EU FP7 CogX
ICT-215181
May 1 2008 (52months)

DR 6.5:
Mixed Initiative Situated Dialogue-Guided Curiosity

Hendrik Zender, Miroslav Janíček, Geert-Jan M. Kruijff

DFKI GmbH, Saarbrücken
⟨zender@dfki.de⟩

Due date of deliverable:	Month 48
Actual submission date:	May 28 2012
Lead partner:	DFKI
Revision:	final
Dissemination level:	PU

In previous years, WP6 investigated how situated dialogue could be used in human-robot interaction to help the robot learn more about its environment. This involved grounding dialogue in multi-agent models of beliefs and intentions, dealing with the uncertainty and incompleteness in these models, and communicating about the content in these models at different levels of granularity. These dialogues were typically tutor-driven. In Year 4, WP6 explored topics that have to do with robot-initiated dialogues. We investigated issues in common ground and transparency that help a robot to make use of its dialogue capabilities to explain its internal state and past actions to its user as well as to learn about the world by asking for missing knowledge or for clarifying uncertain knowledge. Furthermore, Year 4 was used to consolidate longer-term efforts originating in previous project periods, such as a software toolkit for natural language dialogue processing for talking robots.

Executive Summary

One of the objectives of CogX is self-extension. This requires the robot to have an intrinsic motivation – *"curiosity"* – to gather information, in order to acquire new and revise old knowledge. One of the sources of information about the world is dialogue. For dialogue to work, the robot needs to be able to establish with a human some form of mutually agreed-upon understanding, a *common ground*. This requires the robot to provide *transparency* about its actions and goals, as well as *interpreting the intentions* of its interlocutor. The overall goal of WP6 is to develop adaptive mechanisms for situated dialogue processing, to enable a robot to establish such common ground in situated dialogue.

In Year 1, WP6 investigated how a robot could carry out a situated dialogue with a human, about items in the world it needed to learn more about. The robot was able to formulate questions against a multi-agent model of situated beliefs, indicating what it did and did not know – and what it would like to know. The robot was able to represent and reason with uncertainty in experience, but it was relatively fixed in the strategies it would follow to communicate with the human about resolving the uncertainty.

In Year 2, WP6 investigated several issues in how to make dialogue behavior more *adaptive*. This covered several aspects: (1) Making dialogue strategies more adaptive, and (2) varying how much a robot needs to describe to be optimally transparent.

Throughout Years 1 and 2 we assumed the robot to have a fixed set of communicative competences, particularly where it concerned grammatical resources. Practically this meant that, even though the robot was still learning more and more about the world, it already knew how to talk about it. In Year 3, WP6 shifted its focus on taking the CogX objective of self-extension to the realm of situated dialogue processing as well. Making use of large ontologies and on-line resources for modeling common sense indoor knowledge (OMICS, WordNet, Bing image search) provided a powerful means for large-coverage resources for communicating about indoor environments. At the same time, we broadened the scope of self-extension in situated dialogue to the aspect of language acquisition per se.

In Year 4, WP6 investigated issues in *common ground* and *transparency*. We focused on verbalizing a robot's internal state and background knowledge, and integrating that functionality into human-robot interaction. This helps the robot to *explain* its internal state and past actions to its user, and to *learn* about the world by asking for missing knowledge or for *clarifying* uncertain knowledge. The result is a context-adaptive approach for clarification and explanation (Task 6.7), based in a robot's own understanding of what it does (not) know, and what it can (or cannot) attribute to a human it interacts with. The robot can initiate these kinds of dialogue itself, motivated by its own curiosity (Milestone M6.5). Over and beyond these efforts,

we consolidated our longer-term efforts originating in previous project periods, resulting in a software toolkit for natural language dialogue processing for talking robots (TaRoT).

Overall, the work in WP6 has led to a comprehensive theoretical and practical framework for situated dialogue, in which we have paid particular attention to issues in self-extension and introspection in and through dialogue. Our story of the situated nature of spoken dialogue in human-robot interaction is based in a situationally, socially, and epistemically perspectivized (i.e. "asymmetric") notion of common ground (Yr4) and acquisition (Yr3), and a cross-modal view on the relation between linguistic and extra-linguistic information. We consider situated dialogue in the larger context of a collaborative activity, using common ground and cross-modal connections between information to establish intention, intension and denotation for utterances – both for comprehension, and production, and for entities it does or does not know about. The robot can verbalize and clarify this knowledge, in context-adaptive ways, to establish transparency in common ground with the user (introspection), and to drive deliberated, self-motivated forms of learning (self-extension).

Role of Situated Dialogue in CogX

CogX investigates cognitive systems that self-understand and self-extend. In most of the scenarios explored within CogX such self-extension is done in a mixed-initiative, interactive fashion (e.g. the George and Dora scenarios): the robot interacts with a human, to learn more about the environment. WP6 contributes situated dialogue-based mechanisms to facilitate such interactive learning. Furthermore, WP6 explores several issues around the problems of self-understanding and self-extension in the context of dialogue processing. The dialogue capabilities provided by WP6 enable the robot to conduct situated dialogues for interactive learning based on its own curiosity. The models for spatially situated dialogue are grounded in the spatial models developed in WP3 (see also DR.3.3 and DR.3.4, especially [41]). The approach to continual abductive dialogue interpretation has a strong relation to WP4, in that it is inspired by the continual planning approach taken in that WP. At the same time, verbalization of past, ongoing, and past actions is grounded in the planning representations developed in WP4.

Contribution to the CogX Scenarios and Prototypes

The work of WP6 presented in this deliverable, DR.6.5, contributes directly to the George and Dora scenarios, in relation to work performed in WP1 (generating motivation goals from dialogue), WP3 (Qualitative spatial cognition), WP4 (Planning of action, sensing and learning), WP5 (Interactive

continuous learning of cross-modal concepts), and WP7 (Scenario-based integration). In particular, DR.7.3 reflects the work from WP6 in the systematic use of dialogue as a means of knowledge gathering and clarification, and for establishing transparency. DR.7.4 (especially [31]) describes a robot that is capable of continuous learning of visual concepts in dialogue with a tutor. These learning dialogues can be initiated by the human tutor or by the system itself.

1 Tasks, Objectives, Results

1.1 Planned work

The overall goal of CogX is to arrive at a theory of cognitive robots which are capable of self-understanding and self-extension. During the last years, WP6 worked on adaptive mechanisms for situated dialogue processing that would enable a robot to discuss with a human what it did and did not understand about the world. And, thus, through such dialogue, it could gain information to help it learn more. While in the previous years, such dialogues were mainly tutor-driven, the focus in Year 4 is on robot-initiated dialogues. The planned work for WP6 in Year 4 is to support curiosity-driven self-extension through situated dialogue and to provide means for making the robot's self-understanding transparent by verbalizing explanations of its internal states, its decision-making and actions.

Task 6.7 ***Adaptive strategies for clarification and explanation.*** *Towards the end of the project, the robot's learning is primarily curiosity-driven. This is an advance in that it now actively needs to initiate dialogues, if it wants to interact with other agents. We therefore want to investigate (adaptive strategies for) clarification and explanation, more from the engagement-level [30], to address the issue of how to set the context for a clarification request (i.e. scaffolding it), to avoid "out-of-the-blue" behavior.*

Milestone M.6.5 ***Mixed initiative situated dialogue-guided curiosity.*** *The system will be able to initiate and drive situated dialogues for interactive learning based on its own curiosity.*

Objective 2 *Specific representations of beliefs about beliefs for the specific cases of dialogue, manipulation, maps, mobility and some types of vision.* [WPs 2,3,6]

Objective 3 *Representations of how actions will alter the belief state of the cognitive system, and those of other agents, as represented in the first two objectives, i.e. models of the effects of actions on beliefs about space, categorical knowledge, action effects, dialogue moves etc.* [WPs 1,2,3,4,5,6]

Objective 9 *Methods that enable a robot to represent and reason about its beliefs and those of other agents to support natural dialogue and to extend its own abilities and understanding.* [WP 6]

Objective 11 *A robotic implementation of our theory able to complete a task involving mobility, interaction and manipulation, in the face of novelty, uncertainty, partial task specification, and incomplete knowledge.* [WPs 2,3,6,7]

1.2 Actual work performed

In Year 4 we developed new approaches for verbalizing a robot's internal belief state, reflecting what it does and does not know, and what it has done so far. We have integrated this type of verbalization into our general approach to situated dialogue processing. This involves tying it in with our algorithms for situated referring expression generation and resolution, and anchor progression. The primary function of this verbalization is to make the robot's understanding of, and reasoning about, the world more transparent to the user (Task 6.5). This aids building up and maintaining common ground in human-robot interaction, which is crucial if the robot is to successfully use dialogue to obtain information from a human user (Milestone M6.5).

We have achieved this task, and the corresponding milestone, as follows:

1. Building on earlier WP6 work we have further developed our abduction-based approach to continual dialogue understanding (§1.2.1). This makes it possible to reason explicitly with knowledge gaps, be they attributable to the human or the robot itself, and their possible resolution through dialogue as a form of action within a collaborative activity.

2. The very fact that we have knowledge gaps, and the obvious observation that robots and humans experience reality differently, have led us to reconsider the notion of common ground (§1.2.2). Already in earlier work on our abductive model we dropped the assumption on symmetry between dialogue participants. We have now taken this to its logical consequences, and built up a model of common ground which starts from asymmetry in the social, situated, and epistemic factors which play a role in common ground.

3. Finally, it is against this "asymmetric" background that we then consider verbalization of internal state (§1.2.3). This state covers robot beliefs, and past actions leading up those beliefs. The function of such verbalization in dialogue is to explain what the robot believes, and why it believes what it believes. This transparency, realized as a sub-dialogue clarifying what the robot would like to know and why, then provides the scaffolding for whatever question the robot is motivated to ask (mixed-initiative dialogue).

A (partly stylized) example of such a sub-dialogue is as follows.

1. I moved from the lab to the kitchen,

2. to look for cornflakes

3. I have searched for the cornflakes,

4. but I am unable to find them.

5. Could you tell me where the cornflakes are?

The example illustrates the verbalization of different plan/action steps (1,3), reasons (2), and outcomes (4). This provides a background for formulating the final question (5). Annex 1.2.3 describes in detail how the sub-dialogue in (1)–(4) can be generated from a plan and its execution trace.

With this we also answer the reviewers comments, pertaining to WP6:

> *[T]he methods for interpreting and producing referring expressions still need some work and are not yet a particularly convincing showcase of the very interesting model of situated dialogue being developed in the project.*

The work performed in Year 4 in WP6 (and through the integration in WP7) brings back many of the different strands worked on over the years. Using the integrated system functionality, we can now showcase our approach in various complex settings in human-robot interaction for self-extension.

The work performed in this WP meets several main objectives of CogX. The approaches to abductive dialogue interpretation (§1.2.1) and modeling common ground (§1.2.2) make contributions to Objectives 2, 3, 9 by providing methods for representing and reasoning about the beliefs of the robotic agent and other (human) agents it interacts with. The work on situated plan and execution verbalization (§1.2.3) makes further contributions to objectives 3 and 9 by allowing the robotic agent to verbalize its past actions and intentions, thereby allowing a human agent to understand the robot's behavior and beliefs. Transparency of a robot's intentions and actions has been further investigated in the context of a special journal issue (§1.3.2). Additionally, the work on dialogue interpretation and verbalization has been implemented in a software toolkit for talking robots (§1.3.1), which has been deployed in the WP7 integrated systems (cf. Objective 11).

1.2.1 Abductive dialogue interpretation

In task-oriented dialogues between two agents, such as between two humans or a human and a robot, there is more to dialogue than just understanding words. An agent needs to understand what is being talked about, and it needs to understand why it was told something. In other words, what does the speaker *intends* the hearer to do with the information, in the larger context of their joint activity? Language understanding can thus be phrased as an *intention recognition* problem: given an utterance from the human, how do we find the intention behind it?

Abductive Reasoning for Continual Dialogue Understanding

Janíček (Annex 2.1) presents an extended model of the abductive continual approach to situated dialogue understanding. This model draws inspiration from the field of continual planning [6], by explicitly capturing the possible knowledge gaps in such an interpretation. The idea is based on the notion of assertion*, an explicit test for the validity of a certain fact, going beyond the current context. This makes it possible to deal with both uncertainty and incompleteness in situated dialogue processing.*

Let us briefly discuss an example that uses this mechanism. A more detailed example can be found in the article in Annex 2.1.

Suppose that a human user is dealing with a household robot capable of manipulating objects (picking them up, putting them down). The robot and the human are both looking at a table with a mug ("mug_1"), and the human wants the robot to pick up the mug. The human's utterance, "Take the mug" is first parsed and analyzed semantically, and its translation is made part of the abduction context c, within which the robot tries to make sense of the utterance. The inference establishes several alternative proofs, and weighs them by the "costs" (probabilities) for the individual facts and assumptions appearing in a proof. The best proof is the one with the lowest cost. Suppose that the best proof state returned by ABDUCE is the following:

$\text{uttered}(\text{human}, \text{robot}, \text{event}_1)$	$[\textit{proved}]$	(1)
$\text{proposition}(\text{event}_1, \text{take})$	$[\textit{proved}]$	(2)
$\text{intends}(\text{event}_1, \text{human}, I)$	$[\textit{assumed}(p = 0.9)]$	(3)
$\text{relation}(\text{event}_1, \text{patient}, \text{thing}_1)$	$[\textit{proved}]$	(4)
$\text{refers\_to}(\text{thing}_1, X)$	$[\textit{asserted}]$	(5)
$\text{pre-condition}(I, \text{object}(X))$	$[\textit{asserted}]$	(6)
$\text{post-condition}(I, \text{state}(\text{is-holding}(\text{robot}, X)))$	$[\textit{assumed}(p = 0.7)]$	(7)

The proof is an explanation of the event in terms of a partially specified intention I related to the task specified above. An explanation is defined by its pre- and post-condition. The precondition is the existence of an entity X, and the postcondition is the state in which the robot is holding the entity X.

Assumptions are made with an assumability probability according to the beliefs the robot currently maintains, and inferences it can make from more general background knowledge (i.e. a rule base). In our approach the assumability function is manually designed, but it is conceivable to learn or infer it automatically. Reference resolution (i.e., does "the mug" refer to mug_1 or to some other referent?) is therefore essentially treated as an abduction problem.

Note that the proof state contains two atoms marked as assertions. These are the explicit gaps in the proof that make it a *partial interpretation*. They are chosen by the domain engineer, and since they need to be verified (or

falsified) by an external process, they form the interface to external knowledge bases and decision-making, which will select some of the assertions, and tries to verify them.

Suppose that the assertion (5) is tested first. This amounts to resolving the referring expression represented by $thing_1$. Under the open-world assumption, two interpretations are conceivable: the referent is mug_1 or the human might be referring to an object that is not part of common ground, and the reference thus cannot be resolved. The commitment to one of these interpretations is made by taking into account the probabilities and uncertainties about the world that are represented in the robot's beliefs.

The next assertion (6) expressed the presupposition accommodation that there exists an object to which the human is referring. This opens the possibility for further clarification in case the reference in (5) could not be resolved with a sufficient level of confidence.

In case the reference has been resolved to mug_1, but with low confidence, the robot might ask "Did you mean I should take *this* object?" (pointing at the mug, testing the hypothesis $\text{pre}(I, \text{object}, \text{mug}_1)$).

Likewise, in case the robot abductively concludes that the human is referring to an unknown mug, it might ask "Which object did you mean?", prompting the human to give an answer that would ultimately become the proof of the test action for $\text{pre}(I, \text{object}(X))$.

Alternatively, the robot might simply bring the most likely object. The human's acceptance of the choice would then verify the assertion. This is, again, a matter for consideration in the external planning and goal management.

1.2.2 Common ground

In order to interpret what it is that is being communicated, one needs to construct a *meaning representation*. In the processes of constructing meaning, one can appeal to different sources of information. In situated dialogue, these sources at least encompass the situation being described (*focus situation*); any other physical or discursive contexts, or common knowledge (*resource situations*); and the ways in which the communication partners take part in the dialogue (*social situation*) [11]. Figure 1 provides an example.

The social situation in situated dialogue makes it clear that communication partners look at the world from different perspectives. Some of the effort in communicating therefore goes into establishing a *common ground* between partners. The point is to establish a mutual understanding of what is being talked about, what is appealed to notably in reference to the world [8, 7]. This is a dynamic process, in which partners coordinate and align their beliefs [39, 28].

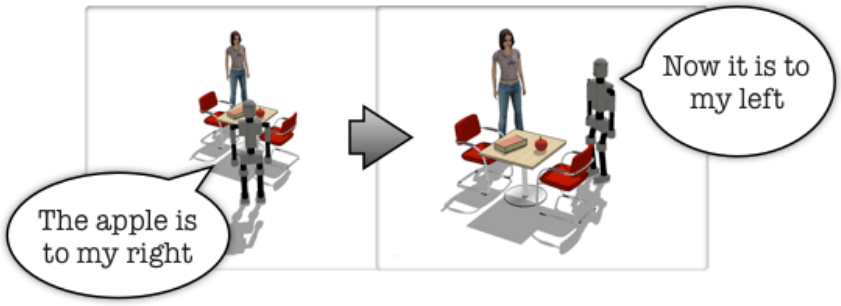

Figure 1: Different situations in meaning: "Now it is to my right" appeals in contrast to the previous situation ("now"), preceding dialogue ("it"), the current situation ("is"), and the current social situation ("to my right").

This need for common ground is not restricted to communication between humans. It holds just as much for human-robot interaction, where understanding and facilitating different perspectives on the world is crucial to establishing an effective collaborative context.

There is no Common Ground in Human-Robot Interaction

There is an inherent asymmetry to situated communication, especially in human-robot interaction. Robots literally see the world differently. This raises an important issue for how to model common ground between a human and a robot. Traditional approaches to common ground adopt a possible worlds-like model theory, on which a shared belief represents the fusion of two private beliefs within a single model. This silently assumes a symmetry in categories against which both private beliefs can be interpreted: A symmetry which cannot be assumed for human-robot interaction. Hence, on such a model, there is no common ground between humans and robots. Kruijff (Annex 2.2) presents an alternative model theory, which captures the inherently subjective nature of experience. It is based on a notion of propositions-as-proofs, turning subjective interpretation as well as the projection to intersubjective verification into a notion of inference or argumentation. Beliefs are arguments, whether private, attributed or shared. This results in a formulation of common ground as a dynamic structure of always argued, but possibly only partially confirmed, or partially assumed, beliefs.

Common ground is a complex notion. Generally speaking, it is about something *between* communicative partners: Namely, "shared understanding." Minimally, this is a shared understanding of the entities which have been talked about, grounded in the (possibly situated) domain of discourse [8, 7, 39]. After entities have been introduced into common ground, they are accessible for reference. Naturally, there can be more to common ground. In

task-oriented domains, common ground provides an interface between the task domain itself, and the communication which mediates and coordinates collaboration in that domain. Typically, this then leads to considering intentions, plans, and tasks to be part of common ground as well. On top of which we can essentially consider all that is implied by the concept of situated meaning [11], illustrated e.g. in Figure 1. (And that would include "commonsense knowledge," e.g. what if the robot was discussing not an edible apple, but a Mac?) As said, common ground is a complex notion.

1.2.3 Verbalization of plans

Interactive intelligent robots need to possess two important features: *autonomy* and *communication skills*. Situated communication can comprise different modalities, like, e.g., spoken dialogue. Autonomy can range from simple reactive control loops to sophisticated goal-directed action planning and execution. If the robot and its user are supposed to engage in some form of collaborative activity – no matter if it is a cognitive assistant whose main purpose is to support its human user, or if it is a curious robot whose main drive is to learn about the world – the human crucially needs to be able to know and understand the robot's actions (past, present, as well as intended ones) in as far as they are relevant for achieving a task at hand; the robot's autonomous decision-making and acting need to be *transparent* to the human.

We consider the verbalization of plans as one way of achieving such a *transparency* in situated discourse. In the context of CogX, we want to endow our robots with the ability to tell a human user about present objectives, what actions were executed in the past and what further actions are planned.

Situated Plan and Execution Verbalization

We present an approach to verbalizing reports of intentions and actions of a planner-enabled agent. We consider the case of an interactive intelligent robot that is endowed with a symbolic AI planner. The robot uses the planner to determine and execute sequences of actions in order to achieve a given goal. Since the robot is operating in a real, physical environment, making use of possibly imperfect sensing and actuating, it is likely to encounter unforeseen events or failures, and therefore needs to re-plan in order to come up with alternative plans for achieving its goal. In order to tell a human about what was planned, what was and was not successfully done, what happened – and why – we present a method for generating natural language reports based on such intended plans and the event structure of their execution. This verbalization is inherently situated in nature. For one, external entities that are used in the planning process refer to things, persons, or locations in

> *the physical environment that the robot and the hearer are situated in. Secondly, the robot's intentions and actions are temporally related to the discourse situation: the robot might report about its current plans and attempts for executing an ongoing task; the robot might explain what it did and what it couldn't successfully do in order to establish common ground for asking the human for help; or the the robot might simply report about past events and give details about why it chose to do what.*

The approach has been implemented on the integrated robotic system Dora [17]. Dora is programmed to exhibit a variety of intelligent behaviors, among which other intelligent mechanisms might arbitrate. However, these decisions might (initially) be intransparent to a human observer. We hence want Dora to provide verbal reports of what it attempted to do and why – thus establishing transparency about the complex spatio-temporal and causal relationships of its actions and action attempts.

