Decalog 2007: Proceedings of the 11th Workshop on the Semantics and Pragmatics of Dialogue, pages 17–24.
Trento, Italy, 30 May – 1 June 2007. Edited by Ron Artstein and Laure Vieu.

Incomplete Knowledge and Tacit Action: *Enlightened Update* in a Dialogue Game

Luciana Benotti
TALARIS Team - LORIA (Université Henri Poincaré, INRIA)
BP 239, 54506 Vandoeuvre-lès-Nancy, France
Luciana.Benotti@loria.fr

Abstract

This paper has two main aims. The first is to show how planning capabilities have been integrated into FrOz, a text adventure game presented in (Koller et al., 2004). Second, we demonstrate that the resulting system offers a natural laboratory for exploring the theory of enlightened update presented in (Thomason et al., 2006). In particular, we shall discuss how this theory applies in a setup with incomplete background knowledge.

1 Introduction

In this paper we investigate, in a simplified and formalised setup, how the information that allows two interlocutors to understand each other correctly is constructed and exploited during a conversation.

Let us start, right away, with an everyday example of the phenomenon we want to investigate. *A few days ago, my mother told my sister: "Please, buy some food for Tiffy." Then my sister took some money from a kitchen drawer, went to the grocery store that is near my primary school, bought a pack of low fat cat food with salmon flavour, and carried the food back home.* And this is exactly how my mother expected her to act. Why? Because both of them know that my sister is always low in cash, that at home there is always money in a particular kitchen drawer, that the grocery store near my primary school is the cheapest one, and that Tiffy is our pet cat, who is getting a bit fat and likes salmon. Is that all? Not quite. They also know that in order to buy something you need money, that in order to open a drawer you need to pull it, and many other things that are usually taken for granted.

Here, my mother and my sister exploited the large amount of information they share in order to leave several actions *tacit*. In conversation, this strategy is not merely valid, it is frequent and pervasive. We are going to investigate it in a 'dialogue game,' a conversational setup simplified in several ways. To start with, i) the interaction is restricted to a set of requests[1] between two interlocutors, with well defined preconditions and effects. Also, ii) the requests can be issued only by one of the interlocutors (who we will call 'the player'), the other (called 'the game') is limited to accepting and executing, or refusing the request. To complete the picture, iii) 'the game' has complete and accurate information about the conversational context (called 'the game scenario'), while 'the player' may have incomplete and even incorrect information.

Our setup is formalised in the implementation of a text adventure engine called FrOz Advanced (FrOzA). Text adventures are computer games that simulate a physical environment which can be manipulated by means of natural language requests (i.e., commands issued to the game). The system provides feedback in the form of natural language descriptions of the game world and of the results of the players' actions. FrOzA extends the text adventure FrOz (Koller et al., 2004) with planning capabilities. This added inference ability allows FrOzA to discover actions left tacit by the player.

This paper has two main aims. The first is to show (in Section 2) how planning capabilities can be integrated into the Description Logic (Baader et al., 2003) based inference architecture provided by FrOz. Second, we wish to demonstrate (in Section

[1] By 'request' we refer to the first part of an adjacency pair (request, acceptance/refusal) as defined in (Clark and Schaefer, 1989)

3) that the resulting system, namely FrOzA, offers a natural laboratory for the theory of *enlightened update* presented in (Thomason et al., 2006). The theory of enlightened update suggests how shared information (usually referred to as *common ground* (Clark, 1996)) is exploited and constructed in the light of a computational framework for reasoning in conversation. We use FrOzA not only to obtain a concrete account of enlightened update theory, but to extend it for handling incomplete background knowledge as well.

2 FrOzA

The architecture of FrOzA is shown in Figure 1; its three main processing modules are depicted as ellipses. The language understanding module parses the command issued by the player and constructs its semantic representation. The language generation module works in the opposite direction, verbalising the results of the execution of the command. The action handling module is in charge of performing the actions intended by the player.

