user utterances by experiment. [Paléologue et al., 2017] is the previous experiment, described in subsection 3.5.2. Independence test produces $\chi^2 = 22.3$ and $p = 1.76 \times 10^{-4}$. Misrecognition rate has increased in experiment #3. . .	71
3.8	Contingency of user utterances misrecognized by NLU, by experiment. [Paléologue et al., 2017] is the previous experiment, described in subsection 3.5.2. Independence test produces $\chi^2 = 2.57$ and $p = 0.632$, which is inconclusive.	72

3.9	Contingency of NLU results by theme, for teaching behaviors with real STT. G stands for successful understanding, A stands for speech recognition error, S for semantic analysis error, U for not understood by the behavior. Independence test produces $\chi^2 = 35.0$ and $p = 4.65 \times 10^{-4}$. NLU results are not independent from the theme. . . .	72
3.10	Result comparisons between [Paléologue et al., 2018] and [Gemignani et al., 2015].	73
3.11	Contingency of speech recognition errors by experiment. Independence test produces $\chi^2 = 41.1$ and $p = 2.30 \times 10^{-8}$	73
3.12	Contingency of NLU by experiment. Independence test produces $\chi^2 = 2.87$ and $p = 0.579$	74
4.1	Usual pragmatic frame involved when asking if an object is interesting, its label, or the label of a location.	95
4.2	Measurements on the collected transcripts, grouped by phase. . . .	108
4.3	Contingency of NLU errors between the the “Regular” and the “Mixed” phases. Independence test produces $\chi^2 = 12.1$ and $p = 1.70 \times 10^{-2}$	108
4.4	Result comparisons between [Paléologue et al., 2018] and [Gemignani et al., 2015].	113
4.5	Rate of successfully taught behaviors ($TS\%$) over attempted teachings, by session.	114
5.1	Recurring pragmatic frame for a robot exploring. The robot probes by looking somewhere for points of interest. The result is provided by the perception system, and then the algorithm confirms by switching the behavior of the robot.	127

Glossary

Action A process through which an agent alters its environment, including itself. ix, x, xiii, 3, 5, 6, 9, 12, 17, 19, 21–24, 26, 31–34, 37, 47, 56, 80–82, 89–92, 96–98, 100, 101, 107, 111, 116, 117, 120, 121, 123, 124, 130, 131, 139–142

Action Primitive An action that cannot be decomposed into sub-actions, but accepts parameters defined in a continuous space. [Skills](#) usually represent action primitives. 56

Agent An individual capable of performing actions. 5, 6, 9

AIBO A small dog-shaped robot designed for home owners, manufactured by Sony. 39

Behavior A behavior is the observable action of an agent given a situation. ix–xii, 3–13, 16–19, 21–27, 29–44, 46–49, 54, 55, 58–66, 68, 69, 72, 74–78, 80–85, 87–98, 100–117, 120–132, 135, 137–140, 142

Behavior Model A behavior model is a formalism to represent behaviors. For example, describing behaviors as lists of elementary action is a behavior model. Describing them as interpolated trajectory curves is another behavior model. 16, 31–35, 55, 58, 77, 91, 102, 139, 142

Behaviorism [...] a systematic approach to understanding the behavior of humans and other animals. It assumes that all behaviors are either reflexes produced by a response to certain stimuli in the environment, or a consequence of that individual's history, including especially reinforcement and punishment, together with the individual's current motivational state and controlling stimuli.

— [Wikipedia contributors, 2019a] xiii, 5, 19

Cognitive System A system that can autonomously collect information from its environment and act back on it through [actions](#). A cognitive system implements an agent, but an agent could be implemented otherwise. ix, 5, 9, 11, 35, 46, 47, 77, 137, 142

Cognitivism An approach to understanding the behavior of humans and other animals that include a model of how they think. In opposition to [behaviorism](#), it accepts that cognitive abilities may develop within subjects. Such abilities are not simply the reinforcement of existing abilities. 5, 30, 137

Composability Composability is a system design principle that deals with the inter-relationships of components. A highly composable system provides components that can be selected and assembled in various combinations to satisfy

specific user requirements.

— [Wikipedia contributors, 2019b] 4, 12, 32, 35, 38

Composition Organization exhibiting the composite pattern: a composite entity (*e.g.* a task) aggregates other equivalent entities, and may be part of equivalent aggregations. 9, 10, 55

Connectionist Knowledge Coined in [Harnad, 1990], opposes *symbolic knowledge* by the fact that information can be stored in a system without being able to refer to it symbolically. Neural networks are a typical example, where the knowledge underlying their effects is spread in the very organization of the networks. This is often referred to a “black box” effect. 7, 30, 31, 141

Dialogue Act A *speech act* conveying information of or the structure of the dialogue. 26

Goal A goal is the statement of a desired condition of an artificial agent. ix, x, 8, 12, 16, 17, 20, 21, 31, 34, 37, 93, 95–100, 102, 116, 121, 123–125, 127, 137, 140, 141

iCub A medium humanoid multi-purpose robot, manufactured by the Istituto Italiano di Tecnologia (IIT). xvii, 138

Instruction A natural language phrase that describes an action to perform. 6, 9, 10, 16, 18, 19, 22, 23, 25, 32

Knowledge Interoperability The property of some knowledge to be interpretable by a variety of systems that may not know about each other. 4, 5

Mixed-Initiative Interaction Interaction that is not driven by a single participant, but alternatively by all of them. 17, 18, 23

NAO A small humanoid robot designed for social interaction, manufactured by SoftBank Robotics Europe. xiv, 29, 39, 138

NAOqi Proprietary middleware distributed with and running on NAO and Pepper robots. “Qi” (from Chinese “氣”, giving “気” in Japanese) means “life force”. ix, x, xv, 10, 27, 46–49, 59, 65, 76, 84, 105, 122, 133–136, 138

Natural Language A language commonly used by humans to communicate with each other. xi, xx, 6, 9, 11, 16, 18, 26, 44

Ontology A set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). — [Gruber, 2009] 5, 84

- Open Scenario** A scenario in which the situations were not predicted, nor the outcome specifically intended. 5, 9, 12, 42, 43, 46, 58, 66, 68, 74, 78, 91, 102, 115
- Pepper** A medium humanoid robot designed for social interaction, manufactured by SoftBank Robotics Europe. xiv, 2, 10, 29, 46, 75, 138
- Perspective Taking** The ability of a subject to evaluate a situation from an external point of view, usually the point of view of a distinct subject. 8, 9
- PR-2** A large humanoid multi-purpose robot, manufactured by Willow Garage. 138
- Pragmatic** Viewed as a relation between the context and the meaning of a speech act. 9, 11, 12, 17–19, 25, 26, 29, 42, 100
- Pragmatic Frame** A pragmatic frame is a negotiated interaction protocol targeted to achieve a joint goal that involves (1) a surface layer, textly, an observable coordinated sequence of pragmatic behaviors in the form of words and actions, (2) a deep structure underlying these behaviors that targets the achievement of one or several joint goals, and (3) a nested cognitive layer that specifies which cognitive operations the frame triggers as it unfolds.
— [Rohlfing et al., 2016] xi, xii, 16–19, 21–23, 25, 29, 31, 61, 77, 95, 126, 127
- Procedural Knowledge** The knowledge of how to achieve a certain process, action or task. It may or may not be represented in the form of symbolic knowledge. 5, 10, 11
- Qi SDK** The [Software Development Kit \(SDK\)](#) allowing development on Pepper robots running 2.9.x versions of NAOqi. 76, 77, 84, 89, 97, 107, 121, 123, 124, 135
- ROS** Open-source middleware widespread in robotics community. ROS stands for “Robotic Operating System”. 10, 97, 138
- Scaffolding** Scaffolding is the process by which a teacher organizes a new skill into manageable steps and provides support such that a learner can achieve something they would not be able to accomplish independently [Greenfield, 1984, Vygotsky, 1978]. A good teacher will scale instruction appropriately and create a good environment for learning the task at hand.
— [Chernova & Thomaz, 2014, p.7] 6, 17–19, 127
- Self-Demonstratino** A demonstration that is performed by the same agent it is addressed to, *e.g.*: performing a choregraphy to rehearse it. 22, 24, 34, 35
- Semantic** Viewed as a relation between a signifier and its meaning. 9–11, 18, 19, 22, 26, 28, 29, 42

- Semantic Frame** A structure representing a situation that can be expressed using natural language. For example the phrases “leaving the room” and “going out of the room” may describe a same semantic frame *go*, with the same slot *provenance* set to “the room”. 11, 12, 18, 29, 31, 51, 56, 58, 62, 77, 90, 94
- Semantic Structure** Often encountered in machine translation, a symbolic representation of a piece of natural language that abstracts away language-specific structural constraints. 11, 51
- Skill** At a the sensorimotor level, the procedural knowledge allowing the proper execution of a task. xiii, 7, 35, 37
- Speaking Floor** In spoken interaction, the abstract direction of attention shared between participants, that support the verbal communication. The alternance of ownership of the floor is called the [speech turn](#). 17, 26, 60, 100
- Speech Act** An action that is performed by the means of an utterance. xiv, 26, 30, 89, 142
- Speech Recognition** Process of recognizing a linguistic or semantic symbol from a natural language utterance. xi, xii, 6, 10, 11, 70–73, 77, 78
- Speech Recognizer** A software component providing speech recognition. 26, 27, 49, 59, 69, 71, 74, 78, 108, 115, 116
- Speech Turn** In spoken interaction, the moment when a participant is conventionally offered to speak, and is meant to be listened to by other participants. xvi, 60
- Spoken Language** A natural language as pronounced or heard by a human. xi, 3–7, 10, 11, 13, 16, 21–24, 26, 32, 34, 36–44, 46, 48, 64, 69, 70, 74, 77, 109, 116, 120–122, 138
- Symbolic Knowledge** Knowledge that can be represented in the form of discreet symbols, on which logical reasoning is possible. xiv, 5, 7, 31, 141
- Task** A symbolic representation of a process, as seen by an artificial agent. 6, 7, 9–11, 35, 38, 55, 56, 128
- Task Plan** An arrangement of tasks in parallel or in sequence. It is a symbolic form of procedural knowledge. xvi, 7, 9–11, 16, 23, 46, 55, 58, 77, 80
- Teaching Behaviors** The act of explicitly teaching a behavior of any kind, including in the form of a [task plan](#), and regardless the modality of communication or demonstration. 4, 5, 10
- Theory of Mind** Theory of mind is the ability to attribute mental states – beliefs, intents, desires, emotions, knowledge, etc. – to oneself, and to others, and to understand that others have beliefs, desires, intentions, and perspectives that are different from one’s own. — [Wikipedia contributors, 2019c] 8, 125, 126

User An individual using or interacting with a tool designed for his or her purpose. [7](#), [10](#)

YARP Open-source middleware used mostly robots from IIT, like the [iCub](#). YARP stands for “Yet Another Robotic Platform”. [138](#), [140](#)

Zero-Shot Learning A property of a machine learning algorithm when it is able to produce a correct output without the need for a single practice. [4](#), [10](#), [11](#), [32](#), [33](#), [37](#), [38](#), [68](#), [92](#)

Acronyms

- ADE** Application Distribution Engine 75, 76
- AI** Artificial Intelligence 5, 6, 9, 95, 96, 122
- API** Application Programming Interface 82, 84, 121, 133, 135, 138
- ASD** Autistic Spectrum Disease 19
- BN** Bayesian Network 34, 139
- CCG** Combinatory Categorical Grammar 27, 28
- CFG** Context-Free Grammar 27
- CNN** Convolutional Neural Network 37
- H** Human 53, 54, 60, 61, 77
- HCI** Human-Computer Interaction 8
- HMI** Human-Machine Interaction 7, 8
- HMM** Hidden Markov Model 28, 139
- HRI** Human-Robot Interaction xi, 2, 5, 7–9, 12, 16, 19, 21–23, 25, 26, 29, 32, 38, 42, 44, 48, 80, 97, 100, 116, 122, 137, 141
- HTN** Hierarchical Task Network 34, 35, 37, 97
- IBL** Instruction-Based Learning 138
- IML** Interactive Machine Learning 7
- IRI** Internationalized Resource Identifier 85, 87, 89, 98
- IRL** Interactive Robot Learning 5, 7–9, 12, 19, 20, 29, 122, 142
- ITL** Interactive Task Learning 5–7, 9, 12, 122
- LfD** Learning from Demonstration 5–7, 9, 12, 21, 130
- LSTM** Long Short-Term Memory 30, 37
- MDP** Markov Decision Process 31, 34, 37, 97, 139
- MTT** Multimodal Recorder 76

NL Natural language 26

NLG Natural Language Generation ix, 26, 29–31, 46, 47, 49, 54, 55

NLI Natural Language Interaction 9, 11, 16, 25, 26, 31, 42, 46, 48

NLP Natural Language Processing 26, 69

NLU Natural Language Understanding ix, xi, xii, 11, 26–29, 31, 42, 46, 47, 49–52, 55, 68, 69, 71–74, 77, 92, 93, 108, 122, 125

PbD Programming by Demonstration 140

PDDL Planning Domain Description Language ix, 37, 97, 98, 100, 105, 124, 125

PNP Petri Network Plan 33, 139

R Robot 54, 60, 61, 77

RNN Recurrent Neural Network 28

ROI Region of Interest 82

SBRE SoftBank Robotics Europe ix, 2–4, 10, 35, 49, 59, 65, 66, 68, 70, 75, 76, 81, 88, 90, 106, 121, 122, 133

SDK Software Development Kit xv

STT Speech-To-Text ix, xi, xii, 28, 29, 47, 49, 66, 68–70, 72, 74, 77

SUS System Usability Scale 41, 75, 103, 116

TTS Text-To-Speech ix, 29, 47, 138

Introduction

Contents

1.1	Motivations: Adapting Social Robots in the Wild	2
1.1.1	Real Software Applications for Social Robots	2
1.1.2	Development of Applications for Social Robots	2
1.1.3	Adapting Social Robots in Real Conditions	3
1.1.4	Challenges of Adapting Social Robots in Real Conditions	4
1.2	Positioning in the Research Landscape	5
1.2.1	Artificial Intelligence	5
1.2.2	Interactive Task Learning	6
1.2.3	Learning from Demonstration	6
1.2.4	Interactive Robot Learning	7
1.2.5	Human-Robot Interaction	8
1.2.6	Positioning	9
1.3	Definitions	9
1.4	Objectives: Teaching Behaviors using Spoken Language in Real Conditions	10
1.5	Contributions	11
1.5.1	A State of the Art	11
1.5.2	A Proof-of-Concept in Homes	11
1.5.3	Teaching behaviors in Rich Scenarios in Homes	12
1.6	Conclusion	12

1.1 Motivations

Adapting Social Robots in the Wild

Social robots are already present in the wild. They can be found in shops [Boom, 2015], tourism activities [Reese, 2016, del Duchetto et al., 2019], care facilities [Satariano et al., 2018, Khosla & Chu, 2013], homes [Richardson, 2017, Reyes, 2016], schools [Gray, 2016, Reddy, 2019], or even in religious services [Peh & Foster, 2017, Löffler et al., 2019].

A significant part of them are **Pepper** robots, made by **SoftBank Robotics Europe (SBRE)**: as of May 2018, 12,000 units have sold in the world [Olson, 2018]. In this section we share various challenges that **SBRE** faces using **Pepper** in the wild, *i.e.* in real conditions.

1.1.1 Real Software Applications for Social Robots

The use cases of such robots range so widely that it is not currently possible for a single robot to cover all of them. Instead, a selection of software applications is deployed on each robot depending on its use cases, *e.g.* introducing itself to people, playing games, taking surveys, informing about cars, wines, locations...

Most applications are tailored to specific use cases. For example, there may be an application for the robot to present shop items, and another to give directions: both applications would certainly be structured differently. Even between two applications for presenting shop items, *e.g.* cars vs. wine, there may be differences on the structure of the information and in the context of interaction.

Applications rely on **Human-Robot Interaction (HRI)** features provided by **Pepper** out of the box: dialogue management, speech recognition and synthesis (with a high-quality voice and accompanied with body language) detection and engagement with humans (looking at them, but not to insistently), and safe full-body animations and locomotion, provided through an agreeable design.

On the other hand, these features remain relatively low-level, so **Pepper** “does nothing” unless applications are installed on it.

1.1.2 Development of Applications for Social Robots

Application development on robots has been established and simplified with time. **Pepper**’s ecosystem includes the graphical programming and animation tool **Choregraphe**, the **Android Studio** plug-in **Pepper SDK**, and **Content Management Systems (CMS)** like **Pepper for Biz** or **P4S**.

These solutions reduced the cost of application development, but still require a specialist to develop them. Non-specialist business **Pepper** owners still rarely intervene in the applications or the contents of their robot. It remains impractical for them to adapt a robot using classical computer interfaces like desktop applications or web pages. This is a missed opportunity to adapt **Pepper** better to their needs.

The difficulty of application development on Pepper also affected private individuals who purchased Pepper. Since the ecosystem of application development for private use has not taken off, it remained difficult for owners to find applications that would augment Pepper’s *behavior*. Overall, Pepper’s *behavior* stagnated. Their owners have become bored by the lack of content, and frustrated by the lack of understanding of the robot: it would not even try to do what the users ask for, and would not remember or learn anything from past interactions.

This situation could be eased if the users could directly teach Pepper what they wanted. Through user interaction, the robot could learn new abilities, accumulate knowledge, and become adapted to its actual use case.

1.1.3 Adapting Social Robots in Real Conditions

SBRE have been working on generic solutions to let the users produce behavioral contents directly through interaction. Two prototype applications, Pepper Play and later Do This Do That, allowed users to make the robot perform specific *actions* on certain events, the way it is done with popular web tools like IFTTT¹. IFTTT stands for “if this then that” and is a simple programming interface that allows pre-programmed functions to be scheduled on particular web events. The prototypes allowed users to define robotic *behaviors* and schedule them in reaction to robotic events. They worked and demonstrated that with a combination of *spoken language* and touchscreen interaction, the Pepper’s *behavior* could be customized. They deployed them to a set of robots lent to some employees, so that they can test new general public applications from their home. This test program is called Pepper@Home.

Confidential results from Pepper@Home highlight that being able to adapt the *behavior* of the robot made the robot more interesting and more unique. The *behaviors* invented were usually social, complementing the overall interaction with the robot. However, Pepper Play and Do This Do That did not solve the issue of a robot that was not able to reuse information that it had been told. The same problem occurred with any other application that was deployed to Pepper@Home: while an application enabled Pepper to understand something, the information would not be reusable to adapt its general *behavior*. It did not scale up.

Overcoming the challenge of adapting the *behavior* of Pepper in that way would be a disruptive change, making the *behavior* of social robots satisfactory for private individuals. It could also make it easier for professional owners to adapt their robots in the field, without the need for a third-party, or an additional computer interface².

Employees who work with a robot may also benefit from adapting it. They could tell the robot what it should do to help them, rather than having no say on the robot’s *behavior*. Thus they may see it more like a partner than like a competitor in the labor market. Caregivers could get more help from robots by providing specific instructions for situations that were unforeseen when the robot

¹<https://ifttt.com/>.

²In addition to the robot itself, which is a computer interface

was initially deployed. The list of new usages goes on, as the full potential of social robots like Pepper is finally realized.

1.1.4 Challenges of Adapting Social Robots in Real Conditions

Why would adapting the **behavior** of a social **behavior** be so difficult? In previous works of research [Lauria et al., 2001] and on prototypes at SBRE, it was already demonstrated that new **behaviors** can be taught using **spoken language**. This is what we call **teaching behaviors** using **spoken language**. In practice it has been done by describing the desired **behavior** just once to the robot. This **zero-shot learning** ability is especially convenient for the users, who otherwise would quickly give up. In addition, it usually requires less computing power than more recent machine learning approaches.

But what is learned is not reusable. A **behavior** is reusable if it can occur in different contexts, with different parameters, of same or different natures. For example, given a **behavior** for greeting Alice, it is more reusable if the robot can also greet Bob. It is even more reusable if it can greet anyone else: *i.e.* if the **behavior** is parametrizable. If the robot can also greet a dog, a statue, or anything else, the **behavior** is more generalized, and its parametrization is more flexible, making it more reusable.

The greeting **behavior** is also made more reusable if the robot can discuss about it: name it, perform it on demand, articulate it with other information, or detail it. Providing details on a known **behavior** and the reasons why it is performed leads to explainability. Explainability is a recommendation for ethical robots [Chatila et al., 2017], and were officially adopted by the OECD [OECD, 2019] and the G20 members [G20, 2019]. [Rudin, 2019] urges the community to use intrinsically explainable models. It could become a requirement for social robots in the future.

To demonstrate reusability, a robot should understand what is the greeting **behavior**, that it can apply to any human, but also to other agents, or representations of it, or any other objects. To be able to generalize this **behavior** – given that it can be generalized – the robot should also be able to identify humans, agents, or objects. And the users should be able to teach who the agents are and what the objects are. If the greeting **behavior** involves locomotion, the robot should also be able to be taught about locations in space. To communicate using **spoken language** about this, the robot should be able to articulate all these notions together. We call this **knowledge interoperability**.

Another dimension of reusability consists in reusing a taught **behavior** in another **behavior**. For example, given a **behavior** for greeting people, the robot could be taught to welcome people home by greeting them and then by inviting them to hang their coat. We call this **composability**.

Research continually makes progress about **teaching behaviors**, but the scenarios of experiments are often closed (see subsection 2.5.3): what is taught is usually reused in the same context, or cannot be relied on for later teachings. Often, robots are not designed to learn or do anything else than what is required to demonstrate

the teaching. For instance, [Mohseni-Kabir et al., 2019] demonstrates a complex teaching scenario and a powerful learning algorithm, but only for the specific task of changing a tire. Whereas HRI at home is an *open scenario*, and social robots in such environments are expected to have other purposes, and learn other things to fulfill them. The expected interaction is richer than what the current state of the art provides, and the learned *behaviors* integrate well that rich interaction.

Therefore it appears that we could make the research progress by achieving the teaching of *behaviors* using *spoken language* in more *open scenarios* and with richer interactions, on a system that can also be taught about its environment. The knowledge, including the *procedural knowledge* within *behaviors*, should remain reusable for the applications providing the teaching *behaviors*. That is the result this thesis aims for.

1.2 Positioning in the Research Landscape

Teaching *behaviors* using *spoken language* to robots has been covered in several complementary research fields. In this section we describe the research landscape from which originates our literature, we detail the notions involved and then position this thesis.

We start first with *Artificial Intelligence* (AI), because it provides the fundamental notions. Then we look at *Interactive Task Learning* (ITL), and then *Learning from Demonstration* (LfD), because they deal with interactive teaching of *behaviors*. Both are part of *Interactive Robot Learning* (IRL), and involve *Human-Robot Interaction* (HRI). Our research overlaps all these fields.

1.2.1 Artificial Intelligence

AI is a field of computer sciences focusing on automating decision-making. It may involve the observation of the world³, its understanding, the production of reusable knowledge, and a mean of action back to the world.

An entity capable of such decision-making is an artificial *agent*. The notion of *agents* is shared with philosophy, grammar, psychology and economics, and refers to a subject capable of *action*. Selecting the adequate *action* for a given situation is called a planning problem, and AI provides solutions for that.

Because AI seeks to implement artificial *agents*, it must describe them from the inside. In psychology, this is said to be a *cognitivist* point of view, – as opposed to the *behaviorist* point of view, that describes *agents* only from the *behavior* they exhibit. It is probably for this reason that the software systems implementing *agents* are called *cognitive systems*.

Some *cognitive systems* come with an *ontology* that structures the *symbolic knowledge* maintained by the system, so that it is *interoperable* with its various components, or across systems [Waibel et al., 2011].

³Regardless it is a virtual or the real world.

AI also borrows the notion of **tasks** from computer sciences: a process that can be scheduled for execution. From a cognitive point of view, an artificial **agent** can perform **tasks** it knows. When a **task** is performed, it may be seen from a behavioral point of view as acting. The difference consists in the fact that an **action** refers to the observed **behavior** of the **agent**, whereas a **task** is a potential that *could* be performed.

1.2.2 Interactive Task Learning

ITL is a sub-field of AI dedicated to allow artificial **agents** to learn new **tasks** through interaction with another **agent**. Usually, this other agent is human, has a role of teacher or oracle and communicates with *written natural language*.

Teaching new **tasks** is indeed a way to teach new **behaviors** to an artificial agent, such as robots, and therefore ITL provides a significant part of our literature.

According to [Laird et al., 2017], early researchers in ITL managed to teach **tasks** to artificial **agents** using *natural language* and with early forms of machine learning. [Simon, 1977, Simon & Hayes, 1976] would be the first attempt, on a Tower of Hanoi puzzle. They associated predefined natural language **instructions** to **actions** regarding the puzzle. Later, [Crangle & Suppes, 1994] would be the first to achieve ITL on robots.

Since then **speech recognition** has matured, allowing such teaching with **spoken language**. ITL used this and started to be applied on robotics, so well that now it meets fields that were once independent.

1.2.3 Learning from Demonstration

LfD is a field of research that focused on making robots able to learn from teacher's *demonstrations*. It sees the learner robot as an individual who learns from its experience, rather than a machine to be programmed. This approach is inspired by developmental psychology, which studies how infants learn with almost no prior knowledge.

[Chernova & Thomaz, 2014] review the research works made in LfD, and describe the teaching process as a looping pipeline (Figure 1.1). The teacher would **scaffold** the teaching interaction, demonstrate the desired outcome (a **behavior** or a target situation), that the learner observes, and analyzes. In return it would practice what it would have learned, so that the human could tell if it is properly acquired. If not, the human would restart the demonstration. We further develop this process in Section 2.1.

LfD distinguish the use **spoken language**, from the demonstrations [Chernova & Thomaz, 2014, p.40], but supports its role in the learning interaction, for scaffolding, or as a form of critique, or as side **instructions**. We included works of LfD in our literature when they exhibited **spoken language**, but it appeared that LfD does not encompass the case when **spoken language instructions** are provided alone, without a demonstration. It appears that for [Chernova & Thomaz, 2014], an **instruction**

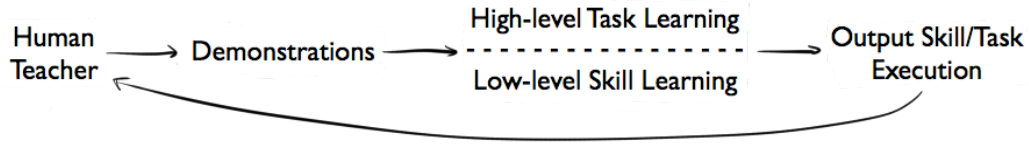


Figure 1.1: Adaptation of “A simplified illustration of the Learning from Demonstration pipeline.”

— [Chernova & Thomaz, 2014, p.3]

alone is not a demonstration.

Acquiring meaning, in any form including language, is called grounding [Harnad, 1990]. Meaning never comes alone, it is always grounded on something otherwise meaningful. Associating demonstrations to *spoken language* is an effective way of grounding and allows the robot to learn both what is demonstrated and the language associated to it. From this point of view, if the teacher meant to teach the language, *e.g.* by miming a known activity and uttering a phrase symbolizing it, the demonstration would be the utterance, not the mime.

About *behavior* learning, [Chernova & Thomaz, 2014] distinguish low-level *skill* learning from high-level *task* learning. How to recognize an object, how to reach it, or how to keep the equilibrium while doing so, are *skills*. Learning them involves learning the right parameters to control live dynamics properly. Whereas *tasks* ignore the details of the dynamics, and instead are a symbolic representation of the possible *behaviors* of the robot⁴ Then they can be composed into *task plans*, potentially hierarchical. This notion of *tasks* is also compatible with ITL, making the exchanges easier between the two fields.

1.2.4 Interactive Robot Learning

What brings together ITL and LfD is that they study how robots can be taught new *behaviors*. But as noted before, *behaviors* involve knowledge of the environment, and therefore social robots should be able to be taught other things.

The recent field of *Interactive Robot Learning* (IRL) covers the general question of how to teach a robot anything. The challenges of IRL are about finding appropriate algorithms to learn from few interactions with inexperienced *users*, and designing *Human-Robot Interaction* (HRI) allowing the robot and the *users* to understand each other [Broekens & Chetouani, 2019]. The work presented in this thesis is a contribution to IRL.

IRL is also a sub-field of *Interactive Machine Learning* (IML), which involves the use of any kind of *Human-Machine Interaction* (HMI), not only robots, and therefore encompasses LfD and ITL. In HMI (as well as in HRI) not only the computer system is studied, but also the human using it, the *user*. This is stressed in [Amershi et al., 2014], a review of IML, that could be well applied to IRL.

⁴This distinction resembles [Harnad, 1990]’s distinction between *connectionist* and *symbolic* knowledge.

The review states that since the system learns from its users, it cannot be studied in isolation. Machine learning often relies on an ideal source of information, but users are not ideal sources of information. Therefore it is important to include the users in the study, in order to design the learner system and the interaction techniques properly.

However our state of the art (in chapter 2) shows that when it comes to IRL, there is an evident lack of real-condition experiments.

1.2.5 Human-Robot Interaction

We end our tour of the research landscape with HRI, the specialization of HMI in robotics. Instead of relying on mice, keyboards, touch inputs and audio visual outputs, HRI relies on:

- A physical embodiment that users can see and touch. See subsection 2.5.1 of the state of the art.
- Seeing and locating the user to observe his or her behavior, social cues including those of his or her engagement. See subsection 3.1.1.
- Identifying the user. See subsection 5.5.4.
- Recognizing and producing speech. See section 2.3.
- Reasoning and making decisions about a non-deterministic environment.

Robots are better suited than avatars for embodying virtual agents. According to [Fridin & Belokopytov, 2014], users are naturally more engaged with a virtual agent embodied in a robot, than the same agent represented by an avatar.

Moreover, robots share the same physical space as their users, making them able to learn things from a point of view that computer cannot have. Our research relies a lot on HRI, so it is important to capture the spirit of the field.

According to [Breazeal, 2004], HRI differs from Human-Computer Interaction (HCI) (both are part of HMI) not only because robots replace computers, but also because robots are seen as agents that have their own goals. Therefore HRI does not only study the performance of the human, but also the performance of the robot, both as part of the human-robot system.

So the goal of the robot should be known, and can be compared to the outcome of the interaction. In turn, this favors certain implementation details of the artificial agent, further developed in section 2.4.

Ultimately, a robot could understand its user's goals, and more naturally collaborate to reach them. This would imply that the robot establishes a theory of mind for its user, and that this is part its knowledge of the environment [Scassellati, 2002].

The theory of mind allows perspective taking, but perspective taking can occur without theory of mind. The problem of perspective taking is involved in the

correspondence problem of LfD [Chernova & Thomaz, 2014, p.17], that could be summarized by the following question: how can what other individuals exhibit be applied on the learner?

Perspective taking is in fact essential for understanding a teaching. It requires a knowledge of the environment, the users, and the ongoing interaction. This constitutes a pragmatic context that is precious to understand properly the semantics in the Natural Language Interaction (NLI). But the pragmatics of the interaction are still tailored by researchers for their purpose. It is often part of the prior knowledge, rather than being automatically analyzed, which makes the robotic behaviors rigid.

1.2.6 Positioning

The problem of making effective teaching behaviors is well covered by IRL, and is detailed by ITL and LfD. Using the notions of tasks and task plans, it is possible to model behaviors that can be reused, by composition.

Providing a rich behavior to the robot in a home context would be interesting for HRI. In real conditions, scenarios are also more open, so achieving consistency and robustness may rely on good knowledge of the state of the art of AI and NLI.

1.3 Definitions

This section defines the most important terms in our research. Definitions are also recalled in the complete glossary.

Action A process through which an agent alters its environment, including itself.

Agent An individual capable of performing actions.

Behavior A behavior is the observable action of an agent given a situation.

Cognitive system A system that can autonomously collect information from its environment and act back on it through actions. A cognitive system implements an agent, but an agent could be implemented otherwise.

Composition Organization exhibiting the composite pattern: a composite entity (*e.g.* a task) aggregates other equivalent entities, and may be part of equivalent aggregations.

Instruction A natural language phrase that describes an action to perform.

Natural language A language commonly used by humans to communicate with each other.

Open scenario A scenario in which the situations were not predicted, nor the outcome specifically intended.

Pragmatic Viewed as a relation between the context and the meaning of a speech act.

Procedural knowledge The knowledge of how to achieve a certain process, action or task. It may or may not be represented in the form of symbolic knowledge.

Semantic Viewed as a relation between a signifier and its meaning.

Speech recognition Process of recognizing a linguistic or semantic symbol from a natural language utterance.

Spoken language A natural language as pronounced or heard by a human.

Task A symbolic representation of a process, as seen by an artificial agent.

Task plan An arrangement of tasks in parallel or in sequence. It is a symbolic form of procedural knowledge.

Teaching behaviors The act of explicitly teaching a behavior of any kind, including in the form of a task plan, and regardless the modality of communication or demonstration.

User An individual using or interacting with a tool designed for his or her purpose.

Zero-shot learning A property of a machine learning algorithm when it is able to produce a correct output without the need for a single practice.

1.4 Objective

Teaching Behaviors using Spoken Language in Real Conditions

We have chosen a straightforward approach for teaching new [behaviors](#): users compose new [behaviors](#) (see [composition](#)) out of existing ones into a [task plan](#), using [instructions](#) in [spoken language](#). The system reuses the knowledge on objects and locations provided by other applications, implementing domain-specific teaching interactions.

We put these learning capabilities together in a social robot, and deployed it in real conditions, in homes, for the first time in the scientific community. [SBRE](#) allowed us to use the Pepper@Home program to achieve this. In return, we were constrained to use [Pepper](#) robots, use its default software, [NAOqi](#)⁵, and refrain from using additional external devices. But our research also benefits from these constraints, by preventing us from relying on assets that are not actually available in real conditions.

⁵As opposed to using [ROS](#), an open-source alternative often favored in the scientific community.

Nonetheless, we plan to demonstrate that our system is able to overcome challenges such as:

- supporting the openness of the home scenario: users may teach Pepper unanticipated **tasks**, To achieve this, we rely on the **semantics** of the **natural language**, use algorithms capable of **zero-shot learning** learning, and which do not prevent explainability.
- supporting a rich set of **behaviors**: basic interaction, and the teaching objects and locations. The basic interaction would represent the variety of usages of Pepper.
- maintaining the consistency of a shared knowledge base, handling **procedural knowledge** as well as knowledge about objects and locations, in a reusable manner.
- merging **behaviors** provided by separate applications, so that they contribute to one another. We present a **cognitive system** to coordinate them, to ensure the overall consistency of Pepper’s **behavior**, while preserving its richness. The system uses planning and applies interaction rules inspired by **pragmatics**.

1.5 Contributions

1.5.1 A State of the Art

Our first contribution is the collection and the review of the state of the art on the teaching of **behaviors** using **spoken language**. Chapter 2 first presents it in comparison to how humans teach **behaviors** to each other. We highlight that the teaching interaction between humans and robots is still too simple in terms of pragmatics.

Then we review how semantics are usually extracted from **spoken language** using **speech recognition** and **Natural Language Understanding (NLU)**, how **cognitive system** usually implement **NLI** and how behavioral knowledge is usually managed in such systems. We finish with a review of experimental setups to highlight the lack of rich and open scenarios, but also the lack of well-shared evaluation methods.

1.5.2 A Proof-of-Concept in Homes

Chapter 3 describes our first experiments demonstrating the teaching of **behaviors** using **spoken language**. We detail how the first versions of our cognitive system and the learning algorithms that support teachings of simplistic **task plan** in real conditions: in homes.

Using a grammar-free speech recognizer and **NLU**, we extract a **semantic structure** that we use to understand the taught contents. Every task, pre-existing or learned, is associated to a **semantic frame**, so that it can be reused in further teachings.

We hypothesize that this technology is sufficient for supporting this teaching in real conditions. We deploy this teachable system in the homes of some users for the first time, allowing a user-centered study. We identify the main caveats of the interaction, including the lack of support for parametrized **behaviors**.

1.5.3 Teaching behaviors in Rich Scenarios in Homes

Chapter 4 presents new **behaviors** developed to support the learning of entities of the world, more precisely: places and points of interests. Such **behaviors** allows robots to produce interoperable and reusable knowledge in open scenarios, such as homes. We adapt the interaction for teaching **behaviors** to take advantage of this knowledge, and support parametrized **behaviors**.

It results in a richer set of **behaviors**, that conflict more directly with each other. Our hypothesis is that the conflicts can be solved, by using an ontology to describe the world, a goal-oriented description of possible **actions**, and a set of rules to prioritize **goals**. An effort was made to make a cognitive system that is domain-independent, and that may remain extensible, and support multimodal interactions in the future.

We introduce the notions of communication acts (a kind of event) and semantic templates (natural language formats representing **semantic frames**) along with our ontology. These notions are used by the goal-selection rules to resolve **goal** conflicts between **behaviors**. It appears that since the tasks the robot performs involve **HRI**, the associated rules define the **pragmatics** of the interaction.

We test our system in homes, with the Pepper@Home population, to prove that our system works. We also look for possible improvements of the teaching mechanism and of the overall user interaction, in comparison with our previous experiments.

Chapter 5 concludes this thesis, and discusses the outcomes of this work, as well as some perspectives we envisioned.

1.6 Conclusion

In this first chapter of this thesis, we draw the big picture in which our research takes place. This work is motivated by the need to adapt the **behavior** of social robots, like Pepper, in real conditions. Despite earlier trials, there are still challenges such as ensuring the interoperability with the software applications already found on the robots, and the **composability** of the taught **behaviors**.

There are also research challenges in the field of **IRL**, which combines **ITL** or **LfD** with **HRI**: there are no prior publication on teaching **behaviors** to robots in real conditions. In addition, we plan to tackle issues that were not tackled before in this context, such as: supporting an **open scenario**, in which the robot already has other purposes than the tasks to teach, ensuring the **behaviors** can reuse the knowledge they produce, and resolving the conflicts between these **behaviors**.

In next chapter, we provide a detailed state of the art of the teaching of [behaviors](#) using [spoken language](#). It should provide the reader the tools to understand the rest of this thesis, as well as it should justify our novelty claims.

State of the Art on Teaching Behaviors using Spoken Language

Contents

2.1	How Humans Teach Behaviors to Humans	16
2.1.1	Common Characteristics of Teaching Interactions	17
2.1.2	Explicit Teaching of Behaviors using Instructions	18
2.2	How Humans Teach Behaviors to Robots	19
2.2.1	Theoretical Specificities	19
2.2.2	Teaching Behaviors to Robots using Spoken Language Instructions	22
2.2.3	Conclusions on How Humans Teach Robots	25
2.3	Natural Language Interaction	26
2.3.1	Tuning Speech Recognition <i>a Priori</i>	27
2.3.2	Performing Natural Language Understanding <i>a Posteriori</i>	28
2.3.3	Natural Language Generation	29
2.3.4	Application in Cognitive Systems	30
2.3.5	Conclusions on Natural Language Interaction	31
2.4	Behavior Models and Learning Techniques	31
2.4.1	Review of Behavior Models and Learning Techniques	32
2.4.2	Choosing a Behavior Model	35
2.4.3	Conclusions on Behavior Models	37
2.5	Experimental Setups and Evaluation	38
2.5.1	Robotic Embodiment	38
2.5.2	Use of Microphones	39
2.5.3	Openness of Experimental Scenarios	40
2.5.4	Evaluation	41
2.6	Conclusion	42

In previous chapter, we explained how we plan to achieve the teaching of **behaviors** using **spoken language**: users provide **instructions** to interactively build **task plans** the robot can execute. But is this approach suitable for human users? And how have roboticists approached teaching **behaviors** using **spoken language**? In this chapter, we answer these questions by presenting the state of the art on teaching **behaviors** using **spoken language**.

We collected every publication exhibiting users teaching **behaviors** to a robot using the **spoken language**¹. This collection is further mentioned as “our literature”, and consists of 20 publications. They are analyzed throughout this chapter, which ends with a summary in table 2.3.

First, in section 2.1, we justify that using instructions is suitable by looking how humans teach each other **behaviors** using the **spoken language**.

Then, until the end of the chapter, we detail how roboticists have achieved teaching **behaviors** using **spoken language**. The **goal** is to analyze the possible technical choices, to justify the ones made in our research, which are presented in next chapters. In each section, we focus on a particular domain.

In section 2.2 we focus on **HRI**, the form of the interaction, and the form of the instructions. We highlight the need (and the lack) of richer interaction patterns, which we implement in chapter 4. We present the notion of **pragmatic frames**, and use it to assess the richness of interaction.

In section 2.3, we look at **NLI** techniques, and evaluate their pros and cons, especially for the potential openness of the interaction. We explain how dialogue systems integrate cognitive systems, and constrain their extensibility.

In section 2.4, we review the **behavior models** used in the literature, and evaluate whether they are compatible with our form of teaching, and with the need to mix **behaviors** from separate applications.

Finally, in section 2.5, we review the experiments presented by the literature, to support the novelty of teaching **behaviors** in real conditions. We found the robots are usually not sufficient to perform the experiments. Additional devices, like microphones, are required. This, in addition to the lack of openness in the experimental scenarios, puts a curb on real-condition experiments. And then we review some evaluation methods that are later used in this thesis.

Let us start this chapter by presenting the natural foundations of the teaching interaction.

2.1 How Humans Teach Behaviors to Humans

Humans routinely teach **behaviors** to each other, by interacting with each other in many ways. For example we can tell each other how to behave, through explanation, dissuasion, or storytelling. We teach a lot of **behaviors** also by demonstration, where

¹We initially collected a larger set of publications exhibiting teaching **behaviors** using **natural language**, *i.e.* not necessarily in the spoken form. Then we narrowed down the study to the spoken form, but we kept some references to that larger literature when useful.

the learners try to imitate the **behavior** they observe. But **behaviors** can also be induced more subtly, or even emerge naturally.

A teaching consists in an explicit act of communication, which should induce, in some way or another, some knowledge transfer to the learner. This is called transfer learning. To teach a **behavior** is specific because the knowledge to transfer is something that the learner can perform, or exhibit².

2.1.1 Common Characteristics of Teaching Interactions

[Chernova & Thomaz, 2014], chapters 2 & 3 present how teachers **scaffold** the learning process of the learner, direct its attention and rely on it to provide demonstrations. In other words, teachers select the form of interaction and its content to help the learner.

In that manner, teachers appear to drive the teaching. But in fact, learners understand that teachers mean to help, and in turn take advantage of it: they can drive the teachers' attention on other matters, and interfere in the turn taking. This is called a **mixed-initiative interaction**, and a good learner would exhibit such initiative.