Dora is equipped with a switching symbolic-probabilistic AI planner [12], able of continual planning and re-planning [6]. The approach is integrated in the general natural language processing sub-system of Dora and shares its linguistic resources with the other dialogue capabilities [19]. The verbalization module is connected to Dora's spatial knowledge base in order to refer to entities in Dora's spatial environment [40, 41, 42].

1.3 Additional work performed

1.3.1 Software toolkit for situated dialogue processing

In CogX, we gathered substantial knowledge and experience in developing functionality for situated dialogue processing. This extended the experience we had already gained in the project preceding CogX, namely CoSy. In Year 4, we dedicated effort to consolidating this knowledge and experience in a toolkit. This toolkit, called the Talking Robots Toolkit or TAROT provides a set of reusable functionalities and resources to build dialogue systems for human-robot interaction.

TAROT– The Talking Robots Toolkit

> TAROT *is an open-source software framework for building spoken dialogue functionality for human-robot interaction.* TAROT *does not impose a specific (cognitive) architecture for building a dialogue system. Its framework allows for multi-threaded (or asynchronous) processing. Processes are defined as glass boxes [25] (not black-boxes) and interact in an event-driven fashion.* TAROT *is written in the Scala programming language and targets the Java platform. Annex 2.4 provides a technical description.*

1.3.2 Outreach: state-of-the-art in expectations, intentions, and actions in human-robot-interaction

Human-robot interaction as a scientific field has received much attention in the past years. The research performed in the context of CogX, especially concerning situated natural language processing and human-robot spoken communication, draws from and directly contributes to this field.

One tangible and relevant outcome of this was that researchers from the CogX consortium (Marc Hanheide, Geert-Jan M. Kruijff, Hendrik Zender) organized the ICRA 2010 Workshop on Interactive Communication for Autonomous Intelligent Robots (ICAIR). Its topics centered around making robots articulate what they understand, intend, and do. Being a successor to the ICRA 2008 Workshop on Social interaction with Intelligent Indoor Robots (SI3R), it attracted researchers from different fields of robotics who work on robots that communicate.

As a follow up to this workshop as well as to the HRI 2011 Workshop on The Role of Expectations in Intuitive Human-Robot Interaction (Verena Hafner, HU Berlin; Manja Lohse, Bielefeld U; Joachim Meyer, Ben-Gurion University of the Negev, Israel; Yukie Nagai, Osaka U; Britta Wrede, Bielefeld U), the editors of the International Journal of Social Robotics proposed to organize a Special Issue on Expectations, Intentions and Actions, for which Marc Hanheide, Manja Lohse, and Hendrik Zender served as guest editors.

Journal of Social Robotics: "Expectations, Intentions & Actions"
This special issue (Annex 2.5) bundles recent advances in embodied and situated social human-robot interaction. The key questions are how meeting or failing to meet the user's expectations influences the efficiency and effectiveness of human-robot interaction; how more effective and efficient interaction with humans can be achieved using modalities available to a robot; how robots can be equipped with models enabling them to understand their users' state of mind; and similarly, how they can make their own expectations and states explicit through eligible communication channels. Each of the seven contributed articles in this issue highlights different aspects around the central theme of expectations, intentions, and actions in human-robot interaction. *The topics covered range from recognition of verbal and non-verbal cues of intentions and expectations, to verbalization and presentation techniques that make internal processing of the robot accessible to the human.*

1.4 Relation to state-of-the-art

Below we briefly discuss how the obtained results relate to the current state-of-the-art. We refer the reader to the annexes for more in-depth discussions.

Abduction Our approach to situated dialogue processing is based on our ongoing research in dialogue as part of continual ("contingency-based") collaborative activity ([23], and [24]). Dialogue modeling is connected to multi-agent models of situation awareness. These models capture beliefs and intentions, and their inherent uncertainty and incompleteness with respect understanding the environment. It is a strongly intention-oriented approach, in the sense of [2, 1, 14, 29, 9]. It accords a strong role to common ground in interaction [21, 20]. Core to the approach is abductive inference. This is inspired by [34, 35, 33], but differs in that our approach does not assume symmetry in understanding between the different dialogue partners. The recent work reported in this deliverable illustrates how we take asymmetry in understanding to dealing with partial information (Annex 2.1), and to a reconsideration of the subjective and intersubjective nature of content in common ground (Annex 2.2)

Common ground Understanding and facilitating *common ground* in human-robot interaction has received substantial attention in recent years [21, 22, 20]. As put forward by several researchers [32, 38, 36, 27], common gound is indeed crucial for establishing an effective collaborative context. Failure to do so typically leads to a breakdown in communication, see e.g. [36, 29]. There is a large body of work on considering intentions, plans, and tasks to be part of common ground [2, 14, 1, 29, 16, 3, 15, 9, 23]. The problem common to all these approaches is that they assume a symmetry between interlocutors: How the speaker sees and talks about the world is how the hearer understands the world. This symmetry-assumption is explicitly stated in e.g. [35]. We can also see it reflected in the formal aspects of the model theories underlying these approaches. Common ground on a belief in proposition p means that, in a single possible worlds model, we can reach a world on which p holds from the worlds on which the private and attributed beliefs about p hold. This is incorrect, as it assumes that the different interlocutors have a single (symmetric) objective model for interpreting. This ignores the fundamentally *subjective* nature of experience, and the inherent differences between humans and robots. The algebraic model theory we propose here overcomes these problems.

Plan and execution verbalization Brenner [4] describes the use of classical AI planning techniques for interpretation and execution of human commands. He sketches how a robot can understand natural language (NL), plan the realization and revise its plans based on new perceptions. Their approach is similar to ours in that the goal is to couple planning symbols to natural language semantics and surface forms. In contrast to the approach presented in this paper, their approach focusses on *understanding* rather than on *generation*.

However, what we are interested in is the generation of natural language from planned and executed goal-directed sequences of actions. One domain where such an approach is chosen is story telling. Telling stories requires methods from many subfields of artificial intelligence (AI), e.g. planning, reasoning about beliefs, and dialogue systems. In [5] an approach to story generation using a continual multi-agent planner is presented. The Virtual Storyteller is another framework that generates simple story texts [37]. It is based on simulation of virtual characters in a story world. An event sequence is captured by a 'Plot Agent' in a formal representation. The representations are similar to the plans in [5], i.e., STRIPS-like [10]. A 'Narrator' component turns the representation into an actual story by selecting the content and processing it with NLG techniques. Story telling and NLG are also brought together in, e.g., [26].

Another system that verbalizes some kind of plans is PROVERB [18]. In this work, mathematical natural deduction (ND) style proofs are verbalized. As input there is a representation of a ND proof. It is processed by a macro-planner to plan output that consists of primitive actions. The actions can be defined as communicative goals they fulfill as well as their possible verbalizations. Subsequently, more detailed linguistic decisions are made in the micro-planning component. Syntactic realizations are done using Tree-Adjoining Grammar (TAG). The final output then is the ND proof in natural language.

When verbalizing plans and actions for autonomous robots, an additional aspect comes into play: the agent's observations are potentially incorrect or incomplete, and execution (as well as execution failures) become a key issue. In such a context it is desirable to have a formal way to determine an explanation of why a plan went wrong and how the problem could be solved. Göbelbecker et al. [13] provide a formalization regarding this issue. The work presented here addresses the prerequisites for informing the user about failures executing the plan and how the user could help to solve the problems. In order to achieve this, we investigate appropriate strategies for a suitable verbalization of the planned and performed actions in a way that is understandable to a human user.

2 Annexes

2.1 Janíček, "Abductive Reasoning for Continual Dialogue Understanding"

Bibliography Miroslav Janíček. "Abductive Reasoning for Continual Dialogue Understanding." In M. Slavkovik and D. Lassiter, editors, *New Directions in Logic, Language, and Computation*. Springer, 2012 (to appear).

Abstract This paper presents a continual context-sensitive abductive framework for understanding situated spoken natural dialogue. The framework builds up and refines a set of partial defeasible explanations of the spoken input, trying to infer the speaker's intention. These partial explanations are conditioned on the eventual verification of the knowledge gaps they contain. This verification is done by executing test actions, thereby going beyond the initial context. The approach is illustrated by an example set in the context of human-robot interaction.

Relation to WP The paper presents an extended and improved version of the approach presented in DR.6.3. It provides the basic inference mechanism for reasoning about beliefs and intentions in the context of dialogue processing. In the context of DR.6.5 it is instrumental in determining the appropriate epistemic context for scaffolding mixed-initiative dialogue for curiosity-driven learning – i.e. which beliefs held by the robot (private or attributed) need to be verbalized, to explain what the robot does or needs to know.

2.2 Kruijff, "There Is No Common Ground In Human-Robot Interaction"

Bibliography Geert-Jan M. Kruijff. "There Is No Common Ground In Human-Robot Interaction." *Manuscript*, 2012.

Abstract There is an inherent asymmetry to situated communication. Those communicating look at the world from different perspectives. This holds particularly true for human-robot interaction. Robots literally see the world differently: They experience reality in fundamentally different ways. This raises an important issue for how to model common ground between a human and a robot. Traditional approaches to common ground adopt a possible worlds-like model theory, on which a shared belief represents the fusion of two private beliefs within a single model. This silently assumes a symmetry in categories against which both private beliefs can be interpreted: A symmetry which cannot be assumed for human-robot interaction. Hence, on such a model, there is no common ground between humans and robots. This paper presents an alternative model theory, which captures the inherently subjective nature of experience. It is based on a notion of propositions-as-proofs, turning subjective interpretation as well as the projection to intersubjective verification into a notion of inference or argumentation. Beliefs are arguments, whether private, attributed or shared. This results in a formulation of common ground as a dynamic structure of always argued, but possibly only partially confirmed or partially assumed beliefs.

Relation to WP Common ground in dialogue indicates a level of mutual understanding between interlocutors, of what is being talked about. In CogX, "what is being talked about" primarily concerns beliefs about experience of an environment, or about inferred (possibly attributable) characteristics of an environment. Common ground can thus be argued to be based on to the ability to align experience and expectations. This is crucial for an effective transfer of information in communication; without it, dialogue is unlikely to yield insights which the robot can use to drive its learning. The problem that now arises in human-robot interaction is that robots and humans experience reality fundamentally differently. A robot cannot simply assume that a human "symmetrically" understands what it is talking about. The robot needs to reason, within its limited capabilities, how the human may understand the environment, and to what extent that might correspond to how it understands the environment itself. The manuscript describes an approach to formulating an algebraic model theory on which we can define a logic for reasoning about different epistemic and situated perspectives, and how they could be aligned.

2.3 Schoch & Zender, "Situated Plan and Execution Verbalisation"

Bibliography Gerald Schoch and Hendrik Zender. "Situated Plan and Execution Verbalisation." *Technical Report* (2012).

Abstract In this paper, we present an approach to verbalizing reports of intentions and actions of a planner-enabled agent. We consider the case of an interactive intelligent robot that is endowed with a symbolic AI planner. The robot uses the planner to determine and execute sequences of actions in order to achieve a given goal. Since the robot is operating in a real, physical environment, making use of possibly imperfect sensing and actuating, it is likely to encounter unforeseen events or failures, and therefore needs to re-plan in order to come up with alternative plans for achieving its goal. In order to tell a human about what was planned, what was and was not successfully done, what happened – and why – we present a method for generating natural language reports based on such intended plans and the event structure of their execution. This verbalization is inherently situated in nature. For one, external entities that are used in the planning process refer to things, persons, or locations in the physical environment that the robot and the hearer are situated in. Secondly, the robot's intentions and actions are temporally related to the discourse situation: the robot might report about its current plans and attempts for executing an ongoing task; the robot might explain what it did and what it couldn't successfully do in order to establish common ground for asking the human for help; or the the robot might simply report about past events and give details about why it chose to do what.

Relation to WP The approach is integrated in the general natural language processing system developed in this WP and shares its linguistic resources with the other dialogue capabilities, cf. Section 2.4. The verbalization module is connected to the spatial representations developed in WP3 in order to refer to entities in Dora's spatial environment. By this it builds upon previous work in this WP on spatially situated generation of referring expressions.

The approach has been implemented on the integrated robotic system Dora (WP7), which is equipped with a switching symbolic-probabilistic AI planner, able of continual planning and re-planning. Thereby, the work presented here has also a close connection to the research on planning performed in WP4.

2.4 Janíček, "Robust situated language processing with TAROT: The Talking Robots Toolkit"

Bibliography Miroslav Janíček. "Robust situated language processing with TAROT: The Talking Robots Toolkit." Manuscript. (2012).

Abstract This document describes TAROT the Talking Robots Toolkit. TAROT is an open-source software framework for building spoken dialogue functionality for human-robot interaction. TAROT does not impose a specific (cognitive) architecture for building a dialogue system. Its framework allows for multi-threaded (or asynchronous) processing. Processes are defined as open "glass" boxes, and can interact in an event-driven fashion. TAROT is written in the Scala programming language and targets the Java platform.

Relation to WP The manuscript describes a toolkit which consolidates the knowledge and experience gained in CogX in developing systems for situated dialogue processing in HRI.

2.5 Hanheide et al., "Expectations, Intentions, and Actions in Human-Robot Interaction"

Bibliography Marc Hanheide, Manja Lohse and Hendrik Zender. "Expectations, Intentions, and Actions in Human-Robot Interaction." *International Journal of Social Robotics*, 4(2):107–108, Springer Verlag, April 2012.

Abstract Human-robot interaction is becoming increasingly complex through the growing number of abilities, both cognitive and physical, available to today's robots. At the same time, interaction is still often difficult because the users do not understand the robots' internal states, expectations, intentions, and actions. Vice versa, robots lack understanding of the users' expectations, intentions, actions, and social signals.

This article constitutes the editorial of a special issue on "Expectations, Intentions & Actions" of the International Journal of Social Robotics. The special issue bundles recent advances in addressing these challenges. The key questions are how meeting or failing the user's expectations influences the efficiency and effectiveness of human-robot interaction; how more effective and efficient interaction with humans can be achieved using modalities available to a robot; how robots can be equipped with models enabling them to understand their users' state of mind; and similarly, how they can make their own expectations and states explicit through eligible communication channels.

Each of the seven articles in the special issue highlights different aspects around the central theme of *expectations, intentions, and actions in human-robot interaction*. The topics covered range from recognition of verbal and non-verbal cues of intentions and expectations, to verbalization and presentation techniques that make internal processing of the robot accessible to the human.

Relation to WP Meeting or failing to meet the users expectations influences the efficiency and effectiveness of human-robot interaction. The article's discussion complements the more fundamental issues of common ground in HRI presented in Annex 2.2.

References

[1] J.F. Allen and D.J. Litman. Discourse processing and common sense plans. In P.R. Cohen, J. Morgan, and M. Pollack, editors, *Intentions and Communication*. The MIT Press, 1990.

[2] J.F. Allen and C.R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15(3), 1980.

[3] N. Blaylock and J. Allen. A collaborative problem-solving model of dialogue. In L. Dybkjr and W. Minker, editors, *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, pages 200–211, 2005.

[4] M. Brenner. Situation-aware interpretation, planning and execution of user commands by autonomous robots. In *Robot and Human interactive Communication*, The 16th IEEE International Symposium, pages 540–545, 2007.

[5] M. Brenner. Creating dynamic story plots with continual multiagent planning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, july 2010.

[6] M. Brenner and B. Nebel. Continual planning and acting in dynamic multiagent environments. *Journal of Autonomous Agents and Multiagent Systems*, 19(3):297–331, 2009.

[7] H. H. Clark and S. A. Brennan. Grounding in communication. In L.B. Resnick, J.M. Levine, and S.D. Teasley, editors, *Perspectives on socially shared cognition*, pages 127–149. APA Books, Washington VA, 1991.

[8] H. H. Clark and C. R. Marshall. Reference diaries. In D. L. Waltz, editor, *Theoretical issues in natural language processing*, volume 2, pages 57–63. Association for Computing Machinery, New York NY, 1978.

[9] G. Ferguson and J. Allen. Mixed-initiative dialogue systems for collaborative problem-solving. *AI Magazine*, 28(2):23–32, 2007.

[10] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[11] J. Ginzburg. Situation semantics and the ontology of natural language. In P. Portner, C. Maierborn, and K. von Heusinger, editors, *The Handbook of Semantics*, pages 830–851. de Gruyter, 2011.

[12] M. Göbelbecker, C. Gretton, and R. Dearden. A switching planner for combined task and observation planning. In *Electronic Proceedings of the Workshop on Decision Making in Partially Observable, Uncertain*

Worlds: Exploring Insights from Multiple Communities at the Twenty-Second International Join Conference on Artificial Intelligence (DMPOUW 2011), 2011.

[13] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel. Coming up with good excuses: What to do when no plan can be found. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI Press, may 2010.

[14] B. J. Grosz and C. L. Sidner. Attention, intention and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[15] B.J. Grosz and L. Hunsberger. The dynamics of intention in collaborative activity. *Cognitive Systems Research*, 7(2-3):259–272, 2006.

[16] B.J. Grosz and S. Kraus. The evolution of sharedplans. In A. Rao and M. Wooldridge, editors, *Foundations and Theories of Rational Agency*, pages 227–262. Springer, 1999.

[17] M. Hanheide, C. Gretton, R. Dearden, N. Hawes, J. Wyatt, A. Pronobis, A. Aydemir, M. Göbelbecker, and H. Zender. Exploiting probabilistic knowledge under uncertain sensing for efficient robot behaviour. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11)*, Barcelona, Catalonia, Spain, July 2011.

[18] X. Huang and A. Fiedler. Proof verbalization as an application of nlg. In *Proceedings oft the 15TH International Joint Conference on Artificial Intelligence (IJCAI*, pages 965–970. Morgan Kaufmann, 1997.

[19] M. Janíček. Tarot: The Talking Robots Toolkit: technical description, version 0.1. Technical report, German Research Center for Artificial Intelligence (DFKI), 2012. CogX project deliverable DR6.5.

[20] S. Kiesler. Fostering common ground in human-robot interaction. In *Proceedings of 14th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN 2005)*, 2005.

[21] G. Klein, P.J. Feltovich, J.M. Bradshaw, and D.D. Woods. Common ground and coordination in joint activity. In W.B. Rouse and K.R. Boff, editors, *Organizational Simulation*, pages 139–184. John Wiley, New York City, NY, 2004.

[22] G. Klein, D.D. Woods, J.M. Bradshaw, R. Hoffman, and P. Feltovich. Ten challenges for making automation a team player in joint human-agent activity. *IEEE Intelligent Systems*, 19(6):91–95, November-December 2004.

[23] G.J.M. Kruijff, M. Janíček, and P. Lison. Continual processing of situated dialogue in human-robot collaborative activities. In *Proceedings of the 19th International Symposium on Robot and Human Interactive Communication (RO-MAN 2010)*. IEEE, 2010.

[24] G.J.M. Kruijff, M. Janíček, and H. Zender. Situated communication for joint activity in human-robot teams. *IEEE Intelligent Systems*, March/April 2012. forthcoming.

[25] G.J.M. Kruijff, P. Lison, T. Benjamin, H. Jacobsson, H. Zender, and I. Kruijff-Korbayová. Situated dialogue processing for human-robot interaction. In H.I. Christensen, G.J.M. Kruijff, and J. L. Wyatt, editors, *Cognitive Systems*, volume 8 of *Cognitive Systems Monographs, COSMOS*, chapter 8, pages 311–364. Springer Verlag, Berlin/Heidelberg, Germany, 2010.

[26] B. Loenneker. Narratological knowledge for natural language generation, 2005.

[27] R.R. Murphy and J.L. Burke. The safe human-robot ratio. In M.J. Barnes and F. Jentsch, editors, *Human-Robot Interactions in Future Military Operations*, Human Factors in Defence, pages 31–49. Ashgate, 2010.

[28] M.J. Pickering and S. Garrod. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27:169–225, 2004.

[29] M. Poesio and D. Traum. Conversational actions and discourse situations. *Computational Intelligence*, 13(3):309–347, 1997.

[30] C. L. Sidner, C. Lee, C. Kidd, N. Lesh, and C. Rich. Explorations in engagement for humans and robots. *Artificial Intelligence*, 166(1-2):140–164, 2005.

[31] D. Skočaj, M. Kristan, A. Vrečko, M. Mahnič, M. Janíček, G.J.M. Kruijff, M. Hanheide, N. Hawes, T. Keller, M. Zillich, and K. Zhou. An integrated system for interactive learning in dialogue with a tutor. *under submission*, 2012.

[32] D. Sofge, M. Bugajska, J. G. Trafton, D. Perzanowski, S. Thomas, M. Skubic, S. Blisard, N.L. Cassimatis, D. Brock, W. Adams, and A. Schultz. Collaborating with humanoid robots in space. *International Journal of Humanoid Robotics*, 2(2):181–201, 2005.

[33] M. Stone. Intention, interpretation and the computational structure of language. *Cognitive Science*, 28(5):781–809, 2004.

[34] M. Stone and R.H. Thomason. Context in abductive interpretation. In *Proceedings of EDILOG 2002: 6th workshop on the semantics and pragmatics of dialogue*, 2002.

[35] M. Stone and R.H. Thomason. Coordinating understanding and generation in an abductive approach to interpretation. In *Proceedings of DIABRUCK 2003: 7th workshop on the semantics and pragmatics of dialogue*, 2003.

