Computational Dialogue Models

Vincenzo Pallotta
Faculty of Computer and Communication Sciences
Swiss Federal Institute of Technology - Lausanne
EPFL IC-ISIM LITH
IN-F Ecublens, 1015 Lausanne (CH)
Tel. +41-21-693 52 97, Fax. +41-21-693 52 78
e-mail: Vincenzo.Pallotta@epfl.ch
web: http://ic.epfl.ch/~pallotta.

MDM.EPFL.LITH.Y1.M6 Deliverable

Multimodal Dialogue Management

Contents

Objectives

The goal of this report is to review the theoretical foundations Dialogue Management Systems, and in particular the problems involved in developing systems that can participate in natural dialogue with humans.

Dialogue models have recently gained a great interest in Computational Linguistics community since they offer a natural framework both for the analysis of human dialogues and the design of man-machine interfaces. Intelligent information exchanging seems to be one of the most challenging tasks among those involving hybrid human-computer interactions. A central issue is how to model various types of interaction among artificial and natural entities at different levels of abstraction. On the one hand, models of interaction are required to better understand the communication phenomena. On the other, suitable languages and paradigms may provide powerful frameworks for developing computer-based applications.

Difficulties in modelling dialogue arise, for instance, from very frequent phenomena such as the use of anaphoric expressions, forms of deixis (e.g. 'now/then', 'here/there', 'I/you', 'this/that'). The use of the contextual information is required for the right interpretation of these linguistic objects. Participants are able to make sense of a dialogue even when little linguistic information is present in the utterances. This is achieved by their cognitive skills: their abilities to perform inferences based on *background knowledge* and *assumptions* on the other participants' mental states. There are situations where the literal meaning is not sufficient to understand the role of the utterance in the dialogue, that is, the corresponding dialogue act cannot be directly recognized by simply its linguistic content: it must be inferred.

In the first part of this report, the current state-of-the-art in Dialogue Management Systems will be presented, and particular attention will be paid to semantic and pragmatic models of dialogue based on the intentions of participants, and to multimodality. The Dialogue Manager (DM) is the program which coordinates the activity of several subcomponents in a dialogue system and its main goal is 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 representation of a user's utterance (or a set of utterances) which has been interpreted by an interpretation module (IM). The representation of the user's utterance contains also a marker which indicates the type of dialogue act that the user intended to perform by means of the utterance.

The recognition of a dialogue act is a crucial issue from which the overall quality of the system depends. In a fixed-initiative dialogue, the DM knows in advance which dialogue act is going to be performed by the user, since the dialogue has to follow a pre-determined schema (e.g. frame-filling, question-answer, finite-state automata, and dialogue grammars). When moving to mixed-initiative dialogues, things become more difficult. Even if there is some predictability coming from the dialogue genre and some initial assumptions, users are free to take the initiative and follow their own interaction patterns. One of the main challenges in computer-based dialogue systems is to find reliable procedures for the automatic recognition and classification of dialogue acts. Some heuristics may guide the recognition of dialogue acts, one of the commonest being that of searching for linguistic markers in the utterance. At this level, recognition should be made during the linguistic processing. In absence of linguistic markers the system needs to rely on another types of heuristic and try to predict the more probable dialogue act from both the content of the utterance and the expectations that can be inferred by combining information from the *current state of the dialogue* and *knowledge about the task domain*.

In the second part, the ViewFinder model [13] is described in detail as an efficient and viable alternative to Plan-based approach in BDI models for modelling Rational Agents. 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 form of reasoning which are necessary in order to assimilate information into knowledge structure from communication. The early implementation of ViewFinder is the ViewGen system [11]. It is intended for use in modelling autonomous interacting agents specifically tailored for modelling agents' mutual beliefs. A belief environment represents each agent belief space and it may use nested environments to represent other's agent beliefs spaces.

The ViewFinder framework provides the foundations for 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. The above issues have been considered in greater detail in the specific case where environments represent agents' beliefs spaces and has been implemented as a PROLOG program (i.e. View-Gen). However, the framework has full generality and it can be applied to model multi-dimensional reasoning systems as those proposed in [36]. Each agent in dialogue has a belief environment which contains attitudes about what other agents believe, want and intend.

ViewFinder has been recently implemented as a computational framework [54], and it is now available as a tool for Computational Dialogue Modelling. A classical example of dialogue problem is modelled and solved within the ViewFinder framework showing how it is possible to avoid the explicit use of Common Ground.

1 General Dialogue Models

1.1 The nature of Human Dialogue

A sequence of isolated utterances that together form a *discourse*. A dialogue is rather a connected sequence of information which provides *coherence* over the utterances, and a *context* for interpreting utterances. Multiple participants engaged in a dialogue are aimed at exchanging information. Interaction is often *goal-driven*. Dialogues are in general imperfect since utterances are often ungrammatical or elliptical. Although, dialogues follow certain *conventions* or *protocols* that participants naturally adopt. One example is that of the turn taking: people seem to know very well when they can take their turn. There is very little overlap (5%) and gaps are often a few 1/10ths of a second. Natural dialogues appear fluid, but it not obvious how humans are able to decide when it is their turn to speak. There are however some extra-linguistic devices that are useful for humans to signal when the turn is released or kept. For instance, verbalized pauses (e.g. eh, uhm) can be understood as a signal for turn retaining, while the use of fillers (e.g. yes, ahah) may signal attention or turn skipping. Dialogue structure can be studied by means of *adjacency pairs*. An adjacency pair is a pair of *dialogue acts* performed by two participants. For instance it could be question-answer, greeting-greeting, offer-acceptance. An theory of conversation based on adjacency pairs has been proposed by Schegloff and Sacks in [58], but fail in modeling another common phenomenon in real dialogue, that of *insertion sequences*. Insertion sequences are adjacency pairs that are embedded (e.g. [question-[question-answer]-answer]. They are a simple form of *sub-dialogue*. The theory of adjacency pairs is not sufficient to model this phenomenon since it could happen that, in a higher level of embedding, the first component of an adjacency pair is not matched.

Other difficulties in modeling dialogue arise from very frequent phenomena such as the use of anaphoric expressions, forms of deixis (e.g. 'now/then', 'here/there', 'I/you', 'this/that'). The use of the contextual information is required for the right interpretation of these linguistic objects.

Participants are able to make sense of a dialogue even when little linguistic information is present in the utterances. This is achieved by their *cognitive skills*, that is their ability to perform *inference* based on *background knowledge* and *assumptions* on the other participants' *mental states*. There are situations where the literal meaning is not sufficient to understand the role of the utterance in the dialogue, that is the corresponding dialogue act cannot directly recognized by simply its linguistic content: it must be inferred.