Judicious initiative involves a form of meta-cognition: the learner must have a model of the interaction, and strategies to act on it: managing the **speaking floor**, asking questions, paying attention to something, etc. [Chernova & Thomaz, 2014, p.11]. A good learner would be capable of learning these strategies through interaction (meta-learning).

Learning these strategies is still difficult to achieve in robotics. The current research is usually focused on achieving certain tasks, given some predefined situation. Chernova & Thomaz invite us to research beyond, and make robots able to recognize and set up the teaching interaction by itself. This is inspiring, and it is possible that our attempt at supporting rich scenarios may lead towards this.

That would require a model of the **pragmatics** of the interaction, and a form of meta-interaction to drive the interaction. On this matter, we have been interested by [Rohlfing et al., 2016], who introduce the notion of **pragmatic frame**, an element of interaction pattern that can be subject to negotiation between participants:

A pragmatic frame is a negotiated interaction protocol targeted to achieve a joint **goal** that involves (1) a surface layer, namely, an observable coordinated sequence of pragmatic **behaviors** in the form of words and **actions**, (2) a deep structure underlying these **behaviors** that targets the achievement of one or several joint **goals**, and (3) a nested cognitive layer that specifies which cognitive operations the frame triggers as it unfolds.

[...]

²But it is also true for any teaching that its success can only be assessed by a **behavior** change of the learner. Therefore any successful teaching results in a **behavior** change.

What we call pragmatic frames are also known as “interactional formats” [Bruner, 1983], p.120, or “frames” [Fillmore, 1985], p. 111; [Tomasello, 2003], p. 25. — [Rohlfing et al., 2016]

[Bruner, 1983] is in fact a study of the emergence of the use of language in infants. Before acquiring language, infants learn to reproduce interactional **behaviors**. Interaction appear to be organized as a sequence of situations, following proto-grammatical rules. The templates of situations are what Bruner calls “interactional formats”.

Infants learn very early to use interactional formats to recognize or build meaning, and perform learning on the outcome, the “slot”. In practice the formats are reinforced interaction after interaction, until meanings build up, and the slot can be understood. It is the interactional format that conveys meaning in the beginning, that is to say it is not carried by the context rather than a symbolic expression. The meaning is conveyed through **pragmatics**.

At the same time, learning and using these formats demonstrate a successful teaching of **behaviors**, by reinforcement. This concurs with the general theory from [Sperber & Wilson, 1987], which states that reinforcement takes place in any teaching.

On the other hand, Fillmore’s frames have rather been used in the field of **semantics**. We develop the use of **semantic frames** in subsection 2.3.2. The latter is of greater importance in our specific case, because we expect the human teacher to provide **instructions** to the robot. This is also why we focus on how humans provide instructions to each other in next subsection.

2.1.2 Explicit Teaching of Behaviors using Instructions

It is commonplace that humans provide **instructions** to each other, *e.g.* to teach recipes or how to do new things. An instruction describes a **behavior** explicitly, so it can be communicated precisely, and thus achieve the teaching.

Teaching with **instructions** obeys to the same rules as any other kind of teaching. Namely:

- the teacher must **scaffold** the interaction with the learner to ensure a good understanding;
- the learner may take **initiatives** to clarify whatever was not understood yet;
- the teaching interaction is prepared using **pragmatic frame** negotiation, involving meta-cognition.

In many cases, the **behavior** can be understood directly from the **natural language** content of the **instructions**, with less or no contextual information. Understanding such **instructions** is therefore more a matter of understanding the **semantics** than the **pragmatics**.

This makes them easier to be written down. For example the website wikiHow [wikiHow Incorporation, b] regroups instructions to learn “how to do anything”.

Providing instructions is indeed a straightforward approach for humans to teach behaviors to each other. We review how it was applied to teach behaviors to robots in subsection 2.2.2.

Nonetheless, there is part of pragmatics and room for reinforcement learning, even when providing instructions.

The book [Bernard-Opitz, 2017] provides a salient example of reinforcement learning on top of instructions. It is dedicated to the education of children with an Autistic Spectrum Disease (ASD), using a behaviorist approach. It presents situations that a child can encounter, several possible behavioral responses, and their consequences.

The teacher is meant to be very explicit on the behavior he or she wants the child to learn, but it is through repetition that the child finally acquires the behavior. This book is also among the rare ones explicitly about “teaching behaviors” to humans³.

As a conclusion of this section, it is natural to teach behaviors using instructions. The learner can leverage semantics to understand them, and requires less contextual information. However pragmatics remain essential to setup the teaching interaction, and scaffold it. This, in turn, involves a form of meta-cognition. We highlighted the theory of pragmatic frame negotiation as a possible path of research.

2.2 How Humans Teach Behaviors to Robots

In the previous section, we borrowed the interaction point of view. We keep this point of view in this section, when we look at how humans teach behaviors to robots. This point of view corresponds to the domain of HRI, which we introduced in subsection 1.2.5.

We start this section by detailing the theoretical specificities of IRL in terms of HRI, so that to have the right tools to review the literature next.

2.2.1 Theoretical Specificities

[Chernova & Thomaz, 2014], chapter 3, provides a piece of state of the art on the teaching of behaviors, and a frame of reference for designing social learner robots. It consists in questions to ask ourselves when designing HRI. We summarize it as follows:

Social interaction. Which social cues and actions are used? What does it inform the teacher and the learner of? Which cues do users prefer?

Motivation for learning. What motivates the learning of the robot? Does it induce any initiative?

³As opposed to “teaching doing something” or “teaching how to do something”. Does the phrase “teaching behaviors” have any implication to the learner’s free will?

Transparency. How does the teacher know about the robot’s understanding? How does it help the learning and the teaching interaction? What form of communication seem the most efficient?

Question asking. Should the robot ask questions when it lacks information, and how? What kind of answer a user can provide, and how to understand the answer?

Scaffolding. How can a robot leverage the proposed scaffolding? How to break down deep teachings into simpler ones?

Directing attention. How is the attention drawn and understood by both parties? How is it used to help the teaching or the questions?

Online vs. batch learning. When does the learning actually occur? How does the learning system’s constraints impact the interaction?

Each question is intended to drive the designer towards a more human-like and effective interaction.

When a user agrees to teach a robot learner, their [goals](#) concur: both want the user to communicate the knowledge to the robot. At this point, a robot relies almost completely on the user. This is why what [\[Amershi et al., 2014\]](#) reminds us is important: in such interaction, the design should focus on effectively getting the user at ease to communicate the knowledge.

To summarize, they highlight that:

- Users are people, not oracles [...]
- People tend to give more positive than negative feedback to learners [...]
- People want to demonstrate how learners should behave [...]
- People naturally want to provide more than just data labels [...]
- People value transparency in learning systems [...]
- Transparency can help people provide better labels [...]

— [\[Amershi et al., 2014\]](#)

It seems to go without saying that users are people, or humans. But in fact we had to wait for [\[Cakmak & Takayama, 2014\]](#) to read a solid evaluation on the effect of instructions provided to human teachers in the context of [IRL](#), *i.e.* an evaluation centered on users. They found that showing videos of examples of interaction is more effective than a manual, or than an interactive tutorial with the robot. An interactive tutorial leads to failures, due to the high uncertainty of understanding the user properly, and should therefore be avoided. Whereas a manual can complement video tutorials without risking additional failures.

Experimenter / programmer actions	Teaching user actions	Robot learner actions	Robot learning
Frame 1			
1. Start	2. Input		Learning
3. End			
Frame 2			
1. Start		2. Perform	
3. End			

Table 2.1: “Basic common pragmatic frames for the category: passive learning.”
— [Vollmer et al., 2016]

The robot should be transparent about its state, and give feedback about any change, or about an absence of change. When something goes wrong or is misunderstood, it is important to give hints to the user, so that he or she can work around the problem. All along, the vocabulary must be chosen carefully.

These conclusions corroborate the assumption used in [Peltason & Wrede, 2012], that the tutor must be continuously informed about the learner’s internal state. It contributes to the regular patterns that can be observed in tutoring interactions. The survey [Vollmer et al., 2016] on the LfD literature identifies these patterns explicitly, and represent them as *pragmatic frames*. This survey reveals that the current experiments do not offer the flexibility the natural interactions usually exhibit.

In a simple case of passive learning, where the teacher demonstrates and the learner tries to imitate, the *pragmatic frames* were presented as quoted in table 2.1.

In the current state of the art, when a robot learner is tested through an experiment, the *pragmatic frames* are predefined, and cannot change dynamically as it was subject to negotiation. As a consequence, the learner has difficulties to repair the interaction, or to understand that the *goal* of the teacher shifted.

Reviews like [Amershi et al., 2014, Chernova & Thomaz, 2014, Vollmer et al., 2016] propose many guidelines, or questions HRI designers should ask themselves. However, the publications we reviewed rarely follow guidelines or answer questions explicitly. In fact, there is no official frame of reference for HRI.

So for the next review of HRI for teaching *behaviors* using the *spoken language*, we designed our own frame of reference. It consists in a set of questions referring to the state of the art presented so far, and highlighting the contributions of this thesis: the robots are in real conditions and are used for other purposes than following orders and being taught *behaviors*; and the teaching is done using instructions. Our questions are:

How did the user and the robot engage in interaction? This question highlights the preset contextual information that both the robot and the user should know. It should also highlight any form of negotiation of interaction, which may be an expression of meta-cognition.

Which pragmatic frames are involved? Knowing the interaction patterns involved allows us to spot whether another study tries to achieve the same form of teaching as us. But also, if there are several pragmatic frames supported by a studied system, it would be interesting to know how the switch of frames occurred. This could be essential to supporting rich scenarios.

What do instructions look like? The term of “instruction” includes a lot of linguistic forms. Knowing the form adopted by a study tells us about the how semantics are leveraged to make the instructions meaningful.

Which prior knowledge is required? In addition to the information preset to put the robot and the user into situation, there is prior knowledge for the user (instructions the *user* may have received) or for the robot (hard-coded in its program). Highlighting them often reveals the limits of teachability of a system.

How is HRI impacted? The previous questions highlighted several points that impact the HRI. This last question intends to draw an overview of the HRI with a given system.

In the next subsection, we ask these questions to each of the publications where a teacher provides spoken language instructions to the robot.

2.2.2 Teaching Behaviors to Robots using Spoken Language Instructions

All the publications we collected about teaching behaviors using spoken language exhibit HRI⁴. But the spoken language did not always consist in instructions.

For instance, in [Mohseni-Kabir et al., 2017, Mohseni-Kabir et al., 2019], the teacher describes the task while demonstrating it. In [Salvi et al., 2012] is similar, but the robot performs a self-demonstration: it replays a recorded sequence of actions and learns from that situation.

But in the rest of the section, we focus only on the publications that exhibited direct instructions, that is to say: [Lauria et al., 2001, Rybski et al., 2007, Rybski et al., 2008, Weitzenfeld & Dominey, 2009, Lallée et al., 2010, Lallée et al., 2012, Forbes et al., 2015, Gemignani et al., 2015, Sorce et al., 2015, Forbes et al., 2015, Petit & Demiris, 2016].

⁴Whereas publications about using natural language often avoided HRI.

How did the user and the robot engage in interaction? None of the experiments situated the teaching within a broader usage context. The human teacher is artificially put in interaction with the robot⁵, and the robot knows in advance it should keep its attention focused on the teacher. Most of the time, the teacher wears a microphone to talk to the robot. We develop this point further in 2.5.2.

However, we found some publications proposing a way to start a new teaching explicitly – which may imply that the robot would be doing something else otherwise. In [Rybski et al., 2008] they introduce the teaching with, for example, “Let me show you what to do when I say dinner is ready”.

Other publications proposed that, but within a larger context where the interaction has several modes [Lallée et al., 2012, Petit & Demiris, 2016]. In [Cakmak & Takayama, 2014, Sorce et al., 2015] it appears more explicitly that some instructions can be used to switch modes⁶.

Finally, in [Scheutz et al., 2017, Scheutz et al., 2018], the user can actively ask to start a new teaching, from within an ongoing teaching, when an action or an object involved is not known.

Which pragmatic frames are involved? The most common modality of learning is passive, involving the **pragmatic frames** described in table 2.1. The other interaction patterns we highlight here are described in detail in [Vollmer et al., 2016].

In [Lallée et al., 2010, Sorce et al., 2015], the robot systematically asks confirmation to ensure the speech recognition was correct, which is a simplified form of **mixed-initiative interaction**. Active learning also appears in [Mohseni-Kabir et al., 2017], where the robot asks questions to check that some **actions** can be regrouped, and in [Scheutz et al., 2017], because the robot identifies the missing objects and tasks autonomously, and ask to be taught about them.

There are also occurrences of exploration learning with user refinements in [Nicolescu & Mataric, 2003, Salvi et al., 2012, Grizou et al., 2013, Grizou et al., 2014, Grizou, 2014, Petit & Demiris, 2016]. The robot perform tasks autonomously, and the teacher tries to correct the robot.

Finally [Sorce et al., 2015] shows an inverted interaction where the robot explains back the **behavior** with the natural language, which is unique in this bibliography.

What do instructions look like? [Lauria et al., 2001] is the earliest demonstration of teaching **behaviors** using **spoken language** instructions. Dialogue 2.1 shows the illustrative example of teaching interaction they chose in their publication.

This form of instruction, in the imperative mood, is typical. Such instructions can be sequenced, and naturally form a **task plan**. This form of instructions is widely spread, we can find them in: [Rybski et al., 2007, Rybski et al., 2008, Weitzenfeld & Dominey, 2009, Lallée et al., 2010, Lallée et al., 2012, Cakmak & Takayama,

⁵This is also highlighted by [Anzalone et al., 2015] on other HRI works.

⁶This ability to switch modes may be seen as a primitive way to negotiate **pragmatic frames**.

USER: Go to the library.

ROBOT: How do I go to the library?

USER: Go to the post office, go straight ahead, the library is on your left.

Dialogue 2.1: Illustrative example of teaching interaction.

— [Lauria et al., 2001]

2014, Forbes et al., 2015, Gemignani et al., 2015, Sorce et al., 2015, Petit & Demiris, 2016, Scheutz et al., 2017, Scheutz et al., 2018]

In general, instructions describe parametrized **actions**. For instance “take the cup” describe an action “to take” parametrized by the object “the cup”. But in [Forbes et al., 2015], they managed to mention objects with phrases like “the tallest object”, hence demonstrating an extra degree of freedom in the form of instruction. In [Gemignani et al., 2015], the user can explicitly teach a parametric **behavior**, allowing the robot to understand many new instructions with a single teaching.

Instructions can also be simpler, like “come”, “here”, “go” [Nicolescu & Mataric, 2003]. This is enough to direct an exploration **behavior**, that serves as a **self-demonstration**, and genuinely accelerates the learning. In [Grizou et al., 2013, Grizou et al., 2014, Grizou, 2014], the instructions can take any form, since the robot learns the instruction (in their case it is either “right”(.*))“wrong”) together with the task to perform.

What prior knowledge is required? Prior knowledge may include procedural knowledge, knowledge of the environment (objects, people), knowledge of the language, and knowledge about interaction state.

In most of the publications, primitive **actions** are predefined and hard-coded. However in [Lallée et al., 2012, Cakmak & Takayama, 2014, Petit & Demiris, 2016, Scheutz et al., 2017, Scheutz et al., 2018] users can teach new **actions** from scratch. This reduces the need for prior procedural knowledge, but relies on knowing in advance how to map instructions and demonstrations to the movement primitives of the robot. This is called the correspondence problem [Chernova & Thomaz, 2014, p. 17], and it is solved *a priori* in these experiments.

Also the words chosen to label these **actions** must be part of a predefined dictionary. In [Lallée et al., 2012, Scheutz et al., 2017, Scheutz et al., 2018] it is possible to learn new objects, and their names, reducing the need for prior knowledge of the environment. In [Scheutz et al., 2017], however, the name of the learned object does not need to be predefined.

[Grizou et al., 2013, Grizou, 2014, Grizou et al., 2014] reach an extreme where nothing is known about the language, and it is the prior knowledge on the task structure that allows the meaning of the language to emerge. [Najar, 2017] demonstrate the same feature, using gestural signs instead of **spoken language**. Indeed, the interaction is always well constrained, and scaffolds the learning of the robot.

From the users’ point of view, there is very little they can do without being

properly instructed before the experiments. In most of the experiments, it is a professional of the laboratory that interacts with the robot, and he or she knows exactly what to do. [Cakmak & Takayama, 2014] studies the impact on that prior knowledge, and reveals that the instructions provided to the user has an impact on the quality of the teaching. [Sorce et al., 2015] made the effort of having naive participants in its experiments, though.

How is HRI impacted? In many works where robots understood instructions in the imperative mood⁷, the HRI was often reduced to a dialogue, *i.e.* to NLI.

However in [Lallée et al., 2012], it was combined with a mean to demonstrate some behaviors, by performing them in front of the robot, or with a mean to teach objects.

Because the robot must be put in a “teaching mode”, and because of how it is presented in the literature, entering the teaching resembles entering a distinct piece of user interface. The transition may not seem natural, but it is at least explicit. It also resembles the use of explicit cues to switch pragmatic frames between humans. This is meta-interaction and [Cakmak & Takayama, 2014, Sorce et al., 2015] exhibit a simple form of it.

In the literature around the use of instructions, we did not find examples where a robot would follow instructions and within the same framework, accept feedback during performance. At best [Gemignani et al., 2015] proposed a way to correct existing behaviors. But there is no application of interaction repair.

However, because teaching behaviors – tasks or skills – involve the teaching of other things in the environment, several publications took advantage of it. In addition to behaviors, [Scheutz et al., 2017] learned objects, words and affordances, [Grizou et al., 2013] learned words and deeper meanings, [Petit & Demiris, 2016] learned words and limbs, [Nicolescu & Mataric, 2003] learned trajectories in space.

In theory these forms of interaction could be multiplied, but it would require a system to support more flexible interaction, like the system presented by [Peltason & Wrede, 2010], focused on describing interaction patterns.

2.2.3 Conclusions on How Humans Teach Robots

We reviewed the literature from the point of view of HRI. We have seen how user engage in interaction, the pragmatics of the interaction, how instructions are provided, what prior knowledge – for the user and the robot –, and the impact on HRI. We found that most often the interaction is simplified: there is always only one user, who enters the teaching immediately, must know everything he or she should do beforehand and has little freedom in the interaction. Nonetheless we highlighted some publications exhibiting ways to control the interaction, an effort to work with naive users, repairing the teachings, or learning about the environment.

We distinguished the publications using instructions from the others, and reported this distinction in table 2.3. And we have reviewed the forms of instructions

⁷As in grammatical mood, used to conjugate of the verb.

these publications supported, and found that instructions are not always used to describe a [behavior](#).

In the next section we go into technical details about how to process natural language in interactive context.

2.3 Natural Language Interaction

Dialogue systems provide [Natural Language Interaction](#) (NLI) to artificial systems. They are also used in robots to support [HRI](#) through [Natural language](#) (NL). NLI shares with [HRI](#) the notion of the [speaking floor](#) and the notion of [actions](#), through [speech acts](#) [[Austin, 1962](#), [Searle, 1965](#)]. [Dialogue acts](#) are a type of [speech acts](#) influencing the dialogue itself, and that can be classified using DAMSL [[Allen & Core, 1997](#)].

The understanding and recognition of [dialogue acts](#) provide the dimension of the [pragmatics spoken language](#) conveys. Information is indeed conveyed by the patterns of dialogues, as well as from the intonations, as observed in [[Bruner, 1983](#)]. But most of the time we need to understand the words and their [semantics](#) to be able to recognize [dialogue acts](#).

In this section we go into the details of [Natural Language Processing](#) (NLP). We explain how to understand natural language (NLU) using a rather [semantic](#) approach, and then generate of a natural language response (NLG).

In the context of robotics, [NLU](#) can effectively ground the [Natural language](#) (NL) to the knowledge a robot may already have, and *vice versa*. Despite the ongoing research on [NLP](#), and there is not yet a general-purpose solution to achieve this grounding. When applied to [spoken language](#), there is also a strong limitation on [speech recognizers](#): they cannot recognize every utterance with ideal precision, and often require to be tuned to recognize certain utterances only.

Roboticians must therefore tailor solutions explicitly for their user cases, and decide how to associate [NL](#) to effects on the robotic system. In particular, they may need to tune the [speech recognizer](#). This tuning can be done following two different paradigms, that are decisive for the rest of [NLI](#):

- The [speech recognizer](#) is tuned to match a grammar or a set of possible utterances. Possible utterances must therefore be known *a priori*, and are intrinsically understood by construction.
- The [speech recognizer](#) transcribes speech. It may be tuned, but all the possible utterances are not known in advance, so [NLU](#) is required to understand the utterance.

The choice of paradigm is often imposed by the technology of the [speech recognizer](#) used. In the rest of this section, we study how the two approaches work, and their impact on [NLI](#).

2.3.1 Tuning Speech Recognition *a Priori*

Some [speech recognizers](#) can be tuned using transcriptions of the utterances to recognize, *e.g.*: “NAO”, “Hello”, “Look at me”. It limits the computational cost of the speech recognition, and makes it suitable for embedded systems.

It also simplifies the process of NLU because by construction, each phrase is associated to a specific meaning, intent, reaction or function, *e.g.*: “NAO” is the name the robot, by saying “Hello” the user means to greet, or “Look at me” is a request for a specific action.

It scales up with the use of grammars, such as [Context-Free Grammar](#) (CFG). We call this approach “generative”. In NAOqi, the dialogue contents can be written using the QiChat [\[Houssin, 2014\]](#) language. It is derived from ChatScript [\[Wilcox, 2011\]](#) and can express rich dialogues using patterned expressions.

Listing 2.1: QiChat example

```
concept:(greetings) [hi hello "hey there"]
concept:(tea) [green black] tea
concept:(hotdrink) [coffee ~tea]

u:(~greetings) ~greetings
u:(do you have _~hotdrink) yes, I have $1
u:(I want to drink something) do you want ~tea?
```

Listing 2.1 declares speech inputs within `u:(...)` statements. These inputs are developed according to the language’s rules, into the list of possible inputs shown in listing 2.2.

Listing 2.2: Developed list of matching inputs.

```
hi
hello
hey there
do you have coffee
do you have green tea
do you have black tea
I want to drink something
```

In the publications exhibiting the teaching of [behaviors](#) to robots, it is difficult to identify when a [CFG](#) is used, because this is an implementation detail. However [\[Lauria et al., 2001, Nicolescu & Mataric, 2003, Rybski et al., 2007, Rybski et al., 2008, Lallée et al., 2010, Lallée et al., 2012, Sorce et al., 2015, Mohseni-Kabir et al., 2017\]](#) may potentially have used it: [CFG-based speech recognizers](#) are easier to use for small and fixed sets of words. Some of the publications mention the use of [speech recognizers](#) like CMU Sphinx-4 [\[Lamere et al., 2003\]](#) or IBM ViaVoice [\[Dyer, 1997\]](#), which support [CFG](#).

Extensions and generalizations of [CFG](#) exists, namely the probabilistic [CFG](#) (PCFG) and the [Combinatory Categorical Grammar](#) (CCG). PCFG was not encountered in our literature, however [CCG](#) are used in [\[Scheutz et al., 2017\]](#): a grammar

of possible utterances described in the standard JSpeech Grammar format. CCG is backed by powerful mathematical models to associate linguistic patterns to semantics, and then to functions [Moortgat, 2011].

But it is also possible to use machine learning directly on the sound signal using Hidden Markov Model (HMM), Recurrent Neural Network (RNN) or similar techniques, and associate it to the desired symbolic output. This is done in [Petit & Demiris, 2016, Scheutz et al., 2017, Forbes et al., 2015, Grizou et al., 2013, Grizou et al., 2014, Grizou, 2014].

The use of “Acoustic DP-Ngrams” in [Scheutz et al., 2017] is particularly interesting because it allows the one-shot learning of a new word that their classical CCG-based technique cannot predict.

Thus, there are a lot of things that can be done using SRs tuned *a priori*. But as the dialogue topics grow open, speech recognition models grow too, to a point that they are destined to become general-purpose, defeating their need of pre-tuning.

2.3.2 Performing Natural Language Understanding *a Posteriori*

SRs with very large models can be used without prior tuning. They are said to recognize the “free speech”, and are often called Speech-To-Text (STT) engines. It usually requires a lot of computing power (typically in the case of RNN). Companies like Nuance, Google, Microsoft or Amazon provide online services for that⁸.

Instead of matching expected inputs, they output a transcription of the recognized speech. So NLU must be performed *a posteriori* on the textual transcription, to produce a semantic representation. This approach was used in [Gemignani et al., 2015, Cakmak & Takayama, 2014].

NLU techniques are numerous, but it is common to encounter the following intermediary results:

- Part-of-speech (POS) tags, representing the grammatical nature of words. The Penn TreeBank specification of POS [Santorini, 1990] is historic, but is specific to the English language. A universal specification exists (i.e. cross-language) [Petrov et al., 2012].
- Dependencies, representing grammatical relations, usually presented in the format defined by Stanford [De Marneffe & Manning, 2008], or and then generalized [De Marneffe et al., 2014].
- Named entities, classified as location, time, ordinal, persons... Introduced by [Grishman & Sundheim, 1996].
- Coreferences, to resolve the target of pronouns. Introduced in [Crystal, 2008].

Stanford’s CoreNLP, presented by [Manning et al., 2014], provides tools to produce these results from texts. RNN are typically appropriate to this task. Some

⁸It is also possible to perform STT in embedded devices with using software like Nuance VoCon Hybrid Speech Recognition, or on desktop using Mozilla DeepSpeech.

models, like DRAGNN [Kong et al., 2017], are freely available. Some online STT providers (namely Microsoft or Google), also provide these results directly from voice recordings, along with the transcription.

Pepper and NAO robots provide STT, and perform NLU to evaluate the emotional state and the engagement strength of the users. We reuse these algorithms called “semantic analysis” and detail them in 3.2.

Semantic models can be found in machine translation [Dorr et al., 1999], and consist in representing the natural language stripped from the actual words, lemmas or lexicons. They are related to the semantic models presented in [Moortgat, 2011].

[Fillmore, 1985] introduce *semantic frames*, a generic semantic model at the origin of [Baker et al., 1998]’s FrameNet. *Semantic frames* consists in a symbolic entity representing a situation, often an action, with parameters, called “slots”. Slots can be filled by various sorts of semantic entities.

FrameNet was constituted from a corpus of texts, and is not directly intended to be usable for *pragmatics*. But the convergence proposed by [Rohlfing et al., 2016] with *pragmatic frames* may be useful to design algorithms reasoning on both on *semantics* and *pragmatics*, and therefore on the interaction and dialogue patterns. This path remains rather unexplored in IRL.

2.3.3 Natural Language Generation

Natural Language Generation (NLG) and Text-To-Speech (TTS) are used in HRI to produce utterances, and thus talk to the users. In an analogy with NLU and speech recognition, we assume that speech could be directly produced from a *semantic* and *pragmatic* representation. But TTS technologies are mature enough to produce utterances that users can understand. This is probably why, in our literature, systems always rely on third-party TTS. We do the same in our research, so in this subsection, we only focus NLG.

NLG has a key role for the explainability of social robots, which is currently a hot issue. To explain a decision made by a robot, the system must not only identify symbolically the reasons of its *behavior*, but also translate these symbols into natural language. [Perera & Nand, 2017] review the recent NLG techniques and classifies them by the following aspects:

1. **Architecture** How components are organized and how the data flows between them.
2. **Content Determination** How to select the information to convey.
3. **Document Structuring** How to group, order, relate information to make it relevant and efficient.
4. **Lexicalization** How sentences and words are chosen to convey the meanings.
5. **Referring Expression Generation** How entities are referred to, using identifying words.

6. **Aggregation** How sentences are articulated with each other, including how words may refer to each other, so that the language remains fluid and avoids redundancies.
7. **Realization** How the text output is technically produced from the intermediate results.

In our research, the most problematic aspects are the architecture and the content determination, because they depend on the rest of the cognitive system. For instance, when [Perera & Nand, 2017] highlight the use of planners to generate sequences of sentences, there is a parallel to draw with planners used with certain *behavior* models (see section 2.4).

On the aspect of content determination, it appears that contents depend on the decision-making process and on the knowledge gathered by the cognitive system. [Perera et al., 2017] illustrate well this relationship on a simpler case of question-answering (QA). They use dependencies (see subsection 2.3.2) to identify question structures and symbols. They translate this information into semantic web queries, and translate back the results into natural language. In this example content is determined by a symbolic knowledge base online, and by hard-coded correspondence between the question and answer structures.

There are in fact various ways to produce texts, including ways that do not provide ways to control the contents in detail. For instance [Durrett et al., 2016] achieve summarization of news articles connectionist machine learning. In that case, the content is determined by the raw input content, and by the model trained with the system – there is no programmable input. Nonetheless it is possible to support programmable input with a *connectionist NLG* model; [Wen et al., 2015] demonstrate this with an *Long Short-Term Memory* (LSTM).

2.3.4 Application in Cognitive Systems

Autonomous robots are usually designed from a *cognitivist* point of view. The software system assessing the robot's situation, and taking decisions for it is said "cognitive", thus drawing an analogy between artificial and biological agents.

In every publication on teaching behaviors using *spoken language*, cognitive systems include a distinct dialogue system. This remains true in the broader literature on teaching behaviors using *natural language*.

Usually, the dialogue system drives the interaction, triggers learning of new behaviors, and the production of behaviors. Dialogues are therefore designed according to these behaviors. This approach is limited: if behaviors involved spoken language, they would interfere with the dialogue system, and it is unclear where *speech acts* fit in this model. This limitation is problematic for social behaviors, which should also participate to drive the interaction.

So the very idea of separating the dialogue system from the rest of the cognitive system is counterproductive to support the learning of social behaviors. Indeed, the

publications in our literature do not intend to demonstrate a solution to extend the NLI through the teaching of new behaviors, so they did not study that limitation.

The support of NLI often lead to using a *symbolic* representations of behaviors. We found some exceptions in [Salvi et al., 2012, Grizou et al., 2013, Grizou et al., 2014, Grizou, 2014]. They use Bayesian Networks or *Markov Decision Process* (MDP) to associate spoken language to behaviors in a black-box manner. In these configurations, the NLU is associated to behaviors using a more *connectionist* approach.

2.3.5 Conclusions on Natural Language Interaction

We reviewed the principles of NLU on which NLI rely, and that we use in chapter 3.

We highlighted the role of speech recognition, and how technical choices around this technology may have an impact on how the cognitive system can reuse the information.

Ideally, to keep the learning as open as possible, NLU should be performed directly on the audio input, but there is currently no immediate solution to extract a grammatical representation of an utterance, without having transcribed it to text (at least partially). Performing NLU on transcriptions may be enough to support the learning of unexpected objects or *actions*, but may lead to lower accuracy.

Extracting grammatical dependencies and a building a semantic model (like *semantic frames*) is common practice for NLU, and brings a good abstraction for learning *behaviors*. We also highlighted that in theory, there could be a way to make *semantic frames* and *pragmatic frames* converge.

Our review on NLG gives another perspective on what is a dialogue. For specialists of that field, this is planning problem: a text is to be built up automatically, so that it conveys a *goal* meaning. This is analogous to cognitive systems finding an action to perform a task satisfying a goal.

Finally we highlighted the interdependence between the dialogues and the task execution in the case of socially capable robots. It appears that separating the dialogue system from the rest of the cognitive system is counterproductive.

In the next section, we focus on the behavior models and their related learning techniques.

2.4 Behavior Models and Learning Techniques

A behavior model is a formalism to represent behaviors. For example, describing behaviors as lists of elementary action is a behavior model. Describing them as interpolated trajectory curves is another behavior model. Learning techniques and behavior models depend strongly on each other: on one hand the learning techniques must be chosen carefully to produce *behaviors* expressed in the given behavior model, on the other hand the behavior model must be chosen carefully to be learnable with existing techniques.

In subsection 2.3.4, we state that social robots should learn social **behaviors**. Learned **behaviors** participate with the dialogues, which cannot be considered separate anymore. The **behavior model** for the learned **behaviors** must be interoperable – if not identical – with the dialogue model. Since the dialogue usually drives the decisions of the robot, such **behavior model** is coupled with the cognitive system’s decision-making engine. In these conditions, the **behavior model** may be a keystone of the cognitive system.

In this section, we review the **behavior** models and the learning techniques used in the literature on teaching **behaviors** using **spoken language**. In this review, we evaluate the **behavior** models according to our objectives (see section 1.4) by checking whether each model:

- is compatible with **zero-shot learning**,
- allows taught **behaviors** to be reused in next teachings (*i.e.* achieving **composability**),
- supports some interoperability with other sources of knowledge in their host system,
- can work in open scenarios,
- can run on a Pepper or a similar robot, or be run on a remote service on the Internet.

We produced a summary of this review in table 2.2.

2.4.1 Review of Behavior Models and Learning Techniques

The first models found in our literature are procedural models: they make explicit when to perform each action. We first review these models, before reviewing alternative models. The history of teaching of **behaviors** using **spoken language** starts with [Lauria et al., 2001]. In their experiment, a fixed dialogue structure drives the interaction, and associates **instructions** to **actions** as prior knowledge. Each action is then associated to a Python function. The teaching is started by giving an unknown instruction, like “go to the library”. Once the teaching is started, every instruction provided is saved, as a sequence, and a Python function is generated from it. The function can be recalled by the dialogue system when the user says the new instruction. Note that the new instruction may have been put in the dialogue structure in advance to enable speech recognition (see subsection 2.3.1), and that therefore this technique may not be suitable for open scenarios.

This technique of associating the dialogue structure to the possible **actions** of the robot is so common that it has been generalized by [Peltason & Wrede, 2010]. They propose the Pamini framework for expliciting these associations, and generate the **HRI behavior** automatically. This framework has not been used to achieve this dialogue / action association in our literature, in [Rybski et al., 2007, Rybski et al., 2008, Weitzenfeld & Dominey, 2009, Lallée et al., 2010, Lallée et al., 2012,

Gemignani et al., 2015, Sorce et al., 2015, Forbes et al., 2015, Petit & Demiris, 2016, Scheutz et al., 2017, Scheutz et al., 2018]. In all these publications, the teaching was performed using instructions, and the **behavior** models were exclusively procedural:

- **Behavior Networks** used in [Rybski et al., 2007, Rybski et al., 2008], it is a graph representing a sequence of **actions**, with possible conditional branching based on the outcome of an action.
- **Percept-Response** used in [Weitzenfeld & Dominey, 2009], directly associates a perception event to a task or a task sequence.
- **Shared Plans** used in [Lallée et al., 2010, Lallée et al., 2012, Sorce et al., 2015], are sequences of **actions** involving several agents (the user and the robot), with possible synchronization points between agents.
- **Task Description Language** A simple language to describe tasks presented only in [Gemignani et al., 2015], allowing sequences and conditionals. It can be translated into **Petri Network Plan (PNP)**.
- **Sequences of Actions** The simplest model to imagine, used in [Forbes et al., 2015, Petit & Demiris, 2016] to represent the highest-level **actions**.
- **Action Script** Another language to describe tasks, found in DIARC [Schermerhorn et al., 2006], and used in [Scheutz et al., 2017, Scheutz et al., 2018].

With these models, **behaviors** can be directly produced from a list of instructions, allowing a quick and **zero-shot learning**. It also makes them all suitable to run in embedded computers.

Some of these models supported some sort of interoperability with the knowledge of the robot. [Lallée et al., 2010] support the teaching of objects, and their reuse in the teaching of **behaviors**. [Lallée et al., 2012] goes beyond by formalizing the object and procedural knowledge, and supporting the exchange of knowledge across robots through a SVN server. [Scheutz et al., 2017] show a robot also learning new words, referring to **actions** or objects, and affordances of objects. Since there are several mechanisms at work, it is possible that they exchange knowledge with each other. They do not demonstrate it, but [Frasca et al., 2018] work with the same **behavior model**, and emphasize interoperability, at least in simulation. In the rest of the literature – exhibiting non-procedural models – only [Forbes et al., 2015] demonstrated some form of interoperability: the **actions** accept any object as parameter, and the object can be retrieved dynamically from the current beliefs on the world, instead of being encoded within the action. This exception tends to show that procedural models promote interoperability⁹.

⁹Could that be explained by the fact that they are symbolic models, and that symbolic knowledge is a mature technology?

These models also potentially allow composition, but it was demonstrated only in [Rybski et al., 2007, Rybski et al., 2008, Lallée et al., 2012, Petit & Demiris, 2016, Scheutz et al., 2018]. In [Petit & Demiris, 2016], composition is not achieved by the high-level action model, but by the stacking of this model on top of another model better suited to learn action primitives.






To learn action primitives, [Petit & Demiris, 2016] use statistics of the effect of the commands sent to limbs, gathered during a phase of motor babbling. Similar models are found in the other publications – which do not interpret directly the **behavior** from **spoken language** instructions, but instead from a demonstration. [Grizou et al., 2013, Grizou et al., 2014, Grizou, 2014] shows a higher-level task babbling, that would be guided by unlabeled utterances. This form of teaching is called “shaping” [Chernova & Thomaz, 2014, p.22], [Najar, 2017]. This interactive game trains an MDP. Similar abilities were demonstrated with MTRNN in [Yamada et al., 2016, Antunes et al., 2018], with written words with no predefined semantics. The robot jointly learns the action and the instruction. The model directly associates the joint commands to the heard speech. This is very powerful but does not run well embedded platforms, nor in open scenarios.

The question of supporting open scenarios seems to divide these models. Procedural models are symbolic and domain-independent, and therefore are suitable for open scenarios. Whereas when a state space needs to be defined, or when the reinforcement of the model is expansive, it prevents an application in open scenarios. For instance [Forbes et al., 2015] use a probabilistic model (in addition to the sequence model of **actions**) to discriminate the parameters of the **actions**, and at the same time, learn the phrases referring to them. It seems that the algorithms used could run on embedded computers, but the state space used in the model must *a priori* include the objects of the world. This is true for all solutions based on a fixed state space. Whereas in [Nicolescu & Mataric, 2003] the teaching relies on demonstrations instead of instructions. Therefore it appears suitable for open scenarios: it uses a one-shot learning technique that can run embedded, and produce **behavior** networks, like [Rybski et al., 2007, Rybski et al., 2008]. It is the same for [Cakmak & Takayama, 2014], who rely on **self-demonstrations**, where the robot records its joint states, as the user moves its arms.

[Salvi et al., 2012] present a singular **behavior model**. They learn affordances of objects instead of procedures. The notion of affordance was introduced by [Gibson, 1977], and is used in robotics to represent the use of an object, and how to use it. Affordance is learned from repeated trials using a BN. And since they include words in the input, they intrinsically associate an action to a word, and *vice versa*.

The last model we encountered in our literature was presented by [Mohseni-Kabir et al., 2017, Mohseni-Kabir et al., 2019]: **Hierarchical Task Network (HTN)**. In comparison to the other symbolic **behavior** models we have reviewed, HTN may be the most powerful: it describes hierarchical action sequences, supporting conditional and intermediate **goals**. That makes it also suitable for automatic planning, using HATP [Alili et al., 2009]. This planning ability is meant to help teamwork between humans and robots [Lemaignan et al., 2017]. But in these experiments,

the robot relies on a lot of high-quality data to recognize the task being performed by the user. It is not currently suitable for embedded computing, and for open scenarios that may involve numerous classes.

See table 2.2 for a summary.  stands for “number of learning shots”,  stands for “supports composition”,  stands for “interoperability”,  stands for “compatible with open scenarios”,  stands for “can run on embedded computer”. We also looked in our literature for demonstrations of the explainability of the models, but found none.

2.4.2 Choosing a Behavior Model

Our review shows that the choice of **behavior model** has a tangible impact on the learning algorithms, on the content of the interaction, and also on the **cognitive system**. However, according to our criteria, our review did not reveal any champion model. There is no widespread **behavior model** either, despite the prevalence of procedural models.

To choose our **behavior model**, the first thing to consider may be whether teaching targets low-level **skills** or high-level **tasks**. Learning both is demonstrated in [Mohseni-Kabir et al., 2019], but it requires too much data, and a tight scaffolding.

For a robot like Pepper in real conditions, learning low-level skill from external demonstrations is difficult: the robot does not capture accurate 3D data, nor has a powerful GPU for feature extraction. **Self-demonstrations** are however possible, and were demonstrated in Pepper Play, an internal prototype at SBRE. Something like what [Cakmak & Takayama, 2014] demonstrated would be possible.

Given good speech recognition abilities, teaching high-level tasks from instructions appears to provide more possibilities, and has been demonstrated for a while [Lauria et al., 2001]. Similar symbolic models appear better suited for embedded computing and open scenarios. They can also support **composability** [Rybski et al., 2007], and interoperability [Lallée et al., 2012].

However we identified another model, **HTN**, that is symbolic and hierarchical (supporting **composability**), but is not restricted to procedural **behaviors**. In combination with a planner, like HATP [Alili et al., 2009]¹⁰, it is possible to automatically deduce a task plan from the definition of a final state, like it is done in [Forbes et al., 2015, Grizou et al., 2014]. This goal-oriented approach has also been identified in [Beetz et al., 2010].

But why would a goal-oriented approach be important, if task plans are already properly defined by the user? Fixed plans assume a perfect predictability of the outcome given an initial situation. But the world is only partially observable, so it can be considered non-deterministic from the performer’s point of view. Goal-oriented **behaviors** support the re-evaluation of the task plans, or at least of the current task, when the world state changes. This makes them more:

¹⁰In addition, HATP support synchronization with the **behaviors** of other agents. That makes of it a good extension of [Lallée et al., 2012]’s shared plans.




Publication					
[Lauria et al., 2001]	0	?	No	?	Yes
[Nicolescu & Mataric, 2003]	1	No	No	?	Yes
[Rybski et al., 2007]	0	Yes	No	?	Yes
[Rybski et al., 2008]	0	?	No	?	Yes
[Weitzenfeld & Dominey, 2009]	0	No	No	No	No
[Lallée et al., 2010]	0	?	Yes	?	Yes
[Lallée et al., 2012]	0	Yes	Yes	?	Yes
[Salvi et al., 2012]	>1	No	No	No	No
[Grizou et al., 2013]	>1	No	No	No	Yes
[Cakmak & Takayama, 2014]	1	No	No	?	Yes
[Grizou et al., 2014]	>1	No	No	No	Yes
[Grizou, 2014]	>1	No	No	No	Yes
[Forbes et al., 2015]	0	No	Yes	No	Yes
[Gemignani et al., 2015]	0	?	No	?	Yes
[Sorce et al., 2015]	0	?	No	?	Yes
[Petit & Demiris, 2016]	0	Yes	No	?	Yes
[Mohseni-Kabir et al., 2017]	1	?	No	No	?
[Scheutz et al., 2017]	1	?	?	No	No
[Scheutz et al., 2018]	1	Yes	?	No	No
[Mohseni-Kabir et al., 2019]	1	?	No	No	No

Table 2.2: Some properties exhibited by the [behavior](#) models and learning techniques used in the teaching of [behaviors](#) using [spoken language](#).

