Frolog: an accommodating text-adventure game

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

Abstract

Frolog is a text-adventure game whose goal is to serve as a laboratory for testing pragmatic theories of accommodation. To this end, rather than implementing ad-hoc mechanisms for each task that is necessary in such a conversational agent, Frolog integrates recently developed tools from computational linguistics, theorem proving and artificial intelligence planning.

1 Introduction

If we take a dialogue perspective on Lewis' (1979) notion of *accommodation* and assume that the state of a dialogue is changed by the acts performed by the dialogue participants, it is natural to interpret Lewis' broad notion of accommodation as *tacit (or implicit) dialogue acts*. This is the approach adopted by Kreutel and Matheson (2003) who formalize the treatment of tacit dialogue acts in the information state update framework. According to them, accommodation is ruled by the following principle:

Context Accommodation (CA): For any move m *that ocurrs in a given scenario* sc_i*: if assignment of a context-dependent interpretation to* m *in* sc_i *fails, try to accommodate* sc_i *to a new context* sc_{i+1} *in an appropriate way by assuming implicit dialogue acts performed in* m*, and start interpretation of* m *again in* sc_{i+1}.

The authors concentrate on the treatment of implicit acceptance acts but suggest that the CA principle can be seen as a general means of context-dependent interpretation. This principle opens up the question of how to find the appropriate tacit dialogue acts. Finding them is an inference problem that is addressed using special-purpose algorithms in (Thomason et al., 2006), where the authors present a unified architecture for both context-dependent interpretation and context-dependent generation. In Frolog, we investigate how this inference process can be implemented using recent tools from *artificial intelligence planning*.

The resulting framework naturally lends itself to studying the pressing problem for current theories of accommodation called *missing accommodation* (Beaver and Zeevat, 2007). These theories can neither explain *why* accommodation is sometimes easier and sometimes much more difficult, nor *how* cases of missing accommodation relate to clarification subdialogues in conversation. We review what Frolog has to offer to the understanding of accommodation in general and missing accommodation in particular in Section 3. But first, we have to introduce Frolog and describe its components, and we do so in Section 2.

2 The text-adventure game

Text-adventures are computer games that simulate a physical environment which can be manipulated by means of natural language requests. The game provides feedback in the form of natural language descriptions of the game world and of the results of the players' actions.

Frolog is based on a previous text-adventure called FrOz (Koller et al., 2004) and its design is depicted in Figure 1. The architecture is organized in three natural language understanding (NLU) modules and three natural language generation (NLG) modules, and the state of the interaction is represented in two knowledge bases (KBs). The two KBs codify, in Description Logic (Baader et al., 2003), assertions and concepts relevant for a given game scenario. The *game KB* represents the true state of the game world, while the *player KB* keeps track of the player's beliefs about the game world. Frolog's modules are scenario-independent; the player can play different game scenarios by plugging in the different information resources that constitute the scenario.

Frolog uses generic external tools for the most heavy-loaded tasks (depicted in grey in Figure 1);

Proceedings of the EACL 2009 Demonstrations Session, pages 1–4,
Athens, Greece, 3 April 2009.

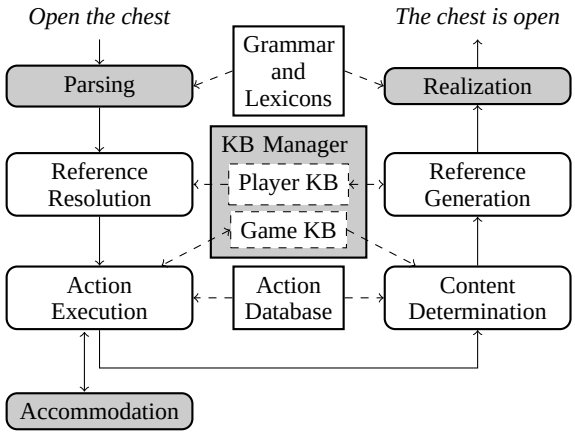

Figure 1: Architecture of Frolog

namely, a generic parser and a generic realizer for parsing and realization, an automated theorem prover for knowledge base management, and artificial intelligence planners for implementing its accommodating capabilities. The rest of the modules (depicted in white) were implemented by us in Prolog and Java. Frolog's interface shows the interaction with the player, the input/output of each module and the content of the KBs.

We now present Frolog's modules in pairs of an NLU module and its NLG counterpart; each pair uses a particular kind of information resource and has analogous input/output.

2.1 Parsing and Realization

The parsing and the realization modules use the same linguistic resources, namely a reversible grammar, a lemma lexicon and a morphological lexicon represented in the XMG grammatical formalism (Crabbé and Duchier, 2004). The XMG grammar used specifies a Tree Adjoining Grammar (TAG) of around 500 trees and integrates a semantic dimension à la (Gardent, 2008). An example of the semantics associated with the player input "open the chest" is depicted in Figure 2.