All three modules make heavy use of inference services (represented as dashed lines in the figure) in order to query and update the components of a game scenario (depicted as rectangles). The processing modules are independent of particular game scenarios; by plugging in a different game scenario the player can play a different game.

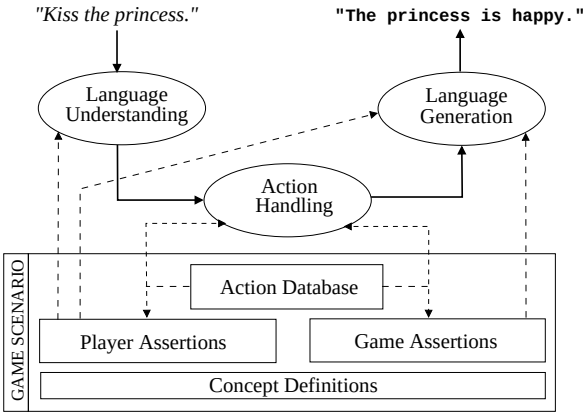

Figure 1: Architecture of FrOzA

In fact, it is in its reasoning abilities that FrOzA extends the original version of the system (FrOz). Thanks to its planning capabilities, FrOzA is able to discover actions intended by the player but left tacit by her. In order to infer these actions, FrOzA uses the planner Blackbox (Kautz and Selman, 1999). Like FrOz, FrOzA uses the theorem prover RACER (Haarslev and Möller, 2001) to query and modify the Description Logic knowledge bases according to the instructions encoded in the action database.

In the rest of the section we will describe the components of FrOzA that are relevant for the purposes of this paper. In Section 2.1 we will describe how FrOzA models a game scenario in its knowledge bases. In Section 2.2 and 2.3 we will explain in detail how actions are handled; in particular, we show how the execution of an action depends on the current game scenario, and how the successful execution changes the scenario. This will pave the way for our discussion of enlightened update in Section 3.

2.1 Modelling a game scenario

FrOzA uses Description Logic (DL) knowledge bases (KB) to codify assertions and definitions of the concepts relevant for a given game scenario. A DL knowledge base is a pair (TBox, ABox) where the TBox is a set of definitions and the ABox a set of assertions about the objects being described in the KB (such objects are usually called individuals). Actually, FrOzA uses two knowledge bases, which share the TBox and differ only in their ABoxes. The common TBox defines the key concepts in the game world and how they are interrelated. Some of these concepts are basic notions (such as *object*) or properties (such as *alive*), directly describing the game world, while others define more abstract notions like the set of all the individuals a player can interact with (the individuals that are *accessible* to the player).

The ABoxes specify properties of particular individuals (for example, an individual can be an *apple* or a *player*). Relationships between individuals are also represented here (such as the relationship between an object and its location).

One of the knowledge bases (the game KB) represents the *true state* of the game world, while the other (the player KB) keeps track of the player's *beliefs* about the game world. In general, the player KB will not contain all the information in the game KB because the player will not have explored the world completely, and therefore will not know about all the individuals and their properties. In fact, it might also be the case that the player KB contains information that is inconsistent with the game KB. The game can deliberately hide effects of an action from the player; pushing a button

might have an effect that the player cannot see.

Crucially, a game scenario also includes the definitions of the actions that can be executed by the player (such as the actions *take* or *eat*). Each action is specified (in the action database) as a STRIPS-like operator (Fikes et al., 1972) detailing its arguments, preconditions and effects. The preconditions indicate the conditions that the game scenario must satisfy so that the action can be executed; the effects determine how the action changes the game scenario when it is executed.

2.2 Handling a single action

In this section we are going to explain in detail how an action issued by the player can change the game scenario.

To illustrate our explanation, let us consider a concrete input and analyse how it is handled by the system. Suppose that the player has just said "Take the key." The semantic representation of this command (obtained by the language understanding module) will be the ground term `take(key1)` (where `key1` represents the only key that the player can see in the current state of the game). This ground term will be passed to the next processing module in the architecture.