Robust They have an opportunity to avoid failures by compensating unexpected state changes.

Efficient They may skip tasks that do not contribute to reaching the desired tasks.

Reusable By expliciting their [goals](#) and effects, [behaviors](#) or their sub-tasks may be reused to participate for other [goals](#).

The evaluation of the current task, or of the current task plan, is traditionally performed using symbolic planners. The dominant model family for planning is STRIPS [Fikes & Nilsson, 1971]. In the International Planning Competition, they use the [Planning Domain Description Language](#) (PDDL) to describe STRIPS problems. It was also used in [She et al., 2014], that supported the teaching of [behaviors](#) using non-spoken natural language. HTN is a long-standing alternative to PDDL, and HATP is a STRIPS planner.

In our literature, this goal-oriented approach was rather solved using MDP [Grizou et al., 2013, Grizou et al., 2014, Grizou, 2014, Forbes et al., 2015] applied to low-level [skills](#). Indeed, MDP is capable of providing a task to perform, given an initial state, a [goal](#) state, and a reward function. MDP is commonly used to solve probabilistic planning problems, but requires training by reinforcement. It is also applicable to learn low-level skills.

Bringing together low-level skills and high-level tasks is a hot topic [Maestre, 2018, Gottstein, 2017, Mohseni-Kabir et al., 2019, Yamada et al., 2016, Xu et al., 2018]. It would allow primitive [actions](#) not to be predefined in robots. This kind of approach has also lead to novel cognitive architectures [Chatila et al., 2018]. Learning skills with minimal prior knowledge is in fact a great challenge in the robotics community [Chernova & Thomaz, 2014, p.50]. The work of [Xu et al., 2018] on Neural Task Programming is an impressive step towards that. They combine a hierarchical state machine with neural networks (an LSTM for the sequence of [actions](#) and a CNN for the parameters) to learn a large variety of [behaviors](#) with few teachings.

2.4.3 Conclusions on Behavior Models

This section reviews which [behavior](#) models are used to teach [behaviors](#) using [spoken language](#). Table 2.2 summarizes that some [behavior](#) models support hierarchical composition of [behaviors](#), interoperability with shared knowledge, [zero-shot learning](#), running on embedded computers, support open scenarios, but none demonstrated supporting all of these constraints together.

A large part of the [behavior](#) models are symbolic, and are bootstrapped with pre-defined associations between tasks and natural language. Few of them rely on a goal-oriented approach. The goal-oriented approach would result in more robust, efficient and reusable [behaviors](#). This approach is not incompatible with hierarchical symbolic [behavior](#) models, and is usually implemented using STRIPS planners. But in our literature, it was only encountered for learning low-level [skills](#), rather

than high-level [tasks](#). The encountered models do not satisfy all the properties we are looking for: [zero-shot learning](#) on embedded computers, [composability](#), interoperability, and compatibility with open scenarios.

In the next section, we focus more on the question of open scenarios, in a review of the experimental conditions found in our literature.

2.5 Experimental Setups and Evaluation

From the beginning (2.2), we highlighted the need for [HRI](#) research to be grounded on situations as real as possible. With Pepper@Home we aim for observing [HRI](#) at home, in people’s everyday life. More generally, the product Pepper targets a large variety of public places. Many of these situations forbid the installation of extraneous devices in the environment. Ideally, the robot should work stand-alone, as a turn-key product.

This constraint makes many findings from our literature difficult to apply. In this section we stress this with a review of the experimental setups, the experimental protocols and the evaluation methods, used for teaching [behaviors](#) using [spoken language](#).

We demonstrate whether (and why) many experiments cannot be directly re-played in real conditions, by checking whether the evaluation methods are transposable, whether extraneous elements are needed, whether the participants can be put in the same conditions in the wild, and whether the interaction scenarios were open¹¹. We intend to justify the novelty of our approach, and our choices in terms of evaluation methods, to the light of the state of the art.

2.5.1 Robotic Embodiment

Like in the rest of this chapter, we focus on the literature presented in table 2.3. We are looking for robotic embodiments that may be compatible with experiments in real conditions, to help identifying the experiments that could be most probably be transposed in real conditions. It is a way to evaluate the novelty of our approach, here on the spectrum of the robotic embodiment.

The literature presents a good variety of robots. We identified 11 different robots in the 20 publications exhibiting the teaching of [behaviors](#) using [spoken language](#).

However, the sensors were often external to the robot. Typically, the microphone: only [Lauria et al., 2001] – surprisingly the oldest experiment – used the on-board microphones of the robots to perform speech recognition. We investigate further this finding in subsection 2.5.2.

For vision too, many robots were extended with external sensors: depth sensors, motion capture devices, and sometimes an additional computer to perform heavy computation. These extensions are not often viable for a robot deployed in real conditions.

¹¹The question of richness was studied in subsection 2.2.2

The list of experiments The following publications do not involve external sensors, apart from the microphone: [Nicolescu & Mataric, 2003, Rybski et al., 2007, Rybski et al., 2008, Lallée et al., 2010, Lallée et al., 2012, Cakmak & Takayama, 2014, Forbes et al., 2015, Gemignani et al., 2015, Scheutz et al., 2017, Scheutz et al., 2018].

In theory, all of these experiments could be run in real conditions. But it is possible that the technical details of the robotic embodiment make it difficult.

It is possible that handmade robots like the ones shown in [Rybski et al., 2007, Rybski et al., 2008, Forbes et al., 2015, Gemignani et al., 2015], or even iCubs from [Lallée et al., 2012, Petit & Demiris, 2016] are not enough encased to face the general public.

It is also possible that big robots like HRP-2 (in [Lallée et al., 2010]), or PR-2 (in [Cakmak & Takayama, 2014, Forbes et al., 2015, Mohseni-Kabir et al., 2017, Mohseni-Kabir et al., 2019, Scheutz et al., 2017, Scheutz et al., 2018]) are too expansive to risk regular deployments in the wild.

And when cheaper robots are used, like NAO in [Sorce et al., 2015]¹² or AIBO in [Weitzenfeld & Dominey, 2009], they are extended with external sensors.

Anyway, the fact that cheaper robots are not used for teaching behaviors using spoken language certainly denotes there is a technical challenge in using them.

Therefore, not only demonstrating such teaching in real conditions is novel, but also using cheaper robots to do so may constitute a contribution.

2.5.2 Use of Microphones

In the previous review, we identified that only [Lauria et al., 2001] used microphones embedded in the robot. In the other studies, there was an external microphone provided to the participant, often in the form of a headset¹³.

In [Gemignani et al., 2015, Forbes et al., 2015], the speech recognition had to be started using a push button. This is not suitable for social robots in real conditions.

However it is understandable that if a study does not require to be performed in real conditions, it is not required to respect this constraint, which adds risks for the study’s success: indeed, speech recognition requires good audio quality to perform well. Also, with good audio input, it is possible to one-shot-train a recognizer for unknown words, like it was done in [Scheutz et al., 2017].

Indeed, working with poor-quality audio signals is a challenge, but it is more important than what it may seem. We humans do not have perfect audio perception neither. We compensate the noise with heuristics, and resolve phonetic ambiguities based on the categories of phonemes we learned from our past experience [Kuhl et al., 1992]. This is called “categorical perception” [Harnad, 1990]. According to

¹²NAO is also found in the literature on the teaching with the natural language in [Yamada et al., 2016]

¹³It was often difficult to assess how audio capture was performed. It seems so common in the community that the papers do not state it. It was sometimes in the illustrations and the side media that the information leaked.

[Oudeyer, 2013], these are in fact cultural phenomena. Therefore we tend to have a perceptual bias on what we are told, influenced by higher-level reasoning. This higher-level reasoning is related to how we manage dialogue-based interactions. A social robot should in theory be capable of such reasoning, and reusing its conversational expertise to improve its speech recognition. Taking up this challenge would also demonstrate the advancement of the conversational skills of the robot.

2.5.3 Openness of Experimental Scenarios

The scenario includes the script which must be followed by participants in an experiment, as well as what the **behavior** experimenters expect from the robot. When the script does not strictly impose a **behavior** to the participant, the participant has some choices in his or her interaction with the robot. The more choices participants are free to have, the more open the scenario is.

On the other hand, the stricter the script, the more predictable the participants are, and the simpler the robot's **behavior** is. It is also easier to prove technical achievements scientifically with strictly-controlled experiments.

In the literature on teaching **behaviors** using **spoken language**, there was in fact little room for initiatives of the participants. There was always a fixed number of tasks to teach. Despite the tasks being more or less diverse, and despite the apparent flexibility of the tested systems, there was no test against unexpected teachings. For this reason we conclude that most of the experimental scenarios are closed.

In [Lauria et al., 2001] and [Forbes et al., 2015], there was not enough details to conclude for sure. But we cannot assume it was open unless it was proven to be, so we also ruled them out.

However we found open scenarios in the broader literature about teaching **behaviors** using natural language, in [Tenorth et al., 2010] and [Nyga, 2017]. They worked on a large set of instructions provided from [wikiHow Incorporation, a] or [wikiHow Incorporation, b]. These online databases gather articles describing in natural language how to do things, written by humans desiring to share their know-how. Because of this, the human authors did not receive instructions specifically for these experiments. [Tenorth et al., 2010] selected 64 articles to teach their system, but then [Nyga, 2017] selected 8,400 of them. By design their system have a predisposition to support open scenarios, in which many teaching contents are possible.

Scenarios can also be made more open if the cognitive system can react properly to the wide variety of interactions found in real conditions. This variety of **behaviors** is what we call the “richness”. It remains a challenge to demonstrate a cognitive system that supports a rich set of **behaviors** in real conditions.

2.5.4 Evaluation

The evaluation methods depend strongly on what is meant to be demonstrated. There was a significant part of the literature on teaching **behaviors** with **spoken language** that only focused in demonstrating that the system did what it was programmed for: [Lauria et al., 2001, Nicolescu & Mataric, 2003, Rybski et al., 2007, Rybski et al., 2008, Lallée et al., 2010, Lallée et al., 2012, Petit & Demiris, 2016, Mohseni-Kabir et al., 2017, Scheutz et al., 2018]. This is a straightforward and qualitative way to demonstrate that a model works. It is usually good to bootstrap new systems, but it does not provide comparative results against other similar models.

In theory, qualitative methods could be used to compare experiments, in order to assess progress or distinguish features, but we found none in that literature. Nonetheless we identified quantitative methods, especially when the research focused on machine learning algorithms, in [Salvi et al., 2012, Grizou et al., 2013, Grizou, 2014, Grizou et al., 2014, Petit & Demiris, 2016]¹⁴. However the performance criteria was often based on the convergence of the algorithm, instead of on the actual **behavior** of the robot. We found that [Yamada et al., 2016]¹⁵ did better by evaluating the similarity of the output **behavior** with a test one.

But the evaluations on the interaction are probably the most interesting to evaluate the teaching of social robots. [Cakmak & Takayama, 2014] did the most thorough study, with a large population, balanced in age and gender, to evaluate the success of the interaction in different conditions (the instructions provided to the participants). They used NASA-TLX [Hart & Staveland, 1988] that measures the ease for performing a task, the teaching in this case. NASA-TLX is recommended by [Chernova & Thomaz, 2014], and can compare to the **System Usability Scale (SUS)** [Brooke, 1986], but differs because it focuses on performing a task, whereas **SUS** provide a coarser-grain evaluation of the usability of a solution.

[Weitzenfeld & Dominey, 2009] introduce an objective measure of the teaching quality: the average training time. [Gemignani et al., 2015] also provide objective measures, that include the number of words, error rates, time of teaching, and difference between the taught **behavior** and the one intended to teach:

- The minimum number of instructions required to teach a specific task (**Min**).
- The average number of instructions not recognized by the automatic speech recognition (**Err**) In this measure, only the instructions that the ASR could not process were considered.
- The average number of instructions misrecognized (**Mis**) by the natural language understanding.

¹⁴[Petit & Demiris, 2016] present both a qualitative evaluation that the system works, and quantitative evaluations on the performance of the machine learning algorithms.

¹⁵This publication is not part of our literature because their uses *written* words for input.

- The average number of corrections needed to modify a wrongly learned task (**Cor**).
- The average time in seconds needed to teach a specific task when no errors or misrecognitions were encountered (**AT no Err**).
- The average time in seconds needed to teach a specific task when errors or misrecognitions were encountered (**AT w Err**).

— [Gemignani et al., 2015]

[Forbes et al., 2015] also provided objective measures of the number of words required for the teaching, and of the time the teaching takes. [Sorce et al., 2015] did also provide a measure of the time for teaching.

As a conclusion there is a recommendation to use NASA-TLX, given that the scenario allows it, for subjective evaluation of the teaching interaction. There is no recommendation for objective measures, but it is common to measure the time for teaching, and the number of words required. Given that two experiments show the teaching of the same behaviors, it could be possible to compare them. But in the current state, experiments are too different to compare.

2.6 Conclusion

To conclude this review chapter, let us recapitulate the take-aways.

Humans naturally use instructions to teach behaviors to each other. This is a form of transfer learning, and it relies mostly on semantics of the natural language.

Pragmatics play a major role by scaffolding the teaching, and depend on the form of interaction. In the literature on teaching robots using the natural language, we distinguished a group of studies that involved spoken language, because they exhibit HRI. We found that the forms of interaction are usually rigid, and that there was a lot of prior pragmatic knowledge. Ideally, the teaching should be able to occur in the middle of other kind of interactions to be used in realistic conditions.

We reviewed the techniques that make NLI possible, with a focus on the speech recognition. We learned that to support an open scenario, we would need a grammar-free recognizer, and perform NLU on transcriptions of the speech.

The cognitive systems of the literature always distinguished NLI from the procedural knowledge, whereas social robots should be capable of being taught new sequences of NLI. We stressed the need for cognitive systems to become more extensible and to allow interoperability between extensions.

We highlighted that symbolic behavior models were suitable in the long run, and that making them hierarchical and goal-oriented eased the reusability of the behaviors. It remains possible to use a connectionist approach to refine the symbolic behaviors in the future.

We studied the experimental setups, and found that no experiment was really transposable to real conditions. The need to use a separate microphone is the major cause for this, but the use of expansive robots or extraneous devices also caused this.

For now, there are no experiment demonstrating the support for open scenarios, like the wild. Finally, we summarized the evaluation methods, and highlighted some that we could reuse in our research.

The next chapter describes our first experiments, that already go beyond the state of the art: we support the teaching of **behaviors** using **spoken language** in real conditions, with nothing other than Pepper robots, in an **open scenario**, with a cognitive system theoretically capable of being extended. It relies on grammar-free speech recognition to build hierarchical symbolic tasks, and is evaluated subjectively and objectively against the users and the taught **behaviors**.

Table 2.3 was compiled to summarize the features found in the state of the art, and compare them to our two publications, [Paléologue et al., 2017, Paléologue et al., 2018]. “Spoken”(.)“Spoken Language”, “Compos.”(.)“Behavior Composition” and “Open”(.)“Open Scenario”.

Publication	Spoken	Compos.	Open	HRI
[Crangle & Suppes, 1994]	No	No	No	No
[Huffman & Laird, 1995]	No	Yes	No	No
[Lauria et al., 2001]	Yes	No	Yes	No
[Nicolescu & Mataric, 2003]	Yes	No	No	Yes
[Rybski et al., 2007]	Yes	Yes	No	Yes
[Rybski et al., 2008]	Yes	No	No	Yes
[Weitzenfeld & Dominey, 2009]	Yes	No	No	Yes
[Arie et al., 2010]	Yes	No	No	Yes
[Lallée et al., 2010]	Yes	No	No	Yes
[Tenorth et al., 2010]	No	Yes	No	No
[Lallée et al., 2012]	Yes	No	No	Yes
[Mohan et al., 2012]	No	No	No	No
[Salvi et al., 2012]	Yes	No	No	Yes
[Mohan et al., 2013]	No	No	No	No
[Grizou et al., 2013] [Grizou et al., 2014] [Grizou, 2014]	Yes	No	No	Yes
[Cakmak & Takayama, 2014]	Yes	No	No	Yes
[She et al., 2014]	No	Yes	No	No
[Mohan & Laird, 2014]	No	No	No	No
[Forbes et al., 2015]	Yes	No	No	No
[Gemignani et al., 2015]	Yes	No	No	Yes
[Sorce et al., 2015]	Yes	No	No	Yes
[Petit & Demiris, 2016]	Yes	Yes	No	Yes
[Yamada et al., 2016]	No	No	No	Yes
[Mohseni-Kabir et al., 2017] [Mohseni-Kabir et al., 2019]	Yes	Yes	No	Yes
[Nyga, 2017]	No	Yes	Yes	No
[Paléologue et al., 2017]	No	Yes	Yes	Yes
[Saponaro et al., 2017]	No	No	No	Yes
[Scheutz et al., 2017]	Yes	?	No	?
[Suddrey et al., 2017]	No	No	No	No
[Scheutz et al., 2018]	Yes	Yes	No	?
[Paléologue et al., 2018]	Yes	Yes	Yes	Yes

Table 2.3: Publications exhibiting teaching behaviors to robots using natural language. They may not necessarily exhibit HRI or spoken language. [Paléologue et al., 2018] is the first to meet all the studied characteristics.

Behavior Composition for a Social Robot in an Open Scenario

Contents

3.1	Extraction of Social Cues with Pepper	48
3.1.1	Awareness of Humans	48
3.1.2	Engagement Assessment	49
3.2	Natural Language Understanding from Speech-to-Text Output	49
3.2.1	Understanding the Natural Language	49
3.2.2	Semantic Expressions	51
3.2.3	Semantic Memory: a Language-Oriented Database	53
3.2.4	Natural Language Generation from Semantic Expressions	54
3.3	Interaction for Teaching Behaviors	55
3.3.1	Extracting Task Teachings	55
3.3.2	Formal Model of Taught Behaviors	56
3.3.3	Interaction Scenario and Patterns for Teaching	59
3.4	Selecting Interactive Behaviors for Teaching Tasks	62
3.4.1	Using a Static Behavior Priority List	62
3.4.2	Using Independent Action Suggestions and Interaction Rules	63
3.5	Proof-of-concepts and Experiment in Homes	65
3.5.1	Preliminary Experiment	66
3.5.2	Simple Task Composition with Manual Transcription	66
3.5.3	Simple Task Composition with Automatic Speech Recognition	69
3.5.4	Task Composition in Home Conditions	74
3.6	Conclusion	77

In this thesis, we demonstrate **Pepper** robots that can be taught new **behaviors** by users, in **open scenarios**. This ability is supported by a novel **cognitive system**, that is able to support a rich set of robotic **behaviors**, provided by separate applications¹.

In this chapter, we introduce that **cognitive system**, and demonstrate that we reach the following objectives:

- Supporting the teaching of new **behaviors**, by composition of **task plans** using **spoken language**.
- Supporting an **open scenario**, where unanticipated **behaviors** may be taught.
- Deploy it to Pepper@Home users, using software based on **NAOqi**.
- Merging a small amount of **behaviors** together, using simple rules of pragmatics.

We take advantage of what chapter 2 taught us about the state of the art:

- Natural language instructions are a natural and straightforward way to transfer **behavior** knowledge.
- Semantic models are good to represent natural language instructions.
- There exist pragmatic models to represent the interaction, and they can be leveraged to disambiguate the purpose of the interaction.
- A proper combination of SR and **NLU** enables the support of unanticipated phrases.
- **NLI** should be provided by the same kind of **behaviors** that can be taught to the system. Dialogue management should not be done by a separate system.
- No other experiment demonstrates an open teaching in real conditions, with no extra device other than the robot.

We build a **cognitive system** to run experiments of growing complexity: from a preliminary experiment focused on the teaching, and with no automatic speech recognition to an experiment in Pepper@Home, where robots have some richer **behavior**, and run without the help of the experimenter.

The first sections of this chapter are ordered approximately according to the flow of information displayed in figure 3.1. Section 3.1 details how engagement is achieved with human users. Then section 3.2 details the semantic analysis of the natural language input for **NLU**, and the challenges we faced switching to **spoken language**. We also detail there how **NLG** is performed, Section 3.3 details the logic of the teaching interaction used in the experiments. Then, in section 3.4, we focus on the decision-making component of the cognitive system, with the intent of supporting **behaviors** provided by external applications.

¹See section 1.4 for details on our objectives

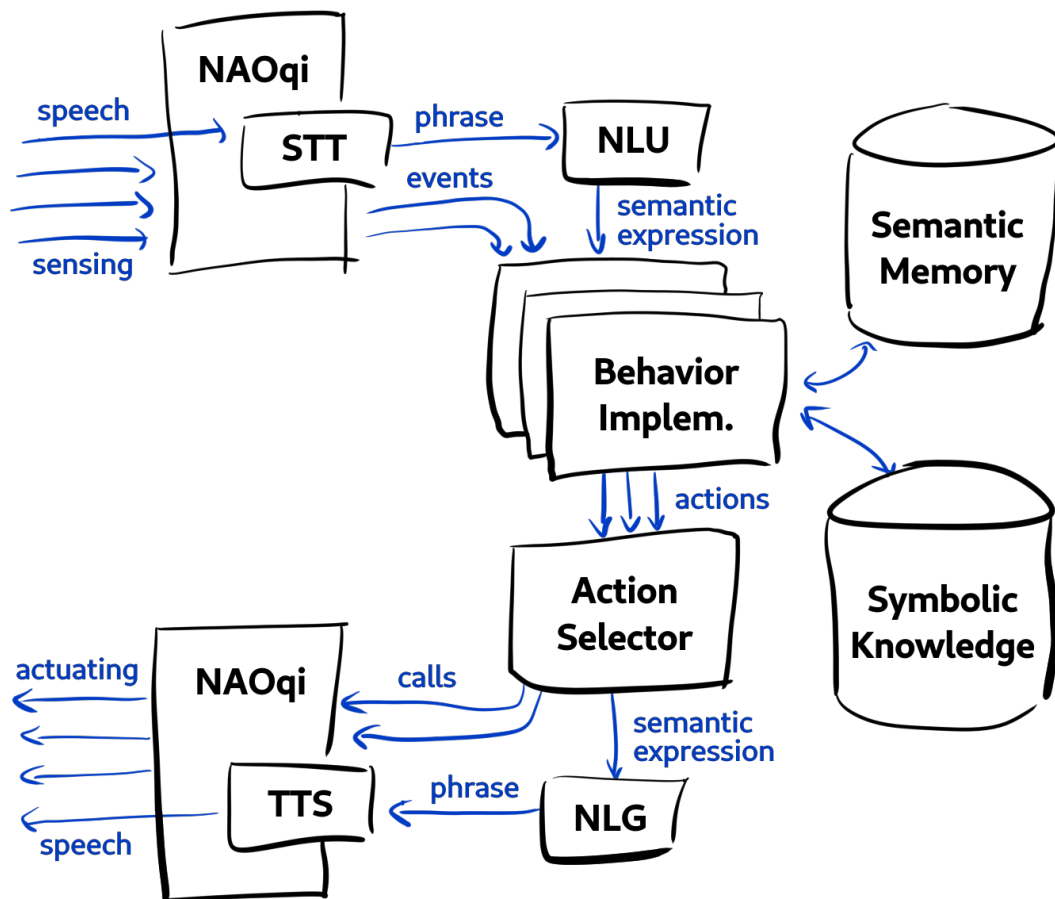


Figure 3.1: Overall information flow within the cognitive system. NAOqi is the interface with the environment, see annex 5.5.6. STT, NLU, NLG and TTS are introduced in section 2.3. Behaviors implementations produce the reasoning of the system: they react processed inputs (events and semantic expressions) may exchange data with databases (semantic memory and symbolic knowledge) and produce NAOqi actions in response, that may be run if selected.

Finally, the last section focuses on the experiments we performed, and evaluates whether the system is effectively capable of:

- Being taught new [behaviors](#) using the [spoken language](#).
- Accepting [behavior](#) labels that are not known *a priori*.
- Reusing formerly taught [behaviors](#) in newly taught [behaviors](#).

This is achieved on a stand-alone Pepper robot at home, with no additional external microphones, and in a cognitive system that exhibits other [behaviors](#) than the teaching interaction itself. These experiments were also an opportunity to learn details on the [NLI](#) side of [HRI](#), like how people naturally split the instructions for the robot to understand them.

3.1 Extraction of Social Cues with Pepper

In subsection [1.1.1](#) we mentioned that Pepper provided features for the detection of humans, and for engaging with them. In this chapter, we rely solely on these built-in features to extract social cues. In this section, we detail which features we use, and the kind of social inputs our system receives.

3.1.1 Awareness of Humans

Pepper can see the humans going around it. As soon as an object appears to stand out from the background, Pepper tries to assess if it can be a human by checking its size, its shape, and by trying to find a face in its upper part.

When Pepper is confident about seeing a human, a human Qi Object (see annex [5.5.6](#) about [NAOqi](#)) is published, and provides access to the following data:

- The 3D location and orientation of the face of the human.
- A 2D picture of the human's face.
- An estimation of his or her gender.
- An age estimation.
- Whether he or she is smiling.
- An estimation of his or her mood and excitement.
- Some indication on where the human's attention is drawn to.
- The human's apparent engagement intent.

This last information on engagement is important to establish a dialogue with humans, and is developed in subsection 3.1.2.

By default, Pepper automatically looks at the humans around. This [behavior](#) always runs in the background, unless a conflicting action performed (for instance, looking at some object). It makes the robot seem aware of them, but also invites humans to engage with it.

3.1.2 Engagement Assessment

When a human draws his or her attention back to Pepper, the engagement is effective. It is assessed by the robot by combining the dynamic of the displacement of the human (going towards the robot) with the direction of his or her attention (towards the robot). [\[Anzalone et al., 2015\]](#) offer a good overview of the techniques used to perform this assessment.

When the engagement is effective, the [speech recognizer](#) is configured to focus especially on the human’s direction, so that to maximize its accuracy. At the same time, Pepper tries to maintain the engagement by maintaining the eye contact (but not too much to avoid gaze aversion), and by orienting its body towards the user.

Disengagement is assessed by body and head movements away from the robot, but also by spotting formal engagement closures, like the utterance of “good bye”. The disengagement evaluation is the outcome of [\[Youssef et al., 2019\]](#)’s work.

In this chapter, we rely completely on these features to assess whether a human is engaged, and express engagement back to him or her. But we do not try to distinguish humans.

In this section we detailed some social cues that we use to engage with users. The features are provided by [NAOqi](#) and are used as is in our research.

3.2 Natural Language Understanding from Speech-to-Text Output

The robotic system has to perform [NLU](#) on the user’s instructions and teachings. Since the scenario must remain open, it cannot rely solely on the tuning of the [speech recognizer](#). Instead we choose to use [STT](#), and perform [NLU](#) on the text result, as explained in 2.3.2. This section presents the tools we use to perform [NLU](#). The semantic expression, the semantic memory, the [NLU](#) and the [NLG](#) algorithms are provided by Jocelyn Martin, software engineer at [SoftBank Robotics Europe \(SBRE\)](#).

3.2.1 Understanding the Natural Language

The [NLU](#) algorithms use dictionaries: for the English language, we use the DELA dictionary from [\[Klarsfeld & Mc Carthy Hammani, 1991\]](#); for the French language, we use the one from [\[Courtois, 1990\]](#); They were downloaded the website of the Lab-

oratoire d'Informatique Gaspard-Monge², and modified when useful for the NLU. They are made accessible by a component named LINGUISTIC DATABASE.

The NLU consists in several operations:

1. First, we chunk the input phrase into a list of words. We use the LINGUISTIC DATABASE to identify groups of words that are exceptionally meaningful together.

E.g.: "greet her" $\xrightarrow{\text{chunk}}$ ("greet", "her")

2. Then, all potential PoS tags are listed for each chunk, ordered by descending probability. Using arbitrary language-specific rules based on N-grams, we rule out impossible combinations. Potential PoS tags may be re-ordered too.

E.g.: ("greet", "her") $\xrightarrow{\text{PoS}_{\text{tag}}}$ ("greet"^{VB}_{root,0}, "her"^{PRP}_{dobj,-1})

3. The dependencies are computed between chunks using N-grams. Only the most probable PoS tag of each chunk is considered. If the computed dependencies appear impossible, related PoS tags are eliminated, and the dependencies are computed again.

E.g.: ("greet"^{VB}_{root,0}, "her"^{PRP}_{dobj,-1}) $\xrightarrow{\text{dependencies}}$ ("greet"^{VB}_{root,0}, "her"^{PRP}_{dobj,-1})

4. Named entities are resolved into semantic symbols. Times, places, agents and arbitrary concepts can be identified.

E.g.:

$$\begin{aligned} & ("greet"^{VB}_{root,0}, "her"^{PRP}_{dobj,-1}) \xrightarrow{\text{semantic}} (me, her, greet), \\ & \quad \text{Agent}(me, self) \wedge \\ & \quad \text{Agent}(her, undetermined) \wedge \text{Word}("her"^{PRP}, her) \wedge \\ & \quad \text{Statement}(greet) \wedge \text{Word}("greet"^{VB}, greet) \wedge \\ & \quad \text{Time}(present, greet) \wedge \text{Request}(action, greet) \wedge \\ & \quad \text{Subject}(me, greet) \wedge \text{Object}(her, greet) \wedge \end{aligned}$$

In figure 3.2 the semantic information is presented in an arbitrary textual form we use in development.

5. Coreferences, expressed by pronouns or implicit, are detected, resolved and developed if the context allows it.

E.g.: without a previous input, we add *coreference*(her, unknown_direction). With the previous input "here is Alice", we replace *agent*(her, undetermined) with *agent*(her, alice), *word*("Alice"^{NNP}, alice).

²<https://infolingu.univ-mlv.fr/DonneesLinguistiques/Dictionnaires/telechargement.html>.

3.2.2 Semantic Expressions

All the results are represented together into a single object, called “semantic expression”³. The semantic expressions can be of one these types:

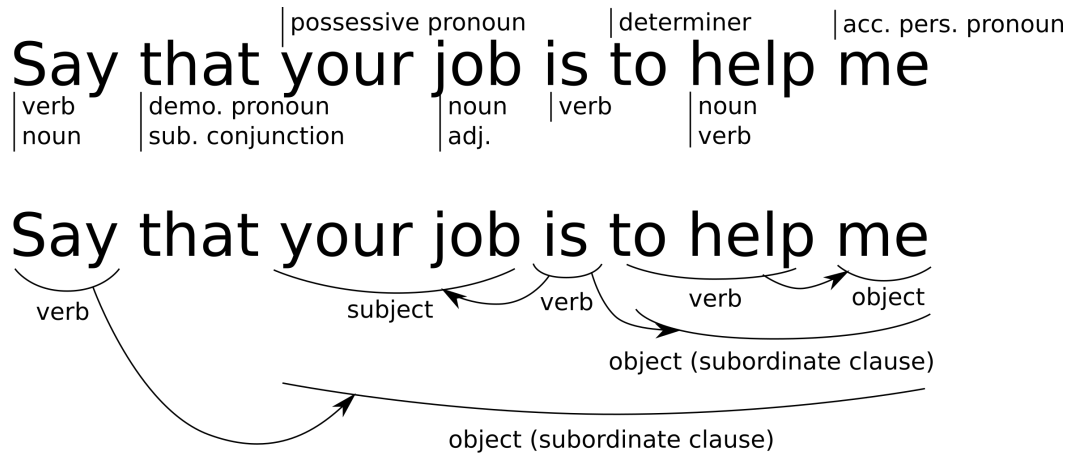
- Grounded on:
 - a statement associating subject, verb and complements of time, place, object, manner, etc...
 - a noun, such as “a banana” or “the Eiffel Tower”.
 - an agent, like “Alice”.
 - a time indication, like “14:00”.
 - a location indication, like “in Paris”.
 - a symbolic notion, like an *agreement*
 - a resource, interpretable by a program.
- Enumeration, with potential conjunctions like “and”, “or”, “then”, etc.
- Conditionals, using conjunctions like “if”, “then”, “else” or “when”.
- Feedback, like “yes” or “okay”.
- Coreferences, like “it”, “him”, or an implicit relation to another statement.
- Comparatives, like in “Paul is taller than Jack”.

In figure 3.2, we illustrate the work of NLU on the sentence “say that your job is to help me”. The root expression is grounded by a statement. The statement is centered on the verb “to say”, at the imperative form. The statement holds for the object complement, an expression interpreted from the text “your job is to help me”. It is grounded by a statement, centered on the verb “to be”, at the present form. The subject is an expression grounded on the noun “job”, which is owned by the recipient of the message. The object is another expression grounded a statement, centered on the verb “to help” at the infinitive form. The object of this statement is an expression grounded on the agent who produced the sentence. Note that statements are adequate to express [semantic frames](#).

By design, semantic analysis does not require pre-configuration. It accepts a large variety of grammatical structures, using any word from the dictionaries. Therefore it is able to support open scenarios, where the input texts cannot be pre-determined.

The semantic expression is a form of [semantic structure](#), can actually be used for machine translation. It can also be stored in a specialized memory, called “semantic memory”.

³In [Paléologue et al., 2017], where this was first presented, we mentioned a “semantic knowledge”. The semantic knowledge was a wrapper around semantic expressions, that allowed them to be stored in and retrieved from the semantic memory (see subsection 3.2.3) This wrapper is not technically useful anymore, so we do not mention it anymore.



```

from:
  extractor: extractorType(written_text) language(en_US)
            fromText("say that your job is to help me")
  author:   userId(currentUser) concept(agent_*, 4)
  time:     +(01/01/2000 00:00:00)
expression:
  statement: word(say, verb) request(action) time(present)
            concept(verb_action_say, 4)
  subject:   userId(me) concept(agent_*, 4)
  object:
    statement: word(be, verb) time(present)
              concept(verb_equality, 4)
              concept(verb_mean, 3)
    subject:   word(job, noun) ref(definite)
              quantity(nb, 1) type(human)
              concept(agent_profession, 3)
    owner:     userId(me) concept(agent_*, 4)
    object:
      statement: word(help, verb)
                concept(sentiment_positive_*, 2)
      receiver:  userId(currentUser)
                concept(agent_*, 4)

```

Figure 3.2: Intermediate and final results of the NLU. Final result is a semantic expression. Adapted from [Paléologue et al., 2017], figure 2.

3.2.3 Semantic Memory: a Language-Oriented Database

The semantic memory is a database of semantic expressions. From this point of view, the use of the term “semantic” seem superfluous, because usual databases can already store semantic symbols. Let us recall that semantic expressions are in fact a representation of grammatical dependencies, augmented with semantic information. What makes a difference with other symbolic structures is in fact its grounding on linguistics.

Therefore, manipulating semantic expressions consists as much in grammatical manipulations as in symbolic manipulations. A database capable of storing, retrieving and querying such data would therefore be characterized more by its affinity with language, than with its affinity with symbols.

The semantic memory is such a database: it leverages the language-oriented structure of semantic expressions to perform queries. It is a language-oriented database.

The storage consists in an enumeration of every semantic expression put in it. As much as possible, semantic expressions are developed before storage. For example the equivalent to “Alice likes pandas and Bob” would be stored as “Alice likes pandas” and “Alice likes Bob”.

Each expression corresponds to a pointer in storage. If requested with an equivalent expression to “Alice likes pandas”, the semantic memory returns that pointer. This is how the direct retrieval of information is done.

Queries can be expressed using semantic expressions. If the semantic expression represents a question, the semantic memory should be able to respond with a semantic expression representing a valid natural language answer.

It supports some usual forms of questions:

- Yes / No, *e.g.*: “Does Alice like Pandas?” produces “Yes”.
- What, *e.g.*: “What does Alice like?” produces “Alice likes pandas and Bob”.
- Who, *e.g.*: “Who likes Bob?” produces “Alice likes Bob”.
- When, *e.g.*: “When does Alice sleep?” produces “Alice sleeps at midnight”.
- Where, *e.g.*: “Where does Alice eat?” produces “Alice eats at the canteen”.
- Why, *e.g.*: “Why does Alice smile?” produces “Alice smiles because she is happy”.

Thus, the semantic memory has the power of expressing arbitrary facts, without requiring an actual semantic representation of them. It is a powerful tool for writing more open dialogues, but also to let end users teach things to the robot.

Nonetheless, semantic information is carried to the semantic memory. This allows inference, so that to support the following dialogue:

HUMAN (H): Alice likes pandas.

H: I am Alice.

H: What do I like?

ROBOT (R): You like pandas.

It is also notable that using a semantic representation within the language-oriented information has an intrinsic effect of perspective taking.

For example the inputs “Your job is to help me” and then “What is your job?” should produce “My job is to help you.”

3.2.4 Natural Language Generation from Semantic Expressions

In the previous sub-section, we referred to semantic expressions using their textual equivalent. In fact, a semantic expression can be translated back to natural language in various ways. The NLG algorithms we use for this are described in this sub-section, using the classification we reported in 2.3.3:

Architecture The algorithms are arranged as a pipeline, from the input (a semantic expression), to the output phrase.

Content Determination The content is determined by the reasoning algorithms that decide the *behavior* of the robot. The NLG algorithms are not involved in this part.

Document Structuring This is also mostly the reasoning algorithms’ responsibility. The NLG algorithms are not meant to significantly alter the meaning of the input semantic expression.

Lexicalization There is a specific algorithm for each type of semantic expression. For enumerations for example, it consists in translating each sub-expression, separated by commas. Between the penultimate and the last sub-expression, the link word is used instead of the comma, *e.g.* “and”, “or”, “then”...

For statements, we take the verb (identified by its lemma) and conjugate it with the person deduced from the subject. Then, from the verb’s lemma, we look up for the right conjugated form. For each type of complement, and according to the verb’s grammatical constraints, we select the preposition before expressing the complement.

Using the dictionaries, we also apply language and word-specific rules. *E.g.* the negation of “can” is “cannot”, and not “can not”.

Referring Expression Generation Semantic entities like agents may be resolved to find a word or a phrase that identify them, from the semantic memory. Whereas places are stored directly in the semantic expressions. Time information is re-expressed using specific rules.

Aggregation The semantic expression is respected, so no extraneous aggregation is performed by the NLG.

Realization The NLG algorithms are hard-coded and called recursively on the content.

Thus, anything that can be expressed with semantic expressions can be translated to natural language. However, the generated phrases may sometimes be imperfect, or not idiomatic. We assume the impact on the understanding should remain negligible.

It is also possible to generate a phrase in a different language. With the help of an additional dictionary of translations, we can use the NLG for the other language, and produce understandable phrases. This achieves a form of translation by semantic transfer, according to [Dorr et al., 1999]’s classification. It is not a powerful translator, and it will not be evaluated, but it potentially allows the content of the semantic memory to remain useful even when the language changes.

In this section we detailed the algorithms we use to perform NLU and NLG. We use a custom data structure to manipulate natural language phrases. Some semantic information is extracted from the phrases, such as entities and semantic frames. This information can be collected in a custom database, specialized for language-oriented information.

3.3 Interaction for Teaching Behaviors

In this section we explain how we use our NLU to build the interaction for teaching behaviors. We detail our behavior model and explain why we think it is sufficient.

As announced in section 1.2.6, behaviors can be modeled as task plans, and be reused by composition. To do so, behaviors must be viewed as tasks, to be reusable in other task plans. We use this model for taught behaviors.

In this chapter, task plans are simple sequences of tasks. Since tasks can be made of task plans, this is a hierarchical model that resembles the behavior networks used in [Rybski et al., 2007, Rybski et al., 2008]⁴. However we have no support for conditional branching.

A “task teaching” is the natural language description of a task and of its associated task plans.

3.3.1 Extracting Task Teachings

In subsection 2.1.2, we show that instructions are a straightforward way of teaching. We choose this approach because it is feasible without the need of advanced sensors required to observe a human’s behavior.

Instructions describe tasks, so that they can be performed. For example “say hello” is an instruction. They can be put in context of a larger task, for example “to make a diabolo, put syrup, and then put sparkling water”. A parent task “to make a diabolo” is decomposed into two instructions “put syrup” and “put sparkling water”.

⁴We identified this model in subsection 2.4.1

We call “task label” the expression declaring the parent task. We call “task description” the set of instructions defining the task. We call “task teaching” the proposition associating a label to a description. A task can be described without a label, but in our work, we choose to make sure taught tasks can be reused in other tasks to teach. For this reason, we do not support task descriptions alone, dissociated from a label.

The semantic expressions introduced in 3.2.2 can represent task teachings. Interpreted altogether, the teaching is unambiguous. However in the experiment presented in subsection 3.5.3, we demonstrated how users would split their teachings into several utterances.

But when the instructions of the task descriptions are taken separately, they can be confused with direct orders, because they are expressed in the imperative form. The context is required for disambiguation. In order to avoid the confusion in practice, we only support instructions expressed in the infinitive form. To remain linguistically correct, the form of the task teaching changed to: “to make a diablo *is to* put syrup and then *to* put sparkling water“. Figure 3.3 shows the full semantic expression extracted for that example.

A teaching can be identified by the verb “to be” at the present tense, associating a task label and a task description. A task label is a statement in the infinitive form. A task description is an enumeration of statements in the infinitive form. A statement in the infinitive form is called a “task declaration”. It is associated to a [semantic frame](#)⁵.

Task declarations can constitute a task label, or be part of a task description. Task descriptions made of a single task declaration are supported, but cannot be distinguished from the task label in a teaching. We call these corner cases “task aliases”.

We do not detail how conditional or event-based instructions should be handled. They can be added later and enrich the task description.

3.3.2 Formal Model of Taught Behaviors

Behaviors taught this way are task plans. A task corresponds to anything that can be executed⁶ by the robot. The functional content of a task may be hard-coded. Such a task is called “primitive”⁷. Or its functional content can be taught, for instance through the interaction we propose in previous subsection 3.3.1.

Task descriptions are the expression of tasks in natural language. Their content – a verb and optional complements – correspond to a [semantic frame](#) [Fillmore, 1985]⁸. We consider that a task t can be associated to a [semantic frame](#) s . This

⁵By construction, any verbal statement corresponds to a [semantic frame](#). Hence a “task declaration” also corresponds to a [semantic frame](#).

⁶In our work, tasks are moreover visibly performed by the robot, *i.e.* they are [actions](#).

⁷Primitive tasks should not be confused with [action primitives](#). A primitive task can be implemented with an action primitive, but can also be implemented by a combination of high-level [tasks](#).