There are several types of dialogues that can be classified considering the *type of mental attitudes* involved and their increasing degree of complexity:

Cooperative dialogues. This class of dialogue seems to be the simplest one since many assumptions can be made about the relevance of the participants contributions. In this case there is no need of recognizing the topic of the conversation which is fixed in advance and the goals between the participants are shared. Examples of this kind of dialogues are information seeking dialogues, assistance or instructional dialogues and in general all dialogues where there are users and service providers interacting together.

Collaborative dialogues. In this case the participant still share the same goals, but may have different starting point. The participant need to accommodate their set of beliefs in order to reach a consensus on a given topic. In general, participants may have contrary beliefs and

the main device for reaching a consensus is negotiation. However, in collaborative dialogues negotiation takes a weaker form since participants already agree in maximizing a common utility function. There could be instead a disagreement on the way the participants believe it might be possible, and on the effort required by each of them in achieving the common objective.

Conflictive dialogues. If we drop together the assumptions that participants have shared beliefs and shared goals we are in the case of conflictive dialogues. The participants are pursuing their own objectives and may share only a subset of their own beliefs. No or few assumptions can be made on how the participants interact in this situation. They will certainly need to find a consensus but the result will depend on the argumentative force of the participants. It is possible that a subset of the participant may have common, or non conflicting goals and thus they can create coalition in order to augment their argumentative power. A rather complex form of negotiation is needed in the general case and sometimes the presence of an arbiter is required in order to better coordinate the dialogue.

Taken for granted that the borderline between the above three classes are not so crisp, we can imagine that there exist different strategies to cope with them. One idea could be that of individuating which are the distinguishing features that allow us to recognize if we are in presence of one of the above dialogue types. The nature of beliefs and goals are clearly the most natural features to consider, but one can realize that these features can be initially only assumed, and must be grounded during the interaction. Starting with the assumption that a dialogue is be cooperative, we might end up with a discussion hardly leading to a consensus. Although this kind of difficulty we can certainly admit that there are some restricted situations in which this taxonomy strictly applies, especially in the first case where the interaction is carried out with a computer application which typically holds a set of goals which are imposed by its designer.

1.1.1 Dialogue control layers

Other than goals and beliefs in the participants' mental states there are some other important aspects of the dialogue that are useful to determine its intrinsic characteristics. The following is a possible classification in terms of control layers of dialogue, that include also the above classification in terms of the participant intentions.

Linguistic structure: Dialogue can be classified in terms of what kind of communicative acts are performed. For instance, an information seeking dialogue will involve questions and answers whereas an instructional dialogue will be likely to contain imperative sentences. Also the discourse structure and how external entities are accessed by referential expression have a central importance.

Intentional structure: As we have already discussed, assumption can be made about the nature of participants beliefs, goals and intentions. In general, the account of mental attitudes can be exploited to determine which are the roles of participants in the dialogue.

Attentional state: If we are able to describe the way participants get into the dialogue in terms of how they contribute to its advancement, then we can also make assumptions on the linguistic structure. For instance, in the attentional state we can consider to monitor the current focus in the dialogue by putting in evidence those objects that are recently referred to by the participants. This information will help us in constraining the set of possible referents when the participant is using anaphoric expressions. Moreover, the attentional state can contain global information about the ongoing dialogue, relating together local attentional structures and also other information contained in the intentional structure. Theories about the modeling of the attentional state have been proposed in [39, 26]. Models of attentional state may include other kinds of information related to the participants, such as, for instance, the initiative, the roles, the social conventions, etc.

Context: Modeling the context in dialogue is maybe the most controversial issue. What actually constitutes a context in a dialogue? One possible answer is: everything that is necessary in order to provide a situated interpretation of the above layers. For instance, it has been remarked in [4] that original speech act theory [6, 60] lacks of treatment of contextuality. In fact, there are very simple cases in which only the context can provide the right identification of the speech act attached to a negative response, like "no, it doesn't" which can be interpreted as a disagreement or agreement when it follows respectively a negative or a positive statement. A definition proposed by Bunt in [20] says that a context is "the totality of conditions that may influence the understanding and generation of communicative behavior". Its vagueness is also its main strength, in fact he continues saying that "it seems hard to determine the boundaries of this notion of context". For the moment we agree with this definition and we make the additional claim that the role of context is of central importance in the whole conception of a dialogue model. Information that can be related to the notion of the context are: the dialogue history, the domain knowledge, the world knowledge and the user models.

1.1.2 Dialogue Acts and Communicative Functions

Utterances and discourses contain descriptions of actions. The speaker's point of view has a central role in the interpretation of utterances and discourses, that is their understanding. In the first person descriptions the agent may express their own desires, goals, reasoning and the precise intentions of a described event or process. In the third person descriptions, we may assign only a conventional interpretation to actions, otherwise we need to express or implicate that the agent of which her action and activities are described, provide us also information about her mental structures (i.e. her mental attitudes). Action's descriptions may involve also subjective evaluation of actions, like for example, we can describe the fact that "John sang a song" by saying that "John spoiled that song".

Pragmatics analyses general principles of purposeful action and it is in line with what we have outlined in the above sections. When considering pragmatic of language, language may be itself considered as a form of activity. In this case we speak of *linguistic pragmatics*.

For Bühler, language is fundamentally a means for the human mind to perform activities, typically communication. Bühler tries to reconcile this notion of language with the mainstream linguistics of the beginning of the XXth century where, following De Saussure, the study of language should preceded that of speech. As already recognized by Plato, language as a tool is the perspective adopted by Bühler [17]:

> "Language is akin to the tool: language belongs to the instruments of life, it is an *organon* like the material instrument, a body-extraneous hybrid; language is like the tool a *purposefully designed mediator*. The only difference is that it is not material things which react to the linguistic mediator, but living beings with whom we communicate."

During communication the speaker performs *linguistic actions* and *acts*. There is a difference between these two terms which has been pointed out also by Bühler. While in the former language is used as a tool for affecting the state of the hearer, the latter is closer to the act of producing a linguistic meaning. It is thus an act which is inherent to the speaking activity and it is independent of the goals of the speaker or the context in which the act is performed.

