Cognitive Dialogue Management

Vincenzo Pallotta

Laboratory of Theoretical Computer Science
Faculty of Information and Computer Science
Swiss Federal Institute of Technology Lausanne

Abstract. Cognitive Dialogue Management is a novel approach to Dialogue Management which proposes a novel architecture for building up advanced interactive interfaces in natural language to computers, that are both flexible and robust. The architecture, based on dialogue, results from the integration of the Information State Dialogue Management model and the ViewFinder framework for the management of mental representations. This paper describes the principles of Cognitive Dialogue Management and provides hints about how metaphors of human information processing can be used both for improving the degree of communication understanding in human-computer interfaces, and for their rational design and development.

1 Introduction

Dialogue Models have recently gained a great interest in Computational Linguistics since they offer a natural framework both for the analysis of human dialogues and the design of man-machine interfaces. A common issue of these tasks is how to model the various types of interaction that happens among artificial and human agents at different levels of abstraction. There are often situations where the literal meaning is not sufficient to understand the role of the utterance in the dialogue, when the corresponding dialogue act cannot be directly recognized by simply looking at its linguistic content, but it must be inferred. This is achieved by means of their *cognitive skills*: their abilities to perform inferences based on *background knowledge* and *assumptions* on the other participants' mental states.

1.1 Cognitive Approaches to Dialogue Management

An important issue in the study of the behaviour of an intelligent agent is the logical link between perception, cognition, and action. Communication among agents using the language is a natural phenomenon, where the speaker acts by affecting the environment in which both the speaker and the hearer are situated. The hearer exploits its perceptive capabilities in order to detect which *changes* have been produced by the speaker's communicative actions. From the hearer's point of view, the environment is made of everything that he/she/(it) is able to perceive, including the speaker agent. This means that the hearer (and symmetrically the speaker) should be able to *access* the changing features of the environment, including the features of the speaker's act of speaking (i.e. the utterance). Not all the environment's features are directly or equally accessible by perception. Moreover, the perception system may be insufficient in extracting all the necessary information about the communicative event or worse, it is not capable of focusing enough on relevant parts of the phenomenon that enable the understanding of the speaker's communication[1].

In this paper we try to provide a more satisfactory answer to this question by presenting a systematic approach to Dialogue Modelling that takes into account the above issues and provides a cognitive model of Natural Language Communication based on the notion of Mental State. If our main goal is to design artificial systems capable of understanding natural language communication, we should at least imagine how cognitive processes can be mapped into a *computational architecture*. Note that we are taking a different perspective than that of the classical Cognitive Science: we do not commit ourselves with the view that human cognitive processes are computational in nature, but we rather propose that intelligent artificial systems should be modeled as cognitive entities.

1.2 Mental States

The notion of a *Mental State* is fundamental and it is advocated by all the people who believe in the potential of computational intelligence as a cognitive process. A mental state is a *container* for some mental or *propositional attitudes* toward some specific content in a given propositional form, which is currently stored in the agent's long term memory. A propositional attitude has at least three components: the *agent* who holds the attitude, the *type* of attitude and its propositional *content*. Examples of propositional attitudes types are belief, desire, intention, commitment, obligation, etc.

Following Allen in [1], we consider the Belief-Desire-Intention (BDI) agent model as the underlying model for a conversational (cognitive) agent. The notion of *mutual belief* is of central importance mutual understanding in communication. A proposition *P* is mutual believed (i.e. *MB(A,B,P)*) by two agents *A* and *B* if *A* believes *P*, *B* believes *P*, *A* believes *B* believes *P*, *B* believes *A* believes *P*, etc. This definition doesn't allow any axiomatic characterization nor computational implementation. Mutual belief can be postulated as primitive operator without reference to simple beliefs as in [9], avoiding infinite recursion. Other problems with mutual beliefs arise when trying to distinguish the pragmatic effect in dialogue: the agent *A* states that some proposition *P* is true, but it is not always the case that the proposition is believed by both agents (e.g. the hearer may have more reliable prior knowledge, for instance during an argument where someone says he/she agrees only to stop arguing).