⁸[Semantic frames](#) were introduced in previous chapter, in subsection 2.3.2.

```

statement:
  word(be, verb) time(present)
  concept(verb_equal_be, 4)
  concept(verb_equal_mean, 3)
subject:
  statement:
    word(make, verb) concept(verb_action, 4)
object:
  word(diabolo, noun) ref(indefinite)
  quantity(nb, 1) type(thing)
object:
  listType(then)
    -> statement:
      word(put, verb)
      object:
        word(syrup, noun) ref(definite)
        quantity(nb, 1) type(thing)
        concept(concrete_*, 4)
    -> statement:
      word(put, verb)
      object:
        word(water, noun) ref(definite)
        type(thing) concept(liquid_*, 4)
      specifier:
        word(sparkling, adjective)
        type(modifier)
        concept(sentiment_positive_*, 3)

```

Figure 3.3: Semantic expression for a sample task teaching: “to make a diabolo is to put syrup and then to put sparkling water”.

association is represented using the predicate $Express(s, t)$. The set of all possible tasks and semantic frames are respectively T and S .

Our teaching interaction is based on a set of known tasks $T_{known} \subset T, n \in \mathbb{N}, |T_{known}| = n$. Each task is associated to a **semantic frame**: $\forall t \in T_{known}, \exists s \in S, Express(s, t)$.

Learning a task $t_{learned}$ from a sequence of known tasks consists in updating T_{known} as follows:

$$\begin{aligned} \forall k \in \mathbb{N}, \exists t_0, \dots, t_k \in T_{known} \\ t_{learned} = (t_0, \dots, t_k), \exists s \in S, Express(s, t_{learned}) \\ T_{known}^{i+1} = T_{known}^i + t_{learned} \end{aligned}$$

In other words, we define a new task $t_{learned}$ as a tuple (t_0, \dots, t_k) of known tasks, and expressed by a **semantic frame** $s \in S$, and put it in T_{known} . Here, this tuple defines the **task plan**.

To bootstrap the system, we define a set of primitive tasks $T_{primitive}$,
 $T_{known}^0 = T_{primitive}, |T_{primitive}| = n_{pri}$
 $\forall t \in T_{primitive}, \exists s \in S, Express(s, t)$

The set of learned tasks $T_{learned}$ contributes to the set of primitive tasks to constitute the set of known tasks, $T_{known} = T_{primitive} \cup T_{learned}$.

The space of possible plans to learn must be large enough so that teachings are not deterministically limited. This is a precondition for supporting **open scenarios**. Given the length of task plan $1 < k < k_{max}$ ⁹, and the number of primitive tasks n_{pri} , there are initially n_{pri}^k possible plans. In total, that makes: $|T_{learnable}^0| = \sum_{k=2}^{k_{max}} n_{pri}^k$. In practice, we do not expect task plans made of more than 5 tasks. Given 10 primitive tasks, the initial number of learnable tasks would be:

$$|T_{learnable}^{i0}| = \sum_{k=1}^5 10^k = 111100$$

When a task composition $t_{learned}$ is learned, it does not add up to n_{pri} , but to n . However, the task is composed of k sub-tasks, and can be used as a shortcut to express longer task plans. It seems that the number of learnable tasks usually grows when a task composition is learned¹⁰. The teacher may not be limited by the size of the space of learnable tasks, and does not limit the application of the **behavior model** to closed scenarios.

As a conclusion, our hierarchical task model allows a quick exploration of all the possible tasks a robot could do. We assume that the limitations of the model may rather consist in:

- the difficulty of defining semantic frames for each task,
- the inadequacy between the known tasks and the tasks users want to teach,

⁹The length of a task plan may be limited by the patience required to teach it to the robot.

¹⁰This could be demonstrated with a simulation on a large random draw of **behaviors**.

- the absence of complex articulations between tasks, such as concurrency or conditionals,
- the lack of support for parameters.

The 3 first limitations do not actually concern our research, and are therefore acceptable. The last limitation impacts the interoperability between [behaviors](#), and is addressed in chapter 4.

3.3.3 Interaction Scenario and Patterns for Teaching

Our objectives, announced in section 1.4, include supporting the openness of the home scenario, and the richness of [behaviors](#) expected of social robots. Pepper@Home is a program organized by SBRE, that lends Pepper robots to employees, in exchange of participating to some experiments. Deploying our system to Pepper@Home robots is part of our objectives.

Pepper@Home robots have basic [behaviors](#) implemented:

- Pepper’s standard [behavior](#) presented in subsection 1.1.1.
- The engagement [behavior](#) presented in subsection 3.1.1.
- General-purpose dialogues including greetings, self-introduction, telling jokes, singing songs, performing dances, and chit chat on various subjects like sports or politics. It is brought by a chatbot written by Yufo Fukuda, distributed through an application called ABC.
- Miscellaneous [behaviors](#) found in other applications, such as a Sport Coach or a game where Pepper imitates animals.

Applications are implemented for NAOqi OS 2.9, which lets the Android system running on the tablet drive the robot¹¹. Certain Android components can be put in the foreground (*a.k.a.* taking the focus) and take control of the screen and of the robot. These components are called Activities, and drive the interaction phases in NAOqi OS 2.9.x.

When ABC runs – or more precisely when its dialogue activity takes the focus – the robot listens to the engaged human, and may react according to the script of a chatbot, whose features are described above. The [behaviors](#) provided by the other applications are not available from ABC, and *vice versa*. For us, ABC is a good starting point for open and rich interaction scenarios, so we have chosen to extend it for our research purpose. (see subsection 3.5.4).

ABC displays blue light indicators to express it is listening to the human. When the robot detects the end of an utterance (based on a timeout after speech is not heard anymore) the listening pauses (along with the feedback), the [speech recognizer](#)

¹¹Applications like “Do This Do That” or “Pepper Play” mentioned in subsection 1.1.3 were made for NAOqi OS 2.5, and are not compatible with NAOqi OS 2.9.

provides textual results, letting the dialogue engine provide a reaction, after which the listening resumes. This mechanism fulfills a role of *speech turn*.

As seen in subsection 2.3.4, providing a separate dialogue system creates constraints on the cognitive system: dialogue contents are privileged in comparison to other behavioral contents, and *behaviors* are preferably driven in a turn-based manner, as responses to the human's input.

To avoid such distinction between dialogues and the other *behaviors*, we removed the forced *speech turn* mechanism: the robot never pauses listening, and may be interrupted at any moment. However the *speaking floor* can still be assessed: when the human finishes talking, a dialogue-based *behavior* can assume the speaking floor is given to the robot, and gives it back to the human when its response was produced.

We expect the teaching of tasks to be possible at any moment during this activity, without compromising the existing activities.

We assume the users may already know how to interact with the robot, before he or she is expected to teach the robot. Users are not expected to be naive. It is noteworthy that some of them are experienced with the robot.

But we do not assume they should guess how to teach the robot. It is useful to teach the users how to teach the robot, and this has been taken advantage of.

We require the users to utter task teachings like explained in subsection 3.3.1. For each utterance contributing to a task teaching, the robot acknowledges, so that the user can go on with the task description or labeling. When the teaching is complete, the robot recalls the whole task. Dialogue 3.1 displays an example teaching, with a task description first, followed by the task label.

HUMAN (H): To move forward.

ROBOT (R): Ok.

H: And to say welcome.

R: Ok.

H: Is to welcome.

R: Ok, to welcome is to move forward and to say welcome.

Dialogue 3.1: Example dialogue for teaching a task plan, terminated by the labeling of the task plan.

We assume that starting and stopping a teaching may not be explicit, because the system may become able to learn tasks during other activities, even though we do not demonstrate such ability in this research. And because of the structure of task teachings, it appeared feasible to recognize it in a context-free manner.

However, if the teaching starts with the task label and ends with a task description, like in 3.2, some ambiguity arises: the task description is an enumeration, and humans do not necessarily end an enumeration with the word “and” coordinating the elements. In that case, the robot should ask explicitly whether the teaching is complete, by saying “is that all?”.

HUMAN (H): To welcome.
 ROBOT (R): Ok.
 H: Is to move forward.
 R: Is that all?
 H: And to say welcome.
 R: Is that all?
 H: Yes.
 R: Ok, to welcome is to move forward and to say welcome.

Dialogue 3.2: Example dialogue for teaching a task plan, started by the labeling of the task plan.

User	Robot
1. Input	2. Learning
	3. Confirm

Table 3.1: Usual [pragmatic frame](#) involved in our teaching of tasks.

Once a task is learned, it is available to execute when ordered to, using the imperative form of the task label. It is also immediately reusable for new teachings. There is no way of correcting the task, but the task can be replaced with a new one.

Tables 3.1 and 3.2 describe the interaction patterns involved in the teaching, expressed using the formalization from [\[Vollmer et al., 2016\]](#).

The scenario and the forms of interaction are not especially rich, however it is designed to be open: taught [behaviors](#) and tasks in plans can be referred to using any verb existing in the dictionaries.

In this section we detailed how task compositions can be taught. Using the semantic analysis on carefully designed instructions, the robot can build a structure representing a task composition, recursively. Then we detailed the interaction pattern we designed for such task teachings.

User	Robot
1. Input	2. Learning
	3. Input query
4. Input	5. Confirm

Table 3.2: Pragmatic frame for asking whether the task description is complete.

3.4 Selecting Interactive Behaviors for Teaching Tasks

By design of the interaction, the teaching **behavior** may enter in conflict with the other **behaviors** of the robot, including the **behaviors** taught by users. The cognitive system should therefore be able to decide which **behavior** is best to perform. We call this task selection.

In this section, we describe how we implemented this mechanism in our first set of experiments. We propose two simplistic solutions, that were sufficient in our simple scenario.

Task selection require that tasks are collected prior to be executed. That is to say that the components implementing the **behaviors** must not directly send commands to the robot, but instead should wrap them in a task, and suggest them to the task selector. A component autonomously suggesting tasks to perform is called a “**behavior**”.

3.4.1 Using a Static Behavior Priority List

Our first cognitive system supporting task selection was presented in [Paléologue et al., 2017], along with figure 3.4. It shows our semantic analysis pipeline (see 3.2) and a reasoning block made of a list of reasoners, in the form of boxes with deep blue background. They attempt to understand the input *a posteriori*, and therefore do not affect each other (as they would if they attempted to configure the SR *a priori*).

Each reasoner accepts a semantic expression for input, and may respond with another semantic expression. Since semantic expressions can be interpreted (see subsection 3.2.4), responses are regarded as task suggestions, and reasoners are regarded as **behaviors**. This equivalence between tasks and semantic expressions allows the “Introspect Behavior” component to describe in natural language any tasks the system has learned: taught **behaviors** are explainable by construction.

The task selection relies solely on the order in which **behaviors** are triggered. We loop over the list of **behaviors** and check for a response for one after the other, If a **behavior** responds, the next **behaviors** are not solicited. The last **behavior**, “Tell if not understood”, always responds that the input was not understood. We often mention it as the fallback **behavior**.

Taught **behaviors** are triggered by the “Obey Order” **behavior**, and therefore are always of higher priority than the other **behaviors** of the system. In case of conflict, if two taught **behaviors** are associated to the same **semantic frame**, the most recently taught **behavior** is triggered in priority.

In the experiments detailed in subsection 3.5.2, we did not encounter any issue related to these technical choices. However we predict that in richer scenarios, conflicts may arise, including between taught **behaviors**. We expect that for these richer scenarios, we may need a more advanced model for task selection.

Indeed, this model does not respect the constraint of independence between **behaviors**. For instance the “Tell if not understood” **behavior** is adequate only after

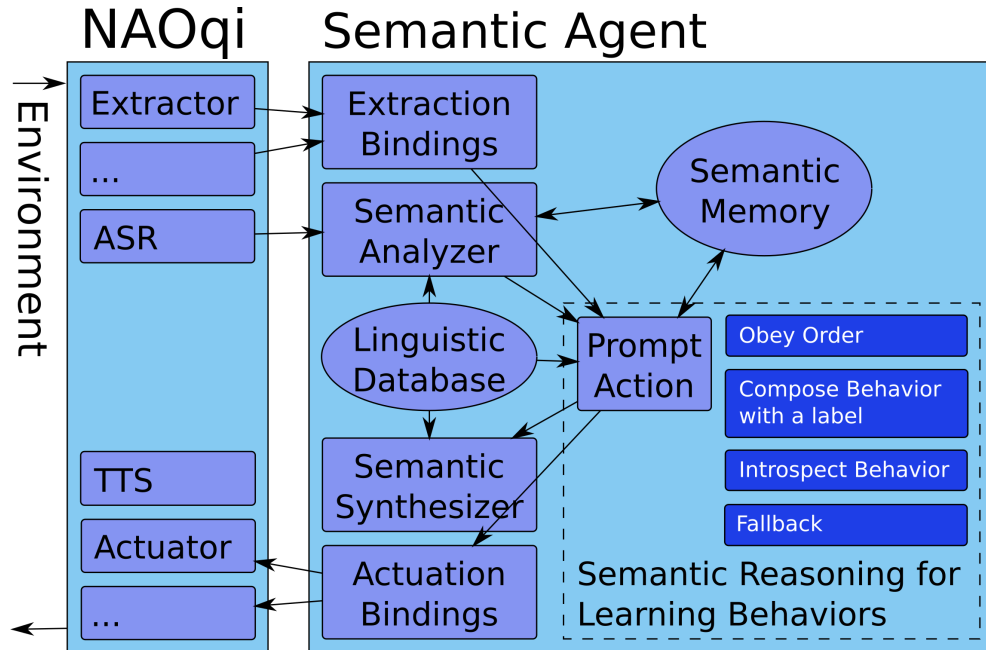


Figure 3.4: Overview of the software architecture with **behaviors** ordered by priority.

— [Paléologue et al., 2017]

the other **behaviors** were triggered. If it was put on top of the priority list, it would make the robot respond that it does not understand, even if the next **behaviors** provide responses based on some understanding.

Finally, triggering **behaviors** one after the other supposes restricting their reaction to a certain form of input. In terms of software architecture, it is better let **behaviors** decide by themselves what they react to, and therefore desolidarize the flow of suggestions from the flow of semantic expressions.

3.4.2 Using Independent Action Suggestions and Interaction Rules

[Paléologue et al., 2018] propose several changes in the cognitive system. The reactions are not produced in a push mode: the **behaviors** are not called directly anymore. Instead, they subscribe to knowledge events – this is called a pull mode – and produce tasks in reaction. Therefore **behaviors** react separately, whereas in [Paléologue et al., 2017] the **behaviors** were only triggered if the previous ones did not respond positively. This difference is illustrated in figures 3.4 and 3.5: in the former, the prompt action centralizes the events and triggers **behaviors** according to its priority list, whereas in the latter, each **behavior** implementation receives knowledge events, with no intermediary.

The tasks are not represented by semantic expressions anymore. Instead, each task encapsulate an arbitrary piece of program. When a task is started, the originating **behavior** is informed, and may execute this piece of program. Figure 3.5

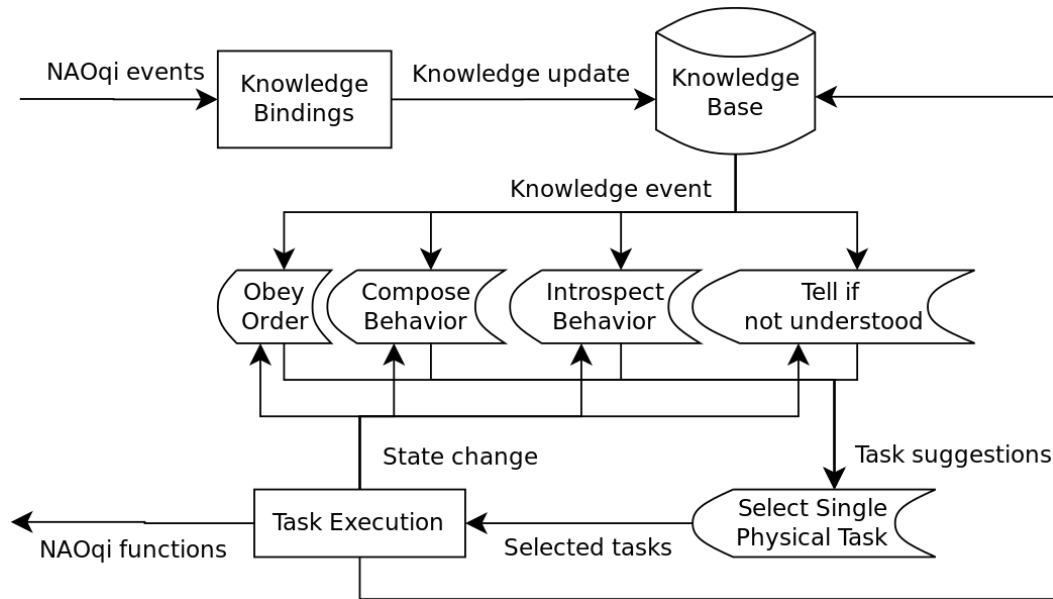


Figure 3.5: Overview of the software architecture with **behaviors** suggesting tasks in reaction.

— [Paléologue et al., 2018]

illustrates this with a state change information propagated from the task execution component to the **behavior** implementations.

Knowledge events are changes in a triple store¹². These events may represent **spoken language** inputs. semantic analysis happens inside each **behavior**¹³, and are omitted in figure 3.5. Behaviors may produce tasks in response, and suggest them to the task selection, instead of performing them immediately.

The task selection consists in taking the first response suggested, for each speech input received. All other responses are ignored. Therefore speech inputs remain a privileged form of input, and action suggestions are meant to be responses to the last speech input. The main improvement is that **behaviors** can be designed more independently.

For instance the fallback **behavior** had to be adapted. It tracks what was heard and said by the robot, and if nothing is responded to some speech input within tenths of second, the **behavior** suggests a task uttering “sorry, I don’t understand”. This rule applies at the interaction level, and is generic enough to not be restricted on a certain topic of discussion.

This cognitive system is first used in the experiment presented in 3.5.3. Because there is no constraint in the organization of the **behaviors**, external **behaviors** can be accepted more easily. Namely, the chatbot used in Pepper@Home (see subsec-

¹²Triple stores are databases storing information in the form of RDF triples [RDF Working Group, 2014].

¹³The results of semantic analysis are cached so that it is not performed several times on the same input.

tion 3.3.3) can integrate the system as a separate *behavior*. This was done in the experiment presented in 3.5.4.

We propose an extension of the rule-based task selection in next chapter 4.4. In the next section, we describe how we put these task selection models in practice in experiments, and the results we produced.

This concludes this section on the novel architecture we propose to manage the task selection in a cognitive system that lets independent components, called *behaviors*, suggest any task at any time. Several task selection schemes are possible, and have been tested in experiments described in next section.

3.5 Proof-of-concepts and Experiment in Homes

In this section, we describe the experiments we held to gradually demonstrate the capabilities of our cognitive systems in the field.

For every experiment, we put together a Pepper with the latest software version (at the time), some *NAOqi* packages providing the service called “SemanticAgent”. The protocols, the binaries and the data analysis are available on GitLab¹⁴.

Participants were always employees of *SoftBank Robotics Europe* (SBRE). They know how to interact with Pepper robots, are professionally involved in the experiments, and live in Paris area, France. Therefore we refrain from generalizing our results to the rest of the population¹⁵.

For evaluation purpose, [Paléologue et al., 2017] propose a set of objective measures adapted from [Gemignani et al., 2015]¹⁶:

- The number of utterances (or instructions) provided by the users (*IC*).
- The average rate of instructions misrecognized (*Mis%*).
- The average time in seconds needed to teach a task, including the time lost with misrecognized instructions or any other error (*AT w Err*).
- The average normalized edit distance (*MED*) between the decomposition in natural language done by the robot and the expected one.
- The percentage of *behaviors* that were successfully taught, among the total number of *behaviors* attempted to be taught (*TS%*).
- The percentage of users who managed to teach *behaviors* (*UTS%*).
- The percentage of users who managed to teach composite *behaviors* (*UTSC%*).

In addition, [Paléologue et al., 2017] introduce subjective measures:

¹⁴<https://gitlab.com/victor.paleologue/teaching-robots-behaviors-experiments>

¹⁵It is not excluded that Pepper robots and their applications could be targeted to a population that is fluent with technology, or have similar professional stakes at play when interacting with Pepper. Like for computers, people may have to learn how to use robots.

¹⁶Original measurements from [Gemignani et al., 2015] can be found in subsection 2.5.4

- How successful the teaching felt. This is the Experienced Teaching Success (*ETS*).
- How easy the interaction felt. This is the Experienced Interaction Ease (*EIE*).

We use these measures to produce measurements in our experiments, in order to evaluate the performance of our system.

3.5.1 Preliminary Experiment

In a first attempt, we put together a version of the SemanticAgent with:

- a fake *STT* input, injectable using Choregraphe’s dialogue widget,
- a simplified semantic analysis, that only supported full teachings in one utterance.
- a simple task selection as described in 3.4.1,
- the 4 *behaviors* described in [Paléologue et al., 2017], subsection 3.2,
- an Android activity providing feedback on what is said or heard.

We solicited 3 *SBRE* employees that knew how to interact with the robot, but who were new to the experiment. We gave them instructions sheets themed by domain of application, among:

- Home
- Healthcare
- Business

The participants have 10 minutes to invent and teach as many *behaviors* as they can. See the protocol for more details online¹⁷.

The experiment immediately showed flaws in the semantic analysis. These flaws prevented the experiment to be held to the end. Also, we collected some feedback on how to improve the instruction sheets.

However it served as the first proof-of-concept that an *open scenario* could be supported by our system: we did not impose the *behaviors* to teach. Instead, participants had to imagine *behaviors* themselves.

3.5.2 Simple Task Composition with Manual Transcription

We setup a version of the SemanticAgent similar to the one used in 3.5.1, with some fixes in the semantic analysis. We provided an updated set of instruction sheets, as shown in 3.6.

¹⁷<https://gitlab.com/victor.paleologue/teaching-robots-behaviors-experiments/tree/master/1%20-%20Preliminary>.

Known Behaviors	Known Behaviors	Known Behaviors
<p>Say *</p> <p><Raise, Lower> your <arms, left arm, right arm, head></p> <p>Move <forward, back></p> <p>Turn <left, right, away></p> <p>Look <left, right, forward></p>	<p>Say *</p> <p><Raise, Lower> your <arms, left arm, right arm, head></p> <p>Move <forward, back></p> <p>Turn <left, right, away></p> <p>Look <left, right, forward></p>	<p>Say *</p> <p><Raise, Lower> your <arms, left arm, right arm, head></p> <p>Move <forward, back></p> <p>Turn <left, right, away></p> <p>Look <left, right, forward></p>
To teach a behavior	To teach a behavior	To teach a behavior
<p>In one sentence, using an <i>existing</i> verb : to + <verb> is to + <verbal group>, ...</p> <p>Examples :</p> <p>To greet is to raise the right arm and say hello</p> <p>To welcome is to move forward and greet</p>	<p>In one sentence, using an <i>existing</i> verb : to + <verb> is to + <verbal group>, ...</p> <p>Examples :</p> <p>To help is to raise the left arm and ask how you can help</p> <p>To care is to move forward and help</p>	<p>In one sentence, using an <i>existing</i> verb : to + <verb> is to + <verbal group>, ...</p> <p>Examples :</p> <p>To promote is to look forward and say that acme has great products</p> <p>To solicit is to move forward and promote</p>

Figure 3.6: Instruction sheets given to participants before starting the experiment. Red: home; green: health care; blue: business. — [Paléologue et al., 2017]

Measure	Home	Care	Business	All
IC	96	130	110	336
Mis%	32.82%	43.04%	21.57%	32.48%
At w Err	104.36 s	130.39 s	82.81 s	105.85 s
MED	0.04	0.21	0.11	0.12
TS%	53.64%	40.28%	80.95%	58.29%
UTS%	100%	100%	100%	100%
UTSC%	0 %	66,67%	66,67%	44,44%
ETS	2 / 5	4 / 5	3 / 5	3 / 5
EIE	2 / 5	2 / 5	3 / 5	2 / 5

Table 3.3: Measurements on the transcripts per theme, on the experiment presented in [Paléologue et al., 2017].

9 SBRE employees participated in the experiment. They did not know how to teach tasks yet, even though they may know how to interact with the robot. The full protocol and the collected data can be found online¹⁸. Table 3.3 summarizes the collected data.

The conclusions published in [Paléologue et al., 2017] are that the system supported the teaching of unexpected composite tasks, given a perfect STT. The constraints of reusability of behaviors within other behaviors, of zero-shot learning open scenario were respected.

The publication includes a user-centered open study highlighting the caveats of the interaction, such as the lack of support for more varied input, for more varied teachings, for parameters in behaviors, and errors in the semantic analysis.

In terms of system design, we mixed the task execution with the dialogue system by accepting executable chunks in the dialogue responses. But the dialogue system still drives the cognitive system, that did not exhibit autonomous behavior. Conflicts between behaviors were therefore avoided.

[Paléologue et al., 2017] also mention the domain-independence of the algorithms used. We assume that domains correspond to themes. Table 3.4 counts how well NLU performed on the user utterances, by theme. Using a χ^2 independence test it appears that we cannot conclude whether the NLU result is independent from the theme.

¹⁸<https://gitlab.com/victor.paleologue/teaching-robots-behaviors-experiments/tree/master/2%20-%20Transcribed>.

Theme	G	S	U	All
Business	67	16	5	88
Care	60	45	11	116
Home	52	21	7	80
All	179	82	23	284

Table 3.4: Contingency of NLU results by theme, for teaching behaviors with perfect STT. G stands for successful understanding, S for semantic analysis error, U for not understood by the behavior. Independence test produces $\chi^2 = 13.4$ and $p = 0.146$, which is inconclusive.

3.5.3 Simple Task Composition with Automatic Speech Recognition

A new version of the SemanticAgent was put together with:

- a real STT input, using Nuance’s service (available in Pepper robots) and a Microsoft online service as a fallback,
- a flexible semantic analysis that also allowed teachings to be split in several utterances,
- a task selector accepting independent action suggestions, with hard-coded rules, as described in 3.4.2,
- the 4 tasks described in [Paléologue et al., 2018].
- the same Android activity providing feedback on what is said or heard.

Switching from textual language to spoken language involves a difficult challenge. Speech recognizers automatically stop capturing audio when a pause is detected. In the case of teaching behaviors, people often pause between instructions, to check if the robot understands them.

The teachings are therefore split into several utterances. The system must be able to understand them in the context of the larger teaching. It has consequences on the NLU, that are detailed in subsection 3.3.1. It also has consequences on the interaction; they are detailed in subsection 3.3.3.

In this experiment, the following hypotheses are made:

1. Pepper’s default speech recognizer provides exploitable results for our teaching algorithm based on NLU¹⁹ and semantic analysis.
2. Users are all able to perform a teaching, given a short set of instructions.

¹⁹[Paléologue et al., 2018] rather mentions NLP, but NLU was meant.

Domain	IC	Mis%	At w Err	MED	TS%	UTS%	UTSC%	ETS	EIE
Home	181	59 %	180 s	0.86	26 %	100 %	0 %	2.3/5	2.3/5
Care	149	53 %	94 s	0.85	55 %	100 %	33 %	1.7/5	2.0/5
Business	150	54 %	105 s	0.40	52 %	100 %	0 %	2.3/5	2.3/5
All	480	55 %	111 s	0.68	46 %	100 %	11 %	2.1/5	2.2/5

Table 3.5: Measurements on the transcripts per theme (*a.k.a.* domain) on the experiment presented in [Paléologue et al., 2018]. The measure *Mis%* includes the speech recognition errors, when the text produced differed from the utterance. At the time it was considered different from *Err*, defined in [Gemignani et al., 2015] as the absence of recognition.

3. Users may naturally attempt to teach tasks in several sentences, and cut their sentences before or after “is to” and between steps of enumerations.
4. Users may naturally attempt to describe tasks first, and then label them, e.g.: “to raise the right arm and to say hello is to greet”.
5. Robotic algorithms written independently can be run in competition without conflict if: they can recognize the context they are relevant with, they suggest reactions of the robot instead of executing them, an algorithm selects the most appropriate reaction.

Hypotheses #1, #2 and #5 refer to our objectives. Hypotheses #3 and #4 were inspired by previous works like [Lallée et al., 2012, Gemignani et al., 2015]. In these studies, the users are forced to split down the teachings in similar ways.

We run the experiment on a new sample of 9 SBRE employees, who did not practice the teaching before, but who knew how to address the robot, but with varying levels of English. The full protocol and the collected data can be found online²⁰. Table 3.5 summarizes the collected data.

Hypotheses #1, #2 and #3 were verified. Hypothesis #4 and #5 were not verified, though not invalidated. Numbers show a high rate of misrecognition of the spoken language. Indeed, the speech recognition is imperfect, and introducing it added a new source of errors. But there are other impactful factors:

- the English level of participants,
- the accent of participants,
- the utterances to recognize are unusual for off-the-shelf STT engines,
- the misunderstanding of who owns the speech floor,

²⁰<https://gitlab.com/victor.paleologue/teaching-robots-behaviors-experiments/tree/master/3%20-%20Automatic>.

Experiment	MisA%	MisS%	MisU%	Mis%	At w Err	MED	ETS	EIE
N-1	0 %	30 %	9 %	39 %	84 s	0.52	3 / 5	3 / 5
N	34 %	11 %	10 %	55 %	112 s	0.68	2 / 5	2.2 / 5

Table 3.6: Comparison of the error measurements between the experiments presented in [Paléologue et al., 2018]. N-1 is [Paléologue et al., 2017], N is [Paléologue et al., 2018]. *MisA%* is the rate of misrecognition error due to the [speech recognition](#). *MisS%* is the rate of misrecognition error due to the semantic analysis. *MisU%* is the rate of misrecognition error due to the proper exploitation of the semantic structure.

Experiment	Mis	¬Mis	All
[Paléologue et al., 2017]	105	179	284
[Paléologue et al., 2018]	236	193	429
All	341	372	713

Table 3.7: Contingency of misrecognized (Mis) user utterances by experiment. [Paléologue et al., 2017] is the previous experiment, described in subsection 3.5.2. Independence test produces $\chi^2 = 22.3$ and $p = 1.76 \times 10^{-4}$. Misrecognition rate has increased in experiment #3.

- the cumulation of errors between interdependent utterances.

[Paléologue et al., 2018] present a comparative table highlighting the differences with the previous experiment. It is reported in table 3.6.

To corroborate the increase of misrecognition rates, we perform the statistical analysis reported in table 3.7. It shows that the misrecognition rate is significantly higher in this experiment than in previous experiment.

However there was an effort done to mitigate that increase, by improving the [NLU](#). To show whether that effort was fruitful, we perform the statistical analysis found in table 3.8. For this analysis, we have to omit the utterances that were not well recognized by the [speech recognizer](#), to compare with previous experiment where the [speech recognizer](#) was perfect. The result is inconclusive.

The publication also stated that we could not conclude on domain-independence. However, by looking at how well [NLU](#) performed by theme, it is possible to draw some conclusion. Assuming that domains correspond to themes, the analysis shown in table 3.9 demonstrates that the two variables are *not* independent.

Finally, we compare our results with [Gemignani et al., 2015] in tables 3.10, 3.11 and 3.12. We need to adapt the data to achieve this comparison. We get the number of [speech recognition](#) errors *Err*, corrections *Cor* and misunderstandings *Mis* by multiplying the published averages with the number of users, 5. We get the

Experiment	Mis(S+U)	\neg Mis(S+U)	All
[Paléologue et al., 2017]	105	179	284
[Paléologue et al., 2018]	85	193	278
All	190	372	562

Table 3.8: Contingency of user utterances misrecognized by **NLU**, by experiment. [Paléologue et al., 2017] is the previous experiment, described in subsection 3.5.2. Independence test produces $\chi^2 = 2.57$ and $p = 0.632$, which is inconclusive.

Theme	G	A	S	U	All
Business	45	43	6	4	98
Care	73	59	13	5	150
Home	75	49	22	35	181
All	193	151	41	44	429

Table 3.9: Contingency of **NLU** results by theme, for teaching **behaviors** with real **STT**. G stands for successful understanding, A stands for **speech recognition** error, S for semantic analysis error, U for not understood by the **behavior**. Independence test produces $\chi^2 = 35.0$ and $p = 4.65 \times 10^{-4}$. **NLU** results are not independent from the theme.

Experiment	IC	Err%	Mis%	At w Err
[Paléologue et al., 2018]	480	34 %	21 %	111 s
[Gemignani et al., 2015]	163	7.98 %	4.29 %	62.4 s

Table 3.10: Result comparisons between [Paléologue et al., 2018] and [Gemignani et al., 2015].

Experiment	Err	¬Err	All
[Paléologue et al., 2018]	163	317	480
[Gemignani et al., 2015]	13	150	163
All	176	467	643

Table 3.11: Contingency of [speech recognition](#) errors by experiment. Independence test produces $\chi^2 = 41.1$ and $p = 2.30 \times 10^{-8}$.

number of instructions *IC* by summing these numbers with the minimum number of instructions.

[Gemignani et al., 2015]’s *Err* is comparable with [Paléologue et al., 2018]’s *MisA* because both characterize the [speech recognition](#) errors. [Gemignani et al., 2015]’s *Mis* corresponds to the union of [Paléologue et al., 2018]’s *MisS* and *MisU*. Therefore the use of the measure *Mis* in [Paléologue et al., 2018] is misleading. Therefore we use the following definitions of specific measures:

- The number of instructions misrecognized due to automatic speech recognition (*Err*), and the corresponding rate (*Err%*).
- The number of instructions misrecognized due to the natural language understanding (*Mis*), and the corresponding rate (*Mis%*).

Measures like *Corr*, *At no Err*, *MED* or *TS%* are not relevant in either of the two publications, so they are ignored in this comparison.

[Speech recognition](#) accuracy appears lower in [Paléologue et al., 2018]. This is demonstrated significant in 3.11, with $p = 2.30 \times 10^{-8}$. The [NLU](#) accuracy appears lower too, but the difference was not demonstrated significant in 3.12. Note that [NLU](#) can only occur on instructions that were well recognized by [speech recognition](#). We cannot conclude either on teaching speed because we only have access to timing averages from [Gemignani et al., 2015].

[Speech recognition](#) accuracy differences may be explained by various factors:

- Users talk through an external, push-to-talk microphone they hold. See subsection 2.5.2.

Experiment	Mis	\neg Mis	All
[Paléologue et al., 2018]	29	288	317
[Gemignani et al., 2015]	7	143	150
All	36	431	467

Table 3.12: Contingency of NLU by experiment. Independence test produces $\chi^2 = 2.87$ and $p = 0.579$.

- The task to teach are pre-defined, so the [speech recognizer](#) can be pre-configured. See subsections 2.5.3 and 2.3.1.
- They used Microsoft Speech Platform, which could offer better performance than Pepper’s Nuance [speech recognizer](#).

Our problem forbids us to apply these simplifications. Our experiment shows that despite these constraints (no pre-defined [behavior](#), no extra device such as held microphone) it is possible to teach composite [behaviors](#) using solely [spoken language](#). Supporting an [open scenario](#), and avoiding the use of a microphone, makes our experiment novel, and proves the feasibility of this approach.

Our approach also includes a design of the cognitive system that does not distinguish dialogue from the other tasks (see subsection 3.4.2). This is also novel, and involved a simple rule to resolve conflict between the normal responses and the fallback ones. It should be tested in more depth in richer scenarios, where the robot is expected to exhibit autonomous [behaviors](#). This is what we do in chapter 4.

Before this, in next subsection, we deploy and test the current system in conditions close to real ones: in homes.

3.5.4 Task Composition in Home Conditions

Our objective is to test in real conditions the system demonstrated earlier in controlled conditions. We put together our software in Pepper@Home robots with:

- a real [STT](#) input (Nuance and then Microsoft as a fallback),
- a flexible semantic analysis that also allowed teachings to be split in several utterances,
- a task selector accepting independent action suggestions, with hard-coded rules, as described in 3.4.2,
- the 4 [behaviors](#) described in [Paléologue et al., 2018].
- a general purpose chatting [behavior](#),

- an Android application named DEF, forked from and resembling ABC, mentioned in subsection 3.3.3, providing feedback on what was said, and on what could be said. It provides an additional activity called “Teaching Experiment”.

We ask 5 participants from Pepper@Home to run the Teaching Experiment activity, and follow the steps. We recall these steps in the protocol of the experiment, published online²¹. They were not allowed to switch app during the experiment. Therefore they could not access to the miscellaneous behaviors found in other applications (animal game, sport coach, etc...). The experiment can be done in French or in English so that to rule out the language from the source of errors. Participants were asked to perform a teaching session with the robot: one phase of 10 minutes of teaching, one phase of interaction with the baseline behaviors only, and one phase of teaching with the baseline behaviors activated. However, due to technical problems in the DEF application, we did not manage to make sure a single full session could be completed. Nonetheless, in this subsection we detail the key points of the experiment.

Pepper@Home applications are usually evaluated using the System Usability Scale (SUS)[Brooke, 1986]. SUS produces a score from a set of 10 subjective questions on a Likert scale. We use it to evaluate the usability of the system for each session, so that we can evaluate whether combining the baseline behaviors and the teaching has an impact on the interaction.

When teaching is involved, we also collect the information required to produce the measurements specified in previous experiments, for further comparison:

- The dialogue history during the experiment, with timestamps.
- The taught behaviors as understood by the robot, in natural language.
- Audio-visual recordings, to check whether users effectively talked to the robot, and what they actually said.

Collecting this data remotely is also a challenge. Even though Pepper@Home users formally agreed that personal data can be collected and processed to improve the robot’s software, we must be respectful of their privacy by:

- Collecting only the required data.
- Transmitting it online without sharing it with third-parties.
- Processing and publishing only anonymized data.

SBRE hosts an online service to manage fleets of Pepper robots, called the Application Distribution Engine (ADE). The Pepper@Home fleet is managed using this tool: we can set up to which system version the robots should update to, and

²¹<https://gitlab.com/victor.paleologue/teaching-robots-behaviors-experiments/tree/master/3%20-%20Automatic>.

which NAOqi applications they should get. Once the participants trigger the updates, the robot retrieves and installs the selected software. On the other hand, the distribution of DEF, the Android application orchestrating the experiment, was less automated: a temporary download link was provided by e-mail to participants, so that they could download it from Pepper’s Android system, and install it manually.

Through the ADE, we distribute the SemanticAgent, providing the teaching behaviors to the robot, through a custom Qi SDK Chat action²². We also distribute the Multimodal Recorder (MTT), an internal tool at SBRE, developed by Robin Beilvert, designed to record raw inputs of the robot on demand.

When DEF starts a phase of the experiment, it configures and starts SemanticAgent’s chat action, as well as MTT, and stops them right at the end of the phase, ensuring that only the experimental phase is recorded. The recordings are left in a folder specific to our experiment, so that it is easily found by the service in charge of uploading them online. The recordings are deleted from the robot once they are uploaded. At the end of each phase, participants must fill in a questionnaire found directly in DEF. DEF collects the questionnaire replies, the dialogue history and the learned behavior from the SemanticAgent, and uploads them online as soon as possible, before deleting them.

The upload destination is a WebDAV server owned by SBRE, and that solely supports authenticated PUT requests on its public endpoint. The upload is encrypted end-to-end, and the uploaded data can only be retrieved from SBRE’s network, by the experimenter and the system administrator. Each has a different private key, in his unique possession. The data collection worked normally for this experiment, but the trials did not complete, resulting in incomplete data.

The collected data nonetheless show successful teachings, including composite ones. Despite the apparently high rate of misrecognition, the system seemed usable. The participants, who are home users of the robot, could discuss in the way they are used to with the robot, while also teaching behavior composition. This success is nuanced by the lack of comparable objective and subjective measurements to perform statistical analysis.

Finally, it is important to note how behaviors have conflicted during these experiments. Pepper’s standard behavior was always going on, including the engagement behavior, and was only interrupted when a physical action was requested to the robot: for instance asking the robot to turn around or to look away naturally interrupted the engagement, which was resumed as soon as possible.

General-purpose dialogues conflicted with the obey behavior: both responded to orders like “raise the left arm”. It appeared that the behavior whose implementation responded the fastest was the one to be selected. Since the obey behavior was running in the same process as the SemanticAgent, and that the general-purpose dialogue implied network communication to take place, the obey behavior was always privileged. This is a lucky coincidence that underlines the need for more advanced

²²https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/reference/chat.html.

rules for task selection in richer scenarios.

Another conflict has been identified in the collected data. The following piece of dialogue is an extract exhibiting it:

HUMAN (H): Être un porte-manteau c’est garder le bras levé.

ROBOT (R): Ok.

R: Je me sers surtout des mains pour exprimer mes émotions.

This piece of dialogue shows a double response of the robot to a teaching. It says “to be a coat hanger is to keep the arm raised”. This is a valid teaching, but it is not seen as a task association. Instead, it is seen as a single task, as part of a longer teaching. But at the same time, the base dialogue has programmed reactions on some keywords of the input. Both responses managed to be performed, revealing an unhandled conflict.

This concludes this section dedicated on experiments. We reviewed how we set up experiments of gradual difficulty, in order to reach our objective: having robots being taught *behaviors* in real conditions, in homes. We identified that our technical choices lead to a rate of *speech recognition* errors higher than in [Gemignani et al., 2015]. This may be an obstacle in the teaching interaction. This rate was proven dependent on the theme, between home, business and health care, despite the cognitive system being designed in a domain-independent manner. We also learned that participants use various ways to introduce the teachings, and expect more feedback from the robot. They also expect the *behaviors* to support parameters, or achieve a specific goal. All of this should be considered to improve the teaching of *behaviors*.

3.6 Conclusion

In this chapter, we presented our approach to achieve the teaching of *behavior* composition using *spoken language* on Pepper robots deployed in homes. Using *NLU* after *STT*, we extract a semantic structure that can be used to describe *task plans*. We formalized our *behavior model* and its ties with *semantic frames*. The interaction was modeled too, using *pragmatic frames*.

We presented a novel *cognitive system* that does not distinguish the dialogue responses from the other behavioral responses. This constraint simplified the collaboration of components designed independently from each other: their interface with the rest of the system consists in suggesting tasks, instead of trying to execute them directly. In return, we need to select the task to perform. This design inspired the new Chatbot API of the *Qi SDK*²³. Chatbots support a simple form of task selection. Using a simple and explainable rule to handle the fallback responses was enough for our scenarios because the general *behavior* of the robot was simple. We expect to need more rules to support richer *behaviors*, in a richer scenario.

²³https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/reference/chatbot.html.