While the center of the discussion for Bühler is about linguistic acts, we are more interested in linguistic actions. In this section we will review some of the classical and current theories of linguistic actions and in particular that of *communicative acts*. Speech act theory was early proposed in philosophy of language by Austin in [6] and later by [60]. The main goal was to consider

utterances as having the status of actions and provide a taxonomy of utterance types according to some identified features. In these works there is a clear departure from the classical studies in linguistics where utterances were classified considering only their syntactic or semantic properties. Semantics were considered as the last stage of analysis, determining their meaning in terms of truth conditions in a model. The shift towards the study of the language usage complementary to the study of language meaning have generated a big debate. While semantics can be said to be the study of linguistic meaning, pragmatics concerns with the study of the speaker's meaning by means of linguistic and extra-linguistic information. A possible solution is that of enriching the semantic representation of a sentence with its pragmatic content as earlier proposed by Morris [52]. Leaving aside any philosophical discussion, one of the most accepted view of pragmatics nowadays is that of language as an activity where people engaged in communication are actually performing actions.

The first formulation by Austin is rather intuitive and puts in evidence the opposition between *performatives* and *constatives* sentences. The former are those that can be used to as means for changing the state of the world, while the latter tends to describe events or states of the world. Compared to constatives, performatives cannot have truth conditions, but they can be only defined by their success or failure in achieving the intended effect. In order to understand the way performatives work one has to consider in which circumstance the performatives may be used and what are the conditions which may guarantee its success. Austin calls them *felicity* conditions. Felicity conditions were divided into three types:

1. appropriateness of circumstance, person, and the conventional procedure used to achieve a conventional effect.
2. correct and complete execution of the procedure
3. the appropriate mental states in which both the speaker and the hearer have to be and the fact that the expected behavior of the participants is realized.

The failure of the first or second condition will cause the whole failure of the speech act, while the failure of the third condition only implies a degrees of mismatch between the expected and the obtained results in the participants.

Austin notice that the difference between performatives and constatives is not strictly reflected at grammatical level distinguishing between declarative and non-declarative sentences. Some declarative sentences may have a performative interpretation in a given context (e.g. "it is cold here"). A question may be interpreted as a constative (e.g. "is she a beautiful girl, isn't it?"). Explicit performatives are the simplest case where seem to describe a certain action of the speaker like, for instance in "I order you...". However, the borderline between performative and constative is not crisp since in the above example, the sentence also describe an event where the subject (i.e. me) is performing an action (i.e. to give an order). Being explicit or not, a performative cannot be simply characterized by a pure representation of an event: its specification must contain also what is the specific action that the performative is supposed to accomplish.

In order to solve this problem Austin proposes a general theory of speech acts which is applicable to all kind of sentences and utterances. Any utterance is produced in order to accomplish three main acts:

locutionary act: the action of producing the utterance (i.e. the linguistic action): the meaning expressed by the utterance is its *propositional content;*

illocutionary act: an action whose direct consequences are transformation between the speaker and the hearer. It is a conventional act, that is the illocutionary act finds their success conditions in the existence of a "social ceremony" which licenses the use of that act, under some circumstances, giving to it the status of a determined action;

perlocutionary act: the context-dependent side effect of the act of producing the sentence which is expected to be obtained in the hearer, provided that the hearer has a sufficient command of the conventional use of the language and the appropriate mental condition.

Searle tries to characterize speech acts by means of rules that govern their use. First he defines the idea of *constitutive* rules for a generic activity, that is the emerging behavior that is needed to identify the act. He defines constitutive rules counter-factually, that is by the fact that their inobservance prevents the recognition of the activity. *Normative* or *regulative* rules are those that prescribe the correct behavior with respect to the considered activity. Searle applies the above two concepts in order to characterize the illocutionary force of a speech act. The rules which determine the illocutionary force of a sentence are constitutive with respect to the use of the sentence. In other words, speech is an illocutionary act when its function is that of affecting the mental state of participants. Normative rules are instead those rules which guarantee that the speaker's intentions and commitments and the hearer's alternatives and obligations be realized. Searle's idea is completed by the fact that linguistic analysis can help in determining the illocutionary act which underlies the surface realization of the speech act. He identifies what he calls *illocutionary force indicating devices* (IFIDs), that is linguistic features which allow the hearer in recognizing the illocutionary force of a speech act.

Instead of entering into the details of the Searle's analysis of speech acts, we will consider in the next section the notion of *dialogue acts* which better accounts the notion of language as action. In particular we will consider a model of interaction in which utterances trigger updates of a dialogue state. The problem of determining the illocutionary and perlocutionary force of an utterance is thus shifted towards the recognition of a dialogue act that is supported by the utterance. The dialogue act is thus treated in a principled way, that is as an update of the dialogue state.

1.1.2.1 Dialogue Acts

The term dialogue act goes back originally to Bunt [18] and can be understood both loosely in the sense of "speech act used in a dialogue" and, in a more specialized 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. Bunt also identifies *local* and *global* aspects of communication. The latter are invariable during the dialogue, while the former are dynamic. The adequate modeling of a dialogue context provides the basis for the study of dialogue acts, that is the combined study of utterance meaning and dialogue mechanisms. Dialogue acts are classified by their *semantic content* and *communicative function*. Semantic content corresponds to propositional content of speech-acts. Communicative function is similar to illocutionary force of speech acts, but its semantics is provided in terms of context changes. Mathematically, a communicative function is a function F that takes a propositional content p and a context Γ and produces an updated context Γ'. The propositional content is computed from a utterance which may also be viewed more abstractly as a contribution of the speaker and which may include multiple modalities of interaction other than natural language. A dialogue act may consist of several utterances. Moreover, one utterance may have several communicative functions at the same time since communication can have several *dimensions* that the speaker can address simultaneously. Compared to speech-act theory where only one speech-act correspond to one utterance (except indirect speech acts[1]), utterances will be considered as carrying several dialogue acts rather than being considered functionally ambiguous. The treatment of indirect speech acts is reduced to a simple consideration: in direct request speech acts the speaker presuppose that the hearer is able to perform the request, while in indirect

[1] Searle in [59] analyses a prototypical situation of the use of indirect speech act is that of indirect request such as "Can you pass me the salt?". In this utterance there are two possible interpretations (i.e. speech-acts): one is to check the ability of the hearer in performing an action, the other is a request for that action.

formulation the condition is *examined*. This perspective may avoid of postulating the existence of an additional illocutionary act by considering the recognition of *additional intentions* in the speaker by the hearer.