1.3 Goals and outline of the paper

We would like to push a little bit further the idea of simulating cognitive processes for language understanding in communication by designing an computational architecture

[1] This standpoint has also driven one of the most successful theory of communication and language understanding, Relevance Theory [25], but has not been yet incorporated in any real dialogue system or natural language interface to computers.

for Cognitive Dialogue Management that allows us to easily deal with mental representations and their operations. Within this perspective, we propose an *Architecture for Dialogue Systems* which is *modular* and is based on a specific decomposition where we have mainly two macro components: the **Robust Interpretation** (RI) and the **Knowledge Assimilation** (KA), also paraphrased as a *perception/action* (PA) component and a *mental state management* (MSM) component. As a metaphor of this dichotomy we illustrate this architecture by two (brain) hemispheres as shown in figure 1. The two hemispheres are loosely coupled and may share some (neural) sub-structures. We decided to model the Knowledge Assimilation component as a Dialogue Management System. For this goal, we consider an efficient dialogue model based on the notion of Information State (IS) and we enhance it by adding a notion of Mental Structure (MS). In this paper we describe a novel architecture for Dialogue Management by extending and implementing the ViewFinder framework [5] and by integrating it within the TRINDI Dialogue Management system [19]. We will show that this solution can be viewed as an efficient and viable alternative to the plan-based approaches to dialogue management [10, 17, 2]. We do not discuss the Robust Interpretation/Perception component here. The interested reader may refer to [3, 23].

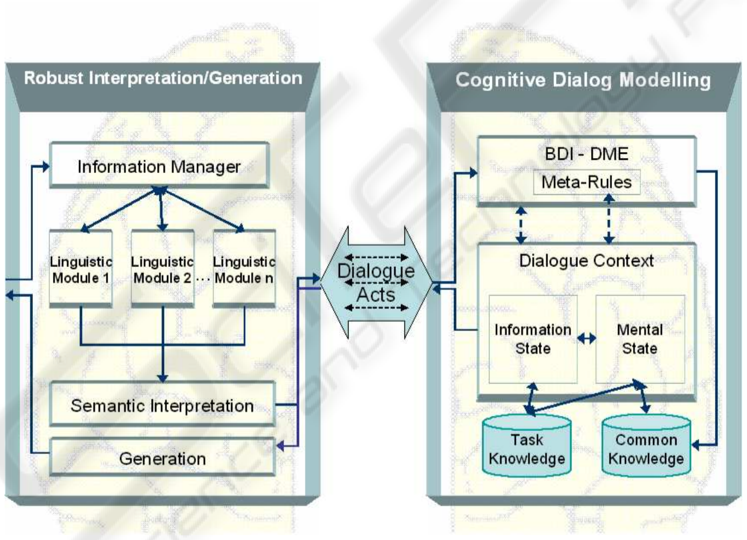


Fig. 1. Cognitive Language Architecture for Dialogue Systems

2 Dialogue Management in TRINDI

The Dialogue Manager (DM) is the program which coordinates the activity of several subcomponents in a dialogue system and, traditionally, it has as its main goal that of maintaining a representation of the current state of the ongoing dialogue. Typically, a DM receives as input a *dialogue act* which contains a semantic representation of a user's utterance (or a set of utterances) which has been interpreted by an interpretation module. The term dialogue act goes back originally to Bunt [7] and can be understood both loosely in the sense of "speech act used in a dialogue" and, in a more specialised sense, as functions which updates the dialogue context. Bunt also claims that the *context-change approach* to dialogue acts can solve difficulties arisen from pure speech-act theory, which concerns only with the assignment of the illocutionary force and the propositional content of utterances and their further classification into a taxonomy. The Bunt standpoint of dialogue acts as context updates is best represented by the Information State approach to Dialogue Management proposed in [11] and implemented in the TRINDI toolkit [19]. The Information State theory of dialogue modelling consists of:

- a description of the *informational components* of the theory (e.g. participants, common ground, linguistic and intentional structure, obligations and commitments, beliefs, intentions, user models, etc.);
- a *formal representation* of the above components (e.g. as a typed feature structure, modal logic, abstract data types);
- a set of *dialogue moves* that will trigger the update of the information state;
- a set of *update rules*, that govern the updating of the information state;
- an *update strategy* for deciding which rule(s) to select at a given point, from the set of applicable ones.