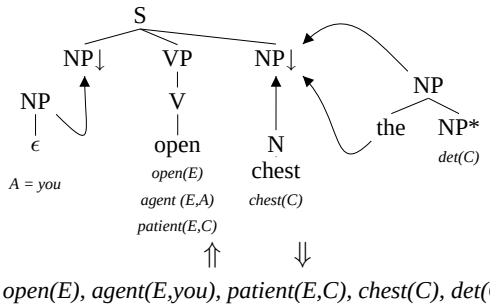

Figure 2: Parsing/realization for "open the chest"

The parsing module performs the syntactic analysis of a command issued by the player, and constructs its semantic representation using the TAG parser Tulipa (Kallmeyer et al., 2008) (illustrated in the Figure 2 by ⇓). The realization module works in the opposite direction, verbalizing the results of the execution of the command from the semantic representation using the TAG surface realizer GenI (Gardent and Kow, 2007) (illustrated in the Figure 2 by ⇑).

2.2 Reference Resolution and Reference Generation

The reference resolution (RR) module is responsible for mapping the semantic representations of definite and indefinite noun phrases and pronouns to individuals in the knowledge bases (illustrated in Figure 3 by ⇓). The reference generation (RG) module performs the inverse task, that is it generates the semantic representation of a noun phrase that uniquely identifies an individual in the knowledge bases (illustrated in the Figure 3 by ⇑). The algorithms used for RR and RG are described in (Koller et al., 2004).

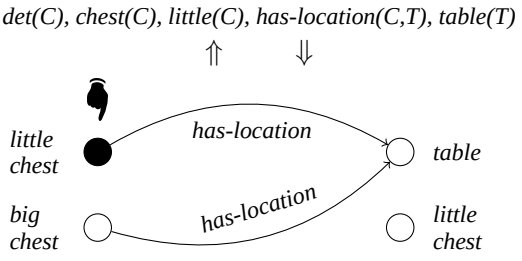

Figure 3: RR/RG for "the little chest on the table"

Frolog uses the theorem prover RACER (Haarslev and Möller, 2001) to query the KBs and perform RR and RG. In order to manage the ambiguity of referring expressions two levels of saliency are considered. The player KB is queried (instead of the game KB) naturally capturing the fact that the player will not refer to individuals he doesn't know about (even if they exist in the game KB). Among the objects that the player already knows, a second level of saliency is modelled employing a simple stack of discourse referents which keeps track of the most recently referred individuals. A new individual gets into the player KB when the player explores the world.

2.3 Action Execution and Content Determination

These two last modules share the last information resource that constitute an scenario, namely, the action database. The action database includes the definitions of the actions that can be executed by the player (such as *take* or *open*). Each action is specified as a STRIPS-like operator (Fikes et al., 1972) detailing its arguments, preconditions and effects as illustrated below. The arguments show the thematic roles of the verb (for instance, the verb open requires a patient and an agent), the preconditions indicate the conditions that the game world must satisfy so that the action can be executed (for instance, in order to open the chest, it has to be accessible, unlocked and closed); the effects determine how the action changes the game world when it is executed (after opening the chest, it will be open).

action: open(E) agent(E,A) patient(E,P)
preconditions: accessible(P), not(locked(P)), closed(P)
effects: opened(P)

Executing a player's command amounts to verifying whether the preconditions of the actions involved by the command hold in the game world and, if they do, changing the game KB according to the effects. After the command is executed, the content determination module constructs the semantic representation of the effects that were applied, updates the player KB with it and passes it to the next module for its verbalization (so that the player knows what changed in the world). For our running example the following modules will verbalize "the chest is open" closing a complete cycle of the system as illustrated in Figure 1.

If a precondition of an action does not hold then Frolog tries to accommodate as we will explain in following section.

3 Accommodation in Frolog

In the previous section we presented the execution of the system when everything "goes well", that is (to come back to the terminology used in Section 1) when the assignment of a context-dependent interpretation to the player's move succeeds. However, during the interaction with Frolog, it often happens that the player issues a command that cannot be directly executed in the current state of the game but needs accommodation or clarification. This is the topic of the next two subsections.

3.1 Tacit acts are inferable and executable: accommodation succeeds

Suppose that the player has just locked the little chest and left its key on the table when she realizes that she forgot to take the sword from it, so she utters "open the chest". If Frolog is in its non-accommodating mode then it answers "the chest is locked" because the precondition *not(locked(P))* does not hold in the game world. In this mode, the interactions with the game can get quite long and repetitive as illustrated below.

Non-accommodating mode	Accommodating mode
P: open the chest	P: open the chest
F: the chest is locked	F: the chest is open
P: unlock it	
F: you don't have the key	
...	

In its accommodating mode, Frolog tries to accommodate the current state sc_i of the game to a new state sc_{i+1} in which the precondition hold, by assuming tacit dialogue acts performed, and starts the interpretation of the command again in sc_{i+1}. That is, the game assumes that "take the key and unlock the chest with it" are tacit acts that are performed when the player says "open the chest".