When a ground term is received by the action handling module, it is matched against the list of action schemas. The action schema that will match the ground term of our example is:

```
action:
  take(X)
preconditions:
  accessible(X),
  takeable(X),
  not(in-inventory(X))
effects:
  add: in-inventory(X)
  del: has-loc(X indiv-filler(X has-loc))
player effects:
  add: in-inventory(X)
  del: has-loc(X indiv-filler(X has-loc))
```

The term `X` in the above schema is a variable that gets bound to the actual argument of the action. In our example, `X` would be bound to the constant `key1`, and thus the preconditions and effects will become ground terms. Once the action schema is instantiated, it is time to check that the action can be executed. An action can be executed if all its preconditions are satisfied in the current game KB. The preconditions can require that individuals belong to certain concepts or that they are related by certain roles. For example, the execution of the action `take(key1)` requires that the key is accessible to the player (`accessible(key1)`), that it is small enough to be taken (`takeable(key1)`) and that it is not carried by the player already (`not(in-inventory(key1))`). The theorem prover RACER is used to query the current *game KB*, thereby checking that the preconditions are satisfied.

If the action can be executed, the *game KB* is updated according to the effects of the action. In our example, the key will no longer be in its original location but it will be carried by the player. The original location of the key is obtained by sending the query `indiv-filler(key1 has-loc)` to RACER. A RACER query is embedded in an action schema when the action depends on properties of individuals not explicitly mentioned in the player command (such as the location of the key).

Once the game executed the action, the player needs to know that the action succeeded. To this end, the player effects in the action schema are communicated to the player by the generation component and asserted in the *player KB*.

If the command cannot be executed in the current game scenario, the first precondition that failed is communicated to the player and both KBs remain unchanged.

2.3 Interpreting the player intention

Now that we know how the actions module handles a simple action, let us explain how *ambiguous commands* and *tacit actions* are handled in FrOzA.

The input of the action module is not a single ground term but a list of possible readings of the input sentence. The list will contain exactly one reading only if the sentence is not ambiguous (as in the example in the previous section). Otherwise, the list will contain one entry for each different reading. For example, the sentence "Unlock the door with the key" is syntactically ambiguous and has two possible readings, one in which the propositional phrase "with the key" modifies the verb "unlock" and another in which it modifies the noun phrase "the door." Sentences can also be referentially ambiguous. For instance, the sentence "Take it" has as many readings as there are salient referents in the game scenario. Each reading is itself a list which represents a sequence of actions to be performed one after the other. For example, every reading of the sentence "Take the key and unlock the door with it" will contain two ground terms, one for each action in the sequence.

If the input sentence has more than one reading, FrOzA decides among them by trying each action sequence in parallel. When an action fails, the entire reading it belongs to is discarded. For example, the reading of the command "Take it and eat it" which resolves both occurrences of "it" to a key, will be discarded because a key is not edible, although it can be taken.

If only one reading succeeds, the game assumes that this is the command the player had in mind, and commits to the end result of the sequence. If more than one sequence is possible, the game reports an unresolved ambiguity. For instance, the game will report an ambiguity if both readings of the command "Unlock the door with the key" are executable in the current game scenario.

The inference capabilities discussed so far are common to FrOz and FrOzA; we now turn to what sets FrOzA apart and will lead us to discuss the theory of enlightened update: *planning capabilities*. Planning is used when no reading is executable, for analysing whether the command includes *tacit actions*. For each failed reading FrOzA tries to find a *sequence of actions* (i.e., a *plan*) which transforms the current game scenario into a scenario where the reading can succeed. If no such plan exists, the reading is discarded, otherwise the plan is concatenated before the reading, enlarging the original sequence of actions. The new list of readings built in this way is reinserted into the action handling module and its execution proceeds as usual.