The information state (IS) is stored internally by an agent (e.g. dialogue system). The information state and all resources are seen as abstract datatypes (i.e. sets, stacks etc.) with related conditions and operations. The IS is composed of a *static* part (SIS), which remains constant during a dialogue. It includes rules for interpreting utterances, updating the *dynamic* part of the information state, and selecting further moves; optionally, move definitions, plan libraries, static databases etc. The dynamic part of IS (DIS) changes over time depending on the occurring events and how these events are treated by the *dialogue move engine* (DME). A typical (minimal) information state structure is showed in figure 2. The main division in the information state is between informa-

$$\begin{bmatrix} \text{PRIVATE :} & \begin{bmatrix} \text{AGENDA : stack(Action)} \\ \text{PLAN : stackset(Action)} \\ \text{BEL : set(Prop)} \\ \text{TMP : } (\textit{same type as SHARED}) \end{bmatrix} \\ \text{SHARED :} & \begin{bmatrix} \text{COM : set(Prop)} \\ \text{QUD : stack(Question)} \\ \text{LU:} \begin{bmatrix} \text{SPEAKER: Speaker} \\ \text{MOVES: assocSet(Move)} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Fig. 2. Information State (Cooper & Larson)

tion which is *private* to the agent and that which is (assumed to be) *shared* between the dialogue participants. Shared information is considered here what has been established (or grounded) during the conversation. Hypotheses about the ungrounded shared IS are kept in a temporary slot in the private IS, until an update rule for grounding is enabled and executed. The “QUD” slot (i.e. the *question under discussion*) provides the underlying mechanism for dealing with sub-dialogues.

Observe that there is no assumption on how the mental state of the agent should look like, except from the fact that it is a collection of (believed) proposition. The extension of IS we propose is one where the “BEL” slot is replaced by a complex dynamic structure which represents the evolution of the agent mental state in terms of *nested mental attitudes* as the one that will be discussed in the next section.

The Dialogue Move Engine updates the IS on the basis of the observed dialogue moves and selects the appropriate moves to be performed. Dialogue Moves (DM) are theoretically defined using preconditions, effects & decomposition, but in practice DMs are not directly associated with preconditions and effects (i.e., output of the interpretation module and input to the generation module).

The main difference between information state approaches and other structural, dialogue state approaches lies on the fact that the latter are based on the notion of “legal” or “well-formed” dialogue, described by some generative formalism (e.g., grammars, finite state automata). Information states can be partially described. Moreover, the motivation for update the information state and selecting a next dialogue move may rely only on part of the information available. The model is simpler than plan-based models, but extensible enough to cope with representations of mental states, as we will discuss later.

3 The ViewFinder framework

In this section we describe a framework for the representation of knowledge assimilated through perception (including communication) and cognitive processes into the mental state of a cognitive agent. In our proposal for Cognitive Dialogue Management we adopt and extend the ViewFinder framework, early proposed by Ballim and Wilks in [4] and further formalized by Ballim in [5]. The main motivation is that language understanding cannot be obtained without a representation of intentionality. Moreover, this representation must be computational if our goal is to build artificial systems capable of natural language processing beyond pure structural linguistic analysis (e.g., phonology, morphology, syntax).

In order to build mental representations of the perceived information, one can assume that there exist some general principles and general representations that are common to all the cognitive agents. It is of fundamental importance the way information is *organized* in cognitive agent’s mental structures. Nonetheless, we believe that feeding mental structures with representations of the acquired information can only be achieved if coupled with some form of linguistic analysis. We do not address this problem here, but the interested reader can look at [24] for the development of this topic.