[36] K. Stubbs, P.J. Hinds, and D. Wettergreen. Autonomy and common ground in human-robot interaction: A field study. *IEEE Intelligent Systems*, 22:42–50, March 2007. Special Issue on Interacting with Autonomy.

[37] I. M. T. Swartjes and M. Theune. The virtual storyteller: story generation by simulation. In *Proceedings of the Twentieth Belgian-Netherlands Conference on Artificial Intelligence, BNAIC 2008, Enschede*, pages 257–265, Enschede, October 2008. University of Twente.

[38] J. G. Trafton, N. L. Cassimatis, M. D. Bugajska, D. P. Brock, F. E. Mintz, and A. C. Schultz. Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 35(4):460–470, 2005.

[39] D. Wilkes-Gibbs and H. H. Clark. Coordinating beliefs in conversation. *Journal of Memory and Cognition*, 31:183–194, 1992.

[40] H. Zender. *Situated Production and Understanding of Verbal References to Entities in Large-Scale Space*, volume 36 of *Saarbrücken Dissertations in Computational Linguistics and Language Technology*. German Research Center for Artificial Intelligence and Saarland University, Saarbrücken, Germany, 2011.

[41] H. Zender. Multi-layered conceptual spatial mapping – representing spatial knowledge for situated action and human-robot interaction. In Y. Amirat, A. Chibani, and G. P. Zarri, editors, *Bridges Between the Methodological and Practical Work of the Robotics and Cognitive Systems Communities – From Sensors to Concepts*, Intelligent Systems Reference Library. Springer Verlag, Berlin/Heidelberg, Germany, to appear 2012.

[42] H. Zender, G.J.M. Kruijff, and I. Kruijff-Korbayová. Situated resolution and generation of spatial referring expressions for robotic assistants. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1604–1609, Pasadena, CA, USA, July 2009.

Abductive Reasoning for Continual Dialogue Understanding

Miroslav Janíček

German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
miroslav.janicek@dfki.de

Abstract. In this paper we present a continual context-sensitive abductive framework for understanding situated spoken natural dialogue. The framework builds up and refines a set of partial defeasible explanations of the spoken input, trying to infer the speaker's intention. These partial explanations are conditioned on the eventual verification of the knowledge gaps they contain. This verification is done by executing test actions, thereby going beyond the initial context. The approach is illustrated by an example set in the context of human-robot interaction.

Keywords: Intention recognition, natural language understanding, abduction, context-sensitivity.

1 Introduction

In task-oriented dialogues between two agents, such as between two humans or a human and a robot, there is more to dialogue than just understanding words. The robot needs to understand what is being talked about, but it also needs to understand why it was told something. In other words, what the human *intends* the robot to do with the information in the larger context of their joint activity.

Therefore, understanding language can be phrased as an *intention recognition* problem: given an utterance from the human, how do we find the intention behind it?

In this paper, we explore an idea inspired by the field of continual planning [8], by explicitly capturing the possible knowledge gaps in such an interpretation. The idea is based on the notion of *assertion*, an explicit test for the validity of a certain fact, going beyond the current context.

The structure of the paper is as follows. After briefly introducing the notion of intention recognition, abduction and situatedness in the next section, we introduce the continual abductive reasoning mechanism in §3, and discuss it on an example in §4, before concluding with a short summary.

2 Background

The idea of expressing *understanding* in terms of intention recognition has been introduced by H. P. Grice [12,20]. In this paper, we build on Stone and Thomason's approach to the problem [23] who in turn extend the work done by Hobbs

and others [13], and base their approach to intention recognition on *abductive reasoning*.

2.1 Abduction

Abduction is a method of explanatory logical reasoning introduced into modern logic by Charles Sanders Peirce [11]. Given a theory T, a rule $T \vdash A \rightarrow B$ and a fact B, abduction allows inferring A as an explanation of B. B can be deductively inferred from $A \cup T$. If $T \nvdash A$, then we say that A is an *assumption*.

There may be many possible causes of B besides A. Abduction amounts to *guessing*; assuming that the premise is true, the conclusion holds too. To give a well-known example:

> Suppose we are given two rules saying "if the sprinkler is on, then the lawn is wet" and "if it rained, then the lawn is wet". Abductively inferring the causes for the fact that the lawn is wet then yields two possible explanations: the sprinkler is on, or it rained.

Obviously, as there may be many possible explanations for a fact, in practical applications there needs to be a mechanism for selecting the best one. This may be done by purely syntactic means (e.g. lengths of proofs), or semantically by assigning *weights* to abductive proofs and selecting either the least or most costly proof [22], or by assigning probabilities to proofs [18]. In that case, the most probable proof is also assumed to be the best explanation. Our approach combines both aspects.

2.2 Intention Recognition

Abduction is a suitable mechanism to perform inferences on the pragmatic (discourse) level. For understanding, abduction can be used to infer the explanation *why* an agent said something, in other words the *intention* behind the utterance.

An intention is usually modelled as a goal-oriented cognitive state distinct from *desires* in that there is an explicit commitment to acting towards the goal and refraining from actions that may render it impossible to achieve [7,10].

For the purposes of this paper, we shall treat intentions as intended actions that have pre- and post-conditions, similar to planning operators in automated planning. Pre-conditions express the necessary conditions *before* the action is executed (and sufficient for its execution), and post-conditions express the necessary conditions *after* the action is executed.

Note that reasoning with intentions allows us to reverse the task, and search for appropriate (surface) presentation of a given intention [24]. Intentions can therefore serve as a middle representational layer and abduction as the inference mechanism by using which we either turn a realisation into an intention, or the other way around.

2.3 Situated Understanding

Suppose that a human user is dealing with a household robot capable of manipulating objects (finding them, picking them up, putting them down). The human wants the robot to bring him the mug from the kitchen, so he instructs the robot by saying:

"Bring me the mug from the kitchen."

Now, what should the robot do? In the beginning, the utterance is just a stream of audio. The robot has to detect voice in the audio data, and if the speech recognition works well enough, it will be able to obtain the surface form of the utterance, i.e. the words that were spoken by the human.

Once the word sequence is recognised, the robot needs to assign linguistic structure to it so that it can reason about its logical structure. The logical structure of the utterance is typically not in any way related to the actual *situated* experience of the robot. The noun phrases "the mug" and "the kitchen" are just referring expressions *standing for* some entities in the real world, and can be manipulated as expressions using logical rules without the need to be concerned about value of the standing-for relation.

However, this relation is absolutely crucial to understanding what the human said and why. Without being able to reduce the referring expressions to the corresponding real-world entities there is no true understanding, and – more importantly – there can be no appropriate reaction, which presumably is one of the reasons why the human uttered the sentence in the first place (i.e. to elicit such a reaction).

Grounding the relation in reality is therefore a crucial task that any cognitive agent has to tackle. However, since all sensory perception is necessarily partial and subject to uncertainty, there is no guarantee that the "knowledge base", a formalisation of the current snapshot of the knowledge about the world, contains the information necessary for such a grounding. In other words, a situated agent cannot afford the luxury of reasoning under closed-world assumption, and has to venture beyond that.

This means that the robot must be able identify its knowledge gaps, and verify or falsify them *while* trying to understand the human's utterance. This implies that the processes of understanding an input and acting on it are interleaved and that there is a bi-directional interface between them.

3 Approach

This paper extends the work of Stone and Thomason on context-sensitive language understanding by explicitly modelling the knowledge gaps that inevitably arise in such an effort due to uncertainty and partial observability. The approach is based on generating partial hypotheses for the explanation of the observed behaviour of other agents, under the assumption that the observed behaviour is

intentional. These partial hypotheses are defeasible and conditioned on the validity (and eventual verification) of their assumptions.

In this section, we examine the abductive reasoning system capable of representing knowledge gaps in the form of partial proofs, how such partial proofs can be generated and verified or falsified, and the semantic framework used in our system to capture linguistic meaning that the system then grounds in reality.

3.1 Partial Abductive Proofs

Our abductive inference mechanism is essentially Hobbs and Stickel's logic programming approach to weighted abduction [13,22] enhanced by a contextual aspect [3] with the weights in the system being assigned a probabilistic interpretation following Charniak and Shimony [9].

Abduction Context. Inference in our system makes use of four ingredients: *facts* (denoted $\mathcal{F}$), *rules* ($\mathcal{R}$), *disjoint declarations* ($\mathcal{D}$) and *assumability functions* ($\mathcal{S}$), collectively called the *abduction context*. The proof procedure uses these iteratively in order to derive proofs of an initial *goal*.

- *Facts* are modalised formulas of the form

  $$\mu : A$$

  where μ is a (possibly empty) sequence of modal contexts, and A is an atomic formula, possibly containing variables.
- *Rules* are modalised Horn clauses, i.e. formulas of the form

  $$(\mu_1 : A_1/t_1) \wedge ... \wedge (\mu_n : A_n/t_n) \rightarrow (\mu_H : H)$$

  where each of the $\mu_i : A_i$ and $\mu_H : H$ are modalised formulas. Each antecedent is annotated by t_i, which determines the way the antecedent is manipulated and is one of the following:
  - *assumable(f)* – the antecedent is assumable under function f;
  - *assertion* – the antecedent is asserted, i.e. identifies a knowledge gap, conditioning the validity of the proof on it being proved in a subsequent reinterpretation (see below).
- *Assumability functions* are partial functions f, $f : \mathcal{P}(\mathcal{F}) \rightarrow \mathbb{R}_0^+$, where $\mathcal{P}(\mathcal{F})$ is the set of modalised formulas, with the additional monotonicity property that if $F \in \text{dom}(f)$, then for all more specific (in terms of variable substitution) facts F', $F' \in \text{dom}(f)$ and $f(F) \leq f(F')$. We also define an empty ("truth") assumability function $\perp$ such that $\text{dom}(\perp) = \emptyset$.

  Since they are partial functions, assumability functions determine both whether a modalised formula may be assumed and the cost of such an assumption. As a special case, the empty assumability function $\perp$ can be used to prevent the formula from being assumed altogether.

– A *disjoint declaration* is a statement of the form

$$\text{disjoint}(\mu : A_1, \ldots, \mu : A_n)$$

which specifies that at most one of the modalised formulas $\mu : A_i$ may be used in the proof. A_i and A_j cannot be unified for all $i \neq j$.

Proof Procedure. The proof procedure is an iterative rewriting process starting from some initial *goal* state. A *proof state* is a sequence of marked modalised formulas (called *queries* in this context)

$$Q_1[n_1], \ldots, Q_m[n_m]$$

The markings n_i are one of the following:

– *unsolved(f)* – the query is yet to be proved and can be assumed if it is in the domain of the assumability function f;
– *proved* – the query is proved in the proof state;
– *assumed(f)* – the query is assumed under assumability function f;
– *asserted* – the query is asserted – its validity is not to be determined in the current context.

Algorithm 1 defines the proof procedure in detail. The top-level function ABDUCE takes an abduction context c and a proof state Π, and returns a set of proof states that

(1) are transformations of Π,
(2) are consistent with c, and
(3) do not contain any *unsolved* queries.

First, the input proof state is checked for validity with respect to the disjoint declarations $\mathcal{D}$ in the function IS-DISJOINT-VALID. If the check turns out to be negative, the proof state is discarded, and ABDUCE returns an empty set.

If Π satisfies the disjointness constraints, the function TF-DUP turns it into a set of proof states where unsolved queries that have already been proved, assumed or asserted are removed. The transformation returns a non-empty set of proof states. This step ensures that no query is examined more than once.

Next, each proof state resulting from TF-DUP is again checked whether it contains an unsolved query. If it does not, then the conditions (1)–(3) above are already fulfilled, and the proof state ends up in the result.

If it does, the proof procedure resolves the proof state against the facts, rules and assumability functions, collecting the results, and recursively calling ABDUCE on them so as to satisfy the above conditions.

Formally, given a proof state

$$\Pi = Q_1[n_1], \ldots, Q_m[n_m]$$

where Q_i is the leftmost query marked (guaranteed to exist at this point) as *unsolved*(f) where f is an assumability function, the transformation rules TF-FACT, TF-RULE and TF-ASSUME each return a (possibly empty) set of transformed proof states, and are defined as follows:

– TF-FACT (resolution with a fact): For all $Q \in \mathcal{F}$ such that the Q and Q_i are unifiable with a most general unifier σ (denoted $\sigma = \text{unify}(Q, Q_i)$), add a new state Π' to the result of the transformation:

$$\Pi' = Q_1\sigma[n_1], \ldots, Q_i\sigma[\mathit{proved}], \ldots Q_m\sigma[n_m]$$

– TF-RULE (resolution with a rule): For each rule $r \in \mathcal{R}$ of the form

$$G_1/t_1, \ldots, G_k/t_k \rightarrow H$$

(with variables renamed so that it has no variables in common with Π) such that there is a $\sigma = \text{unify}(H, Q_i)$, i.e. the rule head is unifiable with the unsolved query, add a new state Π' to the transformation result:

$$\begin{aligned}\Pi' = {} & Q_1\sigma[n_1], \ldots, Q_{i-1}\sigma[n_{i-1}], \\ & G_1\sigma[p_1], \ldots, G_k\sigma[p_k], Q_i\sigma[\mathit{proved}], \\ & Q_{i+1}\sigma[n_{i+1}], \ldots, Q_m\sigma[n_m]\end{aligned}$$

The query markings p_i are derived from t_i for all $i \in \{1, \ldots, k\}$ as follows:

$$\begin{aligned}&\text{if } t_i = \mathit{assumable}(f), \text{ then } p_i = \mathit{unsolved}(f) \\ &\text{if } t_i = \mathit{assertion}, \quad \text{then } p_i = \mathit{asserted}\end{aligned}$$

– TF-ASSUME (assumption): If $Q \in \text{dom}(f)$ such that there is a most general unifier $\sigma = \text{unify}(Q, Q_i)$, add a new state Π' to result of the transformation:

$$\Pi' = Q_1\sigma[n_1], \ldots, Q_i\sigma[\mathit{assumed}(f)], \ldots, Q_n\sigma[n_m]$$

Note that the proof procedure along with the definition of assumability functions ensures that the cost of the proofs are monotonic with respect to unification and rule application, allowing for the use of efficient search strategies.

Knowledge Gaps and Assertions. Our extension of the "classical" logic-programming-based weighted abduction as proposed by Stickel and Hobbs lies in the extension of the proof procedure with the notion of *assertion* based on the work in continual automated planning [8], allowing the system to reason about information not present in the knowledge base, thereby addressing the need for reasoning under the open-world assumption.

In continual automated planning, assertions allow a planner to reason about information that is not known at the time of planning (for instance, planning for information gathering), an assertion is a construct specifying a "promise" that the information in question will be resolved eventually. Such a statement requires planning to be a step in a continual loop of interleaved planning and acting.

By using a logic programming approach, we can use unbound variables in the asserted facts in order to reason not only about the fact that the given assertion will be a fact, but also under-specify its eventual arguments.

Algorithm 1 Weighted abduction

ABDUCE($c = (\mathcal{F}, \mathcal{R}, \mathcal{D}, \mathcal{S}), \Pi = Q_1[n_1], \ldots, Q_m[n_m]$):
  if IS-DISJOINT-VALID($\mathcal{D}, \Pi$) **then**
    $R \leftarrow \emptyset$
    for all $\Pi' \in$ TF-DUP(Π) **do**
      if Π' contains a query marked as *unsolved* **then**
        $H \leftarrow$ TF-FACT($\mathcal{F}, \Pi'$) $\cup$ TF-RULE($\mathcal{R}, \Pi'$) $\cup$ TF-ASSUME($\mathcal{S}, \Pi'$)
        $R \leftarrow R \cup \bigcup_{P \in H}$ ABDUCE(c, P)
      else
        $R \leftarrow R \cup \{\Pi'\}$
      end if
    end for
    return R
  else
    return $\emptyset$
  end if

IS-DISJOINT-VALID($\mathcal{D}, \Pi = Q_1[n_1], \ldots, Q_m[n_m]$):
  for all $d = \text{disjoint}(D_1, \ldots, D_q) \in \mathcal{D}$ **do**
    if $\exists i \neq j \neq k \neq l$ s.t. $\exists \sigma, \sigma'$: $\sigma = \text{unify}(D_i, Q_k)$ and $\sigma' = \text{unify}(D_j, Q_l)$ **then**
      return false
    end if
  end for
  return true

TF-DUP($\Pi = Q_1[n_1], \ldots, Q_m[n_m]$):
  if Π contains a query marked as *unsolved* **then**
    $i \leftarrow \arg\min_{j \in \{1,\ldots,m-1\}}(\exists f$ s.t. $n_j = \textit{unsolved}(f))$
    $H \leftarrow \emptyset$
    for all $s \in \{i+1, \ldots, m\}$ s.t. $\text{unify}(Q_i, Q_s) = \sigma$ **do**
      $H \leftarrow H \cup$ TF-DUP($Q_1\sigma[n_1], \ldots, Q_{i-1}\sigma[n_{i-1}], Q_{i+1}\sigma[n_{i+1}] \ldots, Q_m\sigma[n_m]$)
    end for
    if $H \neq \emptyset$ **then return** H **else return** $\{\Pi\}$ **end if**
  else
    return $\{\Pi\}$
  end if

The proposed notion of *assertion* for our abductive system is based on *test actions* $\langle F \rangle$ [4]. Baldoni et al. specify a test as a proof rule. In this rule, a goal F follows from a state $a_1, ..., a_n$ after steps $\langle F \rangle, p_1, ..., p_m$ if we can establish F on $a_1, ..., a_n$ with answer σ and this (also) holds in the final state resulting from executing $p_1, ..., p_m$.

An assertion is the transformation of a test into a partial proof which assumes the verification of the test, while at the same time conditioning the obtainability of the proof goal on the tested statements. $\mu : \langle D \rangle$ within a proof $\Pi[\langle D \rangle]$ to a goal C turns into $\Pi[D] \rightarrow C \wedge \mu : D$. Should $\mu : D$ not be verifiable, Π is invalidated.

Probabilistic Interpretation. In weighted abduction, weights assigned to assumed queries are used to calculate the overall proof cost. The proof with the lowest cost is the best explanation. However, weights are usually not assigned any semantics, and often a significant effort by the writer of the rule set is required to achieve expected results [13].

However, Charniak and Shimony [9] showed that by setting weights to $-\log$ of the prior probability of the query, the resulting proofs can be given probabilistic semantics.

Suppose that query Q_k can be assumed true with some probability $P(Q_k \text{ is true})$. Then if Q_k is assumable under assumability function f such that $f(Q_k) = -\log(P(Q_k \text{ is true}))$, and under the independence assumption, we can represent the overall probability of the proof $\Pi = Q_1[n_1], ..., Q_n[n_m]$ as

$$P(\Pi) = e^{\sum_{k=1}^{m} cost(Q_k)}$$

where

$$cost(Q_k) = \begin{cases} f(Q_k) \text{ if } n_k = assumed(f) \\ 0 \qquad \text{otherwise} \end{cases}$$

The best explanation Π_{best} of a the goal state G is then

$$\Pi_{best} = \underset{\Pi \text{ proof of } G}{\arg\min} P(\Pi)$$

Exact inference in such a system is NP-complete, and so is approximate inference given a threshold [9]. However, it is straightforward to give an anytime version of the algorithm – simply by performing iterative deepening depth-first search [19] and memorising a list of most probable proofs found so far.

Comparison with Other Approaches. Our system is similar to Poole's Probabilistic Horn abduction [18]. The main difference, apart from the proof procedure which is cost-based in our case, is that we do not include probabilities in our formulation of disjoint declarations. Since we avoid duplicate assumptions, we are able to model the semantics of disjoint declarations with probabilities.

On the other hand, having a general disjoint declaration allows us to define exclusivity rules such as

Algorithm 2 (Nondeterministic) continual abduction

```
CONTINUAL-ABDUCTION(c, Π):
  c = context
  Π = proof

  while Π contains assertion A do
    c' ← TEST-ACTION(c, A)
    H ← ABDUCE(c', A)
    for all Π' ∈ H do
      CONTINUAL-ABDUCTION(c', Π')
    end for
  end while
  return Π
```

$$\text{disjoint}([\text{p}(X, \text{yes}), \text{p}(X, \text{no})])$$

without having to specify the prior probabilities of the disjuncts.

Moreover, in our rule sets for natural language understanding and generation, we need to be able to manipulate logical structure (e.g. logical forms of utterances) efficiently. We have found that the logic-programming-based approach is quite satisfactory in this aspect, since it permits the use of standard Prolog programming techniques. In approaches to probabilistic abduction that are *not* based on logic programming, such as Kate and Mooney's abduction in Markov Logic Networks [15], these techniques are not applicable, which crucially limits their applicability to our domain.

3.2 Generating Partial Hypotheses

For each goal G, a the function ABDUCE returns a set of proofs H, with a total ordering on this set. Due to the use of assertions, some of these proofs may be partial, and their validity has to be verified. The presence of assertions in the proofs means that there is a knowledge gap, namely the truth value of the assertion. Each assertion thus specifies the need for performing a (test) action. This action might require the access to other knowledge bases than the abductive context, as in the case of resolving referring expressions, or an execution of a physical action.