According with Bunt, dialogue acts can be divided in two main categories: *task-oriented* and *dialogue-control* acts. This distinction is mainly based on the observation that only a subset of utterances are strictly relevant to the subject of the dialogue, whereas the remaining are only used for dialogue management purposes (e.g. acknowledgments, self-correction, in making communication smooth and successful). The first category allows us to determine the *type of dialogue* (e.g. information-seeking-and-providing, instructional, negotiation). In the second category we have *feedback* (e.g. negative, positive, completion), *interaction management* (turn taking, pausing, resuming, structuring the discourse, monitoring attention and contact), and *social obligations* (greetings, introducing oneself, thanking, apologizing). The distinction between different types of dialogue acts does not depend uniquely on the type of communicative function. It depends also on the semantic content as for instance when one participant is replying to a question by informing that he/she did not understand the question. There are in fact some communicative functions that can occur in every dialogue act such as questions, informs, answers, verifications, confirmations, etc. These functions can be classified as *informative functions*, which can be in turn sub-classified as *information providing* and *information seeking* functions. There are however communicative functions that are specific to certain dialogue acts and can be considered as representative of them. For instance, it is possible to individuate communicative functions that can be used only in task-oriented acts or only in dialogue-control acts. If we consider as a task that of an information dialogue, we realize that the task-oriented functions coincide with the informative functions.

Each communicative function corresponds to certain observable features of communicative behavior, that is there exists a characteristic set of utterance features such that any utterance having these features can be assigned to that function. A utterance can be *functionally ambiguous* if it has features that belong to more than one function at the same time, say $F_1, \ldots, F_n$. Since communicative functions are organized in a hierarchy it is possible to assign their least upper bound to the sentence (i.e. the most specific function among those that are more general of $F_1, \ldots, F_n$. In case the least upper bound does not exist, the utterance is considered *truly ambiguous*.

1.2 Computational Dialogue Models

1.2.1 Dialogue Systems

What is a Dialogue System? In Dialogue Systems one participant is artificial. It tries to model the dialogue for a task domain and it has extensive knowledge of this domain. In general it (may) have a model of the other participant(s) and it may have ways to cope with unexpected or unusual input. There are several computational models for Dialogue Systems (see [22] for a survey). In this work we will review some models and one representative system for each model.

A Dialogue System must deal at least with the following dialogue tasks:

- Disambiguation
- Confirmation
- Error handling
- Filling in missing information
- Context switching
- Continuation (grounding)

- Database querying or answer generation

The following are examples of application domains where dialogue systems have a major impact:

- Flight and train timetable information and reservation
- Switchboard services
- telephone disconnect ordering
- Automated directory enquires
- Weather information
- Yellow pages enquires
- Appointment scheduling
- Multilingual spoken dialogue real-time translation systems.

1.2.2 Dialogue Management

The Dialogue Manager (DM) is the program which coordinates the activity of several subcomponents in a dialogue system and 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 representation of a user's utterance (or a set of utterances) which has been interpreted by an interpretation module (IM). The representation of the user's utterance contains also a *marker* which indicates the *type* of dialogue act that the user intended to perform by means of the utterance.

The recognition of a dialogue act is a crucial issue from which depends the overall quality of the system. In a *fixed-initiative* dialogue, the DM knows in advance which dialogue act is going to be performed by the user, since the dialogue has to follow a pre-determined schema. When moving to *mixed-initiative* dialogues, things become more difficult. Even if there could be some predictability depending on the dialogue genre and some initial assumptions, users are free to take the initiative and follow their own interaction patterns. One of the main challenges in computer-based dialogue systems is to find reliable procedures for the automatic recognition and classification of dialogue acts.

Some heuristics may guide the recognition of dialogue acts. One of the most common is that of searching for linguistic markers in the utterance. At this level, recognition should be made during the linguistic processing. In absence of linguistic markers the system needs to rely on different type of heuristic and try to predict the more probable dialogue act from both the content of the utterance and expectations the can be inferred combining information from the current state of the dialogue and knowledge about the task domain.

There are several implemented Dialogue Management Systems. We mention here some system which more or less explicitly rely on the above principles:

- TRIPS [2, 1],
- MIT Mercury Dialogue System [61],
- VERBMOBIL [40],
- Stanford MURI system [35],
- ARTIMIS [56, 57].

1.3 Simple Dialogue Models

In this section we consider the following four classes of simple dialogue model:

1. Form filling or frame-based dialogue models
2. Finite-state automata
3. Dialogue Grammars[2]
4. Information State

As presented here, these models provide an increasing expressive powers which is suitable for different types of dialogues. The first three types are rather simple and fairy easy to implement. Since they rely on rigid schema they are well suited for fixed initiative dialogues. Due to their limited power, these three models will not presented in detail here. However, most of the current commercial dialogue systems (e.g. VoiceXML) are based on these models. A special case is that of the Information State approach to dialogue modeling, since it simpler than sophisticated reasoning and planning models, but still more versatile than the above three models.

1.3.1 Information State approach to Dialogue Management

The Bunt standpoint of dialogue acts as context updates is at its best represented by the Information State approach to Dialogue Management proposed in [25] and implemented in the TRINDI system [44]. Based on Ginzburg's notion of *Questions Under Discussion* (QUD), the Information State theory of dialogue modeling 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 data-types (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; also optionally move definitions, plan libraries, static databases etc.

The dynamic part of IS (DIS) changes over time depending on occurring events and how these events are treated by the *dialogue move engine* (DME). A typical (minimal) information state structure is showed in figure 1.1 . The main division in the information state is between information 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 (i.e. grounded)

[2] Finite-state and Grammar-based dialogue models are also referred as *structural dialogue models*.

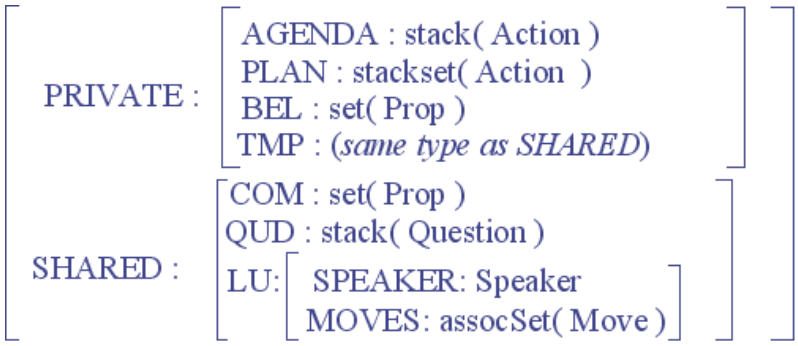

Figure 1.1: Information State (Cooper & Larson)