Partitioned representations have been advocated as a main conceptual tool for structuring information without adding any spurious semantics to its content . The main principle relies on the assumption that information typically has local coherence. In Artificial Intelligence and in particular in Knowledge Representation and Reasoning, the usefulness of partitioned representations have been early recognized by John McCarthy [22]. A survey on this topic [6] classifies frameworks for partitioned representations (or contexts) in two main categories: the *divide-and-conquer* approach and the *compose-and-conquer* approach. The former sees partitioned representations as a way of partitioning a global model of the world into smaller and simpler pieces (examples are the work of Dinsmore [12] and that of McCarthy, formalized in [8]). The latter considers partitioned representations as *local theories* of the world interconnected by a network of relations (as in both Local Model Semantics (LMS) [15] and ViewFinder [5]).

ViewFinder is a formal framework for representing, ascribing and maintaining nested attitudes of interacting agents. Viewpoints on mental attitudes of communicating agents are represented by means of nested typed environments. Operations over typed environment are defined and used to simulate several forms of reasoning, which are necessary in order to assimilate the information into knowledge structures from communication. The early implementation of ViewFinder is the ViewGen system [4] and it is intended for use in modelling of nested beliefs in autonomous interacting agents. The ViewFinder framework provides the theoretical foundations for ViewGen in particular with respect to the following issues related to the manipulation of environments:

- Correspondence of concepts across environments (i.e. intensional objects);
- Operations performed on environments (e.g. ascription, adoption);
- Maintenance of environments.

Relationships between environments can be specified hierarchically or by the explicit mapping of entities. Each environment has associated an axiomatization and a reasoning system.

ViewFinder has been recently implemented as a computational framework [24], and successfully tested on classical knowledge representation problems in dialogue such as the “Three Wise Men” problem [21, 22]. Moreover, plan recognition in ViewFinder has been addressed by Lee in [20]. In the rest of the section we rapidly review the basic notions of ViewFinder. Details are available in [5].

3.1 Backgrounds of ViewFinder

The fundamental notion in ViewFinder is that of *environment* (or *spaces*). Environments are containers for information and can be *nested*. This means that an environment can contain other environments which can in turn contain other environments and so on. An environment contains some propositional content which is assumed to be consistent. The propositional content can be organized with respect to topics. A nested environment is thus a point-of-view with respect to a particular topic, that is, an attitude toward a particular content. A topic itself is a special type of environment.

Different attitudes (e.g. beliefs, goals, intentions) correspond to different *environment types*. Like other environment types, belief spaces can be nested. We can thus represent a point-of-view on a given topic with respect to another nested environments as, for instance, in the expression:

$$Bel(John, wants(Fred, Bel(Anne, P))).$$

Nested environments can be created extensionally or intensionally. Environment operators project the content of an environment onto another environment. Two operators are defined: the *ascription* and the *adoption*. The ascription takes the propositional content of an environment and projects it onto an inner environment. It can be viewed as a rule that dynamically generates or updates the content of nested environments by projecting the content of the outer environment onto the inner environments, provided that none of the inner spaces contain propositional content which is inconsistent with the projected information. Figure 3 shows the prototypical situation with the belief environment type. The second operation on environments is adoption, referred in [4] as

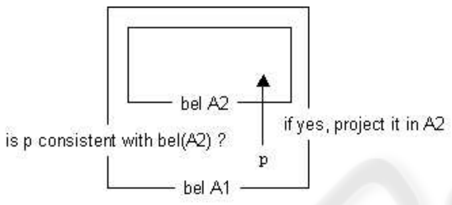


Fig. 3. Default ascription

percolation. Adoption is the ascription's dual operation. Adoption projects information from an inner environment to an outer environment. While the purpose of the ascription algorithm is to allow the dynamic generation of viewpoints, the purpose of the adoption algorithm is to accommodate information in the system's knowledge base (i.e. the mental state) from viewpoints on other agents attitudes. This situation is sketched in figure 4. The adoption operator allows an agent to *extend* its mental state with information

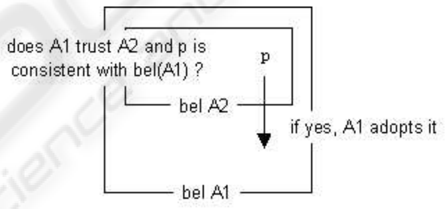


Fig. 4. Default adoption

he gathers from other agents through communication and collaboration, and not only through his own observations of the real world. The adoption operator is triggered when new information becomes available in an environment, for instance after the processing of a dialogue act. The set of new information is *propagated* to other environments using the adoption.