Formally, given an initial goal G and context c, the abduction procedure produces a set H of hypotheses $c : \Pi \rightarrow C \wedge c_i : A_i$, where c_i is a sub-context in which where an assertion $A_i \in \Pi$ may be evaluated. Such proofs are thus both *partial* and *defeasible* — they may be both extended and discarded, depending on the evaluation of the assertions.

The set of possible hypotheses is continuously expanded until the best full proof is found. This process is defined in Algorithm 2.

The algorithm defines the search space in which it is possible to find the most probable proof of the initial goal G. The important point is, however, that

it is just that — a definition. The actual implementation may keep track of the partial hypotheses it defines, and take the appropriate test actions when necessary, or postpone them indefinitely. The cost of *performing* an action is not factored into the overall proof cost.

The partial hypotheses therefore serve as an interface layer between the language understanding and external decision-making processes (such as planning in a robotic architecture).

3.3 Representing Linguistic Meaning

For representing linguistic meaning in our system we use the *Hybrid Logic Dependency Semantics* (HLDS), a hybrid logic framework that provides the means for encoding a wide range of semantic information, including dependency relations between heads and dependents [21], tense and aspect [17], spatio-temporal structure, contextual reference, and information structure [16].

Hybrid Logic. Classical modal logic suffers from a surprising "asymmetry". Although the concept of states ("worlds") is at the heart of model theory, there is no way to directly reference specific states in the object language. This asymmetry is at the root of several theoretical and practical problems facing modal logic [6,1].

Hybrid logic provides an elegant solution to many of these problems. It extends standard modal logic with *nominals*, another sort of basic formulas that explicitly name worlds in the object language. Next to propositions, nominals—and, by extension, possible worlds—therefore become first-class citizens in the object language. The resulting logical framework retains decidability and favourable complexity [2].

Each nominal names a unique state. To get to that state, a new operator is added, the *satisfaction operator*. The satisfaction operator that enables us to "jump" to the state named by a nominal. The satisfaction operator is written $@_i$, where i is a nominal.

Formally, let $\textsc{Prop} = \{p, q, ...\}$ be a set of propositional symbols, $\textsc{Mod} = \{\pi, \pi', ...\}$ a set of modality labels, and $\textsc{Nom} = \{i, j, ...\}$ a non-empty set disjoint from Prop and Mod. We define the well-formed formulas of the basic hybrid multimodal language $\mathcal{L}_@$ over Prop, Mod and Nom as such:

$$\phi ::= i \mid p \mid \neg\phi \mid \phi \rightarrow \varphi \mid \langle\pi\rangle\phi \mid [\pi]\phi \mid @_i\phi$$

A formula $@_i\phi$ states that the formula ϕ holds at the unique state named by i. In more operational terms, the formula $@_i\phi$ could be translated in the following way: "go to the (unique!) state named by i, and check whether ϕ is true at that state".

Hybrid Logic Dependency Semantics. HLDS uses hybrid logic to capture dependency complexity in a model-theoretic relational structure, using ontological sorting to capture categorial aspects of linguistic meaning, and naturally

capture (co-)reference by explicitly using *nominals* in the representation. The dependency structures can be derived from CCG [5], which is the setup used in our system, but other approaches are possible.

Generally speaking, HLDS represents an expression's linguistic meaning as a conjunction of modalised terms, anchored by the nominal that identifies the head's proposition:

$$@_{h:\mathsf{sort}_h}\ (\mathbf{prop}_h \wedge \langle \mathsf{R}_i \rangle\ (d_i : \mathsf{sort}_{d_i} \wedge \mathbf{dep}_i))$$

Here, the head proposition nominal is h. $\mathbf{prop}_h$ represents the *elementary predication* of the nominal h. The dependency relations (such as Agent, Patient, Subject, etc.) are modelled as modal relations $\langle R_i \rangle$, with the dependent being identified by a nominal d_i. Features attached to a nominal (e.g. $\langle$Num$\rangle$ $\langle$Quantification$\rangle$, etc.) are specified in the same way.

Figure 1 gives an example of HLDS representation (logical form) of the sentence "Bring me the mug from the kitchen". The logical form has six nominals, $event_1$, $agent_1$, $person_1$, $thing_1$, $from_1$ and $thing_1$, that form a dependency structure: $event_1$ is the the head of dependency relations Actor (the dependent being $agent_1$), Patient ($thing_1$), Recipient ($person_1$), Modifier ($from_1$), and Subject (the sentence subject, $agent_1$).

Each nominal has an ontological sort (illustrated on $event_1$, the sort is action-non-motion) a proposition (**bring**), and may have features (Mood).

$$
\begin{aligned}
&@_{event_1:\mathsf{action\text{-}non\text{-}motion}}(\mathbf{bring} \wedge \\
&\quad \langle \mathsf{Mood} \rangle\ \mathbf{imp} \wedge \\
&\quad \langle \mathsf{Actor} \rangle\ (agent_1 : \mathsf{entity} \wedge \mathbf{addressee}) \\
&\quad \langle \mathsf{Patient} \rangle\ (thing_1 : \mathsf{thing} \wedge \mathbf{mug} \wedge \\
&\qquad \langle \mathsf{Delimitation} \rangle\ \mathbf{unique} \wedge \\
&\qquad \langle \mathsf{Num} \rangle\ \mathbf{sg} \wedge \\
&\qquad \langle \mathsf{Quantification} \rangle\ \mathbf{specific}) \wedge \\
&\quad \langle \mathsf{Recipient} \rangle\ (person_1 : \mathsf{person} \wedge \mathbf{I} \wedge \\
&\qquad \langle \mathsf{Num} \rangle\ \mathbf{sg}) \wedge \\
&\quad \langle \mathsf{Modifier} \rangle\ (from_1 : \mathsf{m\text{-}wherefrom} \wedge \mathbf{from} \wedge \\
&\qquad \langle \mathsf{Anchor} \rangle\ (place_1 : \mathsf{e\text{-}place} \wedge \mathbf{kitchen} \wedge \\
&\qquad\quad \langle \mathsf{Delimitation} \rangle\ \mathbf{unique} \wedge \\
&\qquad\quad \langle \mathsf{Num} \rangle\ \mathbf{sg} \wedge \\
&\qquad\quad \langle \mathsf{Quantification} \rangle\ \mathbf{specific})) \wedge \\
&\quad \langle \mathsf{Subject} \rangle\ agent_1 : \mathsf{entity})
\end{aligned}
$$

Fig. 1. HLDS semantics for the utterance "Bring me the mug from the kitchen"

Every logical form in HLDS, being a formula in hybrid logic, can be decomposed into a set of facts in the abductive context corresponding to its minimal Kripke model. The resulting set of abduction facts obtained by decomposing the logical form in Figure 1 is shown by Figure 3.

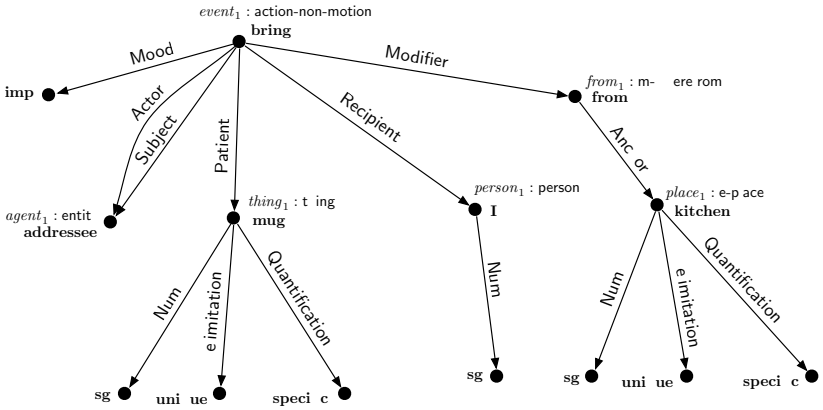

Fig. 2. Minimal model for the hybrid logic formula in Figure 1

HLDS only represents the meaning as derived from the linguistic realisation of the utterance and does not evaluate the state of affairs denoted by it. This sets the framework apart from semantic formalisms such as DRT [14]. The grounding in reality is partly provided by the continual abductive framework by generating and validating (or ruling out) partial abductive hypotheses as more information is added to the system.

4 Example

Let us examine the mechanism in an example introduced in §2.3.

The human's utterance,

"Bring me the mug from the kitchen."

is analysed in terms of HLDS (see Figure 1), and its translation (see Figure 3) is made part of the abduction context c.

The robot tries to make sense of the utterance by proving the goal

$$\text{uttered}(\text{human}, \text{robot}, \text{event}_1)$$

in the abduction context c. Suppose that the best proof state returned by ABDUCE is the following:

$\text{sort}(\text{event}_1, \text{action-non-motion}),$
$\text{prop}(\text{event}_1, \text{bring}),$
$\text{feat}(\text{event}_1, \text{mood}, \text{imp}),$
$\text{rel}(\text{event}_1, \text{actor}, \text{agent}_1),$
$\text{sort}(\text{agent}_1, \text{entity}),$
$\text{prop}(\text{agent}_1, \text{addressee}),$
$\text{rel}(\text{event}_1, \text{patient}, \text{thing}_1),$
$\text{sort}(\text{thing}_1, \text{thing}),$
$\text{prop}(\text{thing}_1, \text{mug}),$
$\text{feat}(\text{thing}_1, \text{delimitation}, \text{unique}),$
$\text{feat}(\text{thing}_1, \text{num}, \text{sg}),$
$\text{feat}(\text{thing}_1, \text{quantification}, \text{specific}),$
$\text{rel}(\text{event}_1, \text{recipient}, \text{person}_1),$
$\text{sort}(\text{person}_1, \text{person}),$
$\text{prop}(\text{person}_1, \text{i}),$
$\text{feat}(\text{person}_1, \text{num}, \text{sg}),$
$\text{rel}(\text{event}_1, \text{modifier}, \text{from}_1),$
$\text{sort}(\text{from}_1, \text{m-wherefrom}),$
$\text{prop}(\text{from}_1, \text{from}),$
$\text{rel}(\text{from}_1, \text{anchor}, \text{place}_1),$
$\text{sort}(\text{place}_1, \text{e-place}),$
$\text{prop}(\text{place}_1, \text{kitchen}),$
$\text{feat}(\text{place}_1, \text{delimitation}, \text{unique}),$
$\text{feat}(\text{place}_1, \text{num}, \text{sg}),$
$\text{feat}(\text{place}_1, \text{quantification}, \text{specific}),$
$\text{rel}(\text{event}_1, \text{subject}, \text{agent}_1)$

Fig. 3. The translation of the hybrid logic formula in Figure 1 into abduction facts

$\text{uttered}(\text{human}, \text{robot}, \text{event}_1)$ [*proved*]	(1)
$\text{prop}(\text{event}_1, \text{bring})$ [*proved*]	(2)
$\text{intends}(\text{event}_1, \text{human}, I)$ [*assumed*(*engagement*)]	(3)
$\text{rel}(\text{event}_1, \text{patient}, \text{thing}_1)$ [*proved*]	(4)
$\text{refers-to}(\text{thing}_1, X)$ [*asserted*]	(5)
$\text{refers-to}(\text{place}_1, P)$ [*asserted*]	(6)
$\text{pre}(I, \text{object}(X))$ [*asserted*]	(7)
$\text{pre}(I, \text{is-in}(X, P))$ [*asserted*]	(8)
$\text{refers-to}(\text{person}_1, \text{human})$ [*proved*]	(9)
$\text{prop}(\text{person}_1, i)$ [*proved*]	(10)
$\text{rel}(\text{event}_1, \text{recipient}, \text{person}_1)$ [*proved*]	(11)
$\text{post}(I, \text{has}(\text{human}, X))$ [*proved*]	(12)

The proof is an explanation of the event (1) in terms of a partially specified intention I (3), defined by its pre- and post-conditions. The pre-conditions are the existence of an entity X (7) and that X is located *in* another entity P (8). The post-condition (12) is the resulting state in which the human has X (12).

The proof appeals to the logical form of the utterance in atoms (2), (4), (10), (11). Also, atom (9) is proved from (1) and (10) (whoever uses "I" refers to themselves), and (12) is a consequence of (2), (9) and (11) (bringing x to a person p means ending up in a state in which p has x).

Atom (3) is assumed under the assumability function *engagement*, which is supplied in the abduction context before calling ABDUCE and specifies the robot's subjective probability of being engaged in a conversation with the particular human at the time the utterance was observed.

Note that the proof state contains four atoms marked as assertions: (5), (6), (7) and (8). These are the explicit gaps in the proof that make it a partial interpretation. They are chosen by the domain engineer, and since they need to be verified (or falsified) by an external process, they form the interface to external knowledge bases and decision-making. Since for now those atoms are marked as asserted, there is nothing more that ABDUCE can do.

The initiative then shifts to an external decision-making process. It selects some of the assertions, and tries to verify them.

A sensible strategy[1] might be to first establish the referent of $place_1$. This could be resolved against the internal knowledge base (in case the robot has been given a tour of the household), or it could trigger the exploration behaviour – in order to resolve $place_1$, the robot could try finding it first. Again, choosing which behaviour is more appropriate depends on the application, and on the planning method that is invoked by the decision-maker in order to verify the assertion.

Once the assertion is verified, the proof is updated accordingly, in our case by replacing all occurrences by replacing the unbound variable P by a unique symbol, for instance by the identifier $id_{kitchen}$ of the topological region in the robot's topological map:

$$\frac{\text{refers-to}(\text{place}_1, id_{kitchen})\ [proved] \quad (6')}{\text{resolves-to-toporegion}(\text{place}_1, id_{kitchen})\ [assumed(topo)]\ (6'')}$$

The atom (6) in the original proof state is expanded by a proof state consisting of queries (6') and (6"), thereby replacing P in the entire proof by $id_{kitchen}$, and adding the cost of assuming (6") to the overall proof cost. This atom is assumed under an assumability function *topo*, supplied as part of the abduction context in which the proof is expanded – i.e. by the external knowledge source. An assumption is added instead of a fact so that the external knowledge base performing this operation can express uncertainty about the resolution result.

The proof is therefore expanded into the following:

[1] Note that the problem of what determing good verification strategies and choosing them is beyond the scope of this paper.

uttered(human, robot, event_1)	[*proved*]	(1)
prop(event_1, bring)	[*proved*]	(2)
intends(event_1, human, I)	[*assumed*(*engagement*)]	(3)
rel(event_1, patient, thing_1)	[*proved*]	(4)
refers-to(thing_1, X)	[*asserted*]	(5)
refers-to(place_1, $id_{kitchen}$)	[*proved*]	(6')
resolves-to-toporegion(place_1, $id_{kitchen}$)	[*assumed*(*topo*)]	(6")
pre(I, object(X))	[*asserted*]	(7)
pre(I, is-in(X, $id_{kitchen}$))	[*asserted*]	(8)
refers-to(person_1, human)	[*proved*]	(9)
prop(person_1, i)	[*proved*]	(10)
rel(event_1, recipient, person_1)	[*proved*]	(11)
post(I, has(human, X))	[*proved*]	(12)

Now there are just three assertions left: (5), (7) and (8). These express the knowledge gaps about the referent of "the mug", the existence of the object, and its location, respectively.

There are, as before, several possible ways of verifying these. The most sensible one would probably be going to the kitchen (i.e. the topological region $id_{kitchen}$) and searching for objects there, which would verify both (8) and (7) and expand them with all objects it finds. There would be many parallel proof states resulting from such an expansion, and the robot would have to prune them down by verifying the remaining assertion (5).

One way of doing that would be to bring all objects one by one to the human, asking "did you mean this one?" Alternatively, the robot might simply bring the most likely object. The human's acceptance of the choice would then verify the assertion. This is, again, a matter for consideration in the higher level of planning and goal management.

5 Conclusion

This paper presents an abductive framework for natural language understanding that is based on abductive reasoning over partial hypotheses. The framework is set within the process of intention recognition.

The abductive framework is contextually-enhanced version of a logic programming approach to weighted abduction with a probabilistic semantics assigned to the weights. Our extension of this framework is in the introduction of the notion of *assertion*, which is essentially a requirement for a future test to verify or falsify the proposition, i.e. to fill a knowledge gap about the validity of the proposition. The hypotheses are therefore defeasible in the sense that the falsification of their assertions leads to a retraction and adoption of an initially less likely alternative.

By explicitly reasoning about these knowledge gaps, the system is able to go beyond the current context and knowledge base, addressing the need for reasoning under the open-world assumption. The responsibility for filling those knowledge gaps then falls to external decision-making processes. These processes can

then use probabilities to express their confidence in the solutions they provide, thereby addressing the need for capturing the ubiquitous uncertainty stemming from unreliable sensory perception and partial observability of the world.

Acknowledgments. The work reported in this paper has been supported by the EU ICT FP7 projects CogX, "Cognitive Systems that Self-Understand and Self-Extend" (project no. 215181) and NIFTi, "Natural Human-Robot Collaboration in Dynamic Environments" (project no. 247870).

The author would like to thank Geert-Jan Kruijff for insightful and inspiring discussions, and the anonymous reviewers for their detailed and very helpful comments.

References

1. Areces, C.: Logic Engineering: The Case of Description and Hybrid Logics. Ph.D. thesis, University of Amsterdam, the Netherlands (2000)
2. Areces, C., Blackburn, P.: Bringing them all together. Journal of Logic and Computation 11(5), 657–669 (2001)
3. Baldoni, M., Giordano, L., Martelli, A.: A modal extension of logic programming: Modularity, beliefs and hypothetical reasoning. J. Log. Comp. 8(5), 597–635 (1998)
4. Baldoni, M., Giordano, L., Martelli, A., Patti, V.: A modal programming language for representing complex actions. In: Proc. of DYNAMICS'98. pp. 1–15 (1998)
5. Baldridge, J., Kruijff, G.J.M.: Coupling CCG and Hybrid Logic Dependency Semantics. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL) (2002)
6. Blackburn, P.: Representation, reasoning, and relational structures: a hybrid logic manifesto. Logic Journal of the IGPL 8(3), 339–625 (2000)
7. Bratman, M.: Intentions, Plans, and Practical Reason. Harvard University Press, Cambridge, MA, USA (1987)
8. Brenner, M., Nebel, B.: Continual planning and acting in dynamic multiagent environments. Journal of Autonomous Agents and Multiagent Systems (2008)
9. Charniak, E., Shimony, S.E.: Probabilistic semantics for cost based abduction. In: AAAI-90 proceedings (1990)
10. Cohen, P.R., Levesque, H.J.: Intention is choice with commitment. Artificial Intelligence 42, 213–261 (March 1990)
11. Fann, K.T.: Peirce's Theory of Abduction. Mouton, The Hague, The Netherlands (1970)
12. Grice, H.P.: Meaning. The Philosophical Review 66(3), 377–388 (1957)
13. Hobbs, J.R., Stickel, M.E., Appelt, D.E., Martin, P.A.: Interpretation as abduction. Artificial Intelligence 63(1–2), 69–142 (1993)
14. Kamp, H., Reyle, U.: From Discourse to Logic. Kluwer, Dordrecht, The Netherlands (1993)
15. Kate, R.J., Mooney, R.J.: Probabilistic abduction using Markov logic networks. In: Proceedings of the IJCAI-09 Workshop on Plan, Activity, and Intent Recognition (PAIR-09). Pasadena, CA, USA (2009)
16. Kruijff, G.J.M.: A Categorial-Modal Logical Architecture of Informativity: Dependency Grammar Logic & Information Structure. Ph.D. thesis, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic (2001)

17. Moens, M., Steedman, M.: Temporal ontology and temporal reference. Computational Linguistics 14(2), 15–28 (1988)
18. Poole, D.: Probabilistic Horn abduction and Bayesian networks. Artificial Intelligence 64(1), 81–129 (1993)
19. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River, NJ, USA, second edn. (2003)
20. Schiffer, S.R.: Meaning. Oxford, Clarendon Press (1972)
21. Sgall, P., Hajičová, E., Panevová, J.: The Meaning of the Sentence in Its Semantic and Pragmatic Aspects. Reidel Publishing Company, Dordrecht, The Netherlands and Academia, Prague, Czechoslovakia (1986)
22. Stickel, M.E.: A Prolog-like inference system for computing minimum-cost abductive explanations in natural-language interpretation. Annals of Mathematics and Artificial Intelligence 4, 89–105 (1991)
23. Stone, M., Thomason, R.H.: Context in abductive interpretation. In: Proceedings of EDILOG 2002: 6th workshop on the semantics and pragmatics of dialogue (2002)
24. Stone, M., Thomason, R.H.: Coordinating understanding and generation in an abductive approach to interpretation. In: Proceedings of DIABRUCK 2003: 7th workshop on the semantics and pragmatics of dialogue (2003)

There Is No Common Ground In Human-Robot Interaction

Geert-Jan M. Kruijff
DFKI GmbH, Saabrücken Germany

***Abstract*—There is an inherent asymmetry to situated communication. Those communicating look at the world from different perspectives. This holds particularly true for human-robot interaction. Robots literally see the world differently: They experience reality in fundamentally different ways. This raises an important issue for how to model common ground between a human and a robot. Traditional approaches to common ground adopt a possible worlds-like model theory, on which a shared belief represents the fusion of two private beliefs within a single model. This silently assumes a symmetry in categories against which both private beliefs can be interpreted: A symmetry which cannot be assumed for human-robot interaction. Hence, on such a model, there is no common ground between humans and robots. This paper presents an alternative model theory, which captures the inherently subjective nature of experience. It is based on a notion of propositions-as-proofs, turning subjective interpretation as well as the projection to intersubjective verification into a notion of inference or argumentation. Beliefs are arguments, whether private, attributed or shared. This results in a formulation of common ground as a dynamic structure of always argued, but possibly only partially confirmed or partially assumed beliefs.**

I. Introduction

Communication is about meaning. You build up meaning for what you hear, you construct meaning for what you intend to convey. And in those processes of constructing meaning, you appeal to different sources of information. In situated dialogue, these sources at least encompass the situation being described (*focus situation*); any other physical or discursive contexts, or common knowledge (*resource situations*); and the ways in which the communication partners take part in the dialogue (*social situation*) [14]. Figure 1 provides an example.