The system was then demonstrated in controlled environments, and then in homes, with no external microphones, in an open scenarios, where the [behaviors](#) to teach were not predefined. The teaching interaction had to support teachings split down in several utterances, cut before or after “is to” (associating a label to a task plan) and between steps of enumerations (describing the task plan). However the [speech recognizer](#) has had difficulties with these conditions, leading to a significantly higher [speech recognition](#) error rate than in a comparable teaching experiment by [\[Gemignani et al., 2015\]](#). To achieve comparison, we needed to clarify the use of certain measures, also avoiding potential misinterpretations.

In next chapter, we push our system further. We adapt it to support parametric tasks, in the same kind of conditions: at home, in an [open scenario](#), on a robot that is designed to a richer set of interactions.

Teaching Behaviors in Rich Scenarios

Contents

4.1	Proposed Behaviors for a Richer Interaction	80
4.1.1	Discover a Point of Interest	82
4.1.2	Label a Point of Interest and a Location	83
4.1.3	Point at a Point of Interest	83
4.1.4	Locate a Point of Interest	83
4.1.5	Visit a Location	84
4.2	Proposed Ontology for Interoperability	84
4.2.1	Social Agents	85
4.2.2	Physical Objects	85
4.2.3	Locations, Areas and Maps	87
4.2.4	Events	87
4.2.5	Communicative Acts	89
4.2.6	Actions	89
4.2.7	Semantic Templates	90
4.2.8	Conclusion on Sharing Knowledge and Behaviors	91
4.3	Teaching Parametrized Behaviors	91
4.3.1	Extracting Parametrized Task Teachings	91
4.3.2	Formal Model for Parametric Behaviors	93
4.3.3	Interaction Patterns for Parametric Behaviors	94
4.4	Rule-Based Task Selection	95
4.4.1	Goal-Oriented Approach and Planning	96
4.4.2	Building Problems from Task Suggestions	97
4.4.3	Rules for HRI Tasks	100
4.4.4	Conclusion on Rule-Based Task Selection	102
4.5	Experiment	102
4.5.1	Experimental Protocol	102
4.5.2	Deployment in Pepper@Home	105
4.5.3	Results and Analysis	106
4.5.4	Conclusion on the Experiment	115
4.6	Conclusion	116

In previous chapters we established a system capable of providing a piece of **HRI**, through which users could teach **behaviors** to robots. They would proceed by composing **task plans** from known tasks, hard-coded or learned. The composition is therefore hierarchical, but remained limited to tasks with no parameters. It was integrated with some baseline **behavior**, and demonstrated the possibility of putting together separate **behaviors**. To achieve this, we implemented the **behaviors** as independent components: each would decide independently whether they should respond to a given situation. In return, they must not perform the responses themselves, but instead suggest tasks to the cognitive system, that would select the right one independently.

In this chapter we rely on the same system, but integrate the teaching with a richer set of **behaviors**. One of our main challenges is to resolve potential conflicts between them, so that they remain functional when put together. Another challenge is to allow them to contribute to each other. For instance if a new skill is provided by some **behavior** implementation, it should be usable in taught **behaviors** too. There is therefore a question of interoperability involved.

In previous experiments, we identified a limitation on parametric **behaviors**: most of **actions** encountered apply on other entities. In other words, tasks often required parameters. For instance, the robot could “say hello”, but not “say hello to Alice”. The robot could “greet”, but not “greet someone”. Worse, there are **actions** that are pointless without parameters, like “go to the kitchen”. Supporting parameters involves maintaining some knowledge on entities of the environment, and having a way to learn them.

For an open environment, entity knowledge cannot be predefined. Therefore a robot must be able to learn about these entities. We propose to let users teach these entities to the robot themselves. These teaching **behaviors** participate with the richness of **behaviors**.

We detail the set of new **behaviors** we introduce in section 4.1. Then in section 4.2 we detail how these new **behaviors** share the entity and procedural knowledge with the rest of the system. This allows the teaching of parametrized **behaviors** described in section 4.3. In section 4.4 we describe how all **behaviors** provide task suggestions, and how task selection is achieved: this is where conflicts are actually solved, using a rule system. Finally in section 4.5 we hold an experiment to validate our system.

4.1 Proposed Behaviors for a Richer Interaction

For 3 years, the Pepper@Home program have collected feedback on apps deployed at participants’ homes. Participants often highlighted the lack of understanding of:

- Space: it is limited to SLAM [Durrant-Whyte & Bailey, 2006], obstacle avoidance and going to the charging station on demand.
- Objects: the robot has no notion of object, cannot locate them, and refer to them.

- People: it is limited to localizing faces.

The example of space is interesting: using SLAM allows the robot to navigate, but the users have no way to talk about the spatial environment with the robot. If the robot was able to put words on the objects and the locations of its environment it can recognize, it could be referred to by users. If the robot accepts instructions applying to objects and locations, the user may be able to leverage this knowledge, and share with the robot a common understanding of objects and locations.

In this section we focus on the **behaviors** we developed to allow this knowledge of objects and locations to be produced and leveraged¹. The detection and the segmentation of objects is not available on Pepper. To simplify, we only identify what we call “points of interests”. A point of interest is an area in space being of higher visual salience, and distinguishable from the background. Each point of interest is considered a physical object.

The proposed **behaviors** allow the robot to, respectively:

- Discover a point of interest.
- Present a discovered point of interest and be told a label for it and for the location it was seen from.
- Point at a point of interest, by a hand gesture.
- Locate a point of interest, by going back to it.
- Visit a location.

Each **behavior** is described in a separate subsection, in a systematic manner, by mentioning:

1. The desired effect of the **behavior**.
2. The decomposition of the **actions** it consists in.
3. The knowledge produced (or destroyed).
4. The conditions for being triggered autonomously.

These **behaviors** are provided by a stand-alone Android application for Pepper, Pepper Explore, developed by Shin Watanabe, employee of **SoftBank Robotics Europe (SBRE)**. The application communicates with our system in the terms described in section 4.2, and thus provides new **behaviors** to the robot. We consider the total **behavior** of the robot significantly richer with this set of **behaviors**, because it provides new cases of conflicts with the **behaviors** presented in previous chapter, in subsection 3.3.3: labeling things is highly contextual, and must be brought autonomously to the user. The labels can then be reused when teaching **behaviors**, to produce parametrized **behaviors**, raising questions of interoperability.

¹It was initially planned to also include **behaviors** designed for learning people through greetings, and let users reuse this knowledge. It may be done in future works, but it has already influenced the ontology and the rules presented in this chapter.

4.1.1 Discover a Point of Interest

This **behavior** consists in exploring the surroundings, in order to discover new points of interest. It performs the following sequence of **actions**:

- Localize and map the immediate surroundings using Pepper’s standard navigation [API](#)².
- Look at a random directions to capture 2D and 3D pictures.
- Analyze pictures to identify a salient point of interest, and estimate its position: we use BRISK [Leutenegger et al., 2011] on the 2D image to identify keypoints. The keypoints are clustered using k-means, for 10 clusters. From the centroid of keypoints of the most dense cluster, we compute a fixed-size **Region of Interest (ROI)**. Using k-means on the depth map and for 2 clusters, we distinguish the foreground data and compute its centroid. This 3D location is remembered as the estimated location of the point of interest.
- Go to that location. The robot cannot physically reach it, so it stops as close as possible to it.
- Look at it and capture 2D and 3D pictures.
- Compare the former pictures with the newer pictures: we use BRISK and k-means to produce 10 clusters, and compare them with a brute-force BRISK matching, to find the most similar cluster. If the matching score is above a certain threshold, the point of interest is confirmed.
- Go back to the initial location.
- Look back to the initial direction.

When the **behavior** finishes successfully, the robot has stored a 2D picture representing the point of interest, the 3D transform of the robot in the world when the picture was taken, and the timestamp of the picture.

When in the context of our system, this **behavior** may suggest a task to perform the exploration autonomously. This happens only when the robot is idle, *i.e.* when no human seem having engaged with the robot for a while. We call this state “idling”, and say the exploration is an “idle task”. Thus, the exploration **behavior** takes advantage of the absence of moving obstacles to maximize its success. The robot explores autonomously every 15 minutes, and does not explore if it has 3 new points of interest (or more) in mind. As a result, the robot appears curious of its environment, but not too much, to avoid making it appear too excited or harassing.

²https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/movement/localization_library.html

4.1.2 Label a Point of Interest and a Location

When a new point of interest is identified, the robot needs to label it to be able to relate to it with a human. This is achieved by asking directly a human about it.

The labeling *behavior* consists in a dialogue in which:

- The robot shows the picture on its tablet, and asks whether the picture is interesting. If the human refutes this, the point of interest is forgotten, and the *behavior* stops there.
- Otherwise, the robot asks what it is, and captures the response, *e.g.* “a bottle”.
- Then, the robot asks for where it was seen, and captures the response, *e.g.* “on the table”.

As a result, natural language labels have been captured: one is to associate with the point of interest, the other is to associate with the location it was seen from. These labels may be reused to look up back the picture and the transform of the point of interest and of the location.

This *behavior* is suggested autonomously when a human comes around and there is at least one unlabeled point of interest.

4.1.3 Point at a Point of Interest

This *behavior* makes the robot point at the desired point of interest.

The procedure is as follows:

- Look at the remembered location of the point of interest. The robot may rotate to achieve this.
- Raise the arm in the same direction as the robot looks.
- Open the hand.

This *behavior* requires the point of interest to have been labeled previously with Pepper Explore. It should be triggered when the user requests it, as a direct instruction, or as part of a *behavior* composition.

4.1.4 Locate a Point of Interest

This *behavior* makes the robot confirm and show the location of a point of interest. It consists in going back to where the point of interest was discovered, and look again at the point of interest. If the robot recognizes the point of interest, it says that it is here. Otherwise, it plays a “sad” animation, hence expressing the failure for potential human observers.

This *behavior* requires the point of interest to have been labeled previously with Pepper Explore. It should be triggered only when the user requests it.

4.1.5 Visit a Location

This *behavior* makes the go back to the specified location. The location is resolved and directly used with a *Qi SDK GoTo* action³. This requires the location to have been labeled previously with *Pepper Explore*. It should be triggered when the user requests it.

In this section we reviewed the *behaviors* we designed to enrich the *behavior* of the robots in *Pepper@Home*. These *behaviors* can produce knowledge on points of interest and locations surrounding the robot. Through dialogue with human users, it can learn their labels. This potentially allows users to order the robot to point at or locate points of interest, or to visit locations, hence creating a teaching and exploration loop that resembles [Nicolescu & Mataric, 2003], and creating a fertile ground for further learning. To allow the system to execute these *behaviors* on direct orders and in new *behaviors*, *Pepper Explore* must be put in relation with our system.

In next section (4.2) we describe how this interoperability with our system is achieved. The adaptation required to accept point of interest and locations as parameters of *behaviors* is discussed in section 4.3. Finally, the conditions for triggering these new *behaviors* may lead to new conflicts that the task selection must cope with. This is discussed in section 4.4.

4.2 Proposed Ontology for Interoperability

As shown earlier in figures 3.1 and 3.5, the *behaviors* are implemented as independent components of the cognitive system. They are interfaced with the system through:

- The *API* of *NAOqi*, the underlying robotic middleware, to sense or prepare commands.
- A knowledge base, acting like a shared database. When the state of the world changes, it produces events. Other components can subscribe to these changes, and publish back processed information.
- Task suggestions: when a *behavior* implementation believes some action must be taken, it suggests a task, that may be selected and performed.

See subsection 3.4.2 for more details on our cognitive system. The knowledge base is symbolic, and is meant to represent the world semantically. For the knowledge to be actually interoperable between *behavior* implementations, we must first agree on an *ontology*⁴.

³https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/movement/reference/goto.html

⁴A set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and rela-

In this section, we summarize this ontology. The full ontology is published online⁵. Each category is represented with an [Internationalized Resource Identifier \(IRI\)](#), under the common root `qiknowledge://`.

Most categories are borrowed from the DOLCE ontology [Gangemi et al., 2002]. The aim of the DOLCE ontology is to be able to categorize everything in the world, including from a robot’s perspective [Borgo et al., 2016]. An effort was made to keep this ontology smaller and simpler than the current IEEE standards (SUMO or CORA). It is currently being standardized by the IEEE.

Figure 4.1 summarizes the place of our categories in the DOLCE ontology. Then in the rest of the section, we describe how we share tasks that could be reused by other [behaviors](#), such as the teaching of [behaviors](#), or the obedience of the robot.

4.2.1 Social Agents

We define the category “Social Agent”, represented by the [Internationalized Resource Identifier \(IRI\)](#) `qiknowledge://agent/social_agent`. This category is borrowed from the DOLCE ontology. It is an agentive social object – it is a social object that has an intentionality. For example, a society, a community, a cartoon character or an author. Humans and fictional characters are social agents. Here we consider that since social robots impersonate characters, we also consider them, in our interpretation of this ontology, as social agents. The robot is represented here as the social agent “self”, or more precisely the [IRI](#) `qiknowledge://agent/social_agent#self`. It is currently the only social agent known to the system.

This could be extended to represent the people the robot meets. When the robot meets someone, he or she is represented programmatically by a “Human” object⁶. This object corresponds to an agentive physical object, which can be located in space, whereas his or her identity would be a social agent.

4.2.2 Physical Objects

We borrow the category of “Physical Object” from the DOLCE ontology [Gangemi et al., 2002]. It corresponds to physical embodiments: a bottle, a house, a human in the flesh. It opposes to social objects: a law, a character, a company; and to abstract entities: a district, an historical fact. The [IRI](#) of this category is `qiknowledge://object/object`.

The exploration [behavior](#) presented in subsection 4.1.1 produces physical object entities when it encounters points of interest in the world. It assumes that points of interest are always carried by physical objects. When a point of interest is given a name, expressed by a phrase, we associate it to the physical object entity with

tionships (or relations among class members).

— [Gruber, 2009]

⁵<https://gitlab.com/victor.paleologue/teaching-robots-behaviors-ontology>.

⁶https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/perception/reference/human.html

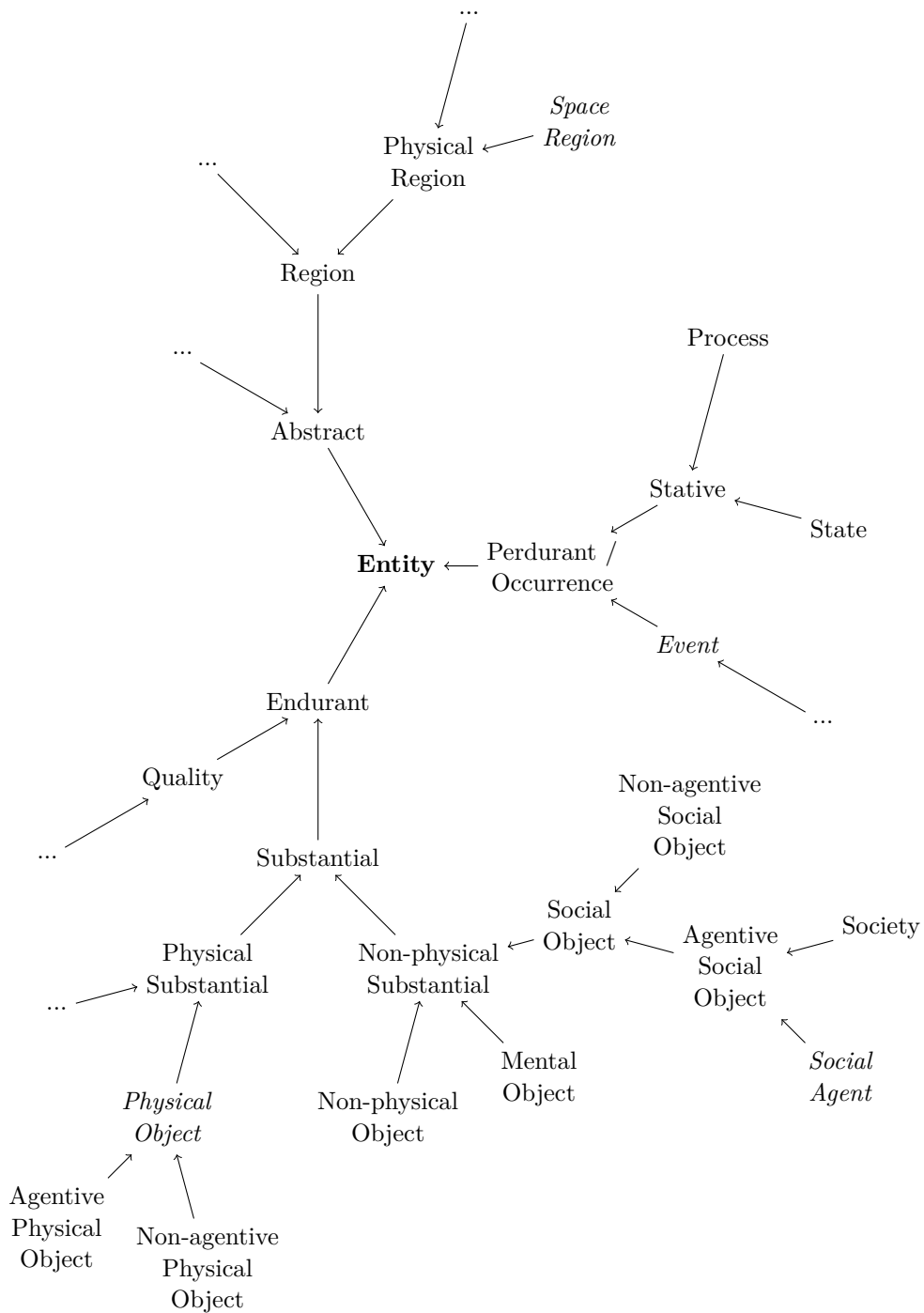


Figure 4.1: Extract of the taxonomy of the DOLCE basic categories presented in [Gangemi et al., 2002]. **Entity** is the root of the taxonomy. We borrowed the categories: *Social Agent*, *Physical Object*, *Space Region* and *Event*.

the predicate “label” of the RDF Schema, from the W3C standard⁷. In practice the label is a localized string, and is meant to be human-readable. Entities of any category can be associated to any number of such labels.

The exploration *behavior* also associates these physical objects with locations. We discuss locations in next subsection.

4.2.3 Locations, Areas and Maps

The exploration *behavior* presented in subsection 4.1.1 computes the locations of points of interest. This location is computed relative to the origin of the map in which the robot localizes itself. It consists in a transform, specifying the coordinates of a point in the map. This is represented using the category “Transform in Map”, with the IRI `qiknowledge://location/transform_in_map`.

The exploration *behavior* also remembers the position of the robot when it came closer to the point of interest. This is also a location expressed as a transform in the map. The *behavior* for labeling the location ask a name for this location. However, this label is not as precise as a transform relative to a map: It is rather referring to some arbitrary area, corresponding to DOLCE’s “Space Region” category. We call this an “Area”, and represent it with the IRI `qiknowledge://location/area`. Areas can be labeled using the “label” predicate of the RDF Schema, like for physical objects.

DOLCE’s specification suggests that the location can be a property of an object. But this is not sufficient to reason on objects that may move in time. For such cases, it is important to distinguish the location of an object from the location in which it has been seen in at a given time. The former is seen from an “endurantist” point of view: the location is intrinsic of the object. The latter is seen from a “perdurantist” point of view: the location is attributed to the event of seeing the object. The next subsection describe the ontology to describe events.

4.2.4 Events

The “Event” category represents occurrences of a situation. They are represented with the IRI `qiknowledge://event/event`. Events may be associated to a time.

When the exploration *behavior* sees an object, it produces two events. One event associates the object with a picture and the location of the robot at the time of occurrence. The other event associates the object with a location. When the user tells the robot the name of the location of the object, it is associated to the location of the second event⁸. Figure 4.2 sums up the knowledge produced by the exploration *behavior*.

Events are also relevant to express utterances occurring during dialogues. We detail these specific events in next subsection.

⁷<https://www.w3.org/TR/rdf-schema/>

⁸It cannot be associated to the location from which the picture was taken, since there is certainty that that location is in the same area as the object.

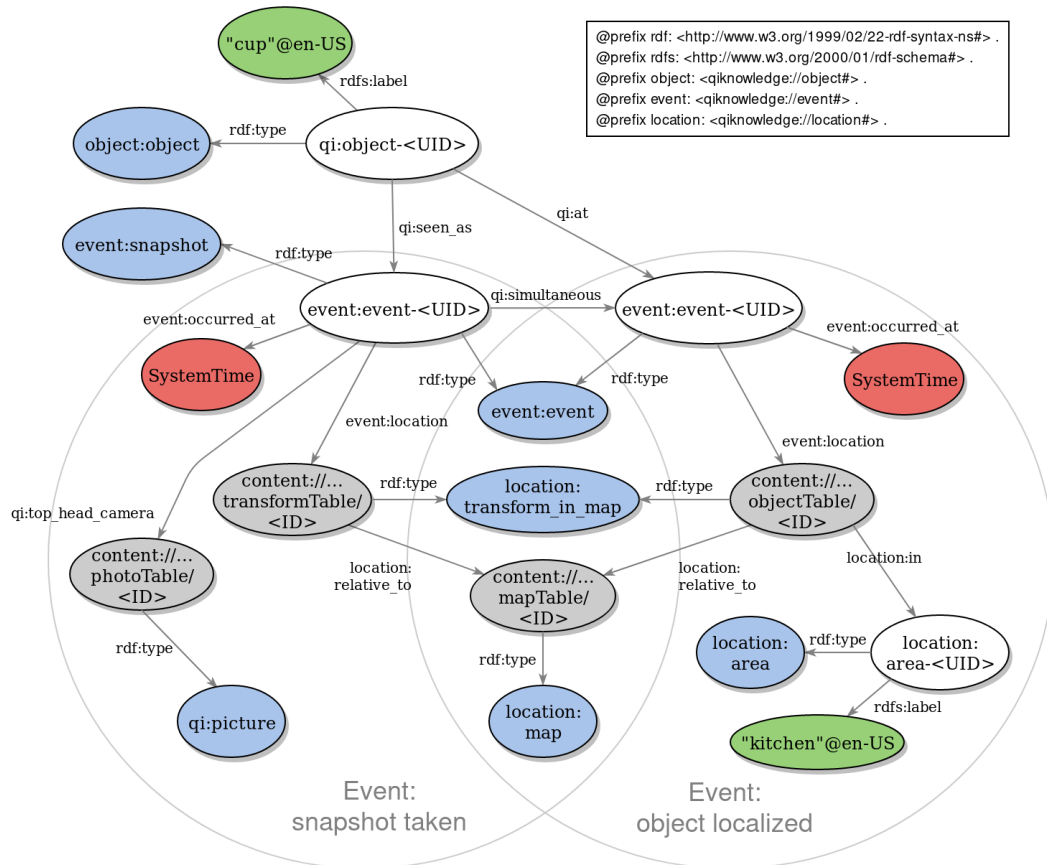


Figure 4.2: Template of knowledge graph produced by the exploration behavior when discovering an object. White nodes are resources, green nodes are localized strings, red nodes are datetime data, grey nodes are resources private to that application. Figure by Shin Watanabe, employee of SBRE.

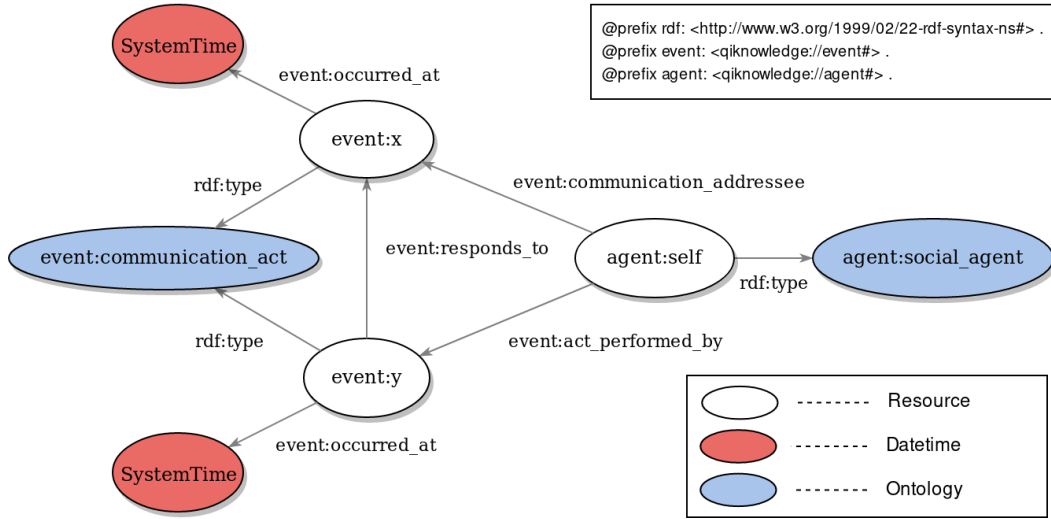


Figure 4.3: Template of knowledge graph produced by dialogues. Communication act **event:x** was addressed to the robot (**agent:self**). The robot performs the communication act **event:y**, as a response to **event:x**.

4.2.5 Communicative Acts

When a user says something to the robot, he or she performs an action directed to the robot. **Speech acts** exhibit this ambivalence between speech and action, but accounts only to one side of it: uttering something can be a form of action. The other side of this ambivalence is that any form of action may communicate information like an utterance does. Sign language is a trivial example. It is also common to consider that most of the communication is non-verbal.

Therefore we adopt the notion of “Communicative Acts”: a specific form of event that represents an attempt at communicating an information, whichever the form it takes. It is represented with the IRI `qiknowledge://event/communication_act`. This notion allows us to consider events like finger-pointing or expressing excitement explicitly within the same framework as dialogues. It complements the efforts we made in previous chapter to avoid a distinction between dialogues and other behaviors.

A communicative act is performed by a social agent and is addressed to another social agent. It can be responded to with any form of act, but for now we consider that all responses are communicative acts. These relations are represented respectively with the following predicates: `qiknowledge://event/act_performed_by`, `qiknowledge://event/communication_addressee`, `qiknowledge://event/responds_to`.

4.2.6 Actions

“Actions” are software objects provided from the **Qi SDK**, and are detailed more in annex 5.5.6. They represent potential **actions** the robot can perform: they encaps-

sulate procedural knowledge. The action is performed by the robot when its “run” method is called. Actions may also accept parameters in the form of properties, that can be set before the action is run. Action objects can be carried around over the network, so they can be shared even between Pepper Explore and our cognitive system. When a **behavior** implementation suggests a task, it shares an action object with our system, which may select it and run it.

For the “obey order” and “compose behavior” **behaviors** to be able to reuse these **actions**, they must always be shared with the system, and be associated with natural language. To do so we introduce a new object type called “Action Factory”. An action factory can create a new action on demand, through its “makeAction” method, so that it can be parametrized for a specific purpose, or for a specific context. It also provides the association with natural language, in the form of “semantic templates”, presented in next subsection.

4.2.7 Semantic Templates

Inspired by the model of **semantic frames**, a “semantic template” associates a situation, or an action, to natural language. It has therefore the same purpose as the task labels used in subsection 3.3.3. It is also based on a human-readable localized string, but adds placeholders for the complements it defines. Placeholders carry coded information, associating a slot name with a code identifying the role of the slot in the sentence. For instance, the semantic template “to visit <object:target>” associates the verb “to visit” with a location complement. The codes are provided by the semantic analysis library presented in subsection 3.2, and is an internal tool at **SBRE**. It includes the following roles:

Object The target of the verb, which is often the object complement.

Receiver The indirect target of the verb, usually an indirect object complement.

Location A location complement.

Time A time complement, locating the statement in time.

Duration A time complement, describing the statement duration.

Cause The cause of the statement.

Manner A manner complement.

The semantic analysis can extract semantic structures from semantic templates. For statements, it consists in an expression centered on the root verb, with child expressions. Each child expression has a distinct role in the statement. The template presented earlier, “to visit <object:target>”, corresponds to the following semantic

structure:

$$\begin{aligned}
 \text{"visit"} < \text{object} : \text{target} > \text{"} \xrightarrow{\text{semantic}} (me, target, visit), \\
 & \quad Agent(me, self) \wedge \\
 & \quad Slot(\text{"target"}, target) \wedge \\
 & \quad Statement(visit) \wedge Word(\text{"visit"}^{VB}, visit) \wedge \\
 & \quad Time(present, visit) \wedge Request(action, visit) \wedge \\
 & \quad Subject(me, visit) \wedge Object(target, visit) \wedge
 \end{aligned}$$

This semantic structure can then be used to compare the template with other phrases. We use these comparison capabilities in subsection 4.3.1, to parametrize behaviors.

4.2.8 Conclusion on Sharing Knowledge and Behaviors

In this section we specified an ontology to allow the sharing of knowledge across the system, and an interface to parametrize actions. Given the behaviors proposed in section 4.1, it is theoretically possible for the robot to perform actions on unpredicted entities, and support an open scenario.

Moreover, since symbolic knowledge systems are domain-independent, it is possible to extend our ontology to new domains in the future. Finally, most of the notions of in our taxonomy are independent from interaction modalities. It could in theory support multimodal interaction.

In next section, we study how to adapt the teaching of behaviors to leverage the parametrization of tasks, and the entity knowledge, to produce parametrized behaviors.

4.3 Teaching Parametrized Behaviors

In this section, we describe how we extend the behavior-teaching interaction presented in 3.3 to support parameters. We first focus on how we leverage the semantic analysis to understand parameters. Then we describe how we adapted our behavior model to support parameters. Finally we review what the teaching interaction looks like in these conditions, to explain how conflicts emerge.

4.3.1 Extracting Parametrized Task Teachings

Semantic templates seem to provide enough information to identify task declarations (see subsection 3.3.1) and their parameters. In this subsection we describe how to achieve this.

Given the example semantic template shown in subsection 4.2.7: a semantic template “visit <object:target>”. It is possible to compare it with the input phrase “visit the kitchen”, and identify that “the kitchen” corresponds to the slot named

“target”. This input phrase corresponds to the following semantic structure:

$$\begin{aligned}
 \text{"visit the kitchen"} &\xrightarrow{\text{semantic}} (me, kitchen, visit), \\
 &Agent(me, self) \wedge \\
 &Word(\text{"kitchen"}^{NN}, kitchen) \wedge \\
 &Reference(DEFINITE, kitchen) \wedge \\
 &Statement(visit) \wedge Word(\text{"visit"}^{VB}, visit) \wedge \\
 &Time(present, visit) \wedge Request(action, visit) \wedge \\
 &Subject(me, visit) \wedge Object(kitchen, visit) \wedge
 \end{aligned}$$

To compare it with the template, we first check that both are statements, and that their root verb has the same lemma, “visit”. Then, we check every child: the input phrase matches the template if it provides all the roles declared in the template. In our example (“visit <object:target>”), the template tells us that the object of the statement should fill in the slot *target*. In the input phrase, the object is *kitchen*, defined by $Word(\text{"kitchen"}^{NN}, kitchen) \wedge Reference(DEFINITE, kitchen)$. Therefore, we have $target = kitchen$.

This semantic representation should correspond to an entity known by the robot. In subsections 4.2.3 and 4.2.3, we specify how object and spatial entities are represented in the knowledge. They may be associated with a natural language label. We continuously gather all the labels of the knowledge perform our NLU on them, to produce a semantic expression. Therefore, input slot values can be compared with existing entities in the knowledge. For instance, we may find a semantic expression equivalent to *kitchen*, and retrieve its corresponding resource node.

At this stage, we can tell which action corresponds to an instruction, and to which parameter it should be applied. The mechanism we use is equivalent to the one presented in [Perera et al., 2015]: they propose frames with slots, and can identify frames and slot values from some natural language instruction. Their frames are analogous to our combination of our semantic templates. And they also use a knowledge base to look up entities labeled with natural language.

To execute the identified action, first we create a new action object from its registered factory (see subsection 4.2.6). Then we set the corresponding properties of the new action with the parameters we identified, in the form of knowledge resource nodes. The name of the slots must match the name of the corresponding property for us to be able to transmit the parameters.

This parametrization mechanism does not compromise the extraction of task teachings we present in previous chapter, subsection 3.3.1. We still define a task teaching as the association of a task label with a task description. A task label consists in an infinitive statement, and the task description consists in an enumeration of infinitive statements. The learning of behaviors is still a zero-shot learning process, consisting in remembering this association of statements. It should be agnostic of the content of the statements. Therefore if a task description now includes statements corresponding to parametrized actions, the learning of behaviors accepts

it, with no change in its implementation.

In opposition, teaching a parametric task requires a change in the learning of **behaviors**: users have to express the existence of a slot in the task label, and how it relates to the tasks of the description. We expect the following example to be usable: “to greet someone is to look at him and to say hello”. In this teaching, the task label “to greet someone” includes the formulation “someone”, that expresses the indetermination of the social agent. Then, the word “him” in the task declaration “to look at him” is a coreference to that “someone”. Given there is a semantic template “to look at <looked:object>”, we can build the template “to greet <someone:object>”, and bind the slot *someone* to the slot *looked* of the sub-task.

Our NLU tool recognizes indeterminate clauses and coreferences, so it allows us to achieve this extraction. We update our task teaching extraction to perform this extraction, and therefore support the teaching of parametric **behaviors**. As a consequence, we must update our formal model. This is the purpose of the next subsection.

4.3.2 Formal Model for Parametric Behaviors

In this subsection, we extend the formal model presented in 3.3.2 to support parametric tasks. The **goal** of this formalization is to clarify our model of **behaviors**, and lay it down in terms that ease its comparison with other models. But we do not take advantage of this formalization in this thesis.

A parametrized task is just a task $t \in T$. A parametric task is a task $t_p \in T_p, T_p \subset T$. It is not fully defined, and therefore cannot be performed without earlier parametrization. Our model in terms that can be compared with existing works in the future

Given the set of all possible entities in the world O . Parametrization is the combination of a task t_p with a tuple of $e \in E$, denoted $(e_0, \dots, e_k), k \in \mathbb{N}$, to produce a new task t . In other words, parametrization is the application defined as $f_p : t_p, (e_0, \dots, e_k) \rightarrow t \in T$. For instance, given $kitchen \in E, visit \in T_p, \exists visit_kitchen \in T, visit_kitchen \notin T_p, f_p(visit, kitchen) = visit_kitchen$. *visit_kitchen* is a distinct non-parametric task, produced by the parametrization of the task *visit* with the entity *kitchen*.

However, given a t_p , f_p is not defined for every tuple (e_0, \dots, e_k) . The set of all possible tuples is denoted \mathcal{E} . Each parametric task has a fixed number of parameters, and for certain parameters, only accept a subset of E . The set of acceptable tuples to fully define a parametric task t is denoted \mathcal{E}_t . The set of all acceptable tuples for all parametric task is denoted \mathcal{E}_{T_p} .

We define the arrival space of the function p as the set of parametrized tasks $T_{parametrized}$ such as: $\forall x \in \mathcal{E}_t, \forall t_p \in T_p, f_p(t_p, x) = t, t \in T_{f_p}$. The number of possible fully parametrized tasks is computed with: $|T_{parametrized}| = |T_p| \times |\mathcal{E}_{T_p}|$. Whenever a parametric task or a new entity is learned, it produces multiple possible parametrized tasks. The number of new tasks grows learning after learning, at the condition that new entities $e_{learned}$ are compatible with existing parametric tasks

t_p , or that new parametric tasks $t_{p,learned}$ are compatible with existing entities e .

Learning an entity consists in defining a new, stand-alone $e_{learned}$, $E^{i+1} = E^i + e_{learned}$. Similarly, learning a task was described in subsection 3.3.2 as updating $T_{known}^{i+1} = T_{known}^i + t_{learned}$, and defining $t_{learned}$ as a sequence of known tasks: $\forall k \in \mathbb{N}, \exists t_0, \dots, t_k \in T_{known}, t_{learned} = (t_0, \dots, t_k)$. This is not sufficient anymore, because it is not applicable to parametric tasks: there is no expression of the required parameters, nor of the parametrization of sub-tasks. like previously, is not sufficient anymore. Formally, it is not applicable if $t_{learned} \in T_p$, or if $\{t_0, \dots, t_k\} \cap T_p \neq \emptyset$.

Instead, the learned task must be potentially parametric, and its parameters must bind all sub-tasks' parameters. This binding can be formalized as a function dispatching the parameters to the sub-tasks: $k \in \mathbb{N}, t \in (t_0, \dots, t_k), x \in \mathcal{E}_{t_{learned}}, d : t, x \longrightarrow y, y \in \mathcal{E}_t$. Therefore we define learned tasks as $t_{learned} = (t_0, \dots, t_k, d)$. They mechanically inherit $\mathcal{E}_{t_{learned}}$ from the sub-tasks. Subsection 4.3.1 explains how we deduce this dispatching function from the natural language.

In subsection 3.3.2 we also defined the association of a task with a **semantic frame** as $s \in S, Express(t_{learned}, s)$, where S is the set of all possible semantic frames. This association is still valid, but we consider it not relevant enough, since tasks are now associated with semantic templates, instead of semantic frames. While each semantic template corresponds to a **semantic frame**, a **semantic frame** may correspond to several semantic templates: “to go to <someplace:location>” and “to walk to <someone:object>” are two distinct semantic templates corresponding to the same semantic frame about moving from one place to another⁹.

Therefore we cannot assume anymore that a **semantic frame** corresponds to a single task. Depending on the complements used, a natural language phrase may match different semantic templates, and therefore different tasks, for the same **semantic frame**.

Note that two semantic templates with the same lemma for their root verb may correspond to different semantic frames. For instance “to go to the Eiffel Tower” vs. “to go mad”. The role of the complements may dramatically alter the meaning of the statement. Similarly, the categories of the entities expected as parameters are significant to identify a **semantic frame**. For instance, to visit a place is not to be interpreted exactly equivalent to visiting someone. The latter is ambivalent, and also corresponds to meeting someone. Therefore the association with a **semantic frame** is really specific to the considered semantic template, that should be as detailed as necessary.

4.3.3 Interaction Patterns for Parametric Behaviors

A priori we consider that the teaching interaction patterns remain similar as in subsection 3.3.3. Due to the new **behaviors** introduced to support parametric **behaviors**, we predict that conflicts may emerge.

⁹Berkeley's FrameNet calls this frame “motion”, see <https://framenet2.icsi.berkeley.edu/fnReports/data/frameIndex.xml?frame=Motion>.

User	Robot
	1. Ask
2. Answer	3. Confirm

Table 4.1: Usual [pragmatic frame](#) involved when asking if an object is interesting, its label, or the label of a location.

First, the teaching instructions require a higher cognitive load for the users. We describe them in subsection 4.3.1. Not only do they have to think about a verb, but also they have to articulate the parameters properly. It may lead to users making pauses during the speech: [Jou & Harris, 1992] show that the pauses in speech production are increased when the speaker’s attention is divided. Pauses in speech are also used to measure the cognitive load [Gorovoy et al., 2010, Khawaja, 2010]. For example, they might cut their sentences between the verb and the complement. They may also do these pauses as an interactive way to make sure the robot understands the teaching. These considerations about pausing are specific to spoken interaction, and do not appear when working solely on transcriptions.

Then, with the [behaviors](#) presented in section 4.1, the forms of interaction should be richer. It is visible at the level of the pragmatic frames. In addition to the frames found in tables 3.1 and 3.2 in previous chapter, the robot should exhibit the frame shown in table 4.1.

Finally, and regardless of the pragmatic frames, the [behaviors](#) for asking about points of interest or locations expect replies like “yes”, “the bottle of oil” or “in the kitchen”. These utterances are hardly understood outside of their context. It is possible that the fallback [behavior](#) expressing the misunderstanding of the robot gets triggered, or the baseline dialogue [behavior](#), instead of the asking [behavior](#). In previous experiment, we encountered this kind of conflict, see subsection 3.5.4. This kind of conflict could be solved by taking advantage of this notion of context, and let the [behavior](#) implementations share their intentions or [goals](#), so that the task selection can solve such conflicts.

In the next section we explain how conflicts are solved by the task selection, using rules that can apply to the [behaviors’ goals](#). These rules are then tested in Pepper robots at home, in section 4.5.

4.4 Rule-Based Task Selection

In section 3.4 we introduced an example interaction rule that could serve action selection in a system where different [behaviors](#) are designed independently. In this section, we explain how we scaled up the task selection: we annotate tasks with [goals](#), and then use a rule-based system to select them: a planner.

Rule-based decision-making is a typical problem of [AI](#). It produces explainable results, and therefore supports meta-cognition: the ability to reason about the

interaction itself. According to [Chernova & Thomaz, 2014, p.11], the best learners are capable of using this form of meta-cognition at their advantage. See our state of the art, in subsection 2.1.1, for more details. This section then describes the rules we developed for our research.

In this section we support the benefits of a goal-oriented approach, and develop how it leads to solving planning problems.

Then this section describes our improved cognitive system integrating the planning system. We detail the changes in how **actions** are suggested, but also how action factories are shared with the system. We face and solve the challenge of putting together task suggestions that have competing **goals**, thanks to a rule system.

Finally, we focus on the rules we defined, and determine what information **behaviors** must share for the planning system to apply the rules.

4.4.1 Goal-Oriented Approach and Planning

The task selection component (see figure 3.5) should select which task is suitable for the current situation. It is allowed to know the state of the world, through the shared knowledge base. It may know of the entities defined in section 4.2, but also of all possible **actions**, and of the currently suggested tasks.

As explained in section 3.4, **behaviors** are the components responsible for knowing whether the conditions demand performing a task. They suggest tasks only when it is relevant for them¹⁰, with no knowledge of each other: they are designed independently.

Conflict arises when several **behaviors** suggest tasks at the same time¹¹. The task selector must decide which one best suits the situation, or in which order to perform them. This cannot happen without extra information about the suggested tasks: why they should run, or what they try to achieve.

In **AI**, these two pieces of information can be formalized under the same notion of **goals**: using logic, the reason why something should be done can always be expressed as a **goal** state of the world. The state of the world can then be changed by performing an action, which has an effect on this state. The goal-oriented approach implies to select the right action to perform given a current state, a **goal** state, and the effects of the **actions**. This kind of problem is well researched in the field of **AI**, and produces explainable solutions. Note however that the explanation is limited, if the reason why **goals** were set cannot be explained.

To apply this in our system, **behavior** implementations should mention the **goals** of the tasks they suggest, and all their effects. This can be achieved for any task suggestion, including hard-coded ones. By default, if a task suggestion does not

¹⁰We view **behaviors** as independent agents that take decisions for themselves. Nonetheless, they are arbitrary programs, that do not require to be “intelligent”.

¹¹Exact simultaneity never really happens. There are always some race conditions that may bring one suggestion before the other. But this state is ephemeral and does not allow anything to visibly happen on the robot.

specify its action's **goals** and effects, a self-fulfilling goal / action pair is generated: the action's effect is to acknowledge the performing of the action, and the **goal** is to acknowledge the performing of the action. Task suggestions have therefore a dual role: we distinguish the **goal** suggestion, from the announcement of a potential action, similar to an action factory. Then, when a **goal** can be fulfilled by an action, this action is selected and scheduled for execution: it becomes a task, that has been accepted by the system.