In order to illustrate the previous behaviour of FrOzA, let us consider again the command "Unlock the door with the key" but now suppose that none of its two readings is executable in the current game scenario. One of the readings fails because there is no "door with the key" in the current game scenario. The other reading cannot be directly executed because the key is not in the player's hands but on a table in front of her. However, for this second reading a plan can be found, namely "to take the key" before unlocking the door; although "take the key" was left tacit by the player, it can be inferred from the game scenario. This plan is concatenated before the original reading and the extended reading is processed again by the action handling module. This time, the input of the action module will be the sequence of actions "Take the key and unlock the chest with it", making explicit the tacit action.

In order to infer tacit actions, FrOzA uses the planning services provided by the planner Blackbox (Kautz and Selman, 1999). Blackbox works by fixing the length of the plan in advance and iteratively deepening it. This behaviour makes it particularly well suited for our needs because it finds optimal plans (minimal in the number of actions) and does it fast. Fast responses are essential for a natural interaction with the player. For a detailed description of the performance of Blackbox in FrOzA see (Benotti, 2006a; Benotti, 2006b). Moreover, optimal plans are crucial, otherwise actions which are executable in the game scenario but completely irrelevant to the player command might be included as tacit actions. For example, a non-optimal planner might not only "take the key" as in our example, but also take and drop other arbitrary objects as well.

The input required by Blackbox are STRIPS-style problems specified in the Planning Domain Definition Language (Gerevini and Long, 2005) which includes the standard elements of a planning specification: the initial state, the available actions, and the goal.

In next section we will present a theoretical account of the intuitions hinted at here by making use of the insights provided by the theory of enlightened update. In particular, we will analyse what information the elements of the planning specifications should contain.

3 Enlightened update in FrOzA

We will now use FrOzA as a laboratory for exploring the theory of enlightened update (Thomason et al., 2006). Using FrOzA we shall construct, step by step, an accurate account of the main principles behind this theory.

The intuition behind enlightened update theory is that when the speaker utters a sentence, as my mother did in our first example, she is not only trying to achieve the obvious effects of the utterance, but is also communicating the ways in which she assumes the world to be, and on which the success of the utterance depends.

Let us make this approach concrete through an example in our game setup. Suppose that the player is inside a room with a locked door while she is holding a golden key in her hands. Then she inputs the command "Unlock the door with the golden key," which is mapped to the semantic representation `unlock(door1 key1)`. The intention

behind this utterance is twofold. It is clear that the player wants the game state to be updated according to the effects of the action, that is, she wants to have the door unlocked. But the player also expects the game to recognise the assumptions she is making and on which the success of the utterance depends. In particular, she assumes that the golden key fits into the door lock.

This strategy for updating the shared knowledge is stated formally as the following principle:

ENLIGHTENED UPDATE (EU): "An agent's public performance of an action [A] that is mutually known to require a commitment C for its successful performance will add to the mutual information the proposition that the agent believes C." (Thomason et al., 2006, p.15).

It is important to notice that in order to be able to perform an EU it must be mutually known that the action, which is being performed publicly, requires its preconditions. In our setup this means that we assume that the exact preconditions required for the successful performance of the action `unlock` are mutually known (by the player and the game). Such an assumption is represented in the action schema below, which specifies the player preconditions to be equal to the original preconditions of the action.

```
action:
  unlock(door1 key1)
preconditions:
  locked(door1), key(key1),
  in-inventory(key1), fits-in(key1 door1)
player preconditions:
  locked(door1), key(key1),
  in-inventory(key1), fits-in(key1 door1)
effects:
  del: locked(door1)
  add: unlocked(door1)
player effects:
  del: locked(door1)
  add: unlocked(door1)
```

After this action (`unlock(door1 key1)`) is executed successfully, the player will believe that "the golden key" is a key, and that it is in her hands, facts that she already knew. However, she will also believe that the door is now unlocked, the obvious effect of the action; and that the golden key fits in the door lock, the assumption she made and was confirmed by the success of the action. This means that, when an action is executed, the *player KB* will be updated not only with the effects of the action but also with its preconditions. When performing this update, the order in which the changes are made is important in order to leave the KB in the intended state. Concretely, the KB should be first updated with the player preconditions and then with the player effects. Otherwise, the preconditions might undo the effects of the action. Moreover, the updates that retract information from the KB have to be performed before the ones that assert information, in order to avoid introducing an inconsistency in the KB.