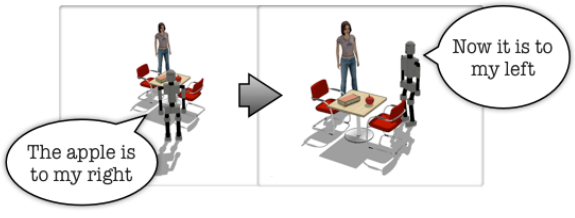

Fig. 1. Different situations in meaning: "Now it is to my right" appeals in contrast to the previous situation ("now"), preceding dialogue ("it"), the current situation ("is"), and the current social situation ("to my right").

The social situation in situated dialogue makes it clear: Communication partners look at the world from different perspectives. They see the world differently. Some of the effort in communicating therefore goes into establishing a common ground between partners. The point is to establish a mutual understanding of what is being talked about, what is appealed to notably in reference to the world [11], [10]. This is a dynamic process, in which partners coordinate and align their beliefs [43], [35].

This need for common ground is not restricted to communication between humans. It holds just as much for human-robot interaction [24], [25], [23]. Understanding and facilitating different perspectives on the world is crucial to establishing an effective collaborative context [38], [40], [39], [32]. Failure to do so typically leads to a breakdown in communication, see e.g. [39], [36].

Yet for all its importance, common ground is typically modeled on a possible worlds-like model theory. And if perspective is modeled explicitly at all as part of common ground, this remains limited to transformations on reference frames. This is inadequate, for a fundamental reason. It disregards the fact that there is an inherent asymmetry between how humans and robots experience reality. This is more than just perspective, or a matter of experience being subjective. Humans and robots categorize experience differently, given sensory input. They individuate that experience differently, to structure it and identify individual referents. And as a result, they construct propositions differently. *We all know that.* But a possible worlds-like model theory does not represent that. On such a theory, a shared belief (proposition) represents the fusion of two private beliefs (propositions) *within* a single model. This silently assumes a symmetry in categories against which both private beliefs (propositions) can be interpreted: A symmetry which cannot be assumed for humans and robots. Hence, on such a model, there is no common ground between humans and robots.

The issue here is not possible worlds, or whether we use static or dynamic models. The point is that we lack a model theory to guide inference for human-robot communication, to direct strategies for a robot to facilitate and adapt to humans, starting from the viewpoint that a human and a robot not only need to coordinate beliefs, they need to coordinate "down to" the propositions they state for their experience (or attribute to others, for their experience). We lack a model theory which is proper in the sense of taking into account the inherent asymmetry in subjective experience, in categorizing experience, and the dynamics of how humans and robots could (even) establish and maintain some form of common ground. All of that means going back to the question of what class of propositions such a model helps describe, and how we can construct a notion of individual on that.

It is such a model theory that this paper provides. The

paper considers the construction of a class of propositions about properties of subjective experience. It follows out the intuitionistic idea of propositions as sets of proofs for the proposition [28], [33], and makes the connection between such a constructivist view on propositions, and the relation between (intentional) phenomenology and logic (Husserl, Peirce, Gödel). It provides an algebraic model theory on which we can consider incompleteness and uncertainty in (models of) experience (i.e. "gaps," [44]). The theory enables non-trivial extensions to propositional models of experience to overcome incompleteness, using a combination of informative approximation [12], axiomatic extension [15], and universe construction [21].

The paper reconstrues both common ground and alignment of perspectives as projections between universes. This intuitively to the idea that an agent can set up an argument for how another might experience a situation, to the extent that this agent can "imagine" this given his own (categorical) knowledge. Some or all of that argumentation might of course be wrong, incomplete, based on assumptions which turn out not to be warranted. But that is precisely the point. The paper starts from subjectivity, assumes there is an asymmetry which almost inevitably gives rise to misunderstanding. The question then becomes how intersubjective understanding (non-misunderstanding) can dynamically arise, for example by filling in the gaps (literally). And with that, the paper hits on perhaps one of the most principal issues in Artificial Intelligence: How an embodied, artificially intelligent agent like a robot can be truly self-aware of its experience, reason with it with respect to itself and to others, and increase its understanding of the world around it.

II. Overview

Common ground is a complex notion. Generally speaking, it is about something *between* communicative partners: Namely, "shared understanding." Minimally, this is a shared understanding of the entities which have been talked about, grounded in the (possibly situated) domain of discourse [11], [10], [43]. After entities have been introduced into common ground, they are accessible for reference. Naturally, there can be more to common ground. In task-oriented domains, common ground provides an interface between the task domain itself, and the communication which mediates and coordinates collaboration in that domain. Typically, this then leads to considering intentions, plans, and tasks to be part of common ground as well; See e.g. [1], [16], [2], [36], [18], [6], [17], [13], [27]. And. On top of which we can essentially consider all that is implied by the concept of situated meaning [14], illustrated e.g. in Figure 1. (And that would include "common sense knowledge," e.g. what if the robot was discussing not an edible apple, but a Mac?) As said, common ground is a complex notion.

Yet, what is in some sense surprising is that the formal underpinnings of practically all computational approaches can be traced back to a system in which one relies on a classical-logical interpretation of the notion of proposition. There *are* different notions of what a proposition is. It all depends on what domain is to be modeled. This can range from mathematical structure (e.g. the intuitionistic notion of proposition-as-sets-of-proofs), to classical logical reasoning (the proposition as truth-statement about structures in reality) and dynamic logical reasoning (the proposition as context-update potential [41], [36]), to experiential observations (e.g. the quantum logic-notion of proposition as experimental statement [3]).

What we are trying to model here is this: How does a robot get from its sensory experience to a proposition? And how would a robot be able to reason about subjective experience of its own, and that of others, to infer how to align these? As a problem statement this goes back to phenomenology, notably to F. Brentano, and later his student, E. Husserl [7]. They argued for a directedness behind experience: There is an intentionality driving the construction of a "symbol" from experience. C.S. Peirce considered this as a sign-process, in which an (intentionally constructed) interpretant mediates between an experiential sign, and the objective aspects of reality this sign is taken to refer to [34]. This sign process combines an interpretation-in-context (the intentional interpretation of a sign under its given appearance), with interpretation against a categorical system (categories, rules, which guide the mediation). Put slightly differently, it combines ventral (categorical) and dorsal (spatiotemporal) forms of interpretation into the formation of an interpretation [9], [20], [19], or more abstractly, a symbol [4], [5].

There are only a few approaches to linking robot sensory experience and symbols which come even close to this idea of symbol formation. The most detailed account is provided by M. Shanahan; See e.g. [29], [30], [31]. The account is based on using abductive reasoning to infer an explanation for a given set of sensory inputs. This is a process of symbol formation. The explanation itself indicates how different low-level sensor data Γ can be drawn together, provided as a logical statement Δ against a theory Σ, such that $\Sigma \wedge \Delta \models \Gamma$. In this Shanahan addresses a problem with logic-based approaches to cognitive robotics [37]. There, this link is not made, and propositions are just considered to be grounded somehow. In Shanahan's abductive approach this is explicitly modeled, through successive layers of (abductive) abstraction; See also n-layered abduction in [22].

This paper presents an approach to providing a model theory on which an abductive approach like Shanahan's can be formulated; See also [27]. However. First of all, the paper focuses on the model theory. We deal with the structure on which logical inference is interpreted, and which thus defines what types of connections can be logically inferred in the first place. Secondly, we adopt the intuitionistic (or constructive) notion of proposition-as-set, and combine it with a notion of strong negation. The constructive notion has a close connection to the phenomenological idea of intentionality in experience. We model a proposition explicitly as the set of possible constructions, the proofs, which bring it about. It is against this constructive idea that we see the notion of negation. Negation itself is a construction, namely the construction that something is not justified or possible [42]. This is a strong notion, going back (in the intuitionistic framework) at least to Kolmogorov [26]. It differs from classical negation, which would stand for

not attempting the construction (which is not a construction by itself). Strong(er) notions of negation have also been deployed more recently in non-monotonic reasoning (e.g. default logic), stable model semantics, and extensions of intuitionistic logic which explicitly model the role of negation in inference by dealing with proofs, disproofs, and dual proofs [**?**].

Those are the essential ingredients. We cast the symbol formation process first of all as a process involving *individuation*, i.e. the construction of a "symbol" as standing for an individually identifiable aspect of reality (Peirce's Object) from sensor data. This symbol formation happens along the lines of categorical reference, and situated frame of reference, with the latter understood in terms of the different forms of situation in the ontology of situated meaning [14]. Particularly the inclusion of a situated frame of reference, in alignment with which the symbol formation process can appeal to specific categorical interpretations, makes symbol formation inherently perspectivized – and thus subjective. Individuation yields a set of constructions against the experiential data. Over these constructions, we can form a completion (under positive proofs and negative disproofs) over the individual, given a category system. Following Peirce (and Shanahan) one could phrase this as symbol formation involving the *abduction* of an individuation, and the *deduction* of a universe of all possible positive and negative consequences of that individuation. Algebraically, we formulate this as universe construction. We do this such that we create the possibility for non-monotonicity, i.e. retracting or changing consequences. From a constructive view, this implies altering proofs or disproofs on the basis of new evidence. Interestingly, coupled to a notion of universe construction this leads to a constructive notion of incompleteness, a "hierarchy" or expansion of gaps in an agents knowledge of an individual. It reveals what is known and can be known (knowability) without falling prey to Fitch's knowability paradox [8]. The construction of a universe (in fact, a stable model) based on individuation under perspective can be extended from one individual, to a structure over multiple individuals. Ultimately, this then provides the basis for the algebraic underpinnings of our notion of common ground. Common ground becomes the projection under "bi-simulation" of universes constructed for the individuated experiences of an individual agent, to universes for another's experiences (under possibly different perspectives). The bi-simulation here is weak, as it is conditional on incompleteness, and assertions. What it basically states is that, following from the agent's subjective experience, and its understanding of someone else's perspectives, it can project an alignment between its own universes (on which it can logically interpret beliefs), and those it can attempt to construct for the other. As the universes on both sides allow for explicit positive and negative constructions, this means we build in the sources for misunderstanding from the very beginning. But it also means that, given external information that is contrary to a positive or negative construction, an explicit trace can be established for how to revise one or more universes to re-align.

Acknowledgments

Research reported of in this paper were partly supported by the EU FP7 IST project "CogX" (Cognitive Systems, #215181).

REFERENCES

[1] J. F. Allen and C. R. Perrault, 'Analyzing intention in utterances', *Artificial Intelligence*, **15**(3), (1980).

[2] J.F. Allen and D.J. Litman, 'Discourse processing and common sense plans', in *Intentions and Communication*, eds., P.R. Cohen, J. Morgan, and M. Pollack, The MIT Press, (1990).

[3] G. Bacciagaluppi, 'Is logic empirical?', in *Handbook of Quantum Logic and Quantum Structures*, eds., K. Engesser, D.M. Gabbay, and D. Lehmann, volume Quantum Logic, 49–78, Elsevier, (2009).

[4] L.W. Barsalou, 'Perceptual symbol systems', *Behavioral & Brain Sciences*, **22**, 577–660, (1999).

[5] L.W. Barsalou, 'Abstraction in perceptual symbol systems', *Philosophical Transactions of the Royal Society of London: Biological Sciences*, **358**, 1177–1187, (2003).

[6] N. Blaylock and J. Allen, 'A collaborative problem-solving model of dialogue', in *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, eds., L. Dybkjr and W. Minker, pp. 200–211, (2005).

[7] F. Brentano, *Psychology from an empirical standpoint*, Routledge, London, 1874/1995. Translated by A. C. Rancurello, D. B. Terrell, and L. L. McAlister.

[8] B. Brogaard and J. Salerno, 'Fitch's paradox of knowability', in *Stanford Encyclopedia of Philosophy*, Stanford University, (2009). Revision July 1 2009, accessed online at February 3 2012.

[9] G. Calvert, 'Crossmodal processing in the human brain: insights from functional neuroimaging studies', *Cerebral Cortex*, **11**, 1110–1123, (2001).

[10] H. H. Clark and S. A. Brennan, 'Grounding in communication', in *Perspectives on socially shared cognition*, eds., L.B. Resnick, J.M. Levine, and S.D. Teasley, 127–149, APA Books, Washington VA, (1991).

[11] H. H. Clark and C. R. Marshall, 'Reference diaries', in *Theoretical issues in natural language processing*, ed., D. L. Waltz, volume 2, 57–63, Association for Computing Machinery, New York NY, (1978).

[12] M. Denecker, V. Marek, and M. Truszczynski, 'Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning', in *Logic-based artificial intelligence*, ed., J. Minker, 127–144, Kluwer Academic Publishers, (2000).

[13] G. Ferguson and J. Allen, 'Mixed-initiative dialogue systems for collaborative problem-solving', *AI Magazine*, **28**(2), 23–32, (2007).

[14] J. Ginzburg, 'Situation semantics and the ontology of natural language', in *The Handbook of Semantics*, eds., P. Portner, C. Maierborn, and K. von Heusinger, 830–851, de Gruyter, (2011).

[15] K. Gödel, 'The modern development of the foundations of mathematics in the light of philosophy (*1961/?)', in *Kurt Gödel Collected Works, Volume III: Unpublished Essays and Lectures*, ed., S. Feferman, 364–387, Oxford University Press, (1995).

[16] B. J. Grosz and C. L. Sidner, 'Attention, intention and the structure of discourse', *Computational Linguistics*, **12**(3), 175–204, (1986).

[17] B.J. Grosz and L. Hunsberger, 'The dynamics of intention in collaborative activity', *Cognitive Systems Research*, **7**(2-3), 259–272, (2006).

[18] B.J. Grosz and S. Kraus, 'The evolution of sharedplans', in *Foundations and Theories of Rational Agency*, eds., A. Rao and M. Wooldridge, 227–262, Springer, (1999).

[19] G. Hickok, 'The functional neuroanatomy of language', *Physics of Life Reviews*, **6**, 121–143, (2009).

[20] G. Hickok and D. Poeppel, 'Dorsal and ventral streams: a framework for understanding aspects of the functional anatomy of language', *Cognition*, **4**(4), 67–99, (2004).

[21] G. Jäger, R. Kahle, and T. Studer, 'Universes in explicit mathematics', *Annals of Pure and Applied Logic*, **109**(3), 141–162, (2001).

[22] *Abductive Inference: Computation, Philosophy, Technology*, eds., J.R. Josephson and S.G. Josephson, Cambridge University Press, 1994.

[23] S. Kiesler, 'Fostering common ground in human-robot interaction', in *Proceedings of 14th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN 2005)*, (2005).

[24] G. Klein, P.J. Feltovich, J.M. Bradshaw, and D.D. Woods, 'Common ground and coordination in joint activity', in *Organizational Simulation*, eds., W.B. Rouse and K.R. Boff, 139–184, John Wiley, New York City, NY, (2004).

[25] G. Klein, D.D. Woods, J.M. Bradshaw, R. Hoffman, and P. Feltovich, 'Ten challenges for making automation a team player in joint human-agent activity', *IEEE Intelligent Systems*, **19**(6), 91–95, (November-December 2004).

[26] A.N. Kolmogorov, 'Zur Deutung der intuitionistischen Logik', *Mathematische Zeitschrift*, **35**, 58–65, (1932).

[27] G.J.M. Kruijff, M. Janíček, and P. Lison, 'Continual processing of situated dialogue in human-robot collaborative activities', in *Proceedings of the 19th International Symposium on Robot and Human Interactive Communication (RO-MAN 2010)*, IEEE, (2010).

[28] P. Martin-Löf, *Intuitionistic Type Theory*, Bibliopolis, Napels, Italy, 1984.

[29] M.P.Shanahan, 'Robotics and the common sense informatic situation', in *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI 96)*, pp. 684–688, (1996).

[30] M.P.Shanahan, 'A logical account of perception incorporating feedback and expectation', in *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR 2002)*, pp. 3–13, (2002).

[31] M.P.Shanahan, 'Perception as abduction: Turning sensor data into meaningful representation', *Cognitive Science*, **29**, 103–134, (2005).

[32] R.R. Murphy and J.L. Burke, 'The safe human-robot ratio', in *Human-Robot Interactions in Future Military Operations*, eds., M.J. Barnes and F. Jentsch, Human Factors in Defence, 31–49, Ashgate, (2010).

[33] B. Nordström, K. Petersson, and J.M. Smith, 'Martin-löf's type theory', in *Handbook of Logic in Computer Science*, eds., S. Abramsky, D.M. Gabbay, and T. S. E. Maibaum, volume 5: Algebraic and Logical Structures, Oxford University Press, (2001).

[34] C.S. Peirce, *Reasoning and the Logic of Things: The Cambridge Conferences Lectures of 1898*, Harvard University Press, 1992. K.L. Ketner, ed.

[35] M.J. Pickering and S. Garrod, 'Toward a mechanistic psychology of dialogue', *Behavioral and Brain Sciences*, **27**, 169–225, (2004).

[36] M. Poesio and D. Traum, 'Conversational actions and discourse situations', *Computational Intelligence*, **13**(3), 309–347, (1997).

[37] R. Reiter, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*, The MIT Press, 2001.

[38] D. Sofge, M. Bugajska, J. G. Trafton, D. Perzanowski, S. Thomas, M. Skubic, S. Blisard, N.L. Cassimatis, D. Brock, W. Adams, and A. Schultz, 'Collaborating with humanoid robots in space', *International Journal of Humanoid Robotics*, **2**(2), 181–201, (2005).

[39] K. Stubbs, P.J. Hinds, and D. Wettergreen, 'Autonomy and common ground in human-robot interaction: A field study', *IEEE Intelligent Systems*, **22**, 42–50, (March 2007). Special Issue on Interacting with Autonomy.

[40] J. G. Trafton, N. L. Cassimatis, M. D. Bugajska, D. P. Brock, F. E. Mintz, and A. C. Schultz, 'Enabling effective human-robot interaction using perspective-taking in robots', *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, **35**(4), 460–470, (2005).

[41] J.F.A.K. van Benthem, *Language in Action: Categories, Lambdas, and Dynamic Logic*, The MIT Press, 1991.

[42] H. Wansing, 'Negation', in *The Blackwell Guide to Philosophical Logic*, ed., L. Goble, 415–436, Basil Blackwell Publishers, Cambridge, MA, (2001).

[43] D. Wilkes-Gibbs and H. H. Clark, 'Coordinating beliefs in conversation', *Journal of Memory and Cognition*, **31**, 183–194, (1992).

[44] J.L. Wyatt, A. Aydemir, M. Brenner, M. Hanhiede, N. Hawes, P. Jensfelt, M. Kristan, G.J.M. Kruijff, P. Lison, A. Pronobis, K. Sjöö, D. Skočaj, and A. Vrečko, 'Self-understanding and self-extension: a systems and representational approach', *IEEE Transactions on Autonomous Mental Development*, **2**(4), 282–303, (2010).

Situated Plan and Execution Verbalisation

Gerald Schoch and Hendrik Zender
Language Technology Lab
German Research Center for Artificial Intelligence (DFKI)

Abstract

In this paper, we present an approach to verbalising reports of intentions and actions of a planner-enabled agent. We consider the case of an interactive intelligent robot that is endowed with a symbolic AI planner. The robot uses the planner to determine and execute sequences of actions in order to achieve a given goal. Since the robot is operating in a real, physical environment, making use of possibly imperfect sensing and actuating, it is likely to encounter unforeseen events or failures, and therefore needs to re-plan in order to come up with alternative plans for achieving its goal. In order to tell a human about what was planned, what was and was not successfully done, what happened – and why – we present a method for generating natural language reports based on such intended plans and the event structure of their execution. This verbalisation is inherently situated in nature. For one, external entities that are used in the planning process refer to things, persons, or locations in the physical environment that the robot and the hearer are situated in. Secondly, the robot's intentions and actions are temporally related to the discourse situation: the robot might report about its current plans and attempts for executing an ongoing task; the robot might explain what it did and what it couldn't successfully do in order to establish common ground for asking the human for help; or the the robot might simply report about past events and give details about why it chose to do what.

1 Introduction

Interactive intelligent robots need to possess two important features: *autonomy* and *communication skills*. Situated communication can comprise different modalities, like, e.g., spoken dialogue. Autonomy can range from simple reactive control loops to sophisticated goal-directed action planning and execution. If the robot and its user are supposed to engage in some form of collaborative activity. No matter if it is a cognitive assistant whose main purpose is to support its human user, or if it is a curious robot whose main drive is to learn about the world, the human crucially needs to be able to know and understand the robot's actions past, present, as well as intended ones in as far as they are relevant for achieving a task at hand; the robot's autonomous decision-making and acting need to be *transparent* to the human.