The problem of finding the right action(s) to fulfill a goal can be solved by symbolic planners. Symbolic planners are sometimes found at the core of some cognitive systems used for teaching **behaviors** (see sections 2.4 and 2.4). They are also found for scheduling HRI tasks, like in [Petrick & Foster, 2016, Sonenberg et al., 2016]. They seem adequate to implement a goal-oriented and rule-based system for task selection.

In subsection 2.4.2, we review some planning solutions, and their corresponding behavior models: Hierarchical Task Network (HTN), Planning Domain Description Language (PDDL), or direct applications of Markov Decision Process (MDP). We choose to work with the Planning Domain Description Language (PDDL) because of its potential support of durative actions [Fox & Long, 2003]¹², preferences and metrics [Gerevini & Long, 2005] so that to weigh in priorities. We choose the planner called fast-downward¹³ because it is actively maintained and has been for several years. However we consider there are potentially better-suited planners presented in [Baier & McIlraith, 2008], a study on planners supporting preferences.

Using a planner may also facilitate the support for goal-oriented task teachings, that we observed in the experiment described in subsection 3.5.2. It means that the user may implicitly describe a complex sequence of **actions**, by describing a **goal** state. Moreover, the sequence of **actions** may be adapted depending on the current state of the world, skipping unnecessary tasks in a plan.

To produce plans, a planner needs the description of all the possible **actions** and of their parameters, the knowledge of an initial situation, and a target situation, the **goals**. We defined in section 4.2 a specification to share parametric **actions**, and ways to share some knowledge on the current state of the world.

In next subsection, we describe how we collected tasks, knowledge and action factories for the planner-based task selector.

4.4.2 Building Problems from Task Suggestions

The notion of **actions** found on Pepper with the Qi SDK integrate an action-centered paradigm¹⁴ that we try to respect as much as possible in our research. Indeed, we try to apply our research in a way that is intrinsically compatible with Pepper. On the other hand, the Qi SDK helps us to share **actions** across components of the

¹²A durative action not only produce effects as a result of completing the **actions**, but also while the action is running.

¹³<http://www.fast-downward.org/>

¹⁴This action-centered paradigm is shared with ROS.


```

(define (domain exploration)
  (:requirements :strips :typing)
  (:types qiknowledge_location_area)
  (:constants visit_event - event)
  (:predicates (occurred ?e - event))
  (:action visit
    :parameters (?area_to_visit - qiknowledge_location_area)
    :precondition ()
    :effect (occurred visit_event)))

```

Figure 4.4: PDDL description of the “visit” action.

system, thanks to a mechanism of “remote objects”, explained in annex 5.5.6.

We propose a component named “Action Pool” to gather **actions** shared by **behaviors**. When **behaviors** register themselves in the system, their action factories and their task suggestions are gathered together in the Action Pool.

Action factories provide some semantic templates (see subsection 4.2.7) and a PDDL description of the action’s preconditions, parameters and effects. To be valid, this PDDL content must describe a full domain, declaring types, predicates, and possibly constants. Figure 4.4 shows an example PDDL description of the action factory for the action “to visit”. Note that it has a self-fulfilling effect, and that it accepts a parameter. The name of the parameter corresponds to a property exposed by the action “to visit”. The type of the parameter is specified as `qiknowledge_location_area`. It is a direct translation of the corresponding IRI of the category of the accepted entity: an area¹⁵ (`qiknowledge://location/area`). The Action Pool associates the action factory with the PDDL action, and then merges the PDDL description with all the other PDDL descriptions already gathered.

Task suggestions however, do not provide semantic templates, and provide a richer PDDL description. Not only they describe an action within its domain, but they must also explicit their goal, and its accompanying PDDL problem. The goal description includes a partial view of the world (the known entities and their current state), and declares what state should be reached. The Action Pool associates each task suggestion with their PDDL action, and merges all the PDDL descriptions together (including the action factory’s).

Our problem of supporting competing, if not conflicting, **behaviors** finally manifests in the technical challenge of merging the PDDL goals: if goals conflict, planning is impossible. This is called an oversubscription problem, and is usually solved using partial satisfaction planning. We propose to use rules to select the most important goals to satisfy. This approach is inspired by the work of [Wilensky, 1981] on meta-planning.

A priori all goals are relevant, and there should be no direct consideration of which behavior suggested it. The system should try to satisfy as many goals as

¹⁵See subsection 4.2.3.

```

template <typename T>
struct SetPriorityComparer
{
    std::vector<T> priorityList;
    bool operator()(const std::set<T>& lhs, const std::set<T>& rhs)
    {
        for (const auto& priority: priorityList)
        {
            if (lhs.count(priority) > rhs.count(priority)) return true;
            if (lhs.count(priority) < rhs.count(priority)) return false;
        }
        if (lhs.size() > rhs.size()) return true;
        if (lhs.size() < rhs.size()) return false;
        return lhs < rhs;
    }
};

```

Figure 4.5: C++ code for sorting all possible combinations of suggested [goals](#), with respect to an ordered list of priorities.

possible, and at best, satisfy all of them. Therefore the first [goal](#) the system should try to achieve is the and-combination of all the [goals](#). More formally, given the set of [goals](#) $G = \{g_0, g_1, \dots, g_k\}$, $k \in \mathbb{N}$, we try to satisfy the goal $\bigwedge G = g_0 \wedge g_1 \wedge \dots \wedge g_k$. Then, and-combinations of all subsets of the [goals](#) can be tried, from the largest to the smallest. The set of all possible [goal](#) combinations is:

$$\mathcal{G} = \bigwedge_{G_{sub} \in \mathcal{P}(G)} G_{sub}$$

Then, we order \mathcal{G} by descending number of involved [goals](#), and arbitrarily if equal. For example, given 3 [goals](#) a , b and c , we would try to plan for the following sequence of [goal](#) combinations: $(a \wedge b \wedge c, a \wedge b, a \wedge c, b \wedge c, a, b, c)$. There are more efficient ways to compute partial satisfaction solutions, but it appeared that this solution was sufficient in our case.

A [goal](#) can be prioritized by starting the planning attempts with the subsets containing that goal. A list of [goals](#) of decreasing priority can be define with a similar algorithm. We compute \mathcal{G} and then order it according to a priority list, computed by the application of rules on the current set of [goals](#) G . It consists in a k-arrangement of G , ordered by decreasing priority. Figure 4.5 details the sorting algorithm we wrote. This ordering maximizes the chances to find a plan respecting the [goal](#) prioritization rules. Nonetheless, the system remains domain-independent: it is agnostic to the details of the rules.

Finally, it appears that rules are defined prior to the [goals](#), and that the [goal](#) selection can only be useful if the [goals](#) are described within a specified ontology. In section 4.2, we specified one to support the contents of our experiment. Action factories must respect this ontology if they want to be used by the system. Tasks

suggestions, and their *goals*, must respect it too to be selected by the rules, but also to benefit from potential *goal* satisfaction maximization. This consists of an additional incentive to respect the interoperability of the system¹⁶.

In next section we describe the various rules we designed for our experiment.

4.4.3 Rules for HRI Tasks

By putting our system together with the *behaviors* enumerated in section 4.1, we faced the challenge of defining the rules properly, and the *goals* accordingly. Together, rules and *goals* strongly impact the *HRI*. The rules we propose here were designed with as least *a priori* as possible on the content of the *behaviors*. As a consequence they focus mostly on considerations of *pragmatics*, defining in more general terms how the interaction should unroll.

4.4.3.1 Responding to Communicative Acts

Our first rule is designed to implement the respect of the *speaking floor*. In the previous chapter – in subsection 3.4.2 – we explained how we avoided having the dialogue management dictating the rhythm of the interaction, and reconstructed the *speaking floor* by selecting only the first response received. In this chapter, we express this rule explicitly, so that it integrates into our system, which is domain-independent.

The previous implementation of this rule relied on the knowledge of task suggestions, which is not accessible anymore from *PDDL* domains. Instead, we make explicit the input communicative acts (see 4.2.5). They appear in the *PDDL* domain as a constant, for instance `input - qiknowledge_event_communication_act`¹⁷. The *actions* producing a response to them make explicit that intent, through their effect. For instance, when action plans to produce a response declared as `response - qiknowledge_event_communication_act`, its effect would state: `(and (qiknowledge_event_responds_to response input)(was_responded_to input))`. Note that the predicate

```
(qiknowledge_event_responds_to
  ?input - qiknowledge_event_communication_act
  ?response - qiknowledge_event_communication_act)
```

directly corresponds to the predicates defined in our ontology.

As a precondition, dialogue response *actions* also state they are not worth being performed if the input communicative act was already responded to: `(not (was_responded_to input))`. This alone allows the planner to select one and only one task to respond to the user's input. The rule identifies these responses, so that they can benefit from prioritization.

¹⁶In software development, the price of complexity is high, and must be balanced by strong incentives, if not necessities.

¹⁷There has been a confusion in the implementation, which mentions communication acts instead of communicative acts.

4.4.3.2 Avoid Misunderstood Input Acts

In this context, replying that the robot did not understand is an application of the response rule described in 4.4.3.1. But replying only as a fallback, when no other response was found, requires an additional rule.

A lack of response means that no **behavior** is considered relevant to respond to it. Note that a communicative act may be responded to by any form of act, and does not necessarily involve a speech.

It can be assumed that a **behavior** refraining from providing a task suggestion in response did not find anything it is designed to process, and does not know what to do about it. In that case, such **behavior** would have not recognized, or in other words, not understood, the input communicative act.

Otherwise, if the **behavior** had understood the input, but did not provide any feedback, it could confuse the users: in human interaction, we expect a minimal response to express good understanding in a discussion. Such response may include staying purposefully still and silent. It is nonetheless an active task, that should be suggested, as a response to the input communicative act.

Therefore we assume it is safe to consider that if a **behavior** had something to do about an input communicative act, it would suggest a task responding to it. And therefore, it is safe to assume that if the “not understood” fallback task is performed, the communicative act has effectively misunderstood.

The predicate to express this is defined as follows: `(was_misunderstood ?c - qiknowledge_event_communication_act)`. We declare the “not understood” task has for effect to declare the input communicative act as misunderstood. We add a rule that produces an additional goal, stating that the input communicative act should not be misunderstood.

4.4.3.3 Staying in Context

The **behaviors** implementations for the basic dialogue, the teaching of **behaviors** and for the labeling of points of interest and locations, all expect to take several speech turns to achieve their purpose. It is important that when a teaching is started, the associated **behavior** keeps some priority. For instance, if a **behavior** autonomously decides to perform some task during a teaching, we would prefer that the teaching finishes before allowing these other tasks to be performed.

We introduce a notion of **behavior** context to achieve that distinction between the tasks. Behavior contexts are entities of type `context`. An action can depend on this context, or lead the context to change, by using the predicate `(in_context ?c context)`. We add a rule privileging the **actions** that do not change this context, so that the interaction tries to preserve its continuity. It is similar to a notion of topics, and to a rule to avoid responses off-topic.

In the current implementation, we automatically attribute a context to every task suggestion: their parent **behavior**. Hence the task selection favors tasks suggested by the same **behavior** that suggested the latest task successfully performed.

For instance, if a **behavior** “Greet people” suggested an action “Say hi”, we would encounter an action `action_say_hi`, with for effect (`in_context greet_people`).

4.4.4 Conclusion on Rule-Based Task Selection

The proposed rule system should support the teaching of parametric **behaviors** in a rich scenario of interaction, implemented by **behaviors** implemented independently. It uses a planner to perform a large part of its symbolic, domain-independent reasoning. The rule system allows a selection of **goals** to be satisfied in priority. That selection is dynamic, and relies solely on the state of the world, and not on prior knowledge of the **behaviors**. In the next and final section, we set up an experiment to demonstrate the system, and check whether its promises were kept.

4.5 Experiment

In this section we describe our experimental protocol, the deployment of the experiment in Pepper@Home, the data collection and our results.

In the previous experiment, in subsection 3.5.4, we intended to demonstrate that a teaching interaction in an **open scenario** was possible, but with a simplified model of **behaviors**. We deployed it in real conditions, and for the first time faced the challenge of having the teaching **behavior** compete with the pre-existing **behavior** of the robot.

In this chapter, we describe a **behavior model** that would accept parameters, and may be more suitable for real-world instructions. We shall test whether it is better accepted, and whether it remains usable despite the complications, using the same measures as in previous experiment.

To support parameters in an **open scenario**, we need to allow the teaching of entities in the world. The behaviors proposed in section 4.1 enable the open learning of objects and locations. Adding these behaviors to the system produces new conflicts that our rules should solve automatically. This experiment should reveal whether other conflicts were left unhandled.

The experiment is held with Pepper@Home users, who live with the robots, with as little intervention from the experimenter as possible. It supports French and English, so that to rule out foreign language proficiency from the source of errors.

The full details of the experiment are online¹⁸.

4.5.1 Experimental Protocol

11 participants interact with a robot according to various scenarios, each tested in a different phase:

1. The robot explores its surroundings and looks for points of interest.

¹⁸<https://gitlab.com/victor.paleologue/teaching-robots-behaviors-experiments/tree/master/5%20-%20Rich%20Interaction>.

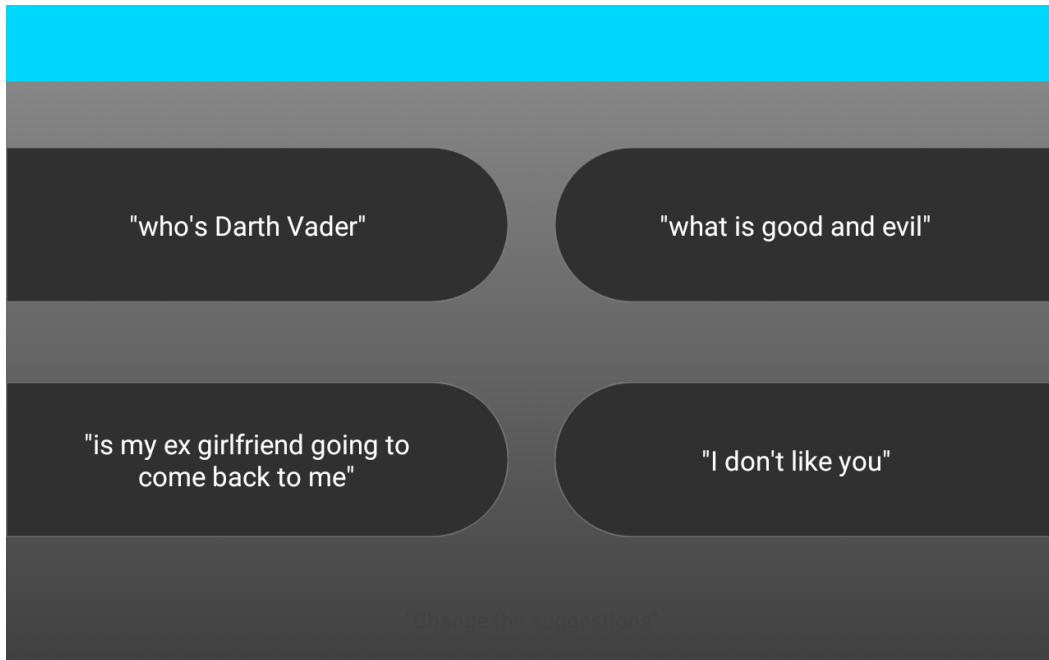


Figure 4.6: Screenshot of the phase 2.a). It runs a regular chat, and displays recommendations about what the users can say to the robot.

- 2.a) *A.k.a.* “Regular”, a regular chat interaction, with the default content that served as a baseline of previous experiment.
- 2.b) *A.k.a.* “Teaching”, the teaching interaction alone, supporting parameters.
- 3. *A.k.a.* “Mixed”, the mix of all the [behaviors](#): regular chat, [behavior](#) teaching, exploration and teaching of points of interest.

In phase 1, the participants are obliged to let the robot run the [behavior](#) for teaching points of interest. Prior to the interaction, they have been told that they would have to let the robot explore, and accept to give their names to the robot. Once the robot knows one person and one point of interest, the phase 1 can finish, and participants can jump to the rest of the experiment. They jump to either phase 2.a) or 2.b), according to a random choice, and then jump to the other phase 2.

In 2.a), participants get accustomed to the baseline chat contents. It looks like the ABC application, and with different colors. See figure 4.6. They discuss for a short period of time with the robot, and then are asked to reply to a questionnaire evaluating the usability of the chat on the [System Usability Scale \(SUS\)](#) [Brooke, 1986]. This scale is based on 10 questions, replied to on a Likert scale, from which a system usability score can be computed. This score is used to perform comparisons, and is not meaningful by itself.

In 2.b), participants learn to teach [behaviors](#) to the robot. Given written instructions on the tablet (see figure 4.7), they are told formulate [behavior](#) teachings, including composite and parametric ones. They are given 10 minutes to teach

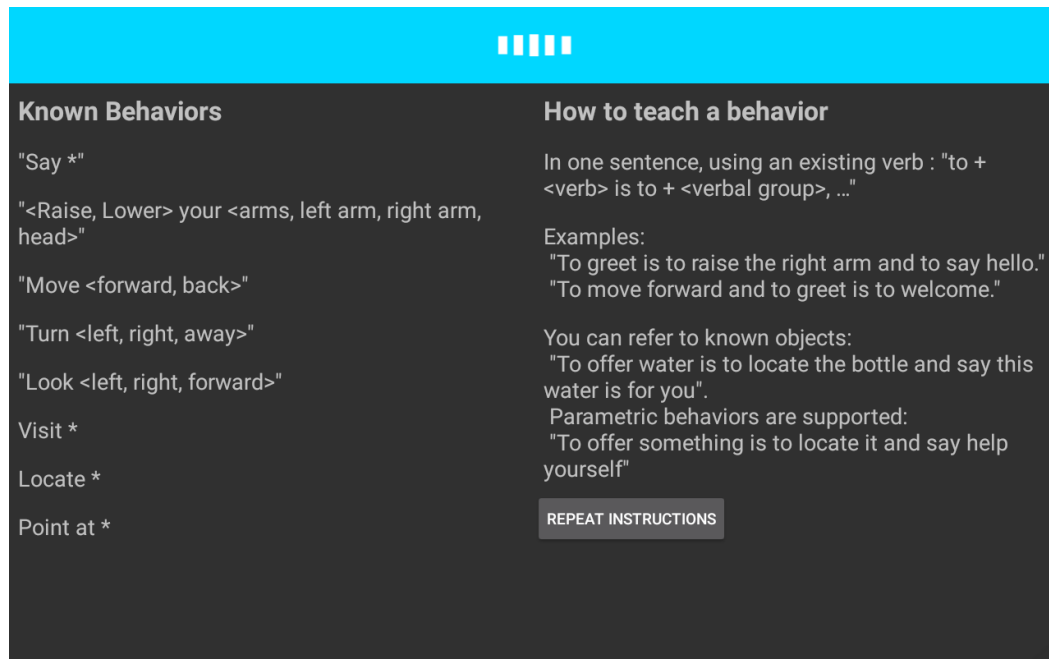


Figure 4.7: Instructions presented to participants in the phase 2.b), to explain how to teach behaviors to the robot.

as many behaviors as they can. We collect all the utterances and the effectively taught behaviors to compute the measures that were used in previous experiments (see section 3.5). We also record the audio to be able to identify speech recognition issues.

In phase 3., the participants are asked to perform some teachings again, this time on a configuration that has all the behaviors active. The screen (figure 4.8) shows both the teaching instructions and the recommendations from the regular chat. We collect the data to perform the same measures as in the two previous phases. The controlled experiment finishes in this phase.

In the last questionnaire presented to the users, we add the following additional survey:

- I feel more connected with the robot now.
- I think it is more worth interacting with the robot now.
- I find the robot entertaining.
- I find the robot more entertaining than before.
- I find the robot useful.
- I find the robot more useful than before.
- What behavior(s) did you refrain to teach?

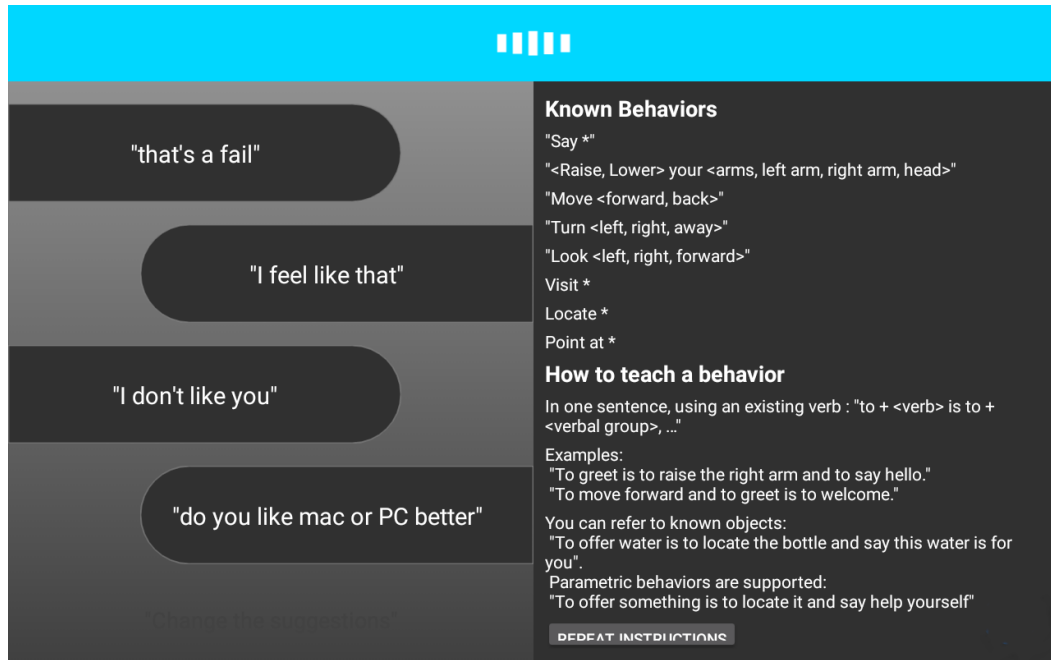


Figure 4.8: Instructions presented to participants in the phase 3, to explain how to teach **behaviors** to the robot, while letting baseline chat suggestions visible.

- What **behavior**(s) would you have liked the robot to know already?
- Any other comment about the **behavior** teaching interaction?

Replies are expected on the Likert scale, between “strongly disagree” and “strongly agree”, to the exceptions of the three last questions, that expect free text answers. Question order is always randomized.

4.5.2 Deployment in Pepper@Home

This experiment targets Pepper@Home users, owning a Pepper 1.8a robot. They are provided installation instructions to bring together all the required components for the experiment.

Prior to our experiment, the **NAOqi** system was updated to the version 2.9.3.114. We compiled all our **NAOqi** packages against corresponding toolchain. When triggering the automatic update of the **NAOqi** applications, we install:

- Version 1.3.7 of the semantic package, providing the right version of the SemanticAgent, implementing our system, and the tool to upload the recordings.
- Version 2.0.0 of the multimodal recorder package, providing the recording tools.
- Version 1.3.3 of the action planning package, providing the ActionPlanning service, producing plans out of **PDDL** descriptions.

Then, when triggering the automatic update of the tablet applications, we install:

- Pepper Explore version 1.5, the autonomous exploration application, that works stand-alone, but also registers a *behavior* in the SemanticAgent.
- DEF, version 2.1.6, providing the updated experimental protocol.

Participants are asked not to try the experiment before all these updates are deployed on their robots. Data collection as soon as possible, and sends the data on a server owned by SBRE, accessible only to the system administrator and the experimenters. The relevant details of this setup are already presented in previous experiment, in subsection 3.5.4.

4.5.3 Results and Analysis

The majority of the participants succeeded in teaching new behaviors to the robot, in real conditions, but not all of them. In this subsection we analyze further the collected data, as well quantitative as qualitative.

We first focus on the impact of mixing the regular chat with the teaching of *behaviors*. Then we evaluate the performance of the teaching observed in this experiment, between the two teaching phases. Qualitative data leads us to highlighting transparency issues, and provide feedback on the regular chat, as well as on the exploration *behavior*. Feedback concur to question the understanding of the interaction by the robot. Finally, we share what the participants said they wanted the robot knew, and what they wanted to teach the robot.

The measurements were produced so that to be comparable with previous experiments. In section 3.5, we define an initial set of measures, that we refine as the analysis goes on. We capitalize on these refinements, and introduce a new refinement about *At w Err*: it has been interpreted as the time between teaching successes, whereas it is not applicable to the phase 3 (“mixed”): people can do different things than than teaching *behaviors*. Instead, we must take into account when new teachings are started, and evaluate the time to achieve a teaching relative to its start¹⁹ When comparing with the *At w Err* of previous experiments, we recompute it accordingly.

In previous chapter we highlight that the *MED* measure on reformulations is not adequate to compare the *behaviors* that the users intended to teach with the *behaviors* the robot actually learned. We propose instead to measure the *MED* on the sequence of tasks, denoted LR_t . We end up with the following grid of measures to express the results:

- The number of utterances (or instructions) provided by the users (*IC*).

¹⁹Given that we track the start of new teachings, we also get to know when teachings are abandoned. Further evaluation of this is possible.

- The number of instructions misrecognized due to automatic speech recognition (*Err*), and the corresponding rate (*Err%*).
- The number of instructions misrecognized due to the natural language understanding (*Mis*), and the corresponding rate (*Mis%*).
- The average time between the beginning of the teaching of a task, and its successful achievement, including the time lost due to any kind of error (*AT w Err*).
- The percentage of **behaviors** that were successfully taught, among the total number of **behaviors** attempted to be taught (*TS%*).
- The Levenshtein ratio between the **behaviors** intended to be taught and the **behaviors** actually learned (*LR_t*).
- The percentage of users who managed to teach **behaviors** (*UTS%*).
- The percentage of users who managed to teach composite **behaviors** (*UTSC%*).

And as subjective measures:

- How successful the teaching felt. It is the Experienced Teaching Success (*ETS*).
- How easy the interaction felt. It is the Experienced Interaction Ease (*EIE*).
- The system usability score (*SU*).

Table 4.2 summarizes these measures on the collected data.

Effect of Mixing on the Regular Chat

Our cognitive system, its goal-oriented task selector, its rule system and our proposed rules and **behaviors** have been mixed with the regular chat, initially provided by the app ABC. That challenging integration has been achieved, without relying on a centralized dialogue system, as most of other cognitive systems do. This could not have been possible if this regular chat content was not provided as **Qi SDK Chatbot**²⁰.

However, the regular chat appears to be affected by being mixed with the other **behaviors**. Indeed, conflicts may have occurred. There have been instructions that triggered both the regular chat and the obey **behavior**, *e.g.* “Regarde à gauche” (look to the left). It appears that the regular chat also provides some basic **actions**. The conflict was properly solved by the rule for staying in context (subsection 4.4.3.3). There were few exceptions where conflicts did not appear solved, but the technical analysis supports that this is a bug, due to a race condition: one **behavior**

²⁰https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/reference/chatbot.html.

Measure	Regular	Teaching	Mixed	All
IC	128	382	315	915
Err%	21.1 %	43.2 %	32.4 %	34.2 %
Mis%	27.5 %	28.5 %	42.2 %	33.0 %
At w Err	N / A	23.0 s	26.2 s	24.0 s
TS%	N / A	39.0 %	25.8 %	34.4 %
LR_t	N / A	54.8 %	66.7 %	58.4 %
UTS%	N / A	70.0 %	33.3 %	72.7 %
UTSC%	N / A	40.0 %	11.1 %	45.5 %
ETS	N / A	1.5 / 5	1.7 / 5	1.6 / 5
EIE	N / A	1.9 / 5	2.1 / 5	2.0 / 5
SU	26.9	17.9	17.9	21.3

Table 4.2: Measurements on the collected transcripts, grouped by phase.

Phase	Mis	¬Mis	All
Regular	60	158	218
Mixed	133	182	315
All	193	340	533

Table 4.3: Contingency of NLU errors between the the “Regular” and the “Mixed” phases. Independence test produces $\chi^2 = 12.1$ and $p = 1.70 \times 10^{-2}$.

suggests a task faster than the other; it is selected, run, but before completion, the other [behavior](#) suggests another task, that has a higher priority, due to the rules. The first task is canceled, and the second is run. However, due to technical reasons, the cancellation is not taken into account fast enough, so the robot had the time to utter its response fully. The conflict was resolved, but too late, so the participants sometimes, but rarely, encountered it.

There is another important source of change: the [speech recognizer](#) is pre-configured in ABC’s regular chat, and this pre-configuration is lost when it is mixed with the teaching, because of constraints of integration. It should have a negative impact on *Err%*. We check this with statistical analysis, but fail to confirm this change is significant ($\chi^2 = 8.17, p = 0.0854 > 0.05$). However, the negative impact on *Mis%* is significant, as shown in table 4.3, ($\chi^2 = 12.1, p = 1.70 \times 10^{-2}$). It can be explained by NLU errors inherent to the teaching of [behaviors](#).

This change also impacted the *SU* negatively. This change is significant ($t = 2.87, p = 1.42 \times 10^{-2}$). We did not achieve a seamless integration of the teaching with the chat. We believe the lower speech recognition accuracy is one of the causes: the more errors, the lower usability score (monotonic relationship). We verify this using a Spearman correlation between *SU* and *Err%* ($r = -0.671, p = 8.55 \times 10^{-3}$) and confirm this correlation.

Teaching Success

The teaching *behavior* allowed the majority of the participants to achieve a teaching within 10 minutes (*UTS%*=70.0%), when explicitly requested to do so in the teaching phase. In the mixed phase, it is normal that fewer participants succeeded in teaching behavior, because they were not explicitly requested to do so. Instead, they could simply chat with the robot if they found it was more comfortable.

However, the rate of successful teachings *TS%* appears different in the different phases. But given our data, this change is not proven significant: $t = 1.46, p = 0.162$ between the teaching and the mixed phase and $t = 0.896, p = 0.383$ between the teaching experiment using spoken language and the teaching phase of the current experiment.

At a glance, it appears that many “successfully” taught *behaviors* are not really what the users meant to teach. Interestingly, half of users happened to teach composite *behaviors* (*UTSC%*=45.5%), demonstrating the reusability of the taught *behaviors*. This success however is to be mitigated: the *LR_t* of 58.4% tells us that of all the procedural knowledge produced by the teaching, less than a half corresponds to what participants intended to teach.

Some participants highlighted that they failed to teach some *behaviors* because of recurrent speech recognition errors. Moreover, they failed to test the taught *behaviors*, because of the speech recognition errors as well. For example, in French, the example provided in the teaching phases included the verb “recevoir” (greet). At the imperative mood, “reçois”, it is homophone to the 3rd person conjugation “reçoit”. Knowing this limitations, some *behaviors* were refrained to be taught, because participants could predict they may be hard to understand at the imperative mood. These participants also tried to work around this issue by using indirect speech, like “j’aimerais que tu reçoives” (I would like you to greet). It appears that supporting a variety of indirect speech may allow users to work around the ambiguities of the *spoken language*.

Some participants visibly tried to close the teaching by themselves, by saying “c’est tout” (that’s all), which mirrors the robot’s utterance “c’est tout?” (is that all?). The teaching interaction can be easily improved by supporting this form of control on the teaching.

Transparency Issues

Teaching should be an iterative process where the subject gets refined, as it is exchanged back and forth between the protagonists. Here the participants often

failed to ask the robot to perform the behaviors while they were teaching them. When they managed to ask the robot to perform a task, the robot would start it with no feedback, so well that the user did not even know it provided a well-formed order to the robot. This is a transparency issue.

Four participants complained of transparency issues. For instance, it is hard to understand when the robot is trying to learn from our instructions, or if it is waiting to execute instructions, or something else. They feel like there are modes, but cannot grasp them. Because of that, they cannot work around the limitations or the misunderstandings of the robot. Often, when the robot believed the instruction could fit in a recipe the user seemed to teach, the robot said “okay”, with no other detail. Users could not tell what was the robot okay with, and could not correct the misunderstanding. By recalling what the robot is okay with, this kind of misunderstandings can be corrected.

This bug, where “okay” is responded indefinitely, shows that the task selection is vulnerable to behaviors abusing one of the system’s rule. It could be improved by providing few exceptional rules to leave locked interaction patterns²¹, associated to some meta-interaction behaviors. In other words, we should be able to talk about the ongoing interaction with the robot.

Most participants assume the responsibility of the good understanding of the robot. They know they are more intelligent than the robot, and accept that the robot cannot guess what they mean. They would provide some feedback saying that they could have achieved the teaching if something was different.

One participant suggested adding some feedback to distinguish the label from the recipe.

Comments About the Regular Chat

The regular chat from ABC had not been scientifically tested yet. This experiment is an opportunity to do so, and collect feedback about it. Two participants said they appreciated to be guided by suggestions in the phases that included the regular chat. It gave them incentives to talk to the robot, at the cost of their imagination. Two participants highlighted that the suggestions were out of context, so they were not understood or were irrelevant. Participants would prefer having an overall understanding of what the robot actually knows about, and what it can do, expressed in a more synthetic manner.

One participant also complained about the inconsistency of the contents. For instance, there had been an issue with the robot proposing to sing songs, whereas they were technically disabled in the application. In addition to this issue, the robot would pretend to understand something, but be incapable of recollecting it. It is a form of contradiction, and it is regularly tracked down by dialogue writers. The more the dialogue content is refined, the fewer contradictions remain. However, the

²¹We find an analogy between these locked interaction patterns, and dead pieces of Petri networks.

ability to recollect requires a deeper understanding of the social interaction, and the ability to build knowledge related to past conversations with users.

Some participants also complained about the quality of the speech recognition in the regular phase. So despite a higher accuracy of the speech recognition in that phase, it still appears problematic.

Feedback on the Exploration Behavior

The exploration [behavior](#) regularly failed because of light conditions in homes. Pepper is a robot designed for businesses, which are often brighter than homes. Therefore the localization action was sometimes not well suited for home. It was worked around by adding light sources in the room of the robot, or by switching rooms the robot.

When it worked, the robot found objects that were often interesting. These objects have been successfully reused in a teaching in only few occasions, although most participants attempted to reuse them. Interoperability has therefore been demonstrated, despite difficult interaction conditions. But only two participants managed to ask the robot to perform [actions](#) on them. Usually, they failed because of speech recognition, and of the transparency of the teaching interaction. When they succeeded, they did not understand that the robot was relocalizing itself before trying to reach the target object: the robot provided no feedback about its current activity, nor acknowledged the order was well understood.

In the mixed phase, the exploration [behavior](#) is triggered autonomously. It happened when the robot lost the user after a while, which should not have happened during the experiment. When this happened, the participants had no idea of what was going on, and hardly failed to repair the interaction.

Engagement and Issues Reading the Room

The robot is not really aware of who it is talking to. Worse, it disengaged from 3 participants, to look somewhere else around. This betrays the robot's lack of understanding of the people surrounding it: it does not really feel the presence of people, nor is able to recognize them. As a result, users cannot build a relationship by interacting with the robot.

The robot pretends it knows about the user, and this is quickly confounded by the user, who end up finding the robot's [behavior](#) deceptive. Instead, if the robot were honest and explicit about its difficulties to understand people, it may give an opportunity for users to repair the engagement. That would require a feature allowing users to draw the attention of the robot.

Thanks to the video recordings, we are able to see the context in which the experiment was held. People usually have their family around, but they try not to disturb the experiment. Sometimes the users do other things during the experiment: they take care of their children, they assist people, comment the interaction with them, or take notes. They are quickly tempted to switch activity, so that performing

the experiment is a genuine effort. Interestingly, in the mixed phase the robot exhibited autonomous [behaviors](#), and happened to ask a participant if an object found was interesting. When this happened, the participant's attention has been drawn back to the robot. But the robot does not realize that: it cannot read the room, and for now has no way to know whether it can draw the user's attention for a teaching, or if it should stay around silently.

What Users Feel and Want

We compute the following statistics expressing specific perceptions about the robot:

- I feel more connected with the robot now: $\mu = 2.00, \sigma = 1.26$
- I think it is more worth interacting with the robot now: $\mu = 2.67, \sigma = 0.816$
- I find the robot entertaining: $\mu = 2.50, \sigma = 1.22$
- I find the robot more entertaining than before: $\mu = 2.67, \sigma = 1.21$
- I find the robot useful: $\mu = 2.33, \sigma = 0.816$
- I find the robot more useful than before: $\mu = 2.67, \sigma = 1.51$

People appear to find the robot slightly more useful, slightly more entertaining. It is overall slightly more worth interacting with the robot, with these new features. Participants suggested the following [behaviors](#) to be already present in future versions:

- Discuss about the robot's [behaviors](#): show them, explain them, forget them, correct them.
- Greetings [behaviors](#): welcoming, saying hi, reply to a smile.
- Look for something on the Internet: a definition, a date, the name of someone.
- Give the weather.
- Talk about food.
- Play music.
- Dance.

They expect to be able to reuse these [behaviors](#) to teach new [behaviors](#). Here are the [behaviors](#) they refrained to teach:

- Check some general information on the Internet and do something about it, *e.g.* to tell the weather, check the outside temperature and say it.
- Greeting the baby of the family.
- “Not safe for work” [behaviors](#). They require privacy guarantees to do so.

Experiment	IC	Err%	Mis%	At w Err	TS%
Teaching	236	44.2 %	43.0 %	23.0 s	34.4 %
[Paléologue et al., 2018]	480	33.8 %	20.4 %	28.7 s	46.4 %
[Gemignani et al., 2015]	163	7.98 %	4.29 %	62.4 s	100 %

Table 4.4: Result comparisons between [Paléologue et al., 2018] and [Gemignani et al., 2015].

Comparison with Previous Experiments

Using the updated measure definitions presented at the beginning of this subsection, we proceed to a comparative analysis of these results with the ones of previous experiments. Therefore some measurements, like *At w Err*, take different values than presented in previous chapter, in subsection 3.5.3. We can compare the teaching phase, which is dedicated to the teaching, with the experiment held in [Paléologue et al., 2018] and [Gemignani et al., 2015]: the user’s only task is to teach behaviors, therefore the instructions are targeted to teaching. *IC* and the error rates *Err%* and *Mis%* therefore relates to these instructions, and not to other unrelated dialogues.

In table 4.4, we recall these measurements together, and add *TS%* to support why *At w Err* appears to be shorter in our experiments than in [Gemignani et al., 2015]’s: it could be due to the fact that our teaching does not support correction, and that the users are allowed to abandon their current teaching. Users actually abandon teachings regularly: 34.4 % of the teachings have been abandoned in this experiment. In [Gemignani et al., 2015], users do not have this liberty, so the teachings are necessarily successful.

What about the raise of *Err%* and *Mis%* between our previous experiment and this experiment? We do not have enough data to conclude whether these changes are significant. As we can see, it is difficult to perform comparisons between experiments without more data. However, these results allow other comparisons to be performed. We have published the annotated transcriptions and their analysis online²².

In theory, all of these objective measures depend on what is being taught to the robot. Because we are in open scenarios, we cannot fix the bias in a certain way, to make sure to know what we are comparing. Ideally, we should be able to study these biases *a posteriori*, by looking at the actual contents of what was taught, and draw more accurate conclusions on this data. Without this, we cannot be sure of which detail makes the significant difference between given works, even with the same measures. There must be a better set of measures: either targeting a more specific performance indicator, on fixed tasks, but a larger variety of them, and

²²<https://gitlab.com/victor.paleologue/teaching-robots-behaviors-experiments/tree/master/5%20-%20Rich%20Interaction>.

Session	TS%	Teaching Attempts
5b594253-7113-4ed5-adaa-bade188e0c9b	0.00%	1
a1560a8d-c1e7-4435-b176-0df6db8556c7	0.00%	6
a7fdd8f6-261e-43f4-ac53-a1f1c223a5e1	0.00%	10
bbf36ca1-957d-493b-a8d3-609be0cb2bc4	25.0%	4
3018b80c-88c6-4d12-b38a-006e03ff2640	28.6%	14
583e730c-d2c3-4604-ad08-e81fe94625e9	28.6%	14
09177fb2-73d1-4830-9db0-bad63d45cf59	33.3%	3
ec3785d2-fd18-4296-90a1-e92e680952ae	45.5%	11
96ff7c2b-0f73-48d1-9c29-e02140931cca	50.0%	6
37740586-ea11-4f56-829e-5b79b7e37372	53.3%	15
361df488-80d8-4677-8a65-a9745cc650f8	71.4%	7

Table 4.5: Rate of successfully taught behaviors (*TS%*) over attempted teachings, by session.

applicable to social robots – [Gemignani et al., 2015]’s pick and place tasks are not suitable for Pepper – or relating to the structure of behaviors, measuring on the performance of changing producing and modifying these structures.

Moreover, the *Mis%* measure is for now a general measure. In our configuration, where behaviors interpret the input independently, we should rather distinguish the interpretation errors of each behavior. There would be a *Mis%* measure for each behavior. We also encountered cases where the behavior could not be performed. For the user, it made no difference with a case of misinterpretation. Behavior failures should also be taken into account, and so should be the erroneous conflict management from the task selector.

User-Centered Study

In this experiment, we managed to have users participating to several phases of teaching, and of interaction. Each recorded session corresponds to a user, that we can therefore characterize. For instance we identify that some users were more proficient than others to achieve the teaching. Table 4.5 presents the *TS%* indicator for each session, and therefore for each user.

It appears that some users managed to become proficient in teaching behaviors, whereas some never succeeded a teaching.

4.5.4 Conclusion on the Experiment

We set up an experiment adapted from the one presented in previous chapter, in subsection 3.5.4. It consists in distributing the software presented in this section to robots in homes, as part of the Pepper@Home project. 11 participants performed the experiment divided several phases. We collect the data for 3 of these phases, exhibiting different **behaviors**:

Regular The regular chat from ABC, serving as a baseline content of what a robot at home is currently capable of doing.

Teaching Our teaching **behaviors**, like presented in previous experiments, but with the support for parametric **behaviors**.

Mixed The mix of the regular chat, the teaching **behaviors**, and the exploration **behaviors** presented in this chapter.

From this data we compute a summary of the performance of the teachings. We demonstrate that the change of the configuration of the **speech recognizer** has a negative impact on the speech recognition error rate *Err%*. In turn *Err%* has a significant impact on *SU*, measuring the system usability.

We identify various caveats in the teaching interaction: lack of transparency and feedback, lack of control, lack of fallback to play around speech recognition errors, lack of some predefined **behaviors**. Some of these issues have been identified since the first experiment in controlled conditions. There is potentially a lot of performance to gain by fixing these issues. This is a favorable cue, that suggests that the teaching interaction can become much more usable in the future.