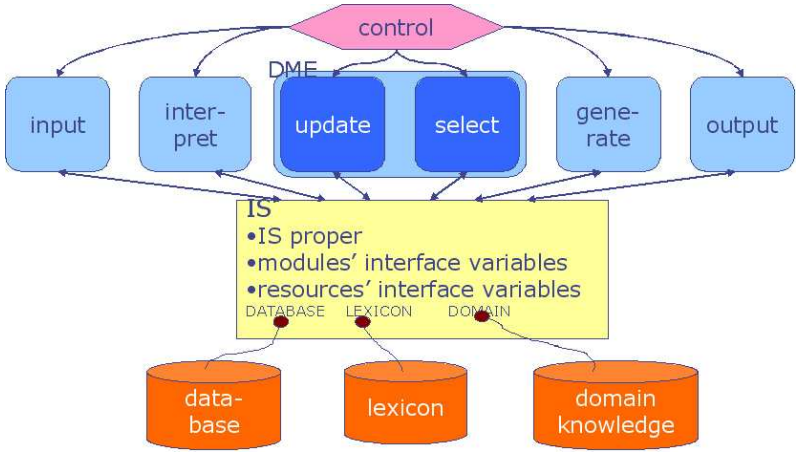

Figure 1.2: Dialogue Move Engine

during the conversation (i.e. the Lewis's "conversational scoreboard" [48]). Hypotheses about the ungrounded shared IS is kept in a temporary slot in the private IS, until an update rule for grounding is enabled and executed.

Observe that there is no assumption on how the belief state of the agent should look like, except from the fact that it is a collection of proposition. For instance, an extension of IS is one where the "bel" slot is replaced by a complex dynamic structure which represent the evolution of the agent mental state in terms of *nested mental attitudes* as the one discussed in section **??**. The "QUD" slot (i.e. the question under discussion) provides the underlying mechanism for dealing with sub-dialogues. In fact, QUD is a *stack of questions*, that is a structure that is similar to the *frame stack* in programming languages for implementing procedure calls.

The Dialogue Move Engine whose general architecture is showed in figure 1.2 , updates the IS on the basis of observed dialogue moves and selects 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 (set of) dialogue move(s) resulting from interpretation result in updates to the IS performed by the DME which:

- selects move(s) to perform
- updates IS based on interpreted user utterances

- updates IS based on generated system utterance
- selection rules (srules): selecting next move(s)
- update rules (urules): updating IS based on performed moves

The term *tacit move* refers to applications of update rules or selection rules. Rules have preconditions and effects.

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. grammar, finite state automata). In other words, in other state-based approaches it is the state itself that licenses the set of allowable next dialogue moves. The "information" is thus implicit in the dialogue state and it is very difficult to recast the information state approach to dialogue state approach since there is no necessary finiteness restriction on information states. Information state are partially described. Moreover, the motivation for update the information state and selecting a next dialogue move may rely on only a part of the information available, rather than on the whole state.

In contrast, dialogue state approaches are easy to compile into information state approaches. The dialogue state is the information state. The dialogue moves are the same move that are used in the dialogue state theory, and transition rules are formulated as updates to a new state given the previous state and the occurrence of an action. Update strategy will mirror that of the transition system (e.g. deterministic, non-deterministic). It is apparent that dialogue state approaches are special case of information state approaches where decision are taken on the whole components of the state and transition are rewriting of the previous state into a new one.

Compared to plan-based approaches, information state approach does not exclude the possibility of including aspects related to intentionality. The model is simpler than general reasoning and planning, but extensible to cope with representations of mental states.

Information state approach to dialogue management reflects the idea of modeling dialogue in cognitive terms using a computational logic-based framework. In fact, It is easy to see the similarity both with knowledge assimilation [28] and active databases [65]. As pointed out by Larsson and Traum in [44], "casting updates of the information state in terms of update rules and strategies that apply the rules under appropriate conditions, provides for a more transparent, declarative representation of system behavior than most procedural program, rendering the resulting dialogue manager easily amenable to experimentations with different dialogue strategies".

An instance of the information state approach to dialogue management is that of the Götemburg Dialogue System GoDiS[3] proposed Larsson and Cooper and described in [43] and [14]. In GoDiS moves are determined by the relation of the content to the domain: a utterance U is an answer if the content A of U is a relevant answer to a question Q in the domain. Moves are not necessarily speech acts. GoDiS has the following basic dialogue moves:

- `ask(Q)`, Q:question
- `answer(A)`, A:answer (proposition or fragment)
- `inform(P)`, P:proposition
- `request-repetition`
- `greet`, `quit`

Interpretation is carried out basically by keyword-spotting and generation is based on (partially) canned text. The domain is represented by Feature-value "semantics". For instance:

[3] A web-based demo of the GoDiS system is available at http://www.ling.gu.se/~peb/dme-demo/frame/

- "to London" interpreted as `answer(to=london)`
- "when do you want to go" interpreted as `ask(X^(to=X))`

Dialogue Plans are used to drive the dialogue. Plans are essentially lists of actions, mainly questions. Plans are static; no dynamic generation of plans. Only the system's moves are represented. Dialogue plans are domain-specific partial specifications of how the system should act in a dialogue. They a specifies *default behavior* and they may be overridden by user initiative. The operational semantics of plan constructs is determined by information state update rules. For instance, the dialogue plans for information seeking of travel information are:

- Find out how user wants to travel
- Find out where user wants to go to
- Find out where user wants to travel from
- Find out when user wants to travel
- Lookup database
- Tell user the price

The language of dialogue plans allows us to build complex plans by plan constructs which include by sub-plan decomposition:

- `action A`;
- `exec(S)` where `S` is a task/sub-plan;
- `sequence <C1,C2,..., Cn>`
- `if P then C` where `P` is a proposition and `C` a construct. The semantics of conditional is: if `P` is (believed to be) true, execute `C`
- `case(<P1, C1>, ...,<Pn-1, Cn-1>, Cn)`

Possible actions in the information-seeking domain are:

- `findout(Q)`: find the answer to Q
- `raise(Q)`: raise Q
- `respond(Q)`: provide answer to Q if there is one in the private beliefs
- `inform(P)`: inform of P

where Q is a question, and P a proposition. Questions can be:

- y/n-question: `?P`
  - Ex. "Do you want a return ticket?", "Do you want to call NN?"
- wh-question: `?λx.P(x)`
  - Ex. "What city do you want to go to?", "Who do you want to search for?"

- alt-question: `{?P1, ?P2, ..., ?Pn}`
  - Ex. "Do you want to search, add to, erase from or change the phone-book?"

The formal representation of plans for the information-seeking task are the following:

`findout(?λx.transport(x))`

`findout(?λx.dest-city(x))`

`findout(?λx.depart-city(x))`

`findout(?return)`

`consultDB(?λx.price(x))`

`respond(?λx.price(x))`

1.4 Dynamic Dialogue Models

1.4.1 States, events, causes

It is apparent that in understanding dialogues one has to face with the problem of understanding the social interaction between the participants. This means that dialogue participants are *doing* something using language and it is possible to provide an interpretation of a linguistic interaction in terms of a theory of language actions. A formal account of a theory of language action has been given by Davidson in [27]. We will look at more informal introduction following [64].