This is the easy case, but what if the action cannot be directly executed (that is, some of its preconditions are false) in the current game scenario? The EU principle extends naturally to cover these cases. And, in fact, these are the cases where the theory of enlightened update is able to bridge the gaps that arise in everyday interactions.

3.1 Enlightened Update with Tacit Actions

To analyse how the EU principle is extended, let us modify our running example a bit in order to return to the game scenario we analysed intuitively in Section 2.3. Suppose that the player does not have a key and she is looking around searching for a way to unlock the door when the game says that there is a golden key lying on a table in front of her. Then she inputs the command "Unlock the door with the golden key." Hence, according to the EU principle, the player knowledge base should be updated with the preconditions of the action. However, one of the preconditions of this action, namely `in-inventory(key1)`, is false in the current game scenario (that is, in both KBs). Clearly, the precondition cannot just be added to the player KB because this will cause a contradiction, but this precondition can be *made* true in the game scenario by *performing the appropriate actions*.

The theory of enlightened update defines the following refinement of the EU pattern to handle exactly this situation:

EU WITH TACIT ACTIONS (EU/TA): "[Assume that] C is a condition that can be manipulated by an audience H. An agent S is observed by H to be doing A while C is mutually known to be false. H then acts [tacitly] to make C true, and S expects H to so act." (Thomason et al., 2006, p.36)

In our example, the player is not holding the key and she knows it, and she is trying to unlock the door anyway, knowing that in order to unlock a door you need to have the key in your hands. Hence, FrOzA should act to make `in-inventory(key1)` true. And it does so by executing tacitly the action `take(key1)`.

We should notice here that an action can be left tacit by the speaker, and recognised correctly by the hearer, only if the *effects* of the action are *mutually known* by the conversation partners.

3.2 Enlightened Update and Incomplete Background Knowledge

In (DeVault and Stone, 2006) the theory of Enlightened Update is implemented and tested in COREF, a conversational agent which uses enlightened update to interactively identify visual objects with a human user. In FrOzA we also implement and test the theory of enlightened update but with an added kind of uncertainty: incomplete background knowledge. In COREF, both interlocutors are assumed to have the same background information. In FrOzA, on the other hand, the game has complete and accurate information about the game world, while the player starts the game without information and acquires it as the game evolves. In this setup, modelling enlightened update highlights the issues involved when one of the interlocutors has incomplete background knowledge. Moreover, it illustrates the point that, as conversation evolves, background knowledge accumulates and that a conversational system can use this information to engage in more flexible and robust conversation.

The key question in FrOzA is how it is able to infer the 'appropriate' tacit actions in a setup with incomplete background knowledge. In principle, it just needs to provide Blackbox with the 'appropriate' inputs mentioned in Section 2.3: the initial state, the goal, and the available actions. However, the question of 'what these three elements should contain' raises a number of subtle issues. Their discussion will highlight the kinds of problems that need to be considered when background knowledge is incomplete.