In this paper we focus on the verbalisation of plans as one way of achieving such a *transparency* in situated discourse. We consider the case of an interactive intelligent robot that is endowed with a symbolic AI planner. The robot uses the planner to determine and execute sequences of actions in order to achieve a given goal. Since the robot is operating in a real, physical environment, making use of possibly imperfect sensing and actuating, it is likely to encounter unforeseen events or failures, and therefore needs to re-plan in order to come up with alternative plans for achieving its goal.

The approach has been implemented on the integrated robotic system Dora [7]. Dora is programmed to exhibit a variety of intelligent behaviours, among which other intelligent mechanisms might arbitrate. However, these decisions might (initially) be intransparent to a human observer. We hence want Dora to provide verbal reports of what it attempted to do and why – thus establishing transparency about the complex spatio-temporal and causal relationships of its actions and action attempts.

Dora is equipped with a switching symbolic-probabilistic AI planner [5], able of continual planning and re-planning [3]. The approach is integrated in the general natural language processing sub-system of Dora and shares its linguistic resources with the other dialogue capabilities [10]. The verbalisation module is connected to Dora's spatial knowledge base in order to refer to entities in Dora's spatial environment [16, 17, 20].

The challenges we address here concern the appropriate verbalisation of plans of autonomous mobile robots. The robot can talk about past events (what was done), current events (what is being done), and intended events (what is expected to be done). Furthermore, this work addresses the prerequisites for informing the user about failures executing the plan and how the user could help to solve the problems, e.g. by changing the state of some objects (for instance, to unlock a door) or to give a modified goal. In order to achieve this, we investigate appropriate strategies for a suitable verbalisation of the planned and performed actions in a way that is understandable to a human user.

The rest of the paper is structured as follows: In Section 2 we provide some relevant scientific background regarding the topic and examine related work. After sketching our general design for the verbalisation of plans and plan executions in Section 3, we then present a system that instantiates this approach in Section 4. Section 5 contains an extensive discussion of an example report generated from realistic data. We conclude in Section 6.

2 Background and Related Work

Brenner [1] describes the use of classical AI planning techniques for interpretation and execution of human commands. He sketches how a robot can understand natural language (NL), plan the realisation and revise its plans based on new perceptions. His approach is similar to ours in that the goal is to couple planning symbols to natural language semantics and surface forms. In contrast to the approach presented in this paper, his approach focusses on *understand-*

ing rather than on *generation*. As a consequence, the approach makes use of a simple grammar that is generated out of the planning domain definitions. The planning domain is parsed and automatically translated into grammar rules and lexical entries. While this can be done in a straightforward way because the components of the planning language are rather similar to grammatical functions – e.g., types correspond to nouns, actions to verbs – the resulting grammar suffers from over-generalisation. It is thus not easily applicable to the verbalisation task.

In contrast, what we are interested in is the generation of natural language from planned and executed goal-directed sequences of actions. One domain where such an approach is chosen is story telling. Telling stories requires methods from many subfields of artificial intelligence (AI), e.g. planning, reasoning about beliefs, and dialogue systems. In [2] an approach to story generation using a continual multi-agent planner is presented. Here a plan is generated that is subsequently instantiated and executed. Then the plan is verbalised, but in a very simple way. The story is told without aggregation or co-references. A typical natural language generation (NLG) treatment would arguably make the text more fluent [13].

The Virtual Storyteller is another framework that generates simple story texts [15]. It is based on simulation of virtual characters in a story world. An event sequence is captured by a 'Plot Agent' in a formal representation. The representations are similar to the plans in [2], i.e., STRIPS-like [4]. A 'Narrator' component turns the representation into an actual story by selecting the content and processing it with NLG techniques. Story telling and NLG are also brought together in, e.g., [11].

Another system that verbalises some kind of plans is PROVERB [9]. In this work, mathematical natural deduction (ND) style proofs are verbalised. As input there is a representation of a ND proof. It is processed by a macro-planner to plan output that consists of primitive actions. The actions can be defined as communicative goals they fulfil as well as their possible verbalisations. Subsequently, more detailed linguistic decisions are made in the micro-planning component. Syntactic realisations are done using Tree-Adjoining Grammar (TAG). The final output then is the ND proof in natural language.

When verbalising plans and actions for autonomous robots, an additional aspect comes into play. In the previously mentioned approaches the planning domain and the planning operators available to the planner faithfully represent the "world" – i.e., the problem of plan execution is not taken into account. Rather, the computed sequence of actions is assumed to lead to the goal without fail. Autonomous mobile robots, like the one we are investigating, however act in the real world. There, the planning domain and the modelling of actions and operations can merely be seen as an approximation of reality. Such robots act in dynamic and only partially observable environments, in which their actions might not inevitably lead to the desired effect. Failures (such as the physical failure of an attempt to perform a particular action, a mismatch between the assumed state of the world with the perceived state of the world, or the possibility that an action did not yield the intended effect in terms of changing the

world state) hence need to be accounted for as part of the design of the planning system. The Continual Planning approach by Brenner et al., e.g., [3] models the ambiguity in assumptions about future states of the world and explicitly allows for re-planning as part of the design. Before and after each action, the planner updates its state representation based on the robot's perception, and evaluates if the planned course of action is still valid and still satisfies the initial goal. If there is a mismatch, re-planning is triggered in order to determine an updated course of action to achieve the goal. The Switching Planning approach, see e.g., [5], combines Continual Planning with Decision-Theoretic Planning, which probabilistically quantifies the uncertainty in the outcome of particular robot actions and in the relation between the perceived world state and the actual state of the world.

In order to achieve intelligent behaviour even if the agent's observations are potentially incorrect or incomplete, it is desirable to have a formal way to determine an explanation of why a plan went wrong and how the problem could be solved. Göbelbecker et al. [6] provide a formalisation regarding this issue. Changing the goal description – i.e., changing the initial state, or the set of planning operators – or re-planning – i.e., to find a new plan to achieve the goal – leads to a solvable goal. It is then a matter of other planning operators and actions to verify if these hypotheses hold. Besides trying to verify these assumptions through the robot's perception, one other possibility is to make use of natural language to engage a human in a dialogue and ask. This is particularly reasonable because the sources of such errors most often lie in the robot's imperfect perceptual capabilities.

3 Design

In this section we sketch a design for a situated verbalisation of plans and plan executions in order to generate a report of what went on.

A *plan* is a sequence of operations that each manipulate the state of the world. Together, these operations contribute to the achievement of a given goal (i.e., state of the world). Typically, such operations are actions that an agent needs to execute in order to alter the world state. Since in our work we focus particularly on autonomous mobile robots, the kinds of agents we are interested in are such robots that operate in a realistic physical environment. For instance, imagine a robot who is given the task of finding a cereal box (more precisely, knowing where an instance of a cereal box is located in the environment and subsequently telling its user where the box is). The robot can then use a planner to determine a sequence of actions that make the goal state true. After this initial plan has been formed (e.g., to go to the kitchen, search the tables in the kitchen, return to the user and report back), the robot can start executing the actions. Since we are dealing with a robot that acts under uncertain sensing and with uncertain information about the world (in other words, its knowledge can be both incomplete as well as incorrect), it might need to find a new sequence of actions in the light of new knowledge. This process is called *re-planning*.

When the robot eventually achieves its goal, a rich structure of planned, intended, assumed, attempted, failed and succeeded actions has led to this state. We therefore do not only consider individual plans, but rather *plan and execution histories* describing intended and executed actions along with their causal dependencies. In addition, these histories contain failed actions with potentially partial explanations of their causes (e.g., the state of the world has changed comparing to the assumptions in the original plan; the execution of an action did not yield the intended effect; or the execution of an action simply went wrong).

Generating a report in natural language (NL) on the base of such a history structure is not straightforward. A plan is merely an ordered sequence of (logical) symbols that somehow describe several steps how a certain goal can be fulfilled. So we need to define an approach how a plan can be mapped to linguistically meaningful symbols. These are used then in the pipeline to generate a report in NL. Concerning this we follow the standard NLG pipeline with *content determination*, *document structuring*, *lexicalisation*, *referring expression generation* (GRE), *aggregation* and *realisation* [13].

Besides the actual plans there are some external resources. First of all we require there to be a *domain definition*. In the domain definition the operators – i.e. actions – that can be used in a plan are defined in terms of some logical statement. This includes:

- the predicate-argument structure of an operator,
- the preconditions in the world so that an action can be executed,
- and the effect to the world by executing the action.

Furthermore predicates and functions are defined for expressing states in the world. Also the parameters – i.e., objects – of the operators are defined in the domain – c.f. [12].

A second resource needs to take the role of a *dictionary*. This dictionary is called the *lexical domain*. Here all the operators in the domain definition are linked to a linguistic expression, a logical form (LF). It is referred to as a *raw logical form* as it includes variables that are going to be instantiated at a later point. Furthermore there is no more information in the raw LFs than the semantic roles objects can adopt in an related expression. Basically raw LFs are the "link" between a plan and its logical expressions to the domain definition in order to map a plan to linguistically meaningful expressions. When the variables of a raw LF are instantiated later we call the new LF a *proto-LF*. A proto-LF is an *underspecified* LF, there is no more information about syntax, tense, or aspect as in a raw LF, as well as the parameters are instantiated with placeholders. Later, when a proto-LF is further processed in the NLG pipeline and all necessary information is added, we refer to the LF as a *full LF*.

We distinguish off-line and on-line processing in our approach. The next two subsections point out what is done in the off-line and on-line steps.

3.1 Off-line processing

As off-line processing we understand work that is done in advance before the actual report generation takes place. These steps are necessary to get a mapping between the logical symbols of a plan to linguistic counterparts.

The first thing to do is to build up the *lexical domain*, i.e., the dictionary with the raw LFs for the operators of the domain. Defining raw LFs is done manually by a human annotator. They are collected in an external resource.[1]

Considering the following fragment of the definition of the operator "move" – leaving out precondition and effect – the annotator of the dictionary needs to decide what syntactic functions the parameters have to take in a NL expression.

(1) Example definition of "move" (fragment)[2]

$$(:\ action\ move \\ :\ parameters\ (?a\ -\ robot\ ?to\ -\ place\ ?from\ -\ place)\ [...])$$

The annotator may interpret the definition of the action "move" that *?a* is the agent, *?to* the goal of the motion, and *?from* the motion's source. So the annotator may decide that the agent takes the role of the subject. A resulting raw LF will reflect these deliberations, cf. Example 6 in Section 5.

This is needed in the on-line processing step then: For every operator – i.e., action, predicate, function – a raw LF should be provided. Though a full coverage is not always possible or desired – i.e., in domain specific cases that are not useful for the verbalisation.

In the domain definition all operators are looked up and extracted automatically. They are stored together with its corresponding annotations, i.e. the raw LFs.

3.2 On-line processing

As on-line processing of a plan we understand plan processing steps that happen at run-time of the system. Using the off-line built resources (domain definition and lexical domain dictionary), the on-line processing part gets a plan or a planning history as input. It then processes the input and, along the NLG pipeline, generates a natural language text that appropriately describes the relevant aspects of what is expressed in the input structure.

First, the plan has to be processed. This can be, e.g., extracting all the planning steps that are going to be verbalised. In the NLG pipeline this is called *content determination – what* is to be said [13]. For each planning step then a suitable raw LF has to be looked up in the lexical domain dictionary. Next, the raw LF is mapped to a proto-LF by instantiating the free variables with the parameters of the planning steps.

[1] It is necessary that the annotator interprets correctly what the writer of the domain wants to express semantically with the definition of a certain operator. If the meaning of an operator is interpreted differently from what was intended by the domain writer, the generated expression of a planning step will of course not properly reflect the intended meaning.

[2] for the full definition refer to Example 4 in Section 5

Steps cannot be expressed in NL randomly as they are just symbols without any linguistic meaning, so we need to find some structure in which order the expressions can be realised – this is called *document structuring* [13]. The following example illustrates this issue:

(2) Example step in a partially ordered plan

4 : *SUCCEEDED move* `0 : D@spatial.sa 5 : C@spatial.sa 6 : C@spatial.sa`

Considering this example there are various terms not possible to map to NL directly. We have a number with colon (*4:*), an expression that says something about the execution status (*SUCCEEDED*) – only now follows the operator (*move*). Then the parameters are listed, in a form that cannot be straightforwardly transformed into NL.[3]

A plan is typically already partially ordered concerning its temporal structure, i.e., the linear order of the steps in the plan correspond to their temporal sequence of execution. This temporal structure in the plan can also be used in NLG. On the other hand, the causal structure – *why* is something executed – often is not obvious for the system. For this concern causal links between the steps can be considered, as well as preconditions and effects of actions. Also the difference between an ongoing and a past activity is to be taken into account. If it is a current plan, the report has to be generated in present tense. Otherwise, if the plan already has been executed, the report has to be generated in past tense. This *situatedness* can be derived from the plan and the *discourse situation*.

Once the content to convey is determined and the document is structured, lexicalisation and GRE can take place. An important step for "good" NL in the report is aggregation. For this aspect, the temporal and causal structure plays an important role, indicating *why* something happened. One can express states after executing an action. Also the messages that are going to be used can be mixed, e.g., they have the same agent and/or objects: *I move from the corridor to the kitchen* and *I search for the cereal box in the kitchen* can be aggregated to *I move from the corridor to the kitchen and search for the cereal box* [13].

After aggregating the messages they can be sent to the realiser. The constructed proto-LFs can be realised to NL with a suitable surface form. Depending on the application scenario, the resulting text can be turned into an audio signal by a text-to-speech engine [14].

4 Implementation

In this section we propose a system that instantiates the approach described above. We show how the system is implemented to generate a NL report from

[3]The overall system that this approach is integrated in is implemented using the CAST framework [8]. Since the generated plans refer to the robot's internal representations of external entities in the world, the parameters reference CAST WorkingMemoryAddresses (like `0 : D@spatial.sa` as their objects).

a plan. For this task we use the *domain definition* and a hand-crafted *lexical domain dictionary* as defined in the previous section.

4.1 PDDL – The Planning Domain Definition Language

The domain definition and the plans itself we use for demonstrating our system are written in PDDL [12]. Because of this we use a PDDL parser to get all relevant information of the domain. The parser we we use is the PDDL4J parser[4], an open source library. With this parser we get all the definitions of actions, predicates, and functions of the domain definition, including parameters, preconditions, and effects. This is essential for mapping planning steps with raw LFs to proto-LFs.

The PDDL4J parser is one of the main components to construct the PDDL domain model of our system. The domain definition file is parsed with it. While constructing the model in which all relevant information is stored the various methods of PDDL4J can be used efficiently – i.e., extracting all the definitions of each operator for later look ups.

Another crucial point where PDDL4J is involved is mapping a raw LF to an instantiated proto-LF. The parameters of a certain operator are extracted out of the domain definition with PDDL4J. As the parameters of a planning step are always in the same order as in the definition, the variables of the raw LF can now be instantiated in a safe way. The result is a proto-LF without any free variables.

4.2 Switching planner

As pointed out above, we have two external resources besides the actual plan – the *domain definition* and the *lexical domain dictionary*. This comes mainly from the construction of the planner we use in the Dora system. It uses two domains: a static domain and a dynamic domain. The static domain is built up in advance and never changes. For this reason we use it as the domain dictionary. We annotate operators with their respective raw LFs. The dynamic domain on the other hand is built dynamically for every run of the planner. It has more operator definitions than the static one. Because the dynamic domain is written in "real" PDDL style we need to use it to apply the PDDL4J parser for getting operator definitions, parameters, preconditions, and effects more easily.

4.3 The lexical domain dictionary

As mentioned in the previous subsection we use the static domain of Dora as the lexical domain dictionary. Here we determine the raw LFs for the operators in a well defined way. Considering the domain definition for the operator "move" we introduced in the previous section, a corresponding raw LF would look like this:

[4]http://math-info.univ-paris5.fr/pellier/research/software

(3) Example raw LF for the operator "move"

$$
\begin{aligned}
@_{p:action-motion_i}(&\mathbf{move} \wedge \\
&\langle Actor\rangle(?actor_i : castreferent \ \wedge \ ?actor) \wedge \\
&\quad \langle Modifier\rangle(from_i : m - wherefrom \ \wedge \ \mathbf{from} \wedge \\
&\quad \langle Anchor\rangle(?from_i : castreferent \ \wedge \ ?from)) \wedge \\
&\quad \langle Modifier\rangle(to_i : m - whereto \ \wedge \ \mathbf{to} \wedge \\
&\quad \langle Anchor\rangle(?to_i : castreferent \ \wedge \ ?to)))
\end{aligned}
$$

This raw LF could state a sentence like *I move from the corridor to the kitchen* when it is getting instantiated and threatened with the proposed NLG methods. We have the variables in there that are indicated with the symbol "?". They need to be instantiated at a later point. The index i is needed to get unique names for the proto-LFs. Otherwise they cannot be realised as they are seen as the same by the system. These indices are instantiated later with the number for a history block and the corresponding step in this block. The expression "castreferent" indicates that this is parameter corresponds to an internal representation stored at the given CAST WorkingMemoryAddress (WMA) which has to be handled with GRE when it comes to realisation in the NLG pipeline.

4.4 Planning history

The plans themselves we use in our system come from a planning history. Such a history consists of all history blocks that the planner has generated while trying to fulfil a goal. Each time, re-planning is necessary (cf. [3]), the previous episode is added as a block to the history. A history block defines the initial state of the world in the beginning of the episode, the current goal, the planning steps for achieving this goal, and a partially ordered planning graph (POPlan). The POPlan in addition shows the actions, i.e., the planning steps[5], with step number and WMA as parameters, needed for GRE later. Also links between the planning steps are listed as a graph structure in the POPlan[6]. Here the planning steps are the nodes and the links are labelled with predicates and functions. The planning graph can be used for determining the causal structure of a plan. The steps indicate what was planned and qualify the execution status (i.e., executed, failed, aborted, pending)[7].

4.5 Content determination

The content determinator of our system is used to derive messages to be realized from a planning episode. For each relevant step, a raw LF can be extracted from the dictionary. The raw LF is then instantiated with the actual parameters occurring in the planning step. The generated message is dependent on the

[5]cf. Example 8 in Section 5
[6]cf. Example 11 in Section 5
[7]cf. Example 10 in Section 5

execution status of the step. If it is *executed*, the POPlan graph can be used to get useful causal links. On the basis of these causal links new messages can be generated (e.g., a message expressing a relevant fact about the world state, or a message expressing a causality that links two or more planning steps). If the execution status of a step is *failed*, the message needs to be qualified with the information that the action expressed in the step was merely attempted but not successfully achieved ("I tried to go to the kitchen but did not reach it."). Additionally, the causal links can be inspected in order to determine a message that expresses a reason for the failure. The messages are expressed by instantiating the raw LFs with the actual parameter referents. Still, some other features of the thus generated proto-LFs are missing. In the subsequent steps the proto-LFs are refined until a fully-specified and well-formed LF is obtained that can be sent to the natural language realiser.

4.6 Refining proto-LFs

In order to turn the proto-LFs in the determined messages into meaningful text, a few more operations need to be performed: the referents need to be expanded to full referring expressions; contextual information (e.g., temporal aspects that depend on the discourse situation) needs to be specified; and finally additional syntactic information that is necessary for a well-formed LF (i.e., the identification of the semantic agent as the syntactic subject of a sentence in active voice) needs to be supplemented.

The approach to referring expression generation makes use of the spatially situated GRE algorithm by Zender et al. [19, 20, 18, 16]. Since the referents are given in the form of a CAST WMAs they can be resolved to internal symbols of the system's spatial representation (cf., e.g., [17]), which is constitues the knowledge base for the GRE module. Other necessary refinement steps of the proto-LF are handled by the TAROT libraries [10]. We will not go into further details of these processes, but instead refer the interested reader to the referenced work.

5 Examples

In this section we show how the system works in a nutshell. We describe all processing steps which have to be conducted to generate a NL report from a plan in PDDL style.

5.1 The domain definition

Assuming we have the resources as indicated in the previous sections. Then the first step is to define raw logical forms for the operators in the domain dictionary. E.g., assuming we have the following definition of the operator "move", defined in the domain definition:

(4) Example definition of the operator "move"

```
(: action move
        : parameters (?a − robot ?to − place ?from − place)
        : precondition (and (or (connected ?from ?to) (connected ?to ?from))
                            (not(done))
                            (is − in?a?from))
        : effect (and (not (is − in ?a ?from))
                      (is − in?a?to)
                      (placestatus?totrueplace)
                      (kd − in − room?a?to)
                      (increase(total − cost)2.0000)
                      (started))
)
```

First there is the kind of operator – *action* – followed by the operators's name *move*. Next the *parameters* are defined, the variables – in our example they are *?a* of the type *robot*, *?to* of the type *place*, and *?from* of the type place, too. These variables are going to be instantiated later in processing the plan.

The next point defined is the *precondition* – one or more states that are necessary that a certain action can be executed. In our example the two places – source and goal of the movement – need to be "connected", slovenly speaking they need to be next to each other. Furthermore the agent needs to be at the place where the source of the movement is. These *predicates* are defined in the same domain definition and look like this:

(5) Example definition of the predicate "is-in"

```
(: predicate [...]
        (is − in ?o − robot ?value − place)
        [...])
```

All predicates are defined in one block – indicated with "[...]" in the example. The predicate itself is defined in one block of brackets inside the predicates definition block. It starts with the predicate's name – *is-in* – followed by the parameters – *?o* of the type *robot* and *?value* of the type *place*.