We have not tapped from this potential since that first experiment. As a result, the teaching interaction remained similar between the first experiments and this experiment. Therefore we have been capable of attempting comparisons between these experiments, but due to insufficient data, we could not draw conclusions. We chose these measures to allow comparisons with some previous works. Now we question them and suggest that future work should focus on teaching specific tasks, or on detailing what operations the teachings produces on the **behavior** knowledge, and on evaluating the performance on these operations.

Most importantly, we demonstrated empirically that our system was capable of:

- Producing **behaviors** that are reused, including in other **behaviors**.
- Producing interoperable knowledge about entities that can be reused in **behavior** teachings.
- Not processing dialogues in dedicated, centralized dialogue system.
- Support unexpected **behaviors** and objects from an **open scenario**.
- Evolving in unknown real homes.

- Exhibit a richer variety of [behaviors](#), and resolve conflicts between independently designed [behaviors](#).

As much as possible, we respected the [HRI](#) recommendation to draw conclusions on empirical data, collected about users in their real environment.

4.6 Conclusion

In this chapter, we explained how we extended the cognitive system presented in the previous chapter. We added [behaviors](#) capable of producing knowledge on physical objects and spatial regions surrounding the robot. It is implemented as a separate application, Pepper Explore, running in the Android system of the tablet, whereas our system is installed directly in the robot.

We defined an ontology based on DOLCE [[Gangemi et al., 2002](#)] to exchange knowledge between [behaviors](#), and with the system. We defined a novel interface to share procedural knowledge with our system: action factories, associated with parameter specifications and semantic templates, to express these [actions](#) in natural language.

The new [behaviors](#) include spoken interaction with humans to label the discovered entities. This interaction conflicts with the other [behavior](#) with the robot. In this chapter we explain that a goal-oriented approach may resolve these conflicts. We set up a symbolic planner, and combine it with a rule system to promote certain [goals](#), in order to resolve the conflicts. We implement 3 rules: responding to communicative acts, avoid misunderstanding communicative acts and staying in context.

This is integrated in our cognitive system, and deployed on Pepper@Home robots, for an experiment. We compare the user experience using [System Usability Scale \(SUS\)](#) [[Brooke, 1986](#)] between the baseline dialogue of the robot, the teaching of [behaviors](#), and the mix of all [behaviors](#). Thanks to quantitative analysis, we determine that the usability of the baseline dialogue is significantly reduced by the mixing with the other [behaviors](#). It appears it is rather due to the difference of configuration of the [speech recognizer](#) than to the rise of conflicts.

Most users (72.7 %) managed to perform the teaching. This is a real step forward for research:

- The robot can be taught new [behaviors](#) by composition of known [behaviors](#) using the [spoken language](#) only.
- The robot has other purposes than being taught [behaviors](#).
- Behaviors are developed independently, and compete with each other, causing conflicts that can be resolved using an explainable rule-based system.
- Behaviors can exchange their knowledge of the world, and their procedural knowledge with each other: they are interoperable.

- Thanks to interoperability, the number of possible **actions** of the robot is multiplied every time a new entity or a new **behavior** is learned.
- The **behaviors** and the system are suitable for open scenarios, for which objects, places, and **behaviors** cannot be programmed in advance.
- The above has been demonstrated in homes of people living with a Pepper robot; in other words, in real conditions.

However, thanks to qualitative observations, we identify various caveats in the teaching interaction: lack of transparency and feedback, lack of control, lack of fallback to play around speech recognition errors, lack of some predefined **behaviors**.

To conclude this thesis, next chapter discusses how these issues may be tackled in the future, and the numerous perspectives this system, and our research, offer.

Conclusions and Perspectives

Contents

5.1	Conclusions and Take-Aways	120
5.2	Generalization on Behaviors	123
5.2.1	Generalization of Implementation	123
5.2.2	Behaviors Everywhere	123
5.2.3	Reasoning on Behaviors	124
5.3	Interaction Repair and Meta-Interaction	125
5.3.1	Interaction Repair	126
5.3.2	Interaction for Defining Goals and Rules	127
5.3.3	Meta-Interaction for Learning to Achieve Goals	127
5.4	Detecting Mistakes and Learning from Them	128
5.4.1	Detecting, Repairing and Learning from User's Mistakes	128
5.4.2	Adapting from Robot's Mistakes and User Corrections	129
5.4.3	Autonomously Making Mistakes for Learning	129
5.5	Next Behaviors and Potential Challenges	130
5.5.1	Correcting Behavior Compositions	130
5.5.2	Teaching Animation Actions	130
5.5.3	Autonomous Recharge	130
5.5.4	Human Greetings and Identification	131
5.5.5	Hush Rule	131
5.5.6	Behavior Modification Evaluation	131

In this conclusion chapter, we recapitulate the objectives this thesis fulfills, and summarize some take-aways.

Then we discuss our model and our cognitive system one last time, to present a possible generalization. This is the subject of section 5.2. It appears it provides an interesting ontology, that may lead to new behavioral, and event learning capabilities.

In section 5.3, we tackle the question of interaction repair. Our results in previous chapter show that a significant part of the interaction with the robot fails. We draw some perspectives about how to reduce interaction failure, and it leads us to a possible generalization on the content of the *behaviors*, and additional opportunities for the robot to learn.

There is indeed a lot to learn from the various errors the robot makes, including from interaction failures. Section 5.4 describes our approach to apply this on a robot, based on our proposed generalization.

Then we propose new *behaviors* to implement in the future to confront our theories, but also, more pragmatically, to improve the experience with Pepper. We predict they would raise some new challenges, related to interesting research subjects.

5.1 Conclusions and Take-Aways

This thesis demonstrates empirically the teaching of new *behaviors* to a robot using *spoken language*, with a unique set of constraints:

- The new *behaviors* are unpredictable: the scenario was open, and the experiment participants have the freedom to invent the *behaviors* to teach.
- The robot not only can be taught *behaviors*, but it also has a rich set of the preexisting *behaviors* of a general-purpose robot, and competing with the teaching.
- The *behaviors* are implemented by separate applications, but nonetheless contributed to the cognitive system with new *actions* reusable in the teaching, and knowledge on entities of the world that could be leveraged by these *actions*.
- The interactions are achieved on an off-the-shelf robot, Pepper, with no extraneous device, in real conditions, close to the wild.

The teaching of *behaviors* using *spoken language* is demonstrated with a population of 11 experienced users, living with a Pepper robot at home. The teaching of *behaviors* is supported by a novel cognitive system that is developed to run on standard Pepper robots. The other *behaviors* are developed as releasable Pepper applications. All binaries are available on direct request to victor (at) paleologue (dot) fr. The cognitive system features:

- An interface to gather **behaviors** and action factories from separate applications. Behaviors may suggest tasks to perform immediately, while action factories provide **actions** on demand.
- An interface to annotate **actions** with natural language, and thus reuse them in instructions.
- A knowledge base and an ontology to share entities between applications, allowing other **behaviors** or **actions** to reuse them as parameters.
- A goal-oriented task planner that selects which task suggestions are most suitable to the current state of the world, and the given **goals**.
- A rule system to prioritize certain **goals**, and guarantee that the robot: replies to users, prefer relevant responses, and avoid switching subjects.
- By design, no distinction between dialogue-based **behaviors** and the other **behaviors**. For the system, speaking is an action like any other.

The fact that **behaviors** are implemented by separate applications that suggest tasks to perform instead of performing them directly stands out as a novelty. It inspired the Chatbot **API** that is already found in the **Qi SDK**¹. It allows a goal-oriented approach to be experimented by **SoftBank Robotics Europe (SBRE)** teams, on Pepper robots. The **behaviors** deployed with the system supports:

- The obedience of the robot: it would perform the **behaviors** it is instructed to.
- The teaching of new **behaviors** to the robot, by composition of existing **behaviors**, using the **spoken language**.
- The introspection of known **behaviors**, that the robot describes in **spoken language**.
- The fallback to a generic answer, where the robot says that it did not understand.
- A baseline dialogue to present the robot, talk about various arbitrary subjects to entertain the user.
- An exploration **behavior**, that makes the robot discover objects in its environment, remembering some of their visual features and their location.
- A labeling **behavior**, taking the initiative to ask the user about a discovered object, and about its location. These objects can then be localized and pointed at on demand, and the locations can be visited.

¹https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/reference/chatbot.html

The [behaviors](#) for obeying, teaching new [behaviors](#) and introspecting [behaviors](#) have already been integrated in [NAOqi](#), but not yet exposed to the public.

This thesis is presented as a contribution to [Interactive Robot Learning \(IRL\)](#), at a cross-roads between [Artificial Intelligence \(AI\)](#), [Interactive Task Learning \(ITL\)](#) and [Human-Robot Interaction \(HRI\)](#). Besides the demonstration of a teaching of [behaviors](#) using [spoken language](#) with unique constraints, and in unique conditions, this thesis contributes by:

- Establishing an updated state of the art on teaching [behaviors](#) to robots using the [spoken language](#).
- Demonstrating that our teaching of [behaviors](#) using the [spoken language](#) was not domain-independent, despite being designed in a domain-independent manner: it could be due to speech recognition, or to a bias in our [Natural Language Understanding \(NLU\)](#).
- Demonstrating that users naturally split down their teaching of [behaviors](#) into several utterances. They cut their sentences before or after “is to” and between steps of enumerations.
- Pepper’s speech recognition error rate is dramatically increased if the speech recognizer is not pre-configured.
- Refining some existing measures, by applying them to more realistic scenarios, showing their maturity and their limits.
- Identifying unsupported forms of interaction: introductions, indirect speech, etc...
- Identifying unsupported forms of teachings: by providing instructions in the imperative form, asking the robot if it knows how to do something...

This thesis has been an opportunity to build an advanced system that moved beyond the state of the art. There has been a lot to gain by applying the constant progress of research in robotics and in [AI](#) on a standard Pepper robot. For the scientific community, because [SBRE](#) provided its expertise to develop maintainable software for Pepper, and an opportunity to deploy the experiments in homes. For [SBRE](#), because the integration of the system is already well advanced, and may be reused in future commercial products.

We introduced this thesis with the expression of [SBRE](#)’s need to allow users to adapt their robot’s [behaviors](#) according to their needs, without requiring the help of an expert developer. We believe that the outcome of this work effectively opens the door to such adaptation, and that the software we produced can be refined to be truly usable, even for home users. Moreover, we show that teaching [behaviors](#) also implies teaching about objects, places, or people, and understanding and applying intelligible rules. All of these features contribute to build the confidence that the robot can understand the world it is thrown into, and that we, users, are in control.

And this is, above all, what the general public expect from robots they may share their lives with.

5.2 Generalization on Behaviors

5.2.1 Generalization of Implementation

In the software implementation, we factorized **behaviors**, tasks and **actions** under a single class, “Action”. This was seen as a generalization of the Qi SDK Actions described in annex 5.5.6.

We consider this class to correspond to a potential procedure, that can be parametrized through public properties, and run on demand by calling the “run” method. It appeared to implement well the **actions** shared through the action factories described in subsection 4.2.6.

The same class also implemented the task suggestions well: they can be run, but not parametrized, because their host **behavior** implementation did it in advance. Because of this “run” method, the same class also can also serve to implement **behaviors**, which are also a procedure, but also exposing task or **goal** suggestions. This generalized implementation suggests a broader generalization is possible.

5.2.2 Behaviors Everywhere

From a psychological point of view, anything an agent visibly does is a **behavior**. In this subsection, we try to apply this in our cognitive system. The **behavior** implementations, that suggest tasks to perform, remain unchanged. But the tasks are not limited to **actions** anymore, and may provide **behaviors** which, if they are selected, may be run. Behaviors may therefore be running, or not. Here we will say that a **behavior** is “active” or “inactive”. On the other hand, action factories may also be able to provide **behaviors** instead of **actions**, so we will call them here “behavior factories”.

This model may support the following use case. The robot knows how to wish a happy birthday to someone. There is a “to wish a happy birthday to someone” **behavior** factory. Alice tells the robot to wish a happy birthday to Bob when the robot meets him. This is done through a “programming” **behavior** and results in the production of a new **behavior**, “wish a happy birthday to Bob when I meet him”, and the suggestion of a goal leading to its selection and therefore activation. When the robot meets Bob, this **behavior** reacts by producing the **behavior** “to wish a happy birthday to Bob”, and suggesting a **goal** leading to its selection. The next day, Alice tells the robot to stop wishing a happy birthday to Bob: the “programming” **behavior** stops to suggest the associated goal, the **behavior** “wish a happy to Bob when I meet him” is deactivated, and may not produce new **behaviors** and **goals**. This generalized model naturally allows our new use case. Figure 5.1 recapitulates this model in action.

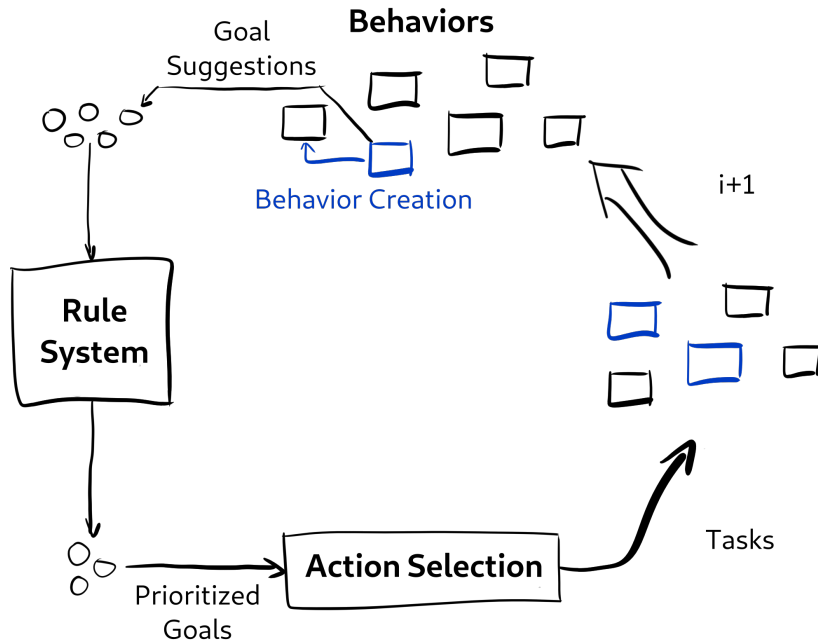


Figure 5.1: Unified model omitting the distinction between **behaviors** and **actions**. Active **behaviors** (blue boxes) may react by creating new **behaviors**, and **goal** suggestions. Goal suggestions are prioritized using the rule system. The action selection produces the tasks to perform: a selection of **behaviors** to activate for next iteration.

This raises again the question of the difference between **actions** and **behaviors**. We published an answer in the repository of the ontology we provided in previous chapter²: a **behavior** is the unit of behavioral knowledge, but its implementation is independent. An implementation may directly actuates the robot when run, like **Qi SDK Actions** do, or instead suggest tasks or **goals** to the system, like our current **behavior** implementations do. The transition to the new model requires to split the production of **behaviors** (**behavior factories**) from the suggestion of **goals** (**goal sources**).

It also allows another use case: the user tells the robot to never communicate with this friend anymore. Given that the birthday **behavior** describes in the **PDDL** that it will produce a communicative act addressed to the friend, and that there is a **behavior** that can suggest the goal of avoiding a communication with the friend, it is possible for the task selection to rule out the birthday **behavior**. This is the basis of the In short, accepting **behaviors** in the task selection system allows a more powerful control of these **behaviors**.

5.2.3 Reasoning on Behaviors

In previous subsection we described a generalized ontology for addressing **behaviors**. That ontology can be expressed in the knowledge of the robot, and therefore can

²<https://gitlab.com/victor.paleologue/teaching-robots-behaviors-ontology>

be articulated with the rest of the ontology, presented in 4.2. It would correspond in DOLCE [Gangemi et al., 2002] to a sub-category of “process”. A *behavior* can be performed and be applied with specific parameters. An instance of performing a *behavior* is an event³ and is performed by a social agent⁴. We call it “behavioral event”, but see it as what is usually called an “activity”. A communicative act⁵ is a particular type of behavioral event, and was designed to be compatible with this generalized ontology. Note that with this ontology, we are able to avoid the distinction of dialogue-based *behaviors*.

We have all the means to reason on the activity of an agent, regardless of it being the robot, or a user. Moreover, with the teaching of *behaviors*, we brought meaning to the *behaviors*, that are grounded on both the instructions and their implementation.

The state of the world and the *goals* remain related to *behaviors* thanks to the PDDL annotation of *behaviors* with effects, and the suggestion of *goals*. On the one hand, *behaviors* can be deduced from the *goals* set in the system, and the observed state of the world. On the other hand, *goals* could be deduced from the past activities. One approach consists in looking for the effects of an agent’s activity on the state of the world. Another approach consists in recognizing the activity, for example from visual observation [Sung et al., 2012], and deduce the *goal* from the past experience of the robot. Deducing the *goals* or intentions of agents is in fact a way to build a *theory of mind* for the robot [Scassellati, 2002]. And this model depends on the *behaviors* the robot knows, and their effects or *goals*.

Reasoning on past activities may also allow a robot to recognize which *behavior* leads to achieving a goal. In theory, a robot could set itself arbitrary *goals*, and try to achieve them, to reinforce this knowledge. This is a form of intrinsic motivation, driving the robot to explore its ability to reach *goals* that were never attempted before. It compares to [Baranes & Oudeyer, 2010]’s autonomous exploration of the sensorimotor space, but differs because the tasks to attempt would be much higher-level in our system. This could be called “behavior babbling”. We believe however that the symbolic model may not be reliable, because any extraction task may fail. Therefore such autonomous learning by the robot may certainly need to be corrected by human intervention. The next section develops the importance of understanding the human inputs, and how to improve that understanding by supporting the repair of interaction.

5.3 Interaction Repair and Meta-Interaction

Our two experiments in homes, respectively analyzed in subsections 3.5.4 and 4.5.3, showed a dramatic amount of misrecognition errors, due to the speech recognition, or due to the NLU. There are technical solutions to reduce the error rate, but a

³See subsection 4.2.4.

⁴See subsection 4.2.1.

⁵See subsection 4.2.5.

cognitive system should not expect the understanding to be perfect. Even humans fail to understand one another, and must compensate this with specific [behaviors](#).

5.3.1 Interaction Repair

Humans can compensate these errors through a form of meta-interaction: they can discuss about previous piece of interaction, identify what went wrong, and suggest a correction. This is called interaction repair, and is missing from our current [behaviors](#).

Typically, in our experiments, we observed participants say things like “Non j’ai pas dit ép pointer” (no, I did not say to blunt), hence trying to correct a past misunderstanding. Participants also missed some feedback from the robot to let them know what it really understood. Every [behavior](#) should be refined to make explicit how they understand what is told to them, and to support corrections on that understanding. For instance, the composition of new [behaviors](#) should make the robot recall each piece of [behavior](#) it understands, and allow users to correct them, like it is done in [Lallée et al., 2010]. Using semantic templates introduced in subsection 4.2.7, we could also identify incomplete instructions, warn the user about it, and look for completion or correction. Proper error handling was explored successfully in [She et al., 2015]. It is also applicable to ambiguous instructions.

Interaction repair is an additional complexity to add to the [behaviors](#) pre-programmed in the robot. To keep the development of such complexity affordable, it may be useful to develop a framework generalizing these pieces of interaction. [Peltason & Wrede, 2010] is an attempt to do so, and provides a general approach on collecting the target information to close a piece of dialogue. There is therefore an incentive to find generalized models to describe interaction patterns, like we attempted to do by describing the pragmatic frames involved in the interaction we designed.

This generalization may potentially go beyond the interactive case. For instance, the exploration [behavior](#) described in subsection 4.1.1 also include some expressive cues: as a side effect of looking around for the object, surrounding users can feel the robot is busy, and trying to look for something. Moreover, when a point of interest is discovered, the robot triggers an animation expressing its satisfaction⁶. Surrounding users may feel this satisfaction, and understand well what the robot is into. This gives us the intuition that pragmatics are involved even if there is no knowledge of surrounding users.

Our intuition is that the pragmatics can be expressed as a form of generic grammar of interaction for extracting or sharing information, that may be applied with non-agentive entities⁷. Based on this hypothesis, we identify a recurring [pragmatic frame](#) in the exploration [behavior](#), shown in table 5.1. It may also be found in dialogues as well as in various forms active observation behaviors.

⁶Note that humans also perform this kind of expressive gestures when they are alone

⁷Note that the [theory of mind](#) [Scassellati, 2002] includes adopting the intentional stance for non-agentive objects. See also [Pérez-Osorio & Wykowska, 2019] for the intentional stance.

World Perception	Robot
	1. Probe
2. Result	3. Confirm

Table 5.1: Recurring **pragmatic frame** for a robot exploring. The robot probes by looking somewhere for points of interest. The result is provided by the perception system, and then the algorithm confirms by switching the **behavior** of the robot.

However, generalizing on the interaction patterns within **behaviors** does not apply to the correction of the task selection itself, that plays a major role on what the robot does.

5.3.2 Interaction for Defining Goals and Rules

It is still only an hypothesis that the rules used for task selection are explainable. It should be demonstrated empirically, thanks to a **behaviors** capable of discussing past task selections, and existing rules. This also implies to put words on all the involved entities, predicates, and rules. This could be done using semantic templates.

Then, if a robot is able to discuss about how its decisions were taken, and that the semantic analysis still supports open scenarios, we are close to being able to define new rules. A user must be able to name a rule, and state which **goal** the rule is meant to favor. Then, the user must be able to order the priority of this rule. This seems very powerful, but we predict that in practice, it is hardly applicable. The priorities of certain **goals** are highly contextual, and are sometimes too hard to express in simple words. Reinforcement learning techniques may be more suitable to learn these priorities, and may not systematically compromise their explainability, as demonstrated in [Keneni et al., 2019].

Nonetheless, a user can set explicit **goals** to the robot, which gives the robot a sense of purpose: what it is meant to do, or learn to do. And meta-interaction can be leveraged to learn autonomously, and better.

5.3.3 Meta-Interaction for Learning to Achieve Goals

[Bruner, 1983] shows that infants manage to learn the language thanks to a shared agreement on the interaction formats (or patterns) with their teacher. Their teacher is aware of these patterns, and respect them to help the infant learner to identify well the thing to learn. This is called **scaffolding** and was introduced in subsection 2.1.1. Infants are able to recognize the interaction patterns, or formats, propose them (by trying to apply them), reason on them (*e.g.* by evaluating their match) and negotiate them, by agreeing or refusing to adopt them. By doing so they build an abstract model of their situation, that serves as a solid matrix for interaction. Through ritualized interaction, they build up their other skills.

A [behavior](#) specialized in meta-interaction, *i.e.* in discussing what the interaction is about, and what it should lead to, could reproduce this negotiation of interaction formats. The users could then announce the interaction format, *e.g.*: “I am going to teach you a move”. This piece of meta-interaction should lead to having some [behaviors](#) promoted as a consequence. Here, a [behavior](#) for learning moves interactively would be granted a temporary priority, as if there was a temporary rule about it. Meta-interaction would therefore allow users to refine the [task](#) selection policies, and thus negotiate the interaction formats.

Using a specific behavior, the robot may correlate sequences of [behaviors](#) with observed effects. For instance, being taught a move leads to the creation of a new behavior in the pool. The robot may initiate certain interactions to test such hypotheses: either by trying to reproduce the sequence of interactive behaviors, or by asking the user about these [behaviors](#). In both cases the robot can train a policy capable of determining a sequence of behaviors to produce these effects, and thus learn how to reach arbitrary goals.

Once the robot has determined the effects of a behavior, it can tell whether a behavior was successfully performed or not. This is explored further in next section, in subsection [5.4.3](#).

5.4 Detecting Mistakes and Learning from Them

In previous section we described how [behavior](#) implementing a meta-interaction can lead to improving the task selection policy. This can be generalized to other forms of knowledge.

5.4.1 Detecting, Repairing and Learning from User’s Mistakes

The simplest mistakes to detect are those who can be deduced from the inconsistency of the user input with the knowledge of the robot. For instance, if the user asks the robot to point at a flower pot, but the robot does not know any flower pot, the robot can deduce that some flower pot exists, but it does not know what it is. It can therefore respond to ask about the flower pot. If the user wants to be helpful, he or she may show or explain what is a flower pot. This kind of teaching situation is ideal: the user and the robot are focused on sharing the understanding of a particular object in space.

Another example consists in trying to correct incomplete or ambiguous instructions. It is straightforward to ask to the user for correction. It appears that certain ambiguities are recurrent, and can be solved probabilistically [[Nyga, 2017](#)]. But by looking for user validation when probabilities are low, or when risk is high, is a good complement. Moreover, it produces data from a real situation, that the instruction interpretation algorithm can train itself on.

As we can see, by reacting properly on user mistakes can lead a robot to improve its skills or knowledge. It is also applicable to mistakes the robot makes.

5.4.2 Adapting from Robot's Mistakes and User Corrections

If an ongoing **behavior** was to be interrupted by the user, the robot should ideally try to understand why. The user may tell what is wrong, and this correction can only be interpreted in the context of the **behavior**. This is a case of interaction repair, and we propose that all **behaviors** try to implement that. With the hope that some framework can be used to simplify the implementation of interaction repair, we can also hope to find a systematic approach to learn from these repairs.

For instance, if during an exploration, a robot went to some place that is disturbing, it should be possible to ask the robot not to go there. This can be shared in the knowledge, so that other **behaviors** may know that this place is special. It can also be used to build an adaptable policy to understand what area exactly is forbidden, or in which situation it is preferable to avoid it. The repair interaction provides a source of validation of any **behavior** that implements it.

Speech recognition too can be trained this way. So long that the robot misunderstands the users, and that the users can correct such misunderstandings, it gathers a data set to train a complementary model to the speech recognition (on a remote server, or when the robot is idle). In the long run, this approach may solve the speech recognition issues (such as the mistranscription of homophones), as the robot gets accustomed to its users speech. Therefore meta-interaction and interaction repair can be leveraged from any **behavior**, to learn from the robot or the user's mistakes.

5.4.3 Autonomously Making Mistakes for Learning

The robot may also learn from autonomous behaviors. Given that the robot knows the effects of **behavior**, either because it has been hard-coded in the **behavior** or because it has been deduced from experience (see subsection 5.3.3), it may discriminate whether a behavior has been performed successfully. Then if a behavior has a high rate of success, it can be used as an oracle to train other algorithms.

For instance, if a "look at" action existed, it would have for effect that the target remains visible. The robot can be programmed to decide autonomously to look at a certain point in space, located using a reliable, but unknown algorithm. It can learn visual cues, and train itself to remember what its visual field looks like. Then, it can move its arm around, and discover the effects of doing so in its field of view. This can be tried with various backgrounds, until the robot can predict the effect of its arm position in its field of view. It can then use this to correct its vision by estimating what the world should look like without its arms in the way.

Given a history of past activities, the robot can recognize the **behaviors** that increased its knowledge of the world, and the quality of this knowledge. It can try to autonomously perform them (using an intrinsic motivation), and refine its skill for finding new and reliable information. This would not only improve the growth of the knowledge of the robot, but also make the robot look curious about its world: physical, abstract and social.

Applied to the physical world, this kind of approach reveals affordances, and can lead to robots learning how to perform physical tasks autonomously, like in [Maestre, 2018]. Applied to the abstract world of knowledge, it can lead to discover new action / knowledge affordances: some knowledge can be used as a parameter of some actions, and this has a predictable effect on the knowledge. This is applicable to the social world too, and may reveal social affordances, that otherwise remain latent in behavior implementations.

5.5 Next Behaviors and Potential Challenges

In the previous sections, we enumerated the many perspectives the pursuit of our research may lead to, sometimes in the long term. In this section, we look at the short term perspectives in practical terms. It consists in behaviors, rules and evaluations methods, that have become affordable thanks to our work.

5.5.1 Correcting Behavior Compositions

Adding interaction repair in the teaching behavior seems to be inevitable to make it really usable. It would consist in being more transparent on what the robot understands during the teaching, and let the users tell the robot when it is wrong. Also, having the robot detect some cases where the input is strange or uncertain, and take the initiative to ask for user correction, could be useful.

The challenge would be to prove that these measures actually improves the success rate of the learning.

5.5.2 Teaching Animation Actions

It seems possible to integrate an adaptation of [Chernova & Thomaz, 2014]'s teaching of movements on Pepper robots, at home. This is a form of Learning from Demonstration (LfD) based on the observation of the position of the joints of the robot, as the human teacher moves the robot's limbs. The robot could therefore learn some primitive behaviors from scratch.

The challenge would reside in the fact that there are several behaviors capable of learning behaviors: one through composition, and this one, through demonstration. It is an opportunity to start working on meta-interaction, since the robot will need to disambiguate which teaching behavior should take over the robot.

5.5.3 Autonomous Recharge

In Pepper@Home, the robot is delivered with a recharge station. It would be useful if the robot could enter and leave the station on command, or autonomously depending on time, battery level, and human presence. This would be an opportunity to discuss event-drive behaviors like "when your battery is lower than 10%, go charge to your station".

This could be a good use case for applying our generalized model on [behaviors](#). The challenge consists in making this novel model work in realistic conditions.

5.5.4 Human Greetings and Identification

This [behavior](#) responds to greetings utterances by greeting back to the user. It tries to track which user is engaged with the robot when it happens, to try to remember that greeting interaction, and not greeting the same user anew. To achieve this the robot must be able to recognize the human.

It requires face recognition to be available on Pepper. We identified that the system proposed by [\[Irfan et al., 2018\]](#) demonstrated an improvement over Pepper’s default face recognition. New users engaging with the robot would be learned as they are greeted back, and are asked their name. Known users are greeted with their name.

The [behavior](#) would produce some knowledge associating encounters, persons and names. Our system should be able to access this knowledge, and understand it according to our shared ontology, described in section 4.2. It would also provide some action factories to leverage this knowledge, and thus multiply the behavioral capabilities of Pepper.

We predict that this [behavior](#) would conflict with the [behavior](#) asking labels for discovered objects. This would require the introduction of new rules about engaging with users: a [behavior](#) which purpose is to acknowledge the engagement is available, it is preferred. The refinement of this rule could be challenging.

5.5.5 Hush Rule

The robot could be temporarily forbidden to perform communicative acts. This use case was encountered regularly in Pepper@Home, and would be easy to implement using our rule system.

The “hush” rule would come along a [behavior](#), capable of discussing with the user whether to turn on and off the rule. When this rule is active, the [actions](#) producing communicative acts are ruled out from the task planning solutions, hence preventing communicative acts to be performed.

The challenge would consist in making the rule system extensible, and to develop a first [behavior](#) to discuss a rule. We can also predict that some exceptions to the rule may be difficult to support. For instance, it is not clear how the robot can ask for a confirmation to stop being hushed, if it is hushed. Also, how to set the priority right, if there were [behaviors](#) producing emergency [behaviors](#), that must absolutely warn the surrounding users.

5.5.6 Behavior Modification Evaluation

We mention in the analysis of the results of previous experiment, in subsection 4.5.4, that the measures we used were perhaps not the best. For instance, they

would not be applicable to evaluate the performance for modifying a [behavior](#) (for correcting it).

The capacity to reason on [behaviors](#) explained in subsection 5.2.3 can also be extended to the benefit of the scientist. By defining a precise ontology of how [behaviors](#) are articulated in compositions, or solicited autonomously by the robot, we would be able to identify the various operations applied when modifying [behaviors](#).

This set of operations can be used to determine quantities of steps to achieve the modification of some procedural knowledge. This is what measures LR_t , used in section 4.5.3. We can therefore estimate a quantity of procedural knowledge a [behavior](#) represents, and distances from a given [behavior](#) to another.

The performance of teaching can then be assessed as the speed or the effectiveness of the system to bridge these distances as it is being taught by users. On the other hand, we can still empirically assess the productivity, and the satisfaction of users when they produce or modify [behaviors](#).

Finally, it can also be used as a basis to assess the complexity of the [behaviors](#) a robot knows, and therefore use it as an intrinsic motivation to drive an autonomous learning of [behaviors](#).

Annex A: NAOqi

In this annex, we present [NAOqi](#) and the [NAOqi](#) framework. The notions involved in [NAOqi](#) should be well understood to understand the technical details of the software produced for this thesis.

[NAOqi](#) is the software driving robots created by [SoftBank Robotics Europe \(SBRE\)](#): NAO, Pepper and Romeo. It is to distinguish from [NAOqi OS](#), the GNU/Linux operating system installed on [SBRE](#) robots, which includes [NAOqi](#).

[NAOqi](#) runs federates a number of services capable of operating [SBRE](#) robots. The services are usually deployed on the robot in the form of [NAOqi](#) packages. Each service provides some [API](#). [NAOqi](#) serves them on the local area network using [libQi](#).

[LibQi](#) is an open source (BSD-3) library providing a base framework to create services that can communicate with one another in-process, between-processes, or on the network. Services can exchange structured data and perform remote procedure calls (RPC) with one another. Service interfaces can be specified formally, so that typed helpers can be generated for client applications.

The [NAOqi](#) framework is the combination of the [libQi](#) framework, the [API](#) provided by the services, typed helpers for [NAOqi](#) services, and some task-specific utilities.

Architecture

[NAOqi](#) relies on a core service, the service directory, listing all the other services it serves. It is analogous to the “master” in ROS, and provides the main endpoint to contact [NAOqi](#) services.

The other services are provided by daemons running on the robot, and are registered to the service directory, but communicate directly with the gateway, the unified endpoint for remote clients. Remote clients connect to this endpoint and can access the registered services without knowing whether they are provided by different processes. This is a micro-service architecture, as found in cloud services.

When a client performs an RPC, the RPC message goes through the gateway, then to the service. When the procedure is done, the service replies to the RPC with a result message to the gateway, and then back to the client. For instance in 2.5.x, to make the robot say something, the client would call the method “say” on the service “ALTextToSpeech”, provided by the daemon “audio”. The call returns some time after the phrase was uttered. See figure 2.

RPC is also used between services, but since nothing forbids several services to be provided by the same daemon, network communication can often be avoided inside [NAOqi](#).

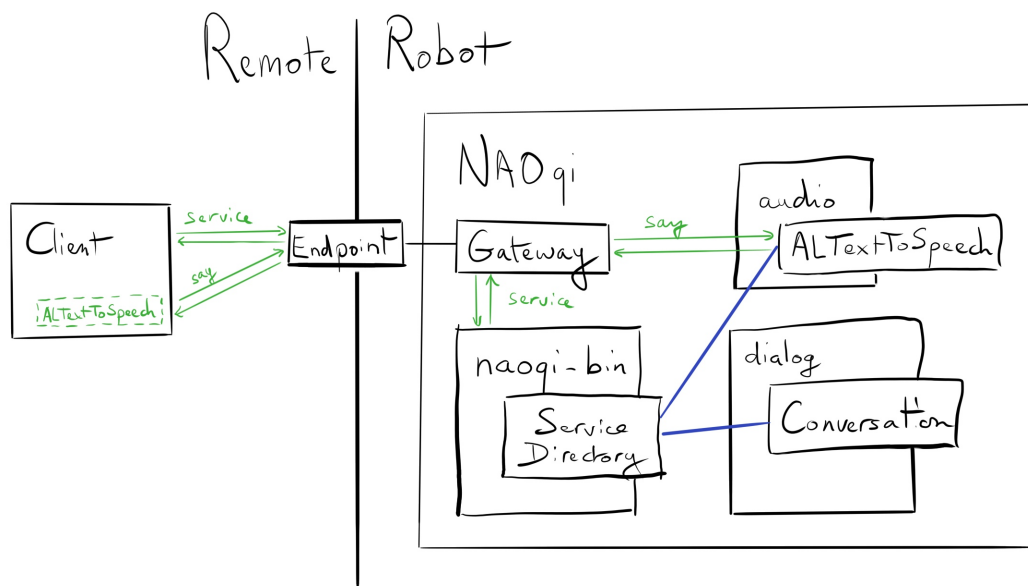


Figure 2: Communication of a RPC from a remote client to a **NAOqi** service. Black boxes right under the **NAOqi** box are daemons. Black boxes emerging from daemons are services. Blue lines represent the registration links. Green arrows represent the RPC calls and their response propagated on the network. Green dotted boxes represent a proxy to a remote object. Here a client gets the service “ALTextToSpeech” provided by the daemon “audio” using a “service” RPC. Then calls “say” on the service.

Objects and Actions

LibQi provides a particular notion of objects that makes it unique. A Qi Object is sort of pointer to an arbitrary object, shared between a service and its clients, through the network. When a client accesses a service, it retrieves a Qi Object pointing to it. That object on the client's side acts like a proxy to the real object: the service hosted in a daemon running on the robot. When a call is attempted on it, it is forwarded to the service using an RPC.

It is in fact the Qi Object mechanism that provides the callable interfaces to the services. Interfaces can include:

- methods to call, with arguments
- signals, providing an observable interface
- properties, exposing data, that may be mutable and observable

But a Qi Object is also a type that can be transferred through RPC, so that a service may return, or accept, other Qi Objects in a RPC.

The notion of Action introduced in software version 2.9.x leverages fully that notion of object. See the getting started page of the [Qi SDK](#)⁸. In 2.9.x, to make a robot say something, the client should rather call the method "makeSay" on the service "Conversation" provided by the daemon "dialog", to retrieve a "Say" action, that is a Qi Object. That object has a method "run" to effectively make the robot utter the desired phrase. See figure 3.

When the method of a Qi Object is called, the RPC message is transmitted to the service that provided the Qi Object, and the execution will take place the service's process.

This mechanism also allows clients to provide Qi Objects to [NAOqi](#) services. This is leveraged in this thesis to extend the system with new [behaviors](#) and new actions.

Sensor Access and Hardware Control

[NAOqi](#) services implement the drivers required to communicate with the hardware of the robot. Sensor data is analyzed by the services, which may translate them into higher-level signals or property, or use it to implement the basic autonomy of the robot.

Remote clients can drive the robot's [behavior](#) by running the actions exposed by the services' [API](#). [NAOqi](#) is the main interface between client applications and the robot's environment.

⁸https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch2_principles/action_getting_started.html

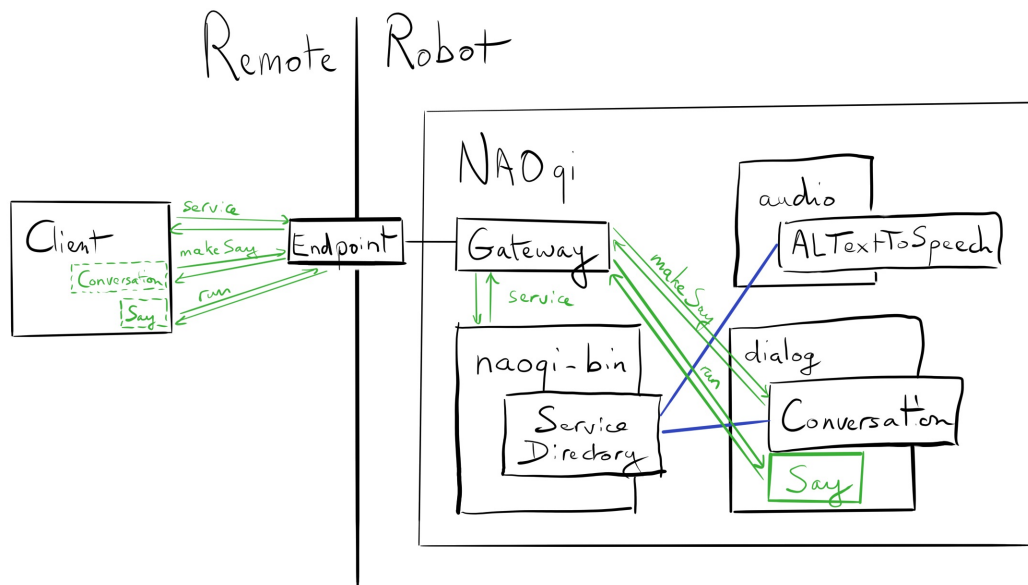


Figure 3: Communication of a RPC from a remote client to a Qi Object provided by NAOqi service. Black boxes right under the NAOqi box are daemons. Black boxes emerging from daemons are services. Blue lines represent the registration links. Green arrows represent the RPC calls and their response propagated on the network. Green boxes represent a non-service Qi Object. Green dotted boxes represent a proxy to a remote object. Here a client gets the service “Conversation” provided by the daemon “dialog” using a “service” RPC. Then calls “makeSay” on the service to get a “Say” object. Then calls “run” on the “Say” object.

Annex B: Review of Cognitive Systems

The software implementation of a piece of HRI is only a part of the software driving the overall **behavior** of a robot, the **cognitive system**. In this section, we make a state of the art on cognitive systems used for teaching **behaviors**, in order to justify the need for a novel one.

A **cognitive system** may constrain the roles of its components, and therefore can prevent certain features to be implemented. It is important to design the **cognitive system** so that it is adequate for the purpose of the robot. In our research, we face particular challenges.

First, the robots should be provided a rich set of interactive **behaviors** from separate applications. Therefore the **cognitive system** driving the overall interaction should be *modular*. Behaviors may know about each other, and may conflict with each other, for example if they produce different reactions to the same situation. Thus there is a need for an *interoperable* interface to make it possible to solve potential conflicts.

Then, the system must accept new interactive **behaviors** created on-the-fly, when users teach them to the robot. That is to say the system’s modularity should also be *dynamic*. The taught **behaviors** may also be interactive, and therefore may conflict with the rest of the **behaviors**, so the conflict management should be *dynamic*.

In this section we show that some of these challenges have been overcome in prior research by existing **cognitive systems**. But there is no existing solution for our set of challenges. First we clarify what a cognitive architecture is in subsection 5.5.6. Then in subsection 5.5.6, we review the **cognitive systems** used in our literature, to look for solutions for our challenges. Finally in subsection 5.5.6 we pinpoint the key design choices and their impact on the robot’s **behavior**. It should justify the solutions we propose in the rest of this thesis or at least give keys to compare them.

What Makes a System “Cognitive”

As introduced in subsection 1.2.1, the term of “cognitive system” is an analogy to the **cognitivist** view of biological agents. Indeed in robotics, since we design the software that drives the **behavior** of the robot – an artificial being with its own **goals**, see subsection 1.2.5 – there is an incentive to adopt this view. Thus the robotic system, and essentially its software parts, is considered a “cognitive system”⁹.

⁹Indeed, cognitive systems are not necessarily robotic.