3.2.1 The initial state

The first question is to decide the information that is needed for the initial state. In FrOzA, two types of information are registered: the objective information in the game KB and a subjective view in the player KB. Which of these should be used in order to discover tacit actions? In fact, we need both. Let us analyse this decision by extending our example once again. Suppose that the golden key, which was lying on the table, was taken by a thief without the player knowing. As a consequence, the key is on the table in the player KB, but in the game KB the thief has it. In this new scenario, the player issues the command "Unlock the door with the golden key." If we included in the initial state the objective information of the game KB, FrOzA would automatically take the key from the thief (for example, by using the steal action) and unlock the door for the player, while the player is not even aware where the key actually was. This is clearly inappropriate. Now, if our initial state includes the information in the player KB, FrOzA would decide to take the key from the table and unlock the door with it. But this sequence of actions is not executable in the game world because the key is no longer accessible (the thief has it). More generally, a sequence of tacit actions found by reasoning over the player KB might not be executable in the game world because the player's KB may contain information that is inconsistent with respect to the game KB. Hence, we need both KBs: we infer the actions intended by the player using the information in her KB but we have to verify this sequence of actions on the game KB to check if it can actually be executed. The *action inference* step is done using the planning services provided by Blackbox on the subjective information, and the *action executability* step is done using the reasoning services provided by RACER on the objective information. In COREF, by way of contrast, once the tacit actions are inferred they do not need to be checked for executability on the objective information. This is because the COREF setup does not allow any of the interlocutors to have a subjective view of the information; both interlocutors are assumed to share the objective information and hence, tacit actions are inferred solely on the basis of objective information.

An interesting consequence of the fact that the FrOzA setup handles incomplete background knowledge is that we can investigate how this background knowledge accumulates and how it affects the interaction. And it turns out that the more the player knows about the game world, the more actions can be left tacit. For example, suppose that after opening the door, the player locked it behind her and continued to the following rooms investigating the game world. After a while she is back and wants to open the door again. This time it is enough for her to say "Open the door", instead of "Unlock the door with the golden key", because she already knows, and the game knows that she knows, which key fits into this lock.

As a consequence, we can drop a simplifying assumption made in (Thomason et al., 2006), namely that whether an action is public or tacit is a *static* matter, corresponding to an arbitrary split in the action database. In FrOzA this distinction is *dynamic* and correlates with the growth of background information.

3.2.2 The goal

The two remaining questions are what the goal and the actions of the planning problem should be. We believe that answering these two questions is also non-trivial, as it was not trivial to define the initial state. However, we have not yet analysed the subtleties involved in these two issues; here we simply present our initial approach.

Let us start defining what the goal should be. According to EU principles, the game should act to make the preconditions of the action true with two restrictions. First, it must be possible for the game to manipulate the preconditions. And second, the action must be mutually known to require its preconditions. Hence, we define the goal as the player preconditions of the action commanded by the player, excluding those that cannot be manipulated by the actions in the action database.

For example, when the player says "Unlock the door with the key" the goal of the planning problem will only include the atoms:

```
locked(door1),
in-inventory(key1),
```

The preconditions that cannot be manipulated by the actions available in the action database, such as `key(key1)` (something that its not a key cannot be transformed into one) and `fits-in(key1 door1)` (if the key does not fit into the lock it is not possible to make it fit) are not included in the goal.

3.2.3 The actions

To complete the picture, the actions available to the planner are all the actions in the game action database. Its preconditions will correspond to the player preconditions and its effects to the player effects. For the moment, we are assuming that the preconditions and the effects of the actions are shared by the game and the player. Hence, the player preconditions and the preconditions of an action coincide; as well as the player effects and the effects. Relaxing this simplifying assumption, would introduce more dynamism in the distinction between tacit and public actions, and hence would better reflect the case of real conversation.

4 Conclusions

In this paper we have described FrOzA and used it to explore the enlightened update theory.

The FrOzA setup shows that enlightened update can be implemented using an off-the-shelf reasoning tool such as Blackbox. At present, the solution provided by this setup is not logically complete because our two inference tools (RACER and Blackbox) work independently and are not capable of sharing information (see (Benotti, 2006b) for the technical details). However, we believe that the present implementation is the kind of laboratory that theories such as enlightened update needs. We leave the study of complete reasoning mechanisms and a comparison between our setup and the one implemented in (Thomason et al., 2006) for further research. We mention in passing that integrating planning capabilities in the framework of a Description Logic reasoner is a topic of current research (see (Baader et al., 2005; Liu et al., 2006)).