Finally, in an action's definition the *effect* of the particular action, i.e., the resulting state, is defined. As in the precondition this is defined in terms of one or more predicates.

5.2 The lexical domain dictionary

After all relevant decisions are taken by the annotator, a resulting raw LF for the introduced action "move" may look like this, including the syntax we have chosen to extract the raw LFs easily:

(6) Example for a raw LF annotation for the action “move”

$$
\begin{aligned}
&;;LF\ move\ =>\\
&\quad @(step_i : action-motion)(move\ \wedge\\
&\qquad < Actor > (?a : castreferent\ \wedge\ ?a)\ \wedge\\
&\qquad < Modifier > (from_i : m-wherefrom\ \wedge\\
&\qquad from\ \wedge\ < Anchor > (?from : castreferent\ \wedge\ ?from))\ \wedge\\
&\qquad < Modifier > (to_i : m-whereto\ \wedge\ to\ \wedge\\
&\qquad < Anchor > (?to : castreferent\ \wedge\ ?to)))
\end{aligned}
$$

An example annotation for the predicate “is-in” looks like the following in our system:

(7) Example for a raw LF annototation for the predicate “is-in”

$$
\begin{aligned}
&;;LF\ is-in\ =>\\
&\quad @(state_i : ascription)(be\ \wedge\\
&\qquad < Cop-Restr > (= var : castreferent\ \wedge\ = var)\ \wedge\\
&\qquad < Cop-Scope > (in_i : m-location\ \wedge\ in\ \wedge\\
&\qquad < Anchor > (= val : castreferent\ \wedge\ = val)))
\end{aligned}
$$

The main difference between an action and a predicate annotation is that the raw LFs of the predicates use “=var” indicating the variable. The reason for this is that in the domain definition predicates have more parameters as this is the design used by the writer of the domain.

This way all predicates and actions are annotated that are wanted to be used in the NLG pipeline. It is up to the writer of the domain dictionary.

5.3 The planning history and NLG

A history contains most often several different blocks of plans. These are separated by keywords so we can extract each plan to handle it for NLG. A single plan may look like this:

(8) Example for a short plan (extract)

actions :
8 : *SUCCEEDED move* `0 : C@spatial.sa F : B@spatial.sa D : B@spatial.sa`
9 : *SUCCEEDED move* `0 : C@spatial.sa H : B@spatial.sa F : B@spatial.sa`
10 : *SUCCEEDED createconesinroom* `0 : C@spatial.sa` *magazine*
`2 : 71@coma H : B@spatial.sa`
11 : *PENDING searchforobjectinroom* `0 : C@spatial.sa` *magazine*
`2 : 71@coma H : B@spatial.sa` *visualobject*1
12 : *PENDING goal*

Each planning step starts with the step number – here we show five steps. The number is followed by a flag that indicates the status of the planning step. We use three execution flags: *SUCCEEDED* means that the particular step has been executed successfully; *PENDING* means that a step is planned but not executed (yet). *FAILED* – not shown in the example – indicates that it has been tried, i.e., by a certain agent, to execute the step but the execution was not successful for some reason – e.g., because the world state has changed or there were wrong assumptions. This can be because a door is unexpectedly locked so that the room behind it cannot be reached.

After the flag comes the operator – here actions. In our example the LF for "move" is extracted. Once the system has found an annotation for the operator the variables in the raw LF are getting instantiated with the parameters in the planning step. For the planning step 8 in our example there are three parameters in the form of WMAs. The order of the variables in the raw LFs for actions does not matter.

When the variables are instantiated we have a *proto-LF*. For "move" it will look this – for the index number we simply use "1" here:

(9) Example proto-LF with "move"

$$
\begin{aligned}
&@(step_1 : action - motion)(move \ \wedge \\
&\quad < Actor > (\texttt{0} : \texttt{C@spatial.sa} : castreferent \ \wedge \ \texttt{0} : \texttt{C@spatial.sa}) \ \wedge \\
&\quad < Modifier > (from_1 : m - wherefrom \ \wedge \\
&\quad from \ \wedge \ < Anchor > (\texttt{F} : \texttt{B@spatial.sa} : castreferent \ \wedge \\
&\qquad \texttt{F} : \texttt{B@spatial.sa})) \ \wedge \\
&\quad < Modifier > (to_1 : m - whereto \ \wedge \ to \ \wedge \\
&\quad < Anchor > (\texttt{D} : \texttt{B@spatial.sa} : castreferent \ \wedge \ \texttt{D} : \texttt{B@spatial.sa}))
\end{aligned}
$$

Then the *state* of the step is imported to the proto-LF. This is done by adding ⟨ExecutionStatus⟩ followed by the current step flag – SUCCESSFUL, PENDING, or FAILED. The reason for this is that we can still apply aggregation to the messages at a later point. Executions states are added like this:

(10) Example proto-LF with "move" and executions status

$$
\begin{aligned}
&@(step01 : action - motion)(move \ \wedge \ < ExecutionStatus > SUCCESSFUL \ \wedge \\
&\quad [...] \ )
\end{aligned}
$$

The states SUCCESSFUL and FAILED are analysed promptly while determining the message. The reason for this is that besides the partially ordered plan as we have in our system we also can use the *graph* for the plan. The graph shows dependencies between the planning steps so that we, e.g., can determine **why** something was or is going to be done, or can be done, or failed while executing.

Getting back to the example 10 with execution status SUCCESSFUL, the content determinator of our system now looks up the dependencies of this step

to steps that where performed before. For this the plan graph is used. Here is an excerpt of the graph for the plan in our example:

(11) Example for a plan graph (excerpt)

links :
8 9 $DEPENDS\ is-in$ 0 : C@spatial.sa $VALUE$: F : B@spatial.sa
9 11 $DEPENDS\ is-in$ 0 : C@spatial.sa $VALUE$: H : B@spatial.sa
9 11 $DEPENDS\ in-room$ H : B@spatial.sa
$MODALITY$: *kval* 0 : C@spatial.sa $VALUE$: *true*
9 10 $DEPENDS\ is-in$ 0 : C@spatial.sa $VALUE$: H : B@spatial.sa
9 10 $DEPENDS\ in-room$ H : B@spatial.sa
$MODALITY$: *kval* 0 : C@spatial.sa $VALUE$: *true*
10 11 $DEPENDS$ *conescreated magazine in* 2 : 71@coma $VALUE$: *true*

The first two numbers indicate that the planning steps with this number – the *nodes* of the plan graph – are connected, meaning the second step causally depends on the first. Then the *label* of the *link* is specified. In our example the labels are always *DEPENDS* – also they can be *THREATENED*, i.e., indicating that the precondition of an operation is violated. After the dependency label the predicate is given, specifying **what** the dependency is actual. The WMAs and parameters following the predicate are the variables of the predicate as defined in the domain definition. They can have different keywords – i.e., VALUE, MODALITY – indicating what the WMAs and parameters actually stand for. This has to be considered when annotating the predicates in the domain dictionary.

The first link in our example states that step 9 is a *child node* of step 8. The execution of step 9 *depends* of that the WMA 0 : C@spatial.sa (the agent) is located at F : B@spatial.sa. This is the precondition to move to another place from there in the following step. This affords, e.g., verbalising a sentence like: "Because I was at place X I could move to place Y."

The same is done for all steps in the plan. Considering the plan graph for step 11 a possible verbalisation can be "After I created view cones in the kitchen I could search for the magazine at place X." These collected messages are expanded to full LFs and given to the surface realisation component of the NLG pipeline in our system.

5.4 Realising the determined messages

Realisation then is executed after lexicalisation and the generation of referring expressions. Grammar rules determine how the sentences are formed [13]. For the extract of our example plan and plan graph, a resulting natural language report in our system is the following:

(12) Verbalisation of the example plan excerpt

1: I moved from place0 to place1.
2: Because that was successful and I was at place1 then,
3: I could move to place2.
4: Next I created viewcones there
5: to be able to search for the magazine in this room.
6: After that I can fulfil the intended goal.

Analysing the verbalisation of the plan we see that it says *what* was done, *why* was it done, *when* was it done, and why the agent was *able* to do it. Also there is a certain degree of aggregation.

Line 1 and line 3 show what was done. For the statement in line 3 also a explanation is given why the movement could be executed – line 2. Line 4 provides aggregation with the temporal expression "next" and indicates *when* the step was executed. This comes from the temporal structure of the partially ordered plan. Line 5 provides an explanation why the agent executed the creation of view cones in line 4. This comes from analysing the plan graph – as well as line 2. Line 6 provides an outlook to the future, indicated with the execution status PENDING – meaning that when step 11 (line 5) can be executed successfully, also PENDING, the agent is able to achieve the intended goal.

6 Conclusion

In the present paper we have proposed an approach for generating reports of plans. We have embedded the proposed approach in the general area of interactive and autonomous robots.

We motivated our approach with a general introduction (Section 1) to the specific challenges involved in the generation of situated natural language for such robots, especially concerning the robots' autonomously determined actions and assumptions. In Section 2 we have given an overview of related work in NLG and planning techniques in artificial intelligence and robotics. We have also discussed other approaches to generating natural language from various structured and unstructured resources for situated and cooperative dialogue systems.

Section 3 describes the general design principles of a system for NLG in the planning environment of the robotics domain. One of the most important points are the resources that are needed for such a task: (1) the domain definition, (2) the lexical domain dictionary, and (3) resources that encode the event structure of what happened (i.e., the plan history). We have shown how these resources have to be processed to get all relevant information for verbalising a plan. Furthermore we have indicated what can be done to get "good" natural language reports. This includes aggregation in terms of sentences mixtures, temporal and causal ordering, and explanations *why* something was done or why it failed. The overall design can be extended for various styles of planning and verbalisation.

Then, in Section 4, we have explained how our approach is implemented in an integrated robotic system. We have shown how the annotation task is done and what considerations have to be made by the human annotator. Also we have indicated how our system processes all given information of the resources automatically. We presented how this information is used in the NLG pipeline and how we finally get to a natural language a report.

Section 5 has provided a detailed example how our system works with real world data – beginning with the annotation and considerations, how the information is processed automatically, and what the results are for a certain plan.

Concluding, our implemented system shows how we can get from some raw data resources constituting a plan to a full report in natural language in good quality.

References

[1] M. Brenner. Situation-aware interpretation, planning and execution of user commands by autonomous robots. In *Proceedings of the 16th IEEE International Symposium Robot and Human Interactive Communication (RoMan)*, pages 540–545, 2007.

[2] Michael Brenner. Creating dynamic story plots with continual multiagent planning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI)*. AAAI Press, July 2010.

[3] Michael Brenner and Bernhard Nebel. Continual planning and acting in dynamic multiagent environments. *Journal of Autonomous Agents and Multiagent Systems*, 19(3):297–331, 2009.

[4] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

[5] Moritz Göbelbecker, Charles Gretton, and Richard Dearden. A switching planner for combined task and observation planning. In *Electronic Proceedings of the Workshop on Decision Making in Partially Observable, Uncertain Worlds: Exploring Insights from Multiple Communities at the Twenty-Second International Join Conference on Artificial Intelligence (DMPOUW 2011)*, 2011.

[6] Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI Press, May 2010.

[7] Marc Hanheide, Charles Gretton, Richard Dearden, Nick Hawes, Jeremy Wyatt, Andrzej Pronobis, Alper Aydemir, Moritz Göbelbecker, and Hendrik Zender. Exploiting probabilistic knowledge under uncertain sensing

for efficient robot behaviour. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI-11)*, Barcelona, Catalonia, Spain, July 2011.

[8] Nick Hawes and Jeremy L. Wyatt. Engineering intelligent information-processing systems with CAST. *Advanced Engineering Informatics*, 24:27–39, 2010.

[9] Xiaorong Huang and Armin Fiedler. Proof verbalization as an application of NLG. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 965–970. Morgan Kaufmann, 1997.

[10] Miroslav Janíček. Tarot: The Talking Robots Toolkit: technical description, version 0.1. Technical report, German Research Center for Artificial Intelligence (DFKI), 2012. CogX project deliverable DR6.5.

[11] Birte Loenneker. Narratological knowledge for natural language generation, 2005.

[12] Drew McDermott, Craig Knoblock, Anthony Barrett, Dave Christianson, Marc Friedman, Chung Kwok, Keith Golden, Scott Penberthy, David E Smith, Ying Sun, and Daniel Weld. PDDL – The Planning Domain Definition Language. *AIPS98 planning committee*, 78(4):1–27, 1998.

[13] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, 1999.

[14] Marc Schröder and Jürgen Trouvain. The german text-to-speech synthesis system MARY: A tool for research, development and teaching. *International Journal of Speech Technology*, 6:365–377, 2003.

[15] I. M. T. Swartjes and M. Theune. The virtual storyteller: story generation by simulation. In *Proceedings of the Twentieth Belgian-Netherlands Conference on Artificial Intelligence, BNAIC 2008, Enschede*, pages 257–265, Enschede, October 2008. University of Twente.

[16] Hendrik Zender. *Situated Production and Understanding of Verbal References to Entities in Large-Scale Space*, volume 36 of *Saarbrücken Dissertations in Computational Linguistics and Language Technology*. German Research Center for Artificial Intelligence and Saarland University, Saarbrücken, Germany, 2011.

[17] Hendrik Zender. Multi-layered conceptual spatial mapping – representing spatial knowledge for situated action and human-robot interaction. In Yacine Amirat, Abdelghani Chibani, and Gian Piero Zarri, editors, *Bridges Between the Methodological and Practical Work of the Robotics and Cognitive Systems Communities – From Sensors to Concepts*, Intelligent Systems Reference Library. Springer Verlag, Berlin/Heidelberg, Germany, under submission 2012. CogX project deliverable DR3.3.

[18] Hendrik Zender, Christopher Koppermann, Fai Greeve, and Geert-Jan M. Kruijff. Anchor-progression in spatially situated discourse: a production experiment. In *Proceedings of the Sixth International Natural Language Generation Conference (INLG 2010)*, pages 209–213, Trim, Co. Meath, Ireland, July 2010. Association for Computational Linguistics.

[19] Hendrik Zender, Geert-Jan M. Kruijff, and Ivana Kruijff-Korbayová. A situated context model for resolution and generation of referring expressions. In *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*, pages 126–129, Athens, Greece, March 2009. Association for Computational Linguistics.

[20] Hendrik Zender, Geert-Jan M. Kruijff, and Ivana Kruijff-Korbayová. Situated resolution and generation of spatial referring expressions for robotic assistants. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1604–1609, Pasadena, CA, USA, July 2009.

Tarot: The Talking Robots Toolkit

Technical description, version 0.1

Miroslav Janíček

TAROT is an open-source software framework for building spoken dialogue functionality for human-robot interaction. TAROT does not impose a specific (cognitive) architecture for building a dialogue system. Its framework allows for multi-threaded (or asynchronous) processing. Processes are defined as glass boxes (rather than black-boxes) and interact in an event-driven fashion. TAROT is written in the Scala programming language and targets the Java platform.

1 Introduction

This document is a technical description of the current version of TAROT, the Talking Robots Toolkit. TAROT is an open-source software project licensed under the Lesser General Public License.

1.1 What is Tarot?

TAROT is an open-source software framework for building natural language interfaces for human-robot interaction. It is asynchronous, event-driven framework, written in the Scala programming language [7] and targets the Java platform.

All software described here can be downloaded from

`http://tarot.opendfki.de`

1.2 About this document

This document is a technical description of Tarot's core concepts, design approach, and an overview of the software packages that together make up the Tarot public distribution.

The structure of this document is as follows: in the next section, we will examine the basic concepts, in §3 address the main implementation design choices, and in §4 describe the distribution that forms the overall TAROT "solution stack".

2 Core concepts

TAROT is a toolkit providing the basic functionality for building natural language interfaces for human-robot interaction. As such, TAROT does not impose any particular processing strategy – this is left to be dealt with by the integrator of the natural language module to a wider system. TAROT is therefore agnostic with regard to the actual cognitive architecture of the robot.

In order to support as wide a range of systems to be integrated with, TAROT has to be

- multi-threaded – a TAROT system must be able to perform multiple actions in parallel;
- event-driven – the system must be able to react to events emitted in the cognitive architecture;
- transparent – the architecture should be a glass rather than black box whenever possible.

Programming multi-threaded systems is fundamentally different from assuming that there is a single execution thread. Since we need TAROT to work with systems capable of performing multiple interleaved tasks, we cannot afford to dodge concurrency issues. However, by imposing a set of good practices, we can limit the dangers to a minimum. These practices in turn guide the design of the fundamental concepts and constraint the properties of the state, and its change and observability.

2.1 State

Defining the notion of state is central to any concurrent application. Typically, concurrency is done by using one the following two models:

- **Shared memory model** – state is explicitly shared between the threads. There has to be access control if the threads are allowed to modify the state. This is usually done using *locking*.
- **Message passing model** – state is implicit in the system configuration, and is changed by components sending messages to each other.

TAROT utilises both approaches. Conceptually, we use the shared memory model (and this is what is exposed to the programmer). However, the values that are available to the threads are *immutable*, which means that they can be shared without the need for access control. Under the hood, however, is a distributed, actor model-based (i.e. message passing, [1]) architecture. State change in TAROT is then done by sending a message to the corresponding state maintainer actor. Note that the API provides an abstraction so that the user does not see that messages are being passed around.

The notion of state as something that can change is embodied in the concept of a *variable*. A variable is an object holding the state, emitting events whenever the state changes, and providing an interface for changing the state. The actual value of the variable is always a pair consisting of the value and its version. *Versions* define a partial ordering over the values of a variable. This allows an application to determine whether one value precedes another, and makes variables and their values referentially transparent.

2.1.1 Variables

TAROT defines three broad classes of variables:

- **Continuous variables** – variables that can only be sampled, but their change can not be directly observed;
- **Discrete variables** – variables that emit events for each value change. We distinguish two sub-classes of discrete variables:
  - **Stateful discrete variables** – discrete variables that hold the value, which can be queried;
  - **Stateless discrete variables** – discrete variables that do not hold the value. Note that since they are discrete variables, the value *change* can be observed.

Continuous variables are used for "volatile" values such as time where it is impossible (or impractical) to observe every change. However, since the value can be queried, every continuous variable can be turned into a discrete variable by querying the value at some sampling rate. For instance, this is – conceptually speaking – what a (digital) audio signal is: sampling of the acoustic pressure as detected by the microphone.

2.1.2 State change observation

Every discrete variable emits events whenever there is a change. Those events are of two kinds:

- **Value change** – emitted whenever the variable value is changed. Depending on the sub-class of the discrete variable, this is one of the following:
  - **New value** (stateful variables) – the new value of the variable.
  - **Value transformation** (stateless variables) – the transformation f, applied to the previous value of the variable, that yields the new value. The transformation f is therefore a partial function $f : A \to A$ where A is the domain of the variable.
- **Constantness** – emitted when a variable is guaranteed not to change any more. This effectively turns a variable into a constant (from this moment on), hence the name.

These events can be listened to using the Observer pattern [3], and listeners are notified asynchronously.

2.2 Performing actions

Actions are an abstraction of an (asynchronously) executed task that changes the state of the system. The action state is exposed in the following variables:

- **Action state** – the state of an FSM describing the current processing state of the action. Valid values are the following:
  - `Pending` – the action has not started yet;
  - `Active` – the action is executing;
  - `Error` – error state;
  - `Finished` – the action finished successfully.

  Transitions of the FSM are depicted in Figure 1.
- Action-dependent **result** – this can be either a discrete (stateless or discrete variable), or a `Future` for a set-once result.

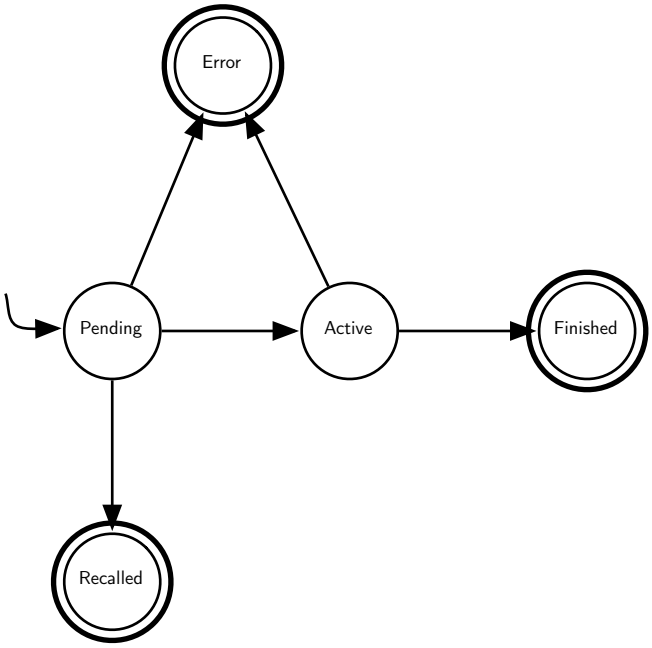

Figure 1: Action state transitions

For an action to be executed, it has to be scheduled by an actuator first.

2.2.1 Actuators

An actuator is an abstraction of the entity executing the action. Actuators are responsible for access control to the underlying resource. The application sends *action requests*

to an actuator, and depending on the nature of the resource, the request is either rejected (in case the actuator is not ready to receive actions), or the action is *scheduled* for execution. TAROT currently defines three classes of actuators:

- **Serial** – actuator operating off a task queue, executing one action at a time;
- **Parallel** – actuator executing all actions in parallel;
- **Single-action** – actuator executing at most one action at a time, with no queue.