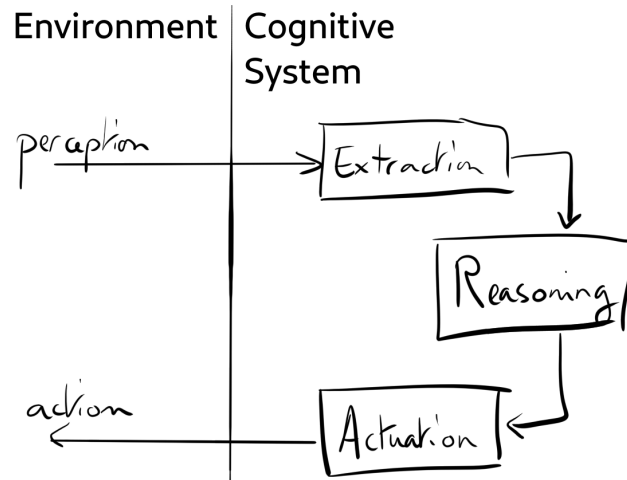


Figure 4: The canonical architecture of a cognitive system.

A cognitive being is able to perceive its environment, process what its senses provide into information, reason on that information with the use of cognitive functions, using past experience and genotypic assets, and act back on the environment. Figure 4 illustrates this. Environment should be taken in a very broad sense, that can include the physical body of the being.

In practice, the extraction and actuation components are provided by frameworks or middleware for robotics, like [NAOqi](#) (found on [Pepper](#) and [NAO](#)), [ROS](#) (found on [PR-2](#)) or [YARP](#) (found on [iCub](#)). They translate the perception and the action into software objects that can be exchanged through their [Application Programming Interface \(API\)](#).

Since using [Pepper](#) is part of our objectives (see section 1.4), we will use the [NAOqi](#) framework, that we present in annex 5.5.6.

Frameworks are somewhat equivalent to one another, so what really differentiate cognitive systems is rather the roles of the software components involved in the reasoning of the system, and the organization of these components. This is what we will focus on in next subsection.

Review of Cognitive Systems

In this review, we display the diversity of cognitive systems encountered in our literature. We group them according to the similarity of their features, and highlight their potential modularity and interoperability.

For the first attempt at teaching behaviors to robots using spoken language, [\[Lauria et al., 2001\]](#) introduces the [Instruction-Based Learning \(IBL\)](#) architecture. Its engine is directly driven by the speech recognizer. As a reaction, it may call the [TTS](#) or order the execution of behavioral processes: either to learn some task plan, or to execute an existing one. A knowledge base shares the known task plans

between the dialogue manager (in charge of interaction) and the robot manager (in charge of the execution of plans). [Weitzenfeld & Dominey, 2009] present a similar system, but with a question system to retrieve information from the robot’s knowledge of the world (excluding the procedural knowledge).

But not all publications share details of the cognitive system they use. Among them, [Nicolescu & Mataric, 2003, Rybski et al., 2007, Rybski et al., 2008, Cakmak & Takayama, 2014] exhibit no specific feature suggesting major differences with [Lauria et al., 2001]. They probably had the dialogue system driving the execution of tasks in a straightforward manner: taught **behaviors** are saved in the database, and associated to pre-defined dialogue inputs. However the **behavior models**, the world and knowledge representations differ. These differences are explored in 2.4.

[Lallée et al., 2010] do not detail their cognitive architecture neither. However [Lallée et al., 2012] continue their work, and present the CHRIS architecture. It is made of modules of “scene perception”, “motor command”(.*)“knowledge base” and “supervision & planning”. Scene perception contributes to the knowledge, but also directly to the motor command, to provide a feedback loop. This is an interesting refinement, that could support, within the same system, an ability to learn high-level task plans, as well as low-level skills.

The supervision & planning module communicates with the knowledge base, reasons on the events, and in return requests motor commands or produces speech. Thanks to a proper specification of the knowledge, including the procedural one, they manage the exchange the knowledge *via* the Internet, and run the same **behaviors** on different robots. However the system remains driven by the dialogue, and the robot is not taught **behaviors** interfering with its overall **behavior**.

[Salvi et al., 2012] present an architecture centered on a **Bayesian Network** (BN). The reasoning component is therefore a sort of black-box. [Grizou et al., 2013, Grizou, 2014, Grizou et al., 2014] use a similar black-box approach, but with a **Markov Decision Process** (MDP). These architectures support a connectionist form of knowledge, in opposition to the symbolic forms of knowledge we encounter in the other publications.

There are other cases of black-box reasoners in the literature on teaching **behaviors** using non-spoken natural language: [Arie et al., 2010, Yamada et al., 2016, Antunes et al., 2018] use neural networks, [Saponaro et al., 2017] put together an **HMM** after a **BN**. The system’s architecture is made simpler and more homogeneous with this black-box approach. But in the current state of the art, it constrains much the richness of the interaction, and is difficult to extend.

But the most common practice is to have a dialogue system designed with a symbolic approach. The dialogue system activates the learning and executing processes. The executing process is setup to execute one **behavior** at a time, according to instructions.

[Gemignani et al., 2015] has a dialogue interface that would produce plans that are then translated to **Petri Network Plan** (PNP), that are then interpreted by the system. The system binds events and data to a state of the world that can trigger transitions, and nodes to **actions** to perform, continuously.

These algorithms on the other hand can potentially be based on a less symbolic approach. For instance [Forbes et al., 2015] uses a state-based dialogue system to drive the robot into the learning phases, provided by a PbD system. But the PbD system relies on state goals, and produces probabilistic estimators to find the actions required to reach them. The overall logic is similar to what is used in [Mohseni-Kabir et al., 2017, Mohseni-Kabir et al., 2019], but not enough details were presented for a more thorough comparison.

[Petit & Demiris, 2016] is driven by a layer of 3 YARP modules, involved gradually at each 3 stages of their experiment. Overall, the auto-biographical information is gathered continually, and stored in a database. Modules use this database for the learning, the speech recognition for the verbal interaction, and send commands to the motors for the exploration. The switch between stages appear procedurally managed. If it is made in an interoperable manner, this stage-switching ability may be a step towards supporting interactive behaviors provided by separate applications.

[Scheutz et al., 2017, Scheutz et al., 2018] used an architecture called DIARC, and presented it in details. Like [Forbes et al., 2015], the dialogue system may feed the execution system with new goals. But according to [Schermerhorn et al., 2006], competing goals are supported. The most important one can be selected autonomously, before the right actions are found to fulfill it.

Other architectures are worth mentioning from the literature on teaching behaviors using the natural language. For instance the SOAR architecture [Laird et al., 1987, Laird, 2012]. This architecture accepts several “problem space computational model (PSCM)” operators [Newell, 1990, Yost, 1993]: given states and operators to alter states, a goal can be reached using various defined paths. When the situation is undecidable, sub-problems with sub-goals are invoked, hence producing a hierarchical behavior. Embodied in a robot, states can be evaluated from the perception of the environment, and operators can act on the environment. [Huffman & Laird, 1995] implements a variant called INSTRUCTO-SOAR. It juxtaposes a dialogue system with SOAR, and can produce new PSCM operators from instructions. That is to say that what is taught to the robot integrates the rest of the behaviors. It is a highly-modular architecture.

Later [Mohan et al., 2012] reuses SOAR for its visual-spatial capabilities and adds a decision-making engine on top of it, with a parallel system for interacting with the instructor. It is very probable that [Mohan & Laird, 2014] uses the same system. However we did not retain these works in our literature, because the interaction with the system is always done using the written form of the language.

[Nyga, 2017] was also excluded for this reason. However the cognitive system they use is also worth mentioning: CRAM [Beetz et al., 2010]. This architecture revolves around a knowledge base and a reasoner. When the system has a goal set (using natural language instructions), the reasoner constructs a plan to reach it. Every step of the plan is goal-oriented, and relies on bindings between the sensors and a symbolic representation of the environment, using “computable predicates”. A computable predicate associates an arbitrary function to a predicate to evaluate

in the knowledge base, driven by an OWL ontology defined by KnowRob [Tenorth & Beetz, 2013].

The use of an ontology defined properly makes a significant difference. It enables a greater interoperability, and also a greater reusability of the knowledge. As a result, they are able to share this knowledge across robots [Beetz et al., 2015], including simulated ones.

With this review we displayed the diversity of cognitive systems found in our literature, and some extraneous related works. It appears that no cognitive system dominates the literature. The observed diversity of approaches could be explained by the diversity of requirements for each study, such as the learning technique, the interaction scenario, or the reuse of existing solutions. Some requirements lead to specific design choices, that we will highlight in next subsection.

Key Design Choices for Cognitive Systems

Designing (or choosing) the right cognitive system is a valuable know-how for research in robotics. In this subsection we detail what we expect from the cognitive system, and the consequences it may have.

The most prominent one is the choice between [symbolic](#) and [connectionist knowledge](#) [Harnad, 1990]. Using a [connectionist knowledge](#) simplifies the architecture, which does not really need to provide functions like planning, explicit data storage, or a pool of [actions](#). In return, it is not as flexible as the [symbolic knowledge](#), which enforces an abstraction to make symbols more reusable.

Planning is also a matter of choice. Several systems merely execute [actions](#) procedurally as the interactive program goes on. This is straightforward, and even helps jumping from one form of interaction to another. But it is limiting the reusability of the [actions](#), because the interactive program must cover all possible cases. The introduction of [goals](#) does not directly solve this, but already bring interoperability to the [actions](#). Generic planning solutions, on the other hand, require situational knowledge to be expressed symbolically.

Expressing the world in terms of symbols appeared frequently in this review. It enabled the robots to learn very directly from the provided instructions, that can be represented using semantic symbols. Symbols are usually more expressive for the humans designing the systems. For this reason it is convenient to use symbols to represent the [goal](#) of a robot. It also supports the need for [HRI](#) to study the satisfaction of the robot's [goals](#).

The use of symbols does not prevent some components to be written using a [connectionist](#) approach. In [Beetz et al., 2010], the “computable predicates” are bridges to the sensible world, for which connectionist approaches work best. These techniques announce promising hybrid architectures [Harmelen et al., 2019].

In terms of modularity, we found that many “modules” were so specific to their cognitive system that we cannot *a priori* trust that they could be replaced by a component from another system. The ways modules communicate with each other

betray inter-dependencies between them. Indeed, no study in [IRL](#) actually tried to test the effective modularity of a system.

In previous subsection ([5.5.6](#)), we highlighted that it is common practice to separate the dialogue system from the rest. Spoken responses are executed differently from other [actions](#) or processes. In theory, uttering something is an action like any other. It is a physical action, that indeed have observable effects in the environment (even more through [speech acts](#), see section [2.3](#)).

So the separation seems unjustified. In practice, it may be easier to conceptualize and implement an interactive system that way: spoken interaction is on one side, and intelligent [behaviors](#) or learning processes are on the other side. But for a social robot, the [behaviors](#) to learn are part of the interactive system, so the separation is in fact counter-productive: the dialogue system itself is made of taught [behaviors](#).

Moreover, this design choice complicates the [cognitive systems](#). It adds up to the list of components, and adds constraints to the planning system. And when the system would be learning social [actions](#) that include speech, there would be two learning components to maintain and coordinate.

Conclusions on Cognitive Systems

Our review of cognitive systems showed how their architecture and the knowledge models depend on each other. It should be helpful to understand the review of [behavior models](#) found in next section.

We also concluded that the current state of the art neither achieves the kind of modularity we expect for a robot that should support multiple applications. Nor the current dialogue management systems are designed to support the teaching of social [behaviors](#).

Bibliography

- [Alili et al., 2009] Alili, S., Alami, R., & Montreuil, V. (2009). A task planner for an autonomous social robot. In *Distributed Autonomous Robotic Systems 8* (pp. 335–344). Springer.
- [Allen & Core, 1997] Allen, J. & Core, M. (1997). Draft of DAMSL: Dialog Act Markup in Several Layers. (pp.32).
- [Amershi et al., 2014] Amershi, S., Cakmak, M., Knox, W. B., & Kulesza, T. (2014). Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4), 105–120.
- [Antunes et al., 2018] Antunes, A., Laflaquière, A., & Cangelosi, A. (2018). Solving Bidirectional Tasks using MTRNN. In *8th Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics* (pp.6). Tokyo, Japan.
- [Anzalone et al., 2015] Anzalone, S. M., Boucenna, S., Ivaldi, S., & Chetouani, M. (2015). Evaluating the Engagement with Social Robots. *International Journal of Social Robotics*, 7(4), 465–478, doi:10.1007/s12369-015-0298-7.
- [Arie et al., 2010] Arie, H., Endo, T., Jeong, S., Lee, M., Sugano, S., & Tani, J. (2010). Integrative learning between language and action: A neuro-robotics experiment. *Artificial Neural Networks - ICANN 2010*, (pp. 256–265).
- [Austin, 1962] Austin, J. L. (1962). *How to do things with words*. The William James Lectures delivered at Harvard University in 1955. Oxford, Great Britain: Oxford University Press.
- [Baier & McIlraith, 2008] Baier, J. A. & McIlraith, S. A. (2008). Planning with Preferences. *AI Magazine*, 29(4), 25, doi:10.1609/aimag.v29i4.2204.
- [Baker et al., 1998] Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1* (pp. 86–90).: Association for Computational Linguistics.
- [Baranes & Oudeyer, 2010] Baranes, A. & Oudeyer, P.-Y. (2010). Maturationally-constrained competence-based intrinsically motivated learning. In *Development and Learning (ICDL), 2010 IEEE 9th International Conference on* (pp. 197–203).: IEEE.
- [Beetz et al., 2010] Beetz, M., Mösenlechner, L., & Tenorth, M. (2010). CRAM - A Cognitive Robot Abstract Machine for everyday manipulation in human environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1012–1017).: IEEE.

- [Beetz et al., 2015] Beetz, M., Tenorth, M., & Winkler, J. (2015). Open-EASE. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1983–1990).: IEEE.
- [Bernard-Opitz, 2017] Bernard-Opitz, V. (2017). *The Cartoon and Script Curriculum for Teaching Social Behavior and Communication*. Lenexa, Kansas: AAPC Pub.
- [Boom, 2015] Boom, D. V. (2015). Pepper the humanoid robot debuts in France. *CNET*.
- [Borgo et al., 2016] Borgo, S., Cesta, A., Orlandini, A., & Umbrico, A. (2016). A planning-based architecture for a reconfigurable manufacturing system. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*.
- [Breazeal, 2004] Breazeal, C. (2004). Social interactions in HRI: the robot view. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(2), 181–186.
- [Broekens & Chetouani, 2019] Broekens, J. & Chetouani, M. (2019). Towards Transparent Robot Learning through TDRL-based Emotional Expressions. *IEEE Transactions on Affective Computing*, (pp. 1–1)., doi:10.1109/TAFFC.2019.2893348.
- [Brooke, 1986] Brooke, J. (1986). *SUS: a "quick and dirty" usability scale*. Usability Evaluation in Industry. London, taylor and francis edition.
- [Bruner, 1983] Bruner, J. S. (1983). *Child's talk: learning to use language*. New York: W.W. Norton, 1 edition.
- [Cakmak & Takayama, 2014] Cakmak, M. & Takayama, L. (2014). Teaching people how to teach robots: The effect of instructional materials and dialog design. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction* (pp. 431–438).: ACM.
- [Chatila et al., 2017] Chatila, R., Firth-Butterfield, K., Havens, J. C., & Karachalios, K. (2017). The IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems [Standards]. *IEEE Robotics & Automation Magazine*, 24(1), 110–110.
- [Chatila et al., 2018] Chatila, R., et al. (2018). Toward Self-Aware Robots. *Frontiers in Robotics and AI*, 5, doi:10.3389/frobt.2018.00088.
- [Chernova & Thomaz, 2014] Chernova, S. & Thomaz, A. L. (2014). *Robot learning from human teachers*. San Rafael, California: Morgan & Claypool.
- [Courtois, 1990] Courtois, B. (1990). Un système de dictionnaires électroniques pour les mots simples du français. *Langue française*, 87(1), 11–22, doi:10.3406/lfr.1990.6323.

- [Crangle & Suppes, 1994] Crangle, C. & Suppes, P. (1994). *Language and learning for robots*. Number no. 41 in CSLI lecture notes. Stanford, California: Center for the Study of Language and Information.
- [Crystal, 2008] Crystal, D. (2008). *A dictionary of linguistics and phonetics*. The language library. Malden, Massachusetts: Blackwell Pub, 6th ed edition. OCLC: ocn187300284.
- [De Marneffe et al., 2014] De Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., & Manning, C. D. (2014). Universal Stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14 (pp. 4585–92).
- [De Marneffe & Manning, 2008] De Marneffe, M.-C. & Manning, C. D. (2008). The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation* (pp. 1–8).: Association for Computational Linguistics.
- [del Duchetto et al., 2019] del Duchetto, F., Baxter, P. E., & Hanheide, M. (2019). Lindsey the Tour Guide Robot - Usage Patterns in a Museum Long-Term Deployment. (pp.8).
- [Dorr et al., 1999] Dorr, B. J., Jordan, P. W., & Benoit, J. W. (1999). A survey of current paradigms in machine translation. In *Advances in computers*, volume 49 (pp. 1–68). Elsevier.
- [Durrant-Whyte & Bailey, 2006] Durrant-Whyte, H. & Bailey, T. (2006). Simultaneous localization and mapping: part I. *IEEE Robotics Automation Magazine*, 13(2), 99–110, doi:10.1109/MRA.2006.1638022.
- [Durrett et al., 2016] Durrett, G., Berg-Kirkpatrick, T., & Klein, D. (2016). Learning-Based Single-Document Summarization with Compression and Anaphoricity Constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1: Long Papers (pp. 1998–2008). Berlin, Germany.
- [Dyer, 1997] Dyer, A. (1997). IBM ViaVoice. *Personal Computer World*, 20(12), 92.
- [Fikes & Nilsson, 1971] Fikes, R. E. & Nilsson, N. J. (1971). Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4), 189–208, doi:10.1016/0004-3702(71)90010-5.
- [Fillmore, 1985] Fillmore, C. J. (1985). Frames and the semantics of understanding. *Quaderni di Semantica*, 6(2), 222–254.
- [Forbes et al., 2015] Forbes, M., Rao, R. P., Zettlemoyer, L., & Cakmak, M. (2015). Robot programming by demonstration with situated spatial language understanding. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2014–2020).: IEEE.

- [Fox & Long, 2003] Fox, M. & Long, D. (2003). PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research*.
- [Frasca et al., 2018] Frasca, T., Oosterveld, B., Krause, E., & Scheutz, M. (2018). One-Shot Interaction Learning from Natural Language Instruction and Demonstration. *Advances in Cognitive Systems*, 6, 1–18.
- [Fridin & Belokopytov, 2014] Fridin, M. & Belokopytov, M. (2014). Embodied Robot versus Virtual Agent: Involvement of Preschool Children in Motor Task Performance. *International Journal of Human–Computer Interaction*, 30(6), 459–469, doi:10.1080/10447318.2014.888500.
- [G20, 2019] G20 (2019). G20 AI Principles. In *G20 Tsukuba*, Ibaraki, Japan: Ministry of Foreign Affairs of Japan.
- [Gangemi et al., 2002] Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., & Schneider, L. (2002). Sweetening ontologies with DOLCE. *Knowledge engineering and knowledge management: Ontologies and the semantic Web*, (pp. 223–233).
- [Gemignani et al., 2015] Gemignani, G., Bastianelli, E., & Nardi, D. (2015). Teaching robots parametrized executable plans through spoken interaction. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (pp. 851–859).: International Foundation for Autonomous Agents and Multiagent Systems.
- [Gerevini & Long, 2005] Gerevini, A. & Long, D. (2005). Plan constraints and preferences in PDDL3. *The Language of the Fifth International Planning Competition. Tech. Rep. Technical Report, Department of Electronics for Automation, University of Brescia, Italy*, 75.
- [Gibson, 1977] Gibson, J. J. (1977). The Theory of Affordances. *Perceiving, acting, and knowing: Toward an ecological psychology*, (pp. 67–82).
- [Gorovoy et al., 2010] Gorovoy, K., Tung, J., & Poupart, P. (2010). Automatic speech feature extraction for cognitive load classification. *CMBES Proceedings*, 33.
- [Gottstein, 2017] Gottstein, R. (2017). *Système délibératif d’un robot autonome : planification probabiliste hiérarchique basée sur des motivations et prise en compte des ressources*. PhD thesis, Université Pierre et Marie Curie.
- [Gray, 2016] Gray, R. (2016). Pepper the ‘emotional’ humanoid becomes first robot to attend SCHOOL. *Mail Online* (2016). <https://www.dailymail.co.uk/sciencetech/article-3540307/Pepper-grows-Emotional-humanoid-robot-enroll-SCHOOL-Japan.html> [Retrieved 2019-06-22].

- [Greenfield, 1984] Greenfield, P. M. (1984). Theory of the teacher in learning activities of everyday life. In B. Rogoff & J. Lave (Eds.), *Theory of the teacher in learning activities of everyday life*. Cambridge, Massachusetts: Harvard University Press.
- [Grishman & Sundheim, 1996] Grishman, R. & Sundheim, B. (1996). Design of the MUC-6 evaluation. In *TIPSTER '96 Proceedings*.
- [Grizou, 2014] Grizou, J. (2014). *Learning from Unlabeled Interaction Frames*. PhD thesis, Université de Bordeaux.
- [Grizou et al., 2013] Grizou, J., Lopes, M., & Oudeyer, P.-Y. (2013). Robot learning simultaneously a task and how to interpret human instructions. In *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on* (pp. 1–8).: IEEE.
- [Grizou et al., 2014] Grizou, J., Lopes, M., & Oudeyer, P.-Y. (2014). Robot Learning from Unlabelled Teaching Signals. In *HRI 2014 Pioneers Workshop*. 00000.
- [Gruber, 2009] Gruber, T. (2009). Ontology. *Encyclopedia of database systems* (2009). Springer-Verlag: New York.
- [Harmelen et al., 2019] Harmelen, F. v., Department of Computer Science, Vrije Universiteit Amsterdam, Netherlands, Teije, A. t., & Department of Computer Science, Vrije Universiteit Amsterdam, Netherlands (2019). A Boxology of Design Patterns for Hybrid Learning and Reasoning Systems. *Journal of Web Engineering*, 18(1), 97–124, doi:10.13052/jwe1540-9589.18133.
- [Harnad, 1990] Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3), 335–346, doi:10.1016/0167-2789(90)90087-6. 03492.
- [Hart & Staveland, 1988] Hart, S. G. & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52 (pp. 139–183). Elsevier.
- [Houssin, 2014] Houssin, D. (2014). QiChat (2014). SoftBank Robotics Europe: Paris. http://doc.aldebaran.com/2-5/naoqi/interaction/dialog/aldialog_syntax_toc.html.
- [Huffman & Laird, 1995] Huffman, S. B. & Laird, J. E. (1995). Flexibly instructable agents. *Journal of Artificial Intelligence Research*, 3, 271–324.
- [Irfan et al., 2018] Irfan, B., Lyubova, N., Ortiz, M. G., & Belpaeme, T. (2018). Multi-modal Open-Set Person Identification in HRI. (pp.5).
- [Jou & Harris, 1992] Jou, J. & Harris, R. J. (1992). The effect of divided attention on speech production. *Bulletin of the Psychonomic Society*, 30(4), 301–304, doi:10.3758/BF03330471.

- [Keneni et al., 2019] Keneni, B. M., Kaur, D., Al Bataineh, A., Devabhaktuni, V. K., Javaid, A. Y., Zaiantz, J. D., & Marinier, R. P. (2019). Evolving Rule-Based Explainable Artificial Intelligence for Unmanned Aerial Vehicles. *IEEE Access*, 7, 17001–17016, doi:10.1109/ACCESS.2019.2893141.
- [Khawaja, 2010] Khawaja, M. A. (2010). *Cognitive load measurement using speech and linguistic features*. PhD thesis, University of New South Wales, Sydney.
- [Khosla & Chu, 2013] Khosla, R. & Chu, M.-T. (2013). Embodying Care in Matilda: An Affective Communication Robot for Emotional Wellbeing of Older People in Australian Residential Care Facilities. *ACM Trans. Manage. Inf. Syst.*, 4(4), 18:1–18:33, doi:10.1145/2544104.
- [Klarsfeld & Mc Carthy Hammani, 1991] Klarsfeld, G. & Mc Carthy Hammani, M. (1991). *Dictionnaire électronique du LADL pour les mots simples de l'anglais (DELASa)*. Rapport Technique du LADL, Université Paris 7, Paris.
- [Kong et al., 2017] Kong, L., Alberti, C., Andor, D., Bogatyy, I., & Weiss, D. (2017). DRAGNN: A Transition-based Framework for Dynamically Connected Neural Networks. *arXiv:1703.04474 [cs]*. arXiv: 1703.04474.
- [Kuhl et al., 1992] Kuhl, P. K., Williams, K. A., Lacerda, F., Stevens, K. N., & Lindblom, B. (1992). Linguistic experience alters phonetic perception in infants by 6 months of age. *Science*, 255(5044), 606–608.
- [Laird, 2012] Laird, J. E. (2012). *The Soar cognitive architecture*. Cambridge, Massachusetts: MIT Press.
- [Laird et al., 2017] Laird, J. E., et al. (2017). Interactive Task Learning. *IEEE Intelligent Systems*, 32(4), 6–21, doi:10.1109/MIS.2017.3121552.
- [Laird et al., 1987] Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33(1), 1–64, doi:10.1016/0004-3702(87)90050-6.
- [Lallée et al., 2012] Lallée, S., et al. (2012). Towards a platform-independent co-operative human robot interaction system: III an architecture for learning and executing actions and shared plans. *IEEE Transactions on Autonomous Mental Development*, 4(3), 239–253, doi:10.1109/IROS.2013.6696343.
- [Lallée et al., 2010] Lallée, S., et al. (2010). Human-robot cooperation based on interaction learning. In *From motor learning to interaction learning in robots* (pp. 491–536). Springer.
- [Lamere et al., 2003] Lamere, P., Kwok, P., Walker, W., Gouvea, E., Singh, R., Raj, B., & Wolf, P. (2003). Design of the CMU Sphinx-4 decoder. In *Eighth European Conference on Speech Communication and Technology*.

- [Lauria et al., 2001] Lauria, S., Bugmann, G., Kyriacou, T., Bos, J., & Klein, E. (2001). Training personal robots using natural language instruction. *IEEE Intelligent systems*, 16(5), 38–45.
- [Lemaignan et al., 2017] Lemaignan, S., Warnier, M., Sisbot, E. A., Clodic, A., & Alami, R. (2017). Artificial cognition for social human-robot interaction: An implementation. *Artificial Intelligence*, 247, 45–69, doi:10.1016/j.artint.2016.07.002.
- [Leutenegger et al., 2011] Leutenegger, S., Chli, M., & Siegwart, R. (2011). BRISK: Binary robust invariant scalable keypoints. In *2011 IEEE international conference on computer vision (ICCV)* (pp. 2548–2555).: Ieee.
- [Löffler et al., 2019] Löffler, D., Hurtienne, J., & Nord, I. (2019). Blessing Robot BlessU2: A Discursive Design Study to Understand the Implications of Social Robots in Religious Contexts. *International Journal of Social Robotics*, doi:10.1007/s12369-019-00558-3.
- [Maestre, 2018] Maestre, C. (2018). *Autonomous and Online Generation of Skills Inferring Actions Adapted to Low-Level and High-Level Contextual States*. PhD thesis, Sorbonne Université, Paris.
- [Manning et al., 2014] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *ACL (System Demonstrations)* (pp. 55–60).
- [Mohan et al., 2013] Mohan, S., Kirk, J., & Laird, J. (2013). A Computational Model for Situated Task Learning with Interactive Instruction. In *International Conference on Cognitive Modeling* Ottawa, Canada.
- [Mohan & Laird, 2014] Mohan, S. & Laird, J. E. (2014). Learning Goal-Oriented Hierarchical Tasks from Situated Interactive Instruction. In *AAAI* (pp. 387–394).
- [Mohan et al., 2012] Mohan, S., Mininger, A., Kirk, J., & Laird, J. E. (2012). Learning grounded language through situated interactive instruction. In *2012 AAAI Fall Symposium Series*.
- [Mohseni-Kabir et al., 2017] Mohseni-Kabir, A., et al. (2017). SLHAP: Simultaneous Learning of Hierarchy and Primitives. In *Companion Proc. ACM/IEEE Int. Conf. on Human-Robot Interaction (HRI)* Vienna, Austria: ACM.
- [Mohseni-Kabir et al., 2019] Mohseni-Kabir, A., et al. (2019). Simultaneous learning of hierarchy and primitives for complex robot tasks. *Autonomous Robots*, 43(4), 859–874, doi:10.1007/s10514-018-9749-y.
- [Moortgat, 2011] Moortgat, M. (2011). Categorical Type Logics. In J. van Benthem & A. ter Meulen (Eds.), *Handbook of Logic and Language (Second Edition)* (pp. 95–179). London: Elsevier.

- [Najar, 2017] Najar, A. (2017). *Shaping robot behaviour with unlabeled human instructions*. PhD Thesis, Université Pierre et Marie Curie, Paris.
- [Newell, 1990] Newell, A. (1990). *Unified theories of cognition*. Number 1987 in The William James lectures. Cambridge, Massachusetts: Harvard University Press.
- [Nicolescu & Mataric, 2003] Nicolescu, M. N. & Mataric, M. J. (2003). Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems* (pp. 241–248).: ACM.
- [Nyga, 2017] Nyga, D. (2017). *Interpretation of Natural-language Robot Instructions*. PhD Thesis, University of Bremen - Institute of Artificial Intelligence, Bremen.
- [OECD, 2019] OECD (2019). Recommendation of the Council on Artificial Intelligence (2019). <https://legalinstruments.oecd.org/en/instruments/OECD-LEGAL-0449> [Retrieved 2019-09-14].
- [Olson, 2018] Olson, P. (2018). Softbank’s Robotics Business Prepares To Scale Up. *Forbes* (2018). <https://www.forbes.com/sites/parmyolson/2018/05/30/softbank-robotics-business-pepper-boston-dynamics/> [Retrieved 2019-09-21].
- [Oudeyer, 2013] Oudeyer, P.-Y. (2013). *Aux sources de la parole: auto-organisation et évolution*. Paris: Odile Jacob.
- [Paléologue et al., 2018] Paléologue, V., Martin, J., Pandey, A. K., & Chetouani, M. (2018). Semantic-based interaction for teaching robot behavior compositions using spoken language. In *International Conference on Social Robotics* (pp. 421–430).: Springer.
- [Paléologue et al., 2017] Paléologue, V., Martin, J., Pandey, A. K., Coninx, A., & Chetouani, M. (2017). Semantic-based interaction for teaching robot behavior compositions. In *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)* Lisbon, Portugal.
- [Peh & Foster, 2017] Peh, C. & Foster, M. (2017). In Japan, robot-for-hire programmed to perform Buddhist funeral rites. *Reuters*.
- [Peltason & Wrede, 2010] Peltason, J. & Wrede, B. (2010). Pamini: A framework for assembling mixed-initiative human-robot interaction from generic interaction patterns. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (pp. 229–232).: Association for Computational Linguistics. 00017.
- [Peltason & Wrede, 2012] Peltason, J. & Wrede, B. (2012). Structuring Human-Robot-Interaction in Tutoring Scenarios. In *Towards Service Robots for Everyday Environments* (pp. 471–482). Springer.

- [Perera & Nand, 2017] Perera, R. & Nand, P. (2017). Recent Advances in Natural Language Generation: a Survey and Classification of the Empirical Literature. *Computing and Informatics*, 36(1), 1–32.
- [Perera et al., 2017] Perera, R., Nand, P., & Naeem, A. (2017). Utilizing typed dependency subtree patterns for answer sentence generation in question answering systems. *Progress in Artificial Intelligence*, 6(2), 105–119, doi:10.1007/s13748-017-0113-9.
- [Perera et al., 2015] Perera, V., et al. (2015). Learning task knowledge from dialog and web access. *Robotics*, 4(2), 223–252.
- [Petit & Demiris, 2016] Petit, M. & Demiris, Y. (2016). Hierarchical action learning by instruction through interactive grounding of body parts and proto-actions. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on* (pp. 3375–3382).: IEEE.
- [Petrick & Foster, 2016] Petrick, R. P. & Foster, M. E. (2016). Using General-Purpose Planning for Action Selection in Human-Robot Interaction. In *2016 AAAI Fall Symposium Series*.
- [Petrov et al., 2012] Petrov, S., Das, D., & McDonald, R. (2012). A Universal Part-of-Speech Tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)* (pp. 2089–2096). Istanbul, Turkey: European Language Resources Association (ELRA).
- [Pérez-Osorio & Wykowska, 2019] Pérez-Osorio, J. & Wykowska, A. (2019). Adopting the intentional stance toward natural and artificial agents. doi:10.31234/osf.io/t7dwg.
- [RDF Working Group, 2014] RDF Working Group (2014). RDF 1.1 Concepts and Abstract Syntax (2014). World Wide Web Consortium. <https://www.w3.org/TR/rdf11-concepts/> [Retrieved 2019-09-29].
- [Reddy, 2019] Reddy, S. (2019). Robots Take a Turn Leading Autism Therapy in Schools. *WSJ* (2019). <https://www.wsj.com/articles/robots-take-a-turn-leading-autism-therapy-in-schools-11558344600> [Retrieved 2019-06-22].
- [Reese, 2016] Reese, H. (2016). How Pepper the robot will become newest crew member of Costa Cruise Line. *TechRepublic*.
- [Reyes, 2016] Reyes, A. R. H. (2016). Pepper: Living with a humanoid robot – 2.O Magazine (2016). <https://2ndopinion.ph/pepper-living-with-a-humanoid-robot/> [Retrieved 2019-06-09].
- [Richardson, 2017] Richardson, H. (2017). Robots 'could solve social care crisis'. *BBC News*.

- [Rohlfing et al., 2016] Rohlfing, K. J., Wrede, B., Vollmer, A.-L., & Oudeyer, P.-Y. (2016). An Alternative to Mapping a Word onto a Concept in Language Acquisition: Pragmatic Frames. *Frontiers in Psychology*, 7, doi:10.3389/fpsyg.2016.00470.
- [Rudin, 2019] Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5), 206–215, doi:10.1038/s42256-019-0048-x.
- [Rybski et al., 2008] Rybski, P. E., Stolarz, J., Yoon, K., & Veloso, M. (2008). Using dialog and human observations to dictate tasks to a learning robot assistant. *Intelligent Service Robotics*, 1(2), 159–167.
- [Rybski et al., 2007] Rybski, P. E., Yoon, K., Stolarz, J., & Veloso, M. M. (2007). Interactive robot task training through dialog and demonstration. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction* (pp. 49–56).: ACM.
- [Salvi et al., 2012] Salvi, G., Montesano, L., Bernardino, A., & Santos-Victor, J. (2012). Language bootstrapping: Learning word meanings from perception–action association. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3), 660–671.
- [Santorini, 1990] Santorini, B. (1990). Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision) (1990).
- [Saponaro et al., 2017] Saponaro, G., Jamone, L., Bernardino, A., & Salvi, G. (2017). Interactive Robot Learning of Gestures, Language and Affordances. In *Grounding Language Understanding GLU2017 August 25, 2017, KTH Royal Institute of Technology, Stockholm, Sweden*.
- [Satariano et al., 2018] Satariano, A., Peltier, E., & Kostyukov, D. (2018). Meet Zora, the Robot Caregiver. *The New York Times*.
- [Scassellati, 2002] Scassellati, B. (2002). Theory of mind for a humanoid robot. *Autonomous Robots*, 12(1), 13–24. 00275.
- [Schermerhorn et al., 2006] Schermerhorn, P., Kramer, J., Brick, T., Anderson, D., Dingler, A., & Scheutz, M. (2006). DIARC: A Testbed for Natural Human-Robot Interaction. *AAAI*, (pp. 1972–1973).
- [Scheutz et al., 2017] Scheutz, M., Krause, E. A., Oosterveld, B., Frasca, T. M., & Platt, R. (2017). Spoken Instruction-Based One-Shot Object and Action Learning in a Cognitive Robotic Architecture. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (pp. 1378–1386).: International Foundation for Autonomous Agents and Multiagent Systems.

- [Scheutz et al., 2018] Scheutz, M., Krause, E. A., Oosterveld, B., Frasca, T. M., & Platt, R. (2018). Recursive Spoken Instruction-Based One-Shot Object and Action Learning. In *IJCAI* (pp. 5354–5358).
- [Searle, 1965] Searle, J. R. (1965). What is a speech act? *Perspectives in the philosophy of language: a concise anthology*, 2000, 253–268.
- [She et al., 2014] She, L., Cheng, Y., Chai, J. Y., Jia, Y., Yang, S., & Xi, N. (2014). Teaching robots new actions through natural language instructions. In *Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on* (pp. 868–873).: IEEE.
- [She et al., 2015] She, L., Jia, Y., Xi, N., & Chai, J. Y. (2015). Exception Handling for Natural Language Control of Robots. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts - HRI'15 Extended Abstracts* (pp. 139–140). Portland, Oregon, USA: ACM Press.
- [Simon, 1977] Simon, H. A. (1977). Artificial Intelligence Systems That Understand. In *IJCAI* (pp. 1059–1073).
- [Simon & Hayes, 1976] Simon, H. A. & Hayes, J. R. (1976). The understanding process: Problem isomorphs. *Cognitive psychology*, 8(2), 165–190.
- [Sonenberg et al., 2016] Sonenberg, L., Miller, T., Pearce, A., Felli, P., Muise, C., & Dignum, F. (2016). Social planning for social HRI. In *2nd Workshop on Cognitive Architectures for Social Human-Robot Interaction*.
- [Sorce et al., 2015] Sorce, M., Pointeau, G., Petit, M., Mealier, A.-L., Gibert, G., & Dominey, P. F. (2015). Proof of concept for a user-centered system for sharing cooperative plan knowledge over extended periods and crew changes in space-flight operations. In *The 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)* (pp. 776–783). Kobe, Japan: IEEE.
- [Sperber & Wilson, 1987] Sperber, D. & Wilson, D. (1987). Précis of relevance: Communication and cognition. *Behavioral and brain sciences*, 10(4), 697–710.
- [Suddrey et al., 2017] Suddrey, G., Lehnert, C., Eich, M., Maire, F., & Roberts, J. (2017). Teaching robots generalizable hierarchical tasks through natural language instruction. *IEEE Robotics and Automation Letters*, 2(1), 201–208.
- [Sung et al., 2012] Sung, J., Ponce, C., Selman, B., & Saxena, A. (2012). Unstructured human activity detection from rgb-d images. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on* (pp. 842–849).: IEEE.
- [Tenorth & Beetz, 2013] Tenorth, M. & Beetz, M. (2013). KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research*, 32(5), 566–590, doi:10.1177/0278364913481635. 00128.

- [Tenorth et al., 2010] Tenorth, M., Nyga, D., & Beetz, M. (2010). Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *2010 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1486–1491).: IEEE.
- [Tomasello, 2003] Tomasello, M. (2003). On the different origins of symbols and grammar. *Studies in the Evolution of Language*, 3, 94–110.
- [Vollmer et al., 2016] Vollmer, A.-L., Wrede, B., Rohlfing, K. J., & Oudeyer, P.-Y. (2016). Pragmatic Frames for Teaching and Learning in Human-Robot interaction: Review and Challenges. *Frontiers in Neurorobotics*, 10, 1–20, doi:10.3389/fnbot.2016.00010.
- [Vygotsky, 1978] Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, Massachusetts: Harvard University Press.
- [Waibel et al., 2011] Waibel, M., et al. (2011). RoboEarth. *IEEE Robotics & Automation Magazine*, 18(2), 69–82, doi:10.1109/MRA.2011.941632.
- [Weitzenfeld & Dominey, 2009] Weitzenfeld, A. & Dominey, P. F. (2009). Coaching Robots to Play Soccer via Spoken-Language. In *RoboCup 2008: Robot Soccer World Cup XII* Suzhou.
- [Wen et al., 2015] Wen, T.-H., Gasic, M., Mrksic, N., Su, P.-H., Vandyke, D., & Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.
- [wikiHow Incorporation, a] wikiHow Incorporation. How to Build a Robot at Home. *wikiHow*. <https://www.wikihow.com/Build-a-Robot-at-Home> [Retrieved 2019-04-24].
- [wikiHow Incorporation, b] wikiHow Incorporation. wikiHow - How to do anything. <https://www.wikihow.com/Main-Page> [Retrieved 2018-04-04].
- [Wikipedia contributors, 2019a] Wikipedia contributors (2019a). Behaviorism. *Wikipedia* (2019a). <https://en.wikipedia.org/w/index.php?title=Behaviorism&oldid=902951188> [Retrieved 2019-06-22]. Page Version ID: 902951188.
- [Wikipedia contributors, 2019b] Wikipedia contributors (2019b). Composability. *Wikipedia* (2019b). <https://en.wikipedia.org/w/index.php?title=Composability&oldid=896608894> [Retrieved 2019-06-09]. Page Version ID: 896608894.
- [Wikipedia contributors, 2019c] Wikipedia contributors (2019c). Theory of mind. *Wikipedia* (2019c). https://en.wikipedia.org/w/index.php?title=Theory_of_mind&oldid=895073941 [Retrieved 2019-05-09]. Page Version ID: 895073941.

- [Wilcox, 2011] Wilcox, B. (2011). Beyond Façade: Pattern Matching for Natural Language Applications (2011). http://chatscript.sourceforge.net/Documentation/Paper-%20Pattern_Matching_for_Natural_Language_Applications.pdf [Retrieved 2019-10-26]. Telltale Games.
- [Wilensky, 1981] Wilensky, R. (1981). Meta-Planning: Representing and Using Knowledge About Planning in Problem Solving and Natural Language Understanding. *Cognitive science*, 5(3), 197–233.
- [Xu et al., 2018] Xu, D., Nair, S., Zhu, Y., Gao, J., Garg, A., Fei-Fei, L., & Savarese, S. (2018). Neural Task Programming: Learning to Generalize Across Hierarchical Tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3795–3802).
- [Yamada et al., 2016] Yamada, T., Murata, S., Arie, H., & Ogata, T. (2016). Dynamical Integration of Language and Behavior in a Recurrent Neural Network for Human–Robot Interaction. *Frontiers in Neurorobotics*, 10, doi:10.3389/fnbot.2016.00005. 00001.
- [Yost, 1993] Yost, G. R. (1993). Acquiring knowledge in Soar. *IEEE Expert*, 8(3), 26–34.
- [Youssef et al., 2019] Youssef, A. B., Clavel, C., & Essid, S. (2019). Early Detection of User Engagement Breakdown in Spontaneous Human-Humanoid Interaction. *IEEE Transactions on Affective Computing*, (pp. 1–1)., doi:10.1109/TAFFC.2019.2898399.