The inference of such tacit dialogue acts is done using artificial intelligence planners. The planning problems are generated on the fly during a game each time a precondition does not hold; the initial state being the player KB, the goal being the precondition that failed, and the action schemas those actions available in the action database. The size of the plans can be configured, when the length is zero we say that Frolog is in its non-accommodating mode. For detailed discussion of the subtleties involved in the kind of information that has to be used to infer the tacit acts see (Benotti, 2007).

Two planners have been integrated in Frolog (the player can decide which one to use): Blackbox (Kautz and Selman, 1999) which is *fast and deterministic* and PKS (Petrick and Bacchus, 2004) which can reason over *non-deterministic actions*. For detailed discussion and examples including non-deterministic actions see (Benotti, 2008).

3.2 Accommodation fails: clarification starts

Tacit acts are inferred using the information available to the player (the player KB) but their execution is verified with respect to the accurate and complete state of the world (the game KB). So

Frolog distinguishes three ways in which accommodation can fail: there is no plan, there is more than one plan, or there is a plan which is not executable in the game world. For reasons of space we will only illustrate the last case here.

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 situation, the player issues the command "Open the chest" and the sequence of tacit acts inferred (given the player beliefs) is "take the key from the table and unlock the chest with it". When trying to execute the tacit acts, the game finds the precondition that does not hold and verbalizes it with "the key is not on the table, you don't know where it is". Such answer can be seen as a clarification request (CR), it has the effect of assigning to the player the responsability of finding the key before trying to open the chest. The same responsability that would be assigned by more commonly used CR that can happen in this scenario, namely "Where is the key?".

In the game, such clarifications vary according to the knowledge that is currently available to the player. If the player knows that the dragon has the key and she can only take it while the dragon is asleep an answer such as "the dragon is not sleeping" is generated in the same fashion.

4 Conclusion and future work

In this paper we have presented a text-adventure game which is an interesting test-bed for experimenting with accommodation. The text-adventure framework makes evident the strong relation between accommodation and clarification (which is not commonly studied), highlighting the importance of investigating accommodation in dialogue and not in isolation.

Our work is in its early stages and can be advanced in many directions. We are particularly interested in modifying the architecture of the system in order to model reference as another action instead of preprocessing references with special-purpose algorithms. In this way we would not only obtain a more elegant architecture, but also be able to investigate the interactions between reference and other kinds of actions, which occur in every-day conversations.

References

F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

D. Beaver and H. Zeevat. 2007. Accommodation. In *The Oxford Handbook of Linguistic Interfaces*, pages 503–539. Oxford University Press.

L. Benotti. 2007. Incomplete knowledge and tacit action: Enlightened update in a dialogue game. In *Proc. of DECALOG*, pages 17–24.

L. Benotti. 2008. Accommodation through tacit sensing. In *Proc. of LONDIAL*, pages 75–82.

B. Crabbé and D. Duchier. 2004. Metagrammar redux. In *Proc. of CSLP04*.

R. Fikes, P. Hart, and N. Nilsson. 1972. Learning and executing generalized robot plans. *AI*, 3:251–288.

C. Gardent and E. Kow. 2007. A symbolic approach to near-deterministic surface realisation using tree adjoining grammar. In *Proc. of ACL07*.

C. Gardent. 2008. Integrating a unification-based semantics in a large scale lexicalised tree adjoininig grammar for french. In *Proc. of COLING08*.

V. Haarslev and R. Möller. 2001. RACER system description. In *Proc. of IJCAR01*, number 2083 in LNAI, pages 701–705.

L. Kallmeyer, T. Lichte, W. Maier, Y. Parmentier, J. Dellert, and K. Evang. 2008. TuLiPA: Towards a multi-formalism parsing environment for grammar engineering. In *Proc. of the WGEAF08*.

H. Kautz and B. Selman. 1999. Unifying SAT-based and graph-based planning. In *Proc. of IJCAI99*, pages 318–325.

A. Koller, R. Debusmann, M. Gabsdil, and K. Striegnitz. 2004. Put my galakmid coin into the dispenser and kick it: Computational linguistics and theorem proving in a computer game. *JoLLI*, 13(2):187–206.

J. Kreutel and C. Matheson. 2003. Context-dependent interpretation and implicit dialogue acts. In *Perspectives on Dialogue in the New Millenium*, pages 179–192. John Benjamins.

D. Lewis. 1979. Scorekeeping in a language game. *Philosophical Logic*, 8:339–359.

R. Petrick and F. Bacchus. 2004. Extending the knowledge-based approach to planning with incomplete information and sensing. In *Proc. of ICP-KRR04*, pages 613–622.

R. Thomason, M. Stone, and D. DeVault. 2006. Enlightened update: A computational architecture for presupposition and other pragmatic phenomena. In *Proc. of Workshop on Presup. Accommodation*.