The whole theory of ViewFinder is based on the notion of *default reasoning*. An agent can always assume that another agent mirrors its mental space except when contrary evidence exists. ViewFinder adopts a different approach to *common*, *shared*, or *mutual knowledge* as found in other theories of mental attitudes [9, 14, 18], since it assumes that knowledge is always shared unless a contrary evidence arises.

Feature-based Topic Language In contrast with ViewGen and the formal model of ViewFinder, we introduce a novel propositional content language for ViewFinder. A *topic* T is associated a set of features identified by a common name, that is $\mathcal{F}_T = \{f_1, \ldots, f_n\}$. Each feature f can take values in a finite domain set (enumeration type) denoted by D_f. A topic can be part of a hierarchy and it inherits features from its parent topics. A feature can be redefined in the child topic. In this sense a topic is a container for structured information.

For example, the topic `WeatherBroadcast`:

```
topic GeneralWeatherBroadcast
  date: [tomorrow, in 2 days, in one week]
  weather: [sunny, cloudy, rainy]
  temperature: [cold, warm, hot]
```

can be specialized to:

```
topic DetailedWeatherBroadcast
  inherits from GeneralWeatherBroadcast
    temperature: [<5C, 5-25C, >25C]
    wind: [N, S, W, E]
    velocity: [0n, 5n, 10n, 15n, 20n, 30n]
```

The notion of stereotypes of ViewGen is also present in ViewFinder. This is a powerful concept for classifying agents and ascribe them some default behavior, knowledge or reasoning capability. We have decided to represent this notion differently by grouping agents into *classes*. Each agent belongs to one or more classes. The appropriate topic definition is selected by an agent A belonging to the class $\mathcal{A}$ and it is denoted by $T(A)$. This parameter allows us to have different definitions of the same topic for different classes of agent. A topic can have more or less features with respect to an agent class, modeling the notion of *agent competency*. We say that the knowledge about a topic T is *complete* if all the features f_i have a unique value, or equivalently if the topic is represented by only one possible combination of feature-values pairs (a tuple)[2]. Conversely, we say that an agent has no opinion about a topic, if he admits that any tuple of the topic space is valid. We call *opinion* (of an agent about a topic) a subspace of the whole topic space.

We have decided to model knowledge about topics as *integrity constraints* over the topic space. An agent builds its opinion about a topic by an elimination process. It interprets the constraints and eliminates all the incompatible tuples from the topic space. Complete knowledge is achieved when the constraints have reduced the topic space to

[2] The term 'complete' has been chosen in analogy to the complete meta-predicate of the Kim and Kowalski's work [18].

only one possible combination of features, whereas inconsistency appears when the constrained topic space is empty. This method allows us to deal with open theories and disjunctive information. From a logic programming point of view, topic features can be viewed as abducible predicates with integrity constraints [16].

Representing opinions The topic language may be sufficient when representing one agent's knowledge about some aspects of the world, but is not expressive enough to capture all the information needed to model other agent's point of view. A topic space allows us to model in a natural way a disjunction of possible topic instances, that we shrink each time we acquire further information, but only in case we have complete knowledge about another agent's opinions. We say that feature f is *undefined* and we write $f = \lambda(A)$ where the parameter A is the only agent capable of providing a value for the feature f.

Ballim and Wilks in [4] have proposed a *taxonomy of competency in believing* showed in figure 5. They consider differently an agent who may be competent in the evaluation of a feature and an agent who actually is an evaluator for that feature. An agent can have competency to evaluate a feature (i.e. belong the appropriate class), and then if he is competent, he can possibly be an evaluator for it. In the positive case, another agent might be aware or not of the way it assigns values to the features. As shown in [24], this topic language allows to cope with all the cases of the belief competency taxonomy described in figure 5.