We have tested the theory of enlightened update with an added kind of uncertainty common in conversation: incomplete background knowledge. This test yielded two interesting consequences. First, the theory applies but raises a number of subtle issues on the kind of required information, and second, the division between tacit and public actions becomes dynamic. In (Thomason et al., 2006) whether an action is public or tacit is a static matter, corresponding to an arbitrary split in the action database. In FrOzA, this distinction correlates with the growth of background information; we believe this to be in line with 'the granularity of conversation' as defined in (van Lambalgen and Hamm, 2004) another point which requires further work.

But more remains to be done. There is a deeper kind of incomplete background knowledge, namely when the action preconditions and effects are not mutually known, i.e. when the task model is not shared by the interlocutors. We believe that accounting with such uncertainty in a conversation is one of the most challenging problems that theories such as enlightened update face nowadays.

Acknowledgements

I would like to thank the members of the Dialogue Discussion Group of the TALARIS Team at LO-

RIA for an enthusiastic welcome, and lively discussions.

References

Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Franz Baader, Carsten Lutz, Maja Milicic, Ulrike Sattler, and F. Wolter. 2005. Integrating description logics and action formalisms: First results. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, Pittsburgh, PA, USA.

Luciana Benotti. 2006a. "DRINK ME": Handling actions through planning in a text game adventure. In Janneke Huitink and Sophia Katrenko, editors, *XI ESSLLI Student Session*, pages 160–172.

Luciana Benotti. 2006b. Enhancing a dialogue system through dynamic planning. Master's thesis, Universidad Politécnica de Madrid.

Herbert Clark and Edward Schaefer. 1989. Contributing to discourse. *Cognitive Science*, 13(2):259–294.

Herbert Clark. 1996. *Using Language*. Cambridge University Press, New York.

David DeVault and Matthew Stone. 2006. Scorekeeping in an uncertain language game. In *The 10th Workshop on the Semantics and Pragmatics of Dialogue (Brandial 2006)*, University of Potsdam, Germany.

Richard Fikes, Peter Hart, and Nils Nilsson. 1972. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288.

Alfonso Gerevini and Derek Long. 2005. Plan constraints and preferences in PDDL3. Technical Report R.T. 2005-08-47, Università degli Studi di Brescia, Italy.

Volker Haarslev and Ralf Möller. 2001. RACER system description. In *Proceedings of International Joint Conference on Automated Reasoning (IJCAR 01)*, number 2083 in LNAI, pages 701–705, Siena, Italy.

Henry Kautz and Bart Selman. 1999. Unifying SAT-based and graph-based planning. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 99)*, pages 318–325, Stockholm, Sweden.

Alexander Koller, Ralph Debusmann, Malte Gabsdil, and Kristina Striegnitz. 2004. Put my galakmid coin into the dispenser and kick it: Computational linguistics and theorem proving in a computer game. *Journal of Logic, Language and Information*, 13(2):187–206.

Hongkai Liu, Carsten Lutz, Maja Milicic, and Frank Wolter. 2006. Reasoning about actions using description logics with general TBoxes. In Michael Fisher, Wiebe van der Hoek, Boris Konev, and Alexei Lisitsa, editors, *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA 2006)*, volume 4160 of *Lecture Notes in Artificial Intelligence*, pages 266–279. Springer-Verlag.

Richmond Thomason, Matthew Stone, and David DeVault. 2006. Enlightened update: A computational architecture for presupposition and other pragmatic phenomena. In Donna Byron, Craige Roberts, and Scott Schwenter, editors, *Presupposition Accommodation*. Ohio State Pragmatics Initiative, draft Version 1.0.

Michiel van Lambalgen and Fritz Hamm. 2004. *The Proper Treatment of Events*. Blackwell.