Serial actuators can be likened to serialisers for thread-unsafe resources, whereas parallel actuators operate on thread-safe resources. Single-action actuators are useful in situations where a task queue makes no sense (such as recording audio), and therefore provide a fail-fast feedback to the caller.

The actuator state is exposed in the following variables:

- **Actuator state** – the state of an FSM describing the current state of the actuator. The values can be the following:
  - `NotStarted` – the actuator has not been started yet;
  - `Starting` – initialising;
  - `Ready` – ready to receive action requests;
  - `Busy` – the actuator is working and *cannot* receive action requests at this time;
  - `Error` – error state;
  - `ShutDown` – successfully shut down.

  For the transition of the FSM, see Figure 2.
- **Currently active actions** – a collection of currently executing actions.

Since they publish the currently-active actions, actuators are *introspectible* – given an actuator, an application can get all the way to the actions it is executing, listen for their state changes etc. This is a crucial element of the system's scalability and flexibility.

2.3 Instantiating systems

Variables, actions and actuators are the basic building blocks of a TAROT *system*. A system is a concrete instantiation of a particular functionality. We use *dependency injection* (DI) to build systems together in a loosely-coupled manner. Using DI to compose functionality makes it possible to program against an interface (rather than against a behaviour), with benefits for code reuse and testability.

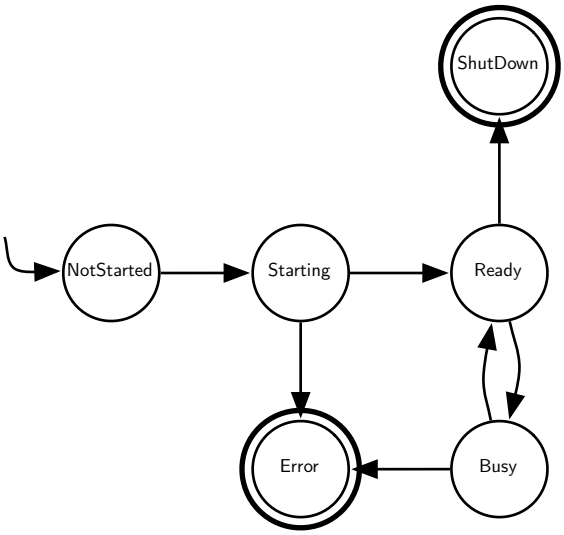

Figure 2: Actuator state transitions

2.3.1 Devices

In order to group actuators belonging conceptually together, TAROT uses the notion of a *device*. Devices do not have any particular functionality of their own, but merely serve as "namespaces" for separating unrelated actuators. The dependency injection for composing functionality into a system is done on the level of devices.

2.3.2 Building systems

To illustrate the principle of combining systems using DI, consider the following example:

```
class Instantiation
    extends AbstractSystemConfiguration
    with HasClockDevice
    with HasAudioCaptureDevice
    with HasSpeechRecogniserDevice
    with HasBasicPhonologicalStringParserDevice
    with HasBasicSurfaceRealiserDevice
    with HasSpeechSynthesiserDevice
    with HasAudioProductionDevice {

  // define functionality of the system here

}
```

Here, we define a class `Instantiation` that has a clock, can capture audio, has a speech recogniser, phonological parser, surface realiser, speech synthesiser and audio production support. The concrete implementation of these is not specified – these are all interfaces. This means that the functionality defined in `Instantiation` will work with any concrete subclass that implements it.

For an example instantiation, we can consider the following source code:

```
val sys = new Instantiation
    with HasSystemClockDevice
    with HasLocalMicrophoneDevice
    with HasSphinx4SpeechRecogniserDevice
    with HasCCGPhonologicalStringParserDevice
    with HasCCGSurfaceRealiserDevice
    with HasMarySynthesiserDevice
    with HasLocalLoudspeakerDevice {

  lazy val sphinxConfig = /* Sphinx4 configuration */

  lazy val ccgParserGrammar = new Grammar(/* URL to the grammar */)
  lazy val ccgRealiserGrammar = ccgParserGrammar

  val maryServerAddress = /* address to the Mary server */

}
```

Now the variable `sys` is holding a concrete instantiation of `Instantiation`, fulfilling its contracts for all the functionality with the local microphone, Sphinx4 ASR engine, OpenCCG parser and surface realiser, Mary TTS speech synthesiser and a local loudspeaker for audio production.

Changing this functionality, for instance transmitting audio data over a network instead of capturing and producing it locally, is would then merely mean changing the corresponding mix-in traits that specify which classes implement the required interfaces. The functionality per se would not change.

TAROT supports Scala's native "cake pattern" for dependency injection, which is a statically typed method of composing dependencies at compile time using mix-in traits. Since it is statically typed, the correctness of this construction is checked by the Scala compiler, and therefore can detect many errors before the code is run, greatly improving the reliability of the system [8].

3 Implementation notes

TAROT targets the Java platform, and is written in the Scala programming language. Thanks to this design choice, the functionality can incorporate existing Java libraries, and can be interfaced to from Java.

Why targeting only the JVM, and why choosing Scala instead of pure Java? What about integration with functionality written in other programming languages and/or running in a distributed manner? Let us briefly examine each of these questions in turn.

Why the Java platform? The Java platform is a well-established multi-platform technology, and the Java programming language is among the most popular programming language as of today. The platform offers an excellent tooling support which greatly increases the programmer productivity. The availability of high-performance virtual machine (VM, or JVM for Java Virtual Machine) implementations, including the standard HotSpot virtual machine, allow programs written for the Java platform to even outpace their counterparts written in C++ for many tasks, and the performance is definitely sufficient for event-driven systems supported by TAROT.

Why Scala? Scala [7] is a general purpose programming language that combines the paradigms of object-oriented and functional programming for the JVM platform. It is a strongly statically typed language with a higher-order type system and type inference (features most commonly seen in functional programming languages such as ML or Haskell).

Its standard distribution features a strong generic data collections library that stresses the need for data structure immutability, reflecting the functional background and making the writing of concurrent applications easier.

What about other programming languages? Not every programming language is strong at everything – some allow you to write a very performant code (C, C++), some offer you a convenient programming environment (Java), some shine at performing certain tasks but it's very hard to apply them outside their typical domain (Prolog), some are statically typed (Java), some are dynamic (Python), and so forth. The idea to use a specialised language for some tasks, and a general purpose language for others, is a tempting one.

Moreover, there may be situations in which an application *has to* interface with functionality written in a different programming language, or even running on a different platform or machine.

How does TAROT fare in this aspect? There are, broadly speaking, three ways to achieve this:

- **Use explicit inter-process communication.** In this setting, the code written in another language is interfaced to using some *protocol*. This completely abstracts away from the runtime environment of the other side, and in general can lead to a well-testable software code. However, the protocol has to be fully specified and conformed to, and offers very little abstraction, which is essential for writing truly complex applications.

  This is certainly supported by TAROT, since it runs on the Java platform.

- **Use a middleware.** Middlewares are an abstraction over inter-process communication protocols, exposing a "native" interface that proxies remote execution environments. This interface is called the remote procedure call interface (RPC). Typical representatives of middlewares are CORBA or ZeroC ICE.

Writing complex and reliable distributed applications using RPC is a very hard task to do correctly. On the one hand, middlewares such as CORBA scale out easily; on the other hand, the data structures passed around in RPC break encapsulation, and ultimately may lead to the adoption of the procedural style of programming. This of course means that the software quality suffers, and maintenance and development cost soar.

In our opinion, this is too high a price to be paid straight away, and therefore we opted for not using any middleware as part of the core TAROT design. However, TAROT's users are free to integrate with the middleware of their choice, making the compromises where *they* need, rather than trying to work around the limitations of the middleware imposed on them from the start.

- **Use a foreign language interface.** A foreign language interface is a way of calling code written in a "foreign" language from the "primary" language. This is usually a one-way relation – the functionality written in the foreign language must conform to constraints imposed by the primary language, but not the other way around.

  This is the common approach interpreted languages (such as Java) take to access the underlying native platform. To this end, the majority of foreign function interfaces specify an intermediate C layer. This approach is supported by TAROT's choice of the JVM platform (and thus, transitively, by Java foreign language interface solutions such as JNI).

4 The Tarot stack

TAROT exposes its functionality in a layered manner, providing a *solution stack* – a collection of software tools stacked on top of that offers a solution to a particular problem, in our case building systems for processing situated natural language. The stack is illustrated by Figure 3.

TAROT is built on top of the Java platform, Scala standard library and the Akka[1] middleware (depicted in the figure by the light red colour). Akka is a toolkit for building massively concurrent systems based on the actor model of concurrency.

In the following, we use the term *modules* to describe software packages that either define some interface or an implementation of an interface, or both. In this sense, modules denote units of software composition and distribution, rather than of run-time functionality.

Let us now examine the stack in more detail.

4.1 Common interfaces

The common interfaces (dark grey in Figure 3) provide the basic functionality without committing to a particular implementation. These modules are structured as follows.

[1] see `http://akka.io`

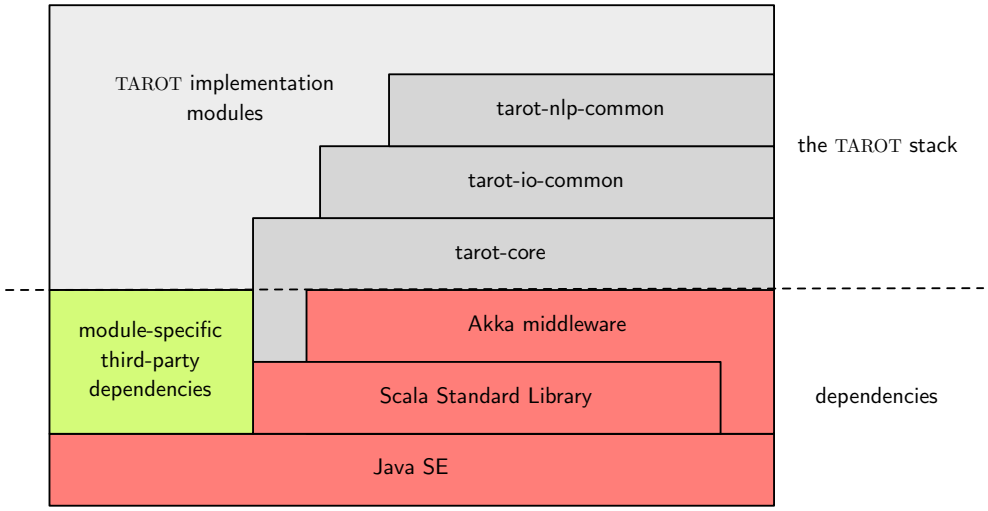

Figure 3: The TAROT software stack

The core functionality as outlined in §2 is implemented in the module `tarot-core`. This module uses the Akka library to implement the built-in concurrency support.

The module `tarot-io-common` implements the basic functionality related to I/O (i.e. interaction with the real world). Currently, these are the following:

- **Audio interfaces** – a definition of *immutable* audio data, and basic conversion methods;
- **Audio capture and production** – i.e. obtaining audio data from the world, and turning it into sound. The module also contains a default implementation using the Java Sound API.
- **Clock** – a device for measuring time and emitting events when a given time passes ("alarm clock"). Again, a default implementation as provided by Java SE is bundled, but the interface can scale to *simulated* clocks as well.

The module `tarot-nlp-common` then defines basic interfaces related to processing natural language. These are currently:

- **Speech recognition** – using the audio interfaces defined in `tarot-io-common`, a speech recognition device provides an interpretation of an audio signal in the form of a best recognised phonological string or word lattice;
- **Parsing** – devices for parsing phonological strings and word lattices into *logical forms* (see below);
- **Logical forms** – logical forms are directed graphs for representing linguistic meaning [4]. The module also contains a variety of convenience transformation operations.

- **Surface realisation** – devices for turning logical forms into phonological strings;
- **Speech synthesis** – conversion of a phonological string in terms of an audio signal.

4.2 Implementation modules

Apart from the interfaces described in the previous section, TAROT also contains default implementations for each of them, using open-source software. These modules are together illustrated by the light grey colour in Figure 3, along with their dependencies in light green.

Currently, the modules in question are the following:

- `tarot-sphinx4` – Sphinx4 speech recognition support. Sphinx4 is an open-source recogniser written in the Java programming language, and its distribution also contains basic acoustic models [6];
- `tarot-openccg` – an implementation of the parsing and surface realisation devices using the OpenCCG library [2];
- `tarot-marytts` – Mary TTS (text-to-speech) [9] support for speech synthesis.

4.3 Resources

We consider linguistic resources to be an auxiliary part of the TAROT stack, in the sense that whenever possible, they should be independent of TAROT but can be well integrated with a TAROT system.

Currently, the only linguistic resource included in the public distribution is the Moloko CCG grammar [5].

5 Conclusion

We have presented TAROT, an open-source software framework for building spoken dialogue functionality for human-robot interaction. The framework supports asynchronous, event-driven processing, but does not impose specific constraints on the part of the wider integrated system's architecture. TAROT is written in the Scala programming language and targets the Java platform.

We have outlined the main concepts of the design approach, and described the contents of the publicly-available distribution of the TAROT solution stack.

5.1 Acknowledgments

The work reported in this paper has been supported by the EU ICT FP7 projects CogX, "Cognitive Systems that Self-Understand and Self-Extend" (project no. 215181) and NIFTi, "Natural Human-Robot Collaboration in Dynamic Environments" (project no. 247870).

References

[1] Gul Abdulnabi Agha. *ACTORS: A Model of Concurrent Computation in Distributed Systems*. PhD thesis, Massachusetts Institute of Technology, 1985.

[2] J. Baldridge and G.J.M. Kruijff. Multi-modal combinatory categorial grammar. In *Proceedings of the 10th conference on European chapter of the Association for Computational Linguistics (EACL 2003)*, pages 211–218. Association for Computational Linguistics, 2003.

[3] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.

[4] Geert-Jan M. Kruijff. *A Categorial-Modal Logical Architecture of Informativity: Dependency Grammar Logic & Information Structure*. PhD thesis, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic, 2001.

[5] G.J.M. Kruijff, P. Lison, T. Benjamin, H. Jacobsson, H. Zender, and I. Kruijff-Korbayová. Situated dialogue processing for human-robot interaction. In H.I. Christensen, G.J.M. Kruijff, and J. L. Wyatt, editors, *Cognitive Systems*, volume 8 of *Cognitive Systems Monographs, COSMOS*, chapter 8, pages 311–364. Springer Verlag, Berlin/Heidelberg, Germany, 2010.

[6] P. Lamere, P. Kwok, E. Gouvêa, B. Raj, R. Singh, W. Walker, M. Warmuth, and P. Wolf. The cmu sphinx-4 speech recognition system. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong*, 2003.

[7] M. Odersky, L. Spoon, and B. Venners. *Programming in Scala*. Artima Inc, 2008.

[8] M. Odersky and M. Zenger. Scalable component abstractions. In *ACM Sigplan Notices*, volume 40, pages 41–57. ACM, 2005.

[9] Marc Schröder and Jürgen Trouvain. The German text-to-speech synthesis system MARY: A tool for research, development and teaching. *International Journal of Speech Technology*, (6):365–377, 2003.

Int J Soc Robot (2012) 4:107–108
DOI 10.1007/s12369-012-0139-x

Expectations, Intentions, and Actions in Human-Robot Interaction

Marc Hanheide · Manja Lohse · Hendrik Zender

Human-robot interaction is becoming increasingly complex through the growing number of abilities, both cognitive and physical, available to today's robots. At the same time, interaction is still often difficult because the users do not understand the robots' internal states, expectations, intentions, and actions. Vice versa, robots lack understanding of the users' expectations, intentions, actions, and social signals.

Many studies have been conducted unveiling the importance of properly designed adaptive human-robot interaction strategies in general and appropriate feedback in particular. Robotic systems have been built that reflect the progress in the different fields in robotics with regard to learning, autonomous behaviours, safe navigation, and manipulation. However, integrated approaches to

- understanding the user and her expectations, intentions, and actions,
- transparently communicating to the user what the robot understood or expected,
- and designing appropriate robot behaviours based on its understanding of the world

are still in their infancy.

This special issue bundles recent advances in addressing these challenges. The key questions are how meeting or failing the user's expectations influences the efficiency and effectiveness of human-robot interaction; how more effective and efficient interaction with humans can be achieved using modalities available to a robot; how robots can be equipped with models enabling them to understand their users' state of mind; and similarly, how they can make their own expectations and states explicit through eligible communication channels.

Each of the seven articles in this issue highlights different aspects around the central theme of *expectations, intentions, and actions in human-robot interaction*. The topics covered range from recognition of verbal and non-verbal cues of intentions and expectations, to verbalisation and presentation techniques that make internal processing of the robot accessible to the human. In the following we give a brief overview of the contents of the articles.

Komatsu, Kurosowa, and Yamada ask "How does the Difference between Users' Expectations and Perceptions about a Robotic Agent Affect Their Behavior?" They call this difference the *adaption gap* and research how it affects the decisions of the users in a game. Their findings show that the participants with positive adaptation gap signs (i.e., the robot performed better than they expected) accepted the robot's suggestions more often than users with a negative adaptation gap sign (i.e., the robot performed worse than they expected). This work shows that expectations indeed play a major role in human-robot interaction.

The paper by Rosenthal, Veloso, and Dey deals with the problem of "Acquiring Accurate Human Responses to Robots' Questions." They identify and evaluate four

This work has been supported by the European Commision under contract number FP7-ICT-215181-CogX.

M. Hanheide
School of Computer Science, University of Lincoln, Lincoln, United Kingdom
E-mail: mhanheide@lincoln.ac.uk

M. Lohse
Research Institute for Cognition and Robotics (CoR-Lab), Applied Informatics, Bielefeld University, Bielefeld, Germany
E-mail: mlohse@techfak.uni-bielefeld.de

H. Zender
Language Technology Lab, German Research Center for Artificial Intelligence (DFKI), Saarbrücken, Germany
E-mail: zender@dfki.de

different types of state information that a robot can use to ground non-supervisors in its state when requesting help. The authors describe what they believe to be the most appropriate combination of information to communicate the robot's intentions and to trigger the right actions by the users.

Two papers in this issue focus on the *analysis of human behaviour* to facilitate *contingency* in interaction with a robot. Contingency is inherently related to expectations in interaction, as contingency can only be achieved if the mutual expectations of the interactants are met. However, the objectives of this analysis of contingency in the two articles differ: While Lohan *et al.* propose a model to detect tutoring turns in an interactive learning setting informed by adult-infant interaction, Lee *et al.* investigate the potential of multimodal cues to help turn-taking in natural interaction.

Focusing on the perception of intention and emotions, Yohanan and MacLean define different categories of *intent* in the interaction with an affective artificial creature. Those intents are related to emotional states that are haptically conveyed to a robot. The contribution proposes patterns of gesture use for emotional expression as well as a touch dictionary used to convey those different intents.

The paper by Lemaignan *et al.* entitled "Grounding the Interaction: Anchoring Situated Discourse in Everyday Human-Robot Interaction" presents a robotic system that can engage in *situated dialogues* with a human about shared tabletop environments. The system comprises perceptual modalities that allow it to build symbolic representations of the environment. One of the key aspects of the approach is that the system is capable of *perspective taking* – effectively allowing the robot to take into account the human's perspective as well as its own when interpreting and generating utterances.

Finally, Salem *et al.* present work on the generation of behaviour by the robot to facilitate the understanding of robot's intentions and expectations. They propose a novel *gesture generation* mechanism evaluated on a humanoid platform to produce *co-verbal* hand and arm gestures.

Some of these papers are extended versions of work presented in two preceding workshops, others have been submitted to an open call for papers. This special issue appears as a follow-up to the workshops on *Interactive Communication for Autonomous Intelligent Robots (ICAIR)*, held at the ICRA 2010 conference in Anchorage, AK, USA, and on *the Role of Expectations in Intuitive Human-Robot Interaction*, held at the HRI 2011 conference in Lausanne, Switzerland.

Therefore, we wish to express our gratitude to our colleagues who helped us organise these workshops as well as this special issue. Many thanks go to the co-organisers of our workshops: Verena Hafner (Humboldt University Berlin), Geert-Jan M. Kruijff (DFKI Saarbrücken), Joachim Meyer (Ben-Gurion University of the Negev), Yukie Nagai (Osaka University), and Britta Wrede (Bielefeld University). Special thanks go to Professor Shuzhi Sam Ge, who encouraged us to prepare this special issue based on the workshop outcomes, and to the reviewers for their support in maintaining the high standard of this issue: Kai Oliver Arras, Cindy L. Bethel, Laura Boccanfuso, Heriberto Cuayáhuitl, Frank Hegel, Thomas Holz, David O. Johnson, Takayuki Kanda, Alexandra Kirsch, Nate Koenig, Min Kyung Lee, Ross Mead, Yukie Nagai, Hirotaka Osawa, Julia Peltason, Annika Peters, Karola Pitsch, Astrid Marieke von der Pütten, Bogdan Raducanu, Subramanian Ramamoorthy, Laurel Riek, Pericle Salvini, Sven R. Schmidt-Rohr, Elin Anna Topp, Alan Richard Wagner, Yueh-Hsuan Weng, and Britta Wrede.