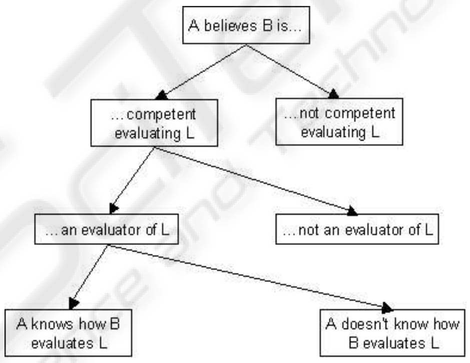

A believes B is...
...competent evaluating L
...not competent evaluating L
...an evaluator of L
...not an evaluator of L
A knows how B evaluates L
A doesn't know how B evaluates L

Fig. 5. The Ballim and Wilks's taxonomy of Competency in Believing

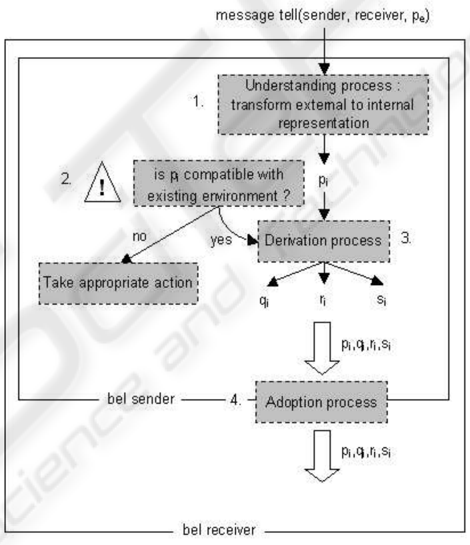
3.2 Assimilating mental attitudes from communication

In this section we tackle the problem of assimilating knowledge from communications between agents in ViewFinder. We followed the ideas both from the Lee's [20] and Dragoni's works [13]. Assimilating new information requires to solve three problems:

1. decide for a communication language understood by the two agents speaking,
2. represent the knowledge in each mental states
3. transfer the knowledge contained in the message into the mental state.

We already provided a solution for the second issue with the topic language and we have decided to adopt for the content level the same representation we used for topics, that is, a set of constraints on the topic space. In other words, topic language can be used also as the propositional content representation in dialogue acts. Since the moment a dialogue act is taken in charge by the Dialogue Manager until the whole information contained is digested, four steps have occurred:

1. the *translation* of the message content into the mental state representation language.
2. the comparison of the actual mental representation of the sender with the message sent for *consistency check*.
3. the *derivation* of new information (i.e. constraints) using the actual mental state and the message received;
4. the *adoption* of the propagated of new information in the whole system (i.e., forward-chaining).



message tell(sender, receiver, p_e)
1.
Understanding process : transform external to internal representation
2.
is p_i compatible with existing environment ?
p_i
no
yes
Derivation process
3.
Take appropriate action
q_i
r_i
s_i
p_i,q_i,r_i,s_i
bel sender
4.
Adoption process
p_i,q_i,r_i,s_i
bel receiver

Fig. 6. The Knowledge Assimilation Process

This process is showed in figure 6. Once the receiver has decoded the message (i.e., extracted the dialogue act), she can use it and update her mental state. Its ability of managing viewpoints is very interesting at this point. The receiver builds her mental representation of the sender, then she compares the received message with her point of view on the sender. If she detects any inconsistency she can take the appropriate action, for instance to inform the sender of the inconsistency, to ask her for more information or to reject the message. If the message is compatible, she assimilates it. The integration process takes place in the point of view of the sender seen by the receiver. The algorithm, depending on the content language, generates new content. In our solution the process uses a *propagation mechanism* to generate new constraints. The receiver reasons on her viewpoint on the sender and derives constraints that should be consistent within the sender's mental state. At the end of the derivation process, the point of view of the sender seen by the receiver is updated with all this new information. This new information can then be back-projected (from inner to outer environments) to the receiver using adoption provided that the receiver trusts the sender, and if the new information is consistent with the already present propositional content.