First we need to outline some basic notion in a theory of action that are necessary for the interpretation of language use as in interaction phenomenon. The first claim is that actions cannot understood without a link to some cognitive notions, in particular those related to mental concepts like those related to Intentionality. This also means that a theory of action cannot be understood in purely behavioral terms even if actions and act are components of rational agents behavior.

The notion of action is strongly related to that of *event*. A preliminary definition is that an action is an event caused by a human being (or in general by an agent). An event can be defined is terms of change of state or as a transition from one state to another state of the world. States (or situations) are snapshots of the reality at a given time. The notion of time appears and we should give an account of it. For the moment let imagine that there is a temporal order between time-points and that it is possible to look at the world as it is at a given time-point. If there is a difference between states at two different time-points (one preceding the other), we can imagine that a *changing event* has occurred between the two time-points. More formally, we can represent state descriptions by means of world-time-point pairs where worlds can be expressed, for instance, as set of properties. We can also enrich this description by adding more information on the world component such as the fact that there is a set of possible worlds attached to a given time-point, one of which is the real one.

Differences between successive states of the world $\langle w, t_i \rangle$ and $\langle w, t_{i+1} \rangle$ can be characterized by differences in the properties of w holding at different time-points. These differences may involve also the appearance of a new object or the disappearance of some others, or the begin or the end of a relation between objects. In general we assume that, if no event occurs between two time-points, all the properties, object and relations persist.

If an event e begins at t_i and ends at t_{i+1} then the states $\langle w, t_i \rangle$ and $\langle w, t_{i+1} \rangle$ are called respectively, the initial and the final state for the event e. The identification of events depends on the description language we adopt, how expressive it is and the granularity we choose in order to model the interaction between agents that inhabit a given environment. For instance, we can assign a label

to events or consider the as n-ary predicates having as arguments the object and the properties they affect. We can also try to build an ontology of events considering some inherent features of events, like for instance the way changes in the world are performed (e.g. gradually, instantaneously, having the change of feature during or at the end of the event's duration intervals).

There is an important consideration about events: a distinct event unifies a possibly infinite series of intermediate *phases*. But we are only interested at the difference between the initial and the final state, since it is that difference that is *relevant* for our understanding of the event in the framework of a situation or a scenario. The classification of event type should be also based on cognitive and conventional or social categories if our goal is not only to understand the physical changes over time, but we want to explain relationships between the causes of the events and the mental state of those agents who have caused the events.

A *compound event* is an event which can be decomposed into sub-events, occurring in sequence, but that are perceived as a atomic at a certain granularity of representation. This implies that we can consider some other situations in which those sub-events can occur independently and have their own status of atomicity and thus being decomposed in other sub-events. In compound events, the initial state corresponds to the initial state of the first occurring sub-event and the final state corresponds to the final state of the last occurring sub-event. A compound event can be *continuous* if sub-events are linked together by identifying the initial state of one sub-event with the final state of the preceding sub-event. In case sub-events do not completely cover the entire duration of the compound event, but they can periodically occur, then the compound event is said to be *discontinuous*.

A *process* can be related to continuous compound events since they can occur during a time period in which intermediate events can or cannot be distinguished. What counts in the perception and description of a process are the properties of changes rather than the differences between the initial and final states. The focus of attention in the description of a process is about what is happening during a period of time. We take an event and we make an abstraction of what can be the sub-event structure and the consequences of its occurrence in a given world in terms of initial and final states. This means that processes can be taken as more general concepts and that events can be identified as being parts of a process by distinguishing initial states, final states and some changes between them. For instance, a "talk" can be thought of as a process, while "starting to talk" or "stopping to talk" are the starting and the ending events of that process.

A *series* of events can be a set of possibly independent events while a *sequence* of events are not only linearly ordered but also linked together by some *dependency* relations. One of the most important dependency relation between events is that of causality. The causality relation may have several forms depending how this relation holds between events in a given sequence:

- each event causes the immediately successive event
- a causally related subsequence of events causes a successive event (or a sequence of events)
- an independent series of events causes an event (or a sequence of events)

Each event may thus have a simple cause or a compound cause and the caused event (or the sequence of events) is called *consequence*. A *direct* consequence follows immediately the causing event, otherwise it is *indirect*. For instance, the event of a falling glass directly causes the event of the breaking glass. The fact that I will have to clean all the pieces of the broken glass or the event of looking for another one are indirectly caused by the falling glass event.

Causality here is not defined as a relation between events and states, but rather between events. Another notion of causality can be expressed between events and states considering the fact that an event causes a state transition by changing some features of the state in which the event occurs. The final state of a causal sequence of events is called *result*. This is the ordinary notion of causality, but here we will distinguish between the consequence and the result of an event. A

state can be also "caused" as a result of an event by preventing a state transition as for instance, when I hold a glass preventing a state transition into a state where the glass is broken. A sequence of events that are related by this notion of causality is called a *course of events*. Courses of events are temporally ordered in a way such that if A causes B, then A must precede in time B. A course of events is also part of a set of *possible courses of events* one of which we consider as the *actual* one and the remaining are the *alternatives*. The indirect results of courses of events are called *ramifications*.

Causes can be characterized by consequence or results and by some conditions on consequences. An event A is a sufficient condition for the event B, if whenever A occurs in a state $\langle w, t_i \rangle$ then it is necessary that B occurs in each successive state resulting from the execution of A in $\langle w, t_i \rangle$, that is B is unavoidable after the occurrence of B. Conditions can be weakly sufficient for an event when the caused event occurs only in some of the resulting states, depending on the *force* of the causing event, that is if the the cause is sufficient in at least one, many, the majority or all the possible successive states after the occurrence of the causing event.

The notion of event and causality have been considered so far without any relation with the individual which may trigger the event (and thus their consequences and results). If we would like to express the fact that an individual, a person for instance, may cause an event, our definition of causality must be revised since only events can cause other events to occur. Intuitively, *actions* may be part of an event, and conversely an action may imply a course of events for its realization. Strictly speaking, an action may contain several events, but only a subset of it directly causes the changes we are interested in. In other words we can consider that there are some event that are related to an action but only a small subset are sufficient for the result which is intended for that action. For example, if we want to open a door there is an event of "opening the door" which causes the change of state from one where the door was closed to one in which the door is open. Moreover, there are another event which is related to the fact that an agent has pushed the door, which in turn has caused the occurrence of the opening event. The opening event might have been caused by another event, let's say the air movement caused by the wind. In the first case we say that an agent has performed the action of opening the door, but in the second case we would not claim that the wind has performed that action.