4 Integration of DME and ViewFinder

We describe, as the last step, the integration of Information State and Mental State in TRINDI Dialogue Management. We add a new element to the private component of IS, an instance of the new "Env" datatype, representing the outer mental environment of the agent. The ViewFinder system's environment provides more context for the interpretation and assimilation of dialogue acts in the Information State (including updates of the ViewFinder environment itself). The following is the description of the Env TRINDI datatype with its operations and conditions:

```
type Env
operations:
   tell( A1, A2, Topic, Features, Values )
   untell( A1, A2, Topic, Features, Values )
   ask( A1, A2, Topic, Features, Response )
   cleanKB
   loadEnv(EnvSignature)
conditions:
   instanceOf(TopicName, TopicType);
   topicDef(TopicName, AgentClass, TopicDef);
   isa(TopicName, ParentTopicName);
   agentDef(AgentClass);
   adopts(AgentName1, AgentName2, TopicName);
   ascribes(AgentName1, EnvType1, AgentName2, EnvType2).
```

The operations and conditions mirror the ViewFinder operations on the content of environments expressed in the topic language. We added the following Dialogue Moves to the basic set of TRINDI moves in order to model dialogue acts having content expressed in the ViewFinder topic language:

```
ask(Agent, Topic, Features),
askif(Agent, Topic, Features, Values),
answer(Agent1, Agent2 Topic, Features, Values),
inform(Agent, Topic, Features,Values).
```

The update rules for grounding and planning are not discussed here. We only show the rule for answer integration, which is triggered when the dialogue manager decides to assimilate the information obtained from the successful processing of the answer dialogue move in the Env componente of the private information state:

```
RULE : integrateOwnAnswer
CLASS : integrate
PRE:
   valRec( SHARED.LU.SPEAKER, PRIVATE.NAME )
   assocRec( SHARED.LU.MOVES,
             answer(PRIVATE.NAME, A, Topic, Features, Values), true)
EFF:
   popRec( SHARED.QUD )
   tellRec( PRIVATE.ENV, PRIVATE.NAME , A, Topic, Features, Values)
   setAssocRec(SHARED.LU.MOVES, answer(P), true)
```

5 Conclusions and future work

In this paper we described an architecture for Cognitive Dialogue Management based on the integration of TRINDI Dialogue Management toolkit with the ViewFinder framework for Mental Spaces Management. The Information State approach to Dialogue Management has been adopted by integrating the Information State in the the TRINDI Toolkit. Information State, as initially proposed in the TRINDI framework, does not suffice for taking into account different dimensions of a dialogue. We also need to consider the evolution of the system's view about the speaker's mental state, which can be reported and simulated within an appropriate data structure. The structure we proposed to integrate into the Information State is that of *Nested Typed Environments* provided by *ViewFinder*. In order to cope with frequent cases of partial knowledge about the world and to avoid problems related to the construction of mutual knowledge, we proposed an extension of ViewFinder by a systematic account of operations between mental spaces and by introducing a new content representation language based on integrity constraints over set of features. The choice of the operations and the content language allows us to robustly and efficiently deal with complex epistemic problems in communication, such as the "three wise men" problem.

Summing up, in the notion of Cognitive Dialogue Modelling we incorporate a set of tools helpful to achieve robust human-computer interfaces where understanding from communication is an essential factor. Understanding can be partial in some circumstances, but the system should be able to robustly react and complete the missing information by contextual knowledge. Context must include also a model of the users, and in particular a model of their mental state. Moreover, the design of artificial dialogue system should be based on a model of cognitive processing rather than on the simple extraction of information from sophisticated input devices. We proposed to extensively exploit Computational Logic to conceive and soundly implement inference systems as the basis of a *computational cognition*. Cognition in artificial systems needs not to mirror human cognition, but human cognition metaphors can still be used and adapted to computer systems.