1.4.1.1 Actions, agents, plans

Actions are predicated usually by agents capable of some form of *Intentionality*. Actions imply some mental behaviors or at least presuppose some mental preconditions. An agent can unintentionally cause an event like for instance, when a human unconsciously moves the body or their parts triggering some changes in the current state of the world. In this case the notion of *doing* seems more appropriate. Analogously, an agent can be doing something by intentionally avoiding of performing an action since she wanted or intended to prevent the occurrence of the consequences caused by that action. In the other cases, an agent that is performing an action is typically involved in a *body moving event*. In summary, an agent is performing an action only in the case what is doing has been intentionally performed.

We introduced some new concepts like those of intention and agent. Having the intention of doing something can be regarded as an event that causes another event, that of doing. In this case intentions must entail a state change. This change should happen somewhere in the world and the place in which it is most likely to happen is the agent's mind. There is anyway a big difference between a body-moving event and a mental event. We may have the intention of performing an action, but I can decide of not doing it. If intentions are the primary causes of actions then the following situation would never happen. This means that there should be some conditions (other than that of sufficient) for the occurrence of some event which is related to an intention since these events not occur accidentally. Conversely, if intentions are changes in the agent's mental state under the conscious intervention of the agent, they must be considered as actions. In this case, there must be always an intention which causes an intention, which in turn is caused by

another intention and so on. A further problem is related to the fact that an agent may decide to not performing an action even if she as the intention of doing it (e.g. when the action is in conflict with social conventions or it may be the case that the action cannot be actually performed in the actual state of the world). Thus intentions require some kind of intermediate concept to link together the notion of the change of the agent's mental state and the decision of performing some related actions.

In any case, we can further distinguish two kind of action: those which are based only on some particular activity and those which also have an event as their consequences. As for events, actions can be perceived at different levels of granularity and thus *compound actions* are those actions which can be decomposed into *sequences of actions*. As for compound events, there are some intermediate results between each occurrence of the actions that are part of a compound action. The final result of an action is a subset of the union of all intermediate results, namely those which are relevant for that action. This set can be related to the intention of an action, since this set can be what has really motivated the agent in having the intention of performing that action. For instance, if I open the door I will only have the intention of performing an action which leads to a state in which the door is open, rather than having the intention of performing all the sub-actions (like moving the arm, the finger, etc...) that are necessary to perform the compound action of opening the door. This is, of course, a matter of granularity. If the event associated to an action satisfies the conditions of being a process, the this event is called *activity*. An activity can be considered as in the case of processes, as a compound action in which the intermediate states are not distinguishable.

What we are interested in when performing an action are some aspects of the resulting world. This means that an action can be related to a certain goal. Whereas an intention is strictly related to an action, a goal can be possibly not strictly related to the action itself, but the execution of an action may result into a state in which the goal can be satisfied[4]. The assertion of a goal is a mental event in which the agent answers to the question of "why an action should be performed". We will see later that goals and intentions have the same status in the ontology of mental structures (they cannot mutually defined).

The *failure* or the *success* of an action can be defined in terms of both goals and intentions. In both cases we are interested in conditions on the consequences or on the resulting states. An action is said to be fully *successful* if their consequences or final results correspond to those of its goal. Since the consequences of an action may not only be under the agent's control, an action may only partially succeed with respect to a given goal. In that case we can consider a weaker notion of success and distinguish between success with respect to an intention (I-success) and with respect to a goal (G-success). An agent may successfully perform an action with respect to her intention, but she may fail in achieving a goal since the world may unpredictably react to the occurrence of that action. The achievement of a goal by means of an action implies both an attempt of performing that action for which the agent had the intention and the fact that the world is actually affected in a way which satisfies the goal. For instance, I may have as a goal that the member of the class I am teaching understand the content of my lecture. For that I have the intention of giving a clear lecture and I will attempt in doing my best. If I fulfilled my goal depends not only how my judgment on the quality of the lecture (which can be successful from my point of view) but also on the judgment provided by the students (which can be somewhat unpredictable). Activities are not qualified and identified only by the kind of the agent's body-movement, but also by the objects and their produced changes. The final state of these changes in the considered objects constitute the *I-result* of an action.

One can argue that an action that fails is not an action.With the above definition of success we are able to accommodate this perspective by saying that an action maybe I-successful without being G-successful. The converse may hold as well since it may happens that some agent's goals can be fulfilled by chance or by some external events without having to perform the action which could

[4] In some cases goals and intentions can be the same when, for instance, the intended result of an action is itself the satisfaction of the agent's desire and it is not related to further consequences of that action.

have led to the fulfillment in absence of those events. For example, I might want to close the door when the wind does it without my intervention. In that case, the action of "closing the door" has I-failed, while it has G-succeeded.

We can restrict ourselves in considering some local properties of actions and considering intentions only within the scope of a certain goal for that action and thus requiring that an action is G-successful only if it is also I-successful. This will allows us to identify necessary conditions for the fulfillment of a goal that are those imposed by the successful fulfillment of the related intentions.

There are cases in which it does not make sense of considering only the I-success of an action and in general we can further classify action distinguishing those for which we require only the I-success and those for which both I-success and G-success are necessary. We will call henceforth the former *acts* and the latter *actions*. In the case of actions, the G-success may require the co-occurrence of additional consequences that might not be under the direct control of the agent. As for actions, the notion of *compound act* can be defined as a sequence of acts such that the result of the preceding act is the condition for the I-success of the next act. The main difference between a compound act and a sequence of acts is that for the former there exists a *global intention* from which all the intermediate intentions depend. The I-success of a compound act depends on the I-success of all constituent sub-acts, but only the final global result is identified in the global intention. Global intentions will be called also *plans*.

A plan is a course of actions which is intended to change the state of the world in a clearly identified manner. The development of acts is determined by the plan but also by the actual properties of the state in which the planned actions will be executed. This means that a plan is a flexible collection of actions which can be revised during their execution. The agent who is carried out a plan may have several alternatives (or in the worst case no alternative at all) at a given stage of the execution of the plan. The agent can make a choice among the alternative and commit herself to one of them. A possible definition of intention is the result of this decision [23].