References

1. J. Allen. *Natural Language Understanding*. Benjamin/Cummings, Menlo Park, 1987.
2. J. Allen and C.R. Perrault. Analyzing intention in dialogues. *Artificial Intellicence*, 15(3):143–178, 1980.
3. A. Ballim and V. Pallotta. Robust methods in analysis of natural language data. *Natural Language Engineering*, 8(2), 2002.
4. A. Ballim and Y. Wilks. *Artificial Believers*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1991.
5. A. Ballim. *ViewFinder: A Framework for Representing, Ascribing and Maintaining Nested Beliefs of Interacting Agents*. Ph.d., University of Geneva, 1992.
6. P. Bouquet, C. Ghidini, F. Giunchiglia, and E. Blanzieri. Theories and uses of context in knowledge representation and reasoning. Technical Report 0110-28, IRST, October 2001. "citeseer.nj.nec.com/bouquet01theories.html".
7. H. Bunt. Conversationa principles in question-answer dialogus. In D. Krallmann, editor, *Zur Theory der Frage*, pages 119–141. Narr Verlag, 1979.
8. S. Buvač, V. Buvač, and I.A. Mason. Metamathematics of contexts. *Fundamenta Informaticae*, 23(3), 1995.
9. P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.
10. P.R. Cohen and C.R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.
11. R. Cooper. Information states, attituds and dialogue. In *Proceedings of the Second Tblisi Symposium on Language, Logic and Computation*, Tblisi, September 1997.
12. J. Dinsmore. *Partitioned Representations: A study in mental representation, language understanding and linguistic structure*, volume 8 of *Studies in cognitive systems*. Kluwer, 1991.
13. A.F. Dragoni, P. Giorgini, and L. Serafini. Mental states recognition from communication. *Journal of Logic and Computation*, 12(1):119–136, February 2002.
14. R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about knowledge*. MIT Press, 1996.
15. C. Ghidini and F. Giunchiglia. Local model semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.
16. A. C. Kakas, R. Kowalski, and F. Toni. The role of abduction in logic programming. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5, pages 235–324. Oxford University Press, 1998.
17. H. Kautz. A formal theory of plan recognition and its implementation. In J.F. Allen, H.A Kautz, R.N. Pelavin, and J.D. Tennenberg, editors, *Reasoning About Plans*, chapter chapter 2, pages 69–126. Morgan Kaufmann, 1991.
18. J. S. Kim and R. A. Kowalski. An application of amalgamated logic to multi-agent belief. In M. Bruynooghe, editor, *Second Workshop on Meta-Programming in Logic META90*, pages 272–283, Dept. of Computer Science, 1990. Katholieke Univ. Leuven.
19. S. Larsson and D. Traum. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural Language Engineering*, 6(3 & 4):323–340, 2000.
20. M. Lee. *Belief, Rationality and Inference: A general theory of Computational Pragmatics*. Ph.d. thesis, University of Sheffield, 1998.
21. J. McCarthy. Formalization of two puzzles involving knowledge. http://www-formal.stanford.edu/jmc/puzzles/puzzles.html, 1987.
22. J. McCarthy. Notes on formalizing contexts. In Ruzena Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 555–560, San Mateo, California, 1993. Morgan Kaufmann.

23. V. Pallotta and A. Ballim. Agent-Oriented Language Engineering for robust NLP In A. Omicini, P. Petta, R. Tolksdorf, Eds., *Proceedings of Second International Workshop, ESAW 2001, Engineering Societies in the Agents World II*, pages 86–104. (Springer LNAI 2203). Prague, Czech Republic, July 7, 2001.
24. V. Pallotta. *Cognitive Language Engineering: Towards Robust Human-Computer Interaction*. Ph.d. thesis, Swiss Federal Institute of Technology Lausanne (EPFL), August 2002.
25. D. Sperber and D. Wilson. *Relevance: Communication and Cognition*. Language and Thought Series. Harvard University Press, Cambridge, MA, 1986.