1.4.1.2 Mental Structures of Action

We have seen in section 1.4.1.1 that a characterization of actions and acts cannot be made without considering mental concepts like intentions, goals, commitments, etc.. In this section we would like to introduce some fundamental aspects about what are the mental structures required to support the notions of actions and acts. Events take the status of actions or acts when there is some kind of awareness by the agent of what are the consequence of that event in a given situation. Since acts and actions imply changes both in the agent's body and in the properties and relations between objects of the environment, the agent must *know* the actual state of her body (included her mental state) and of the concerned objects. The agent should be also informed about all the possible changes of the world that are compatible with, for instance, the nature's physical and biological laws. This means that the agent should have a fairy rich data base which contains consistent information of her *knowledge* and *beliefs*. The difference between knowledge and belief is rather controversial, but what we can say at this stage is that knowledge is a collection of propositions whose truth is guaranteed by conventionally accepted truth criteria, whereas beliefs are propositions where only the agent commits herself with their truth, or following Dretske [32], "knowledge as justified true belief".

Knowledge and beliefs can be combined with several inference rules which allow the agent to derive new information from old information. Inference rules can be of different nature (e.g. deductive, inductive, adductive), but in general they should be sound, that is the derived information should be consistent with the old one. Rules may be also considered as a part of the agent's knowledge or belief and they can be also acquired or changed over time. In this case the agent needs an explicit representation of the inference system (the way the rules are stored and used), resulting to a higher-order of inference rules. Thus the agent mental space can be organized into several meta-levels of representation and abstraction. For the moment we restrict ourselves to the simplest

form of mental organization, that of a fixed set of inference rules and a dynamic knowledge and beliefs base.

Typically actions are aimed at changing (part of) the world in a way which ought to be with respect to the agent's *desires* and *goals*. If an agent desires or wants that the state s being realized, she must believe that at that moment it is not the case. It could be also the case that the state s is impossible and thus it can never be achieved. In this case the agent may only desire this state. In case the agent may choose among the desired states some which are (in my opinion) possible (i.e. beliefs), then the agent can build the goal of achieving s. There are some situations in which an agent may have a goal which is not among her desires when, for instance, the agent adopts someone else's desires or she has to follow the will of someone else (e.g. a blackmail).

An agent may desire p, knowing that p has an *undesirable consequence* q. If the agent believes that p is less desirable than q is undesirable, the agent will not desire p anymore. In general, an agent *prefers* p rather than q if p is more desirable between p and q. This definition of preference is based on the agent's desires, but if we consider real possibilities things become more complex. It may be possible that while the agent prefers p rather than q, the agent may also believe that consequences of p are more undesirable than those of q. In this case the agent will *reasonably* prefer q rather than p.

1.4.1.3 Rational Agent Theory

The above discussion allows us to introduce some fundamental notions which are related to the *rational* behavior of agents. A rational behavior implies that the agent's desires are controlled by inferences that can be made from the agent's knowledge and beliefs about the possible consequences of actions, environmental conditions, other agents' goals and desires, etc.

There are two possible behavior with respect to a goal or a desired state. One possibility is to wait until the desired state is achieved by the ordinary course of events, the other is to act in such a way that one believes the desired state will eventually realize. The latter behavior is defined as *pro-active*. If the agent reasonably believes that there is no other undesirable consequences in achieving the desired state, then the agent may transform her desire into an goal. More precisely, a goal is a mental state in which the agent has a representation of the state of affair or a future event together with the belief that there is the need of some action in order to obtain it. If there are some *alternatives* on how to fulfill a goal, the agent has to make a *choice*. This decision can be made rationally, that is by computing some preferences and taking into account what are the risks of failure for the different alternatives. An *optimal* decision is the one that chooses an alternative which maximally fulfill the agent's desires.

After the decision on how to achieve a goal, the agent must commit herself to a intention of an action or to a sequence of actions, i.e. a plan. Whereas goals and plans can be formulated before actions and sequences of actions, intentions are contextually determined, that is they are build taking into account a given situation. For instance, if I have a plan of going to Paris, I already know the sequence of actions needed to achieve this goal. In contrast I do not know how these action will be actually carried out in a specific situation, as for instance, in which seats I will sit in the train. Often, plans need to be partially specified since at the moment of the formulation the necessary information is not yet available. In some other case we are forced to revise our plans when we subsequently find ourselves in a situation which renders the original plan inadequate.

The success or the failure of an action may depend not only on the agent's intention of performing that action but also on the agent's *ability* or *capability* in the successful execution of that action. The difference between ability and capability lies on the fact that the former is dependent on the actual situation while the latter strictly depends on the agent. The set of agent's abilities contains all the acts that an agent is able to perform in a particular time given the appropriate conditions. The set of capabilities can be viewed as of a larger set which contains, of course abilities, plus those actions the agent is capable of performing in other possible circumstances.

The set of agent's abilities and capabilities can be further restricted by conditions that are outside the agent's control, as for instance, physical constraints (e.g. body or environmental hindrances), psychological constraints (e.g. fair, shame), social constraints (e.g. norms, obligations, rules, permissions), etc. In this perspective a doing or a process become an act or an activity only if the agent is able to *control* it, that is the agent is able to start and end them whenever she wants, under certain conditions.

An agent may act in a way such that a certain event, a process, or some other agent's action is changed from its normal course, by stopping it, or changing some of its properties (e.g. slowing down it, speeding up it). Within this type of acts there are the *preventing acts*, that is the agent's attempt to prevent that some event happens by changing the situation in a way such that the event cannot occur. In other words, the agent knows that without her intervention the event will eventually occur. The decision made of acting in order to prevent the occurrence of a future event is based on a *counterfactual* reasoning, that is an action is successful if its expected consequences would never been obtained without the execution of that action.

There are some circumstance that in order to prevent some known consequences the agent decides of not performing an action that would have necessarily caused those consequences. From another perspective the agent prefers the normal course of events she is observing without pro-actively intervene to change it. The agent is thus *responsible* of her *non-action*, and sometimes this may imply a behavior which is not consistent with some social conventions or norms. Social behavior may require that agent's action is not only driven by the agent's goals and intentions, but also by *social rules*. The *adoption* and the *respect* of these social rules or obligations allows the agent of being admitted to a *social community*. However, the agent may have good reasons to decide to break these rules by simply avoiding to act accordingly. In general, "non-acting" requires itself a rational decision and it can be itself observable since the agent may act in a different way of that which is prescribed by social conventions. "Non-actions" can be further classified in two behaviors: *abstention* and *allowance*. The former is based on the fact that the agent refuses to act as prescribed by some social convention while in the latter the agent avoids to perform acts that could to some extent change the natural course of events, that is she leaves things will happen[5].

The above description of rational behavior is just a conjecture of how, for instance, the human rational behavior can be in reality. As remarked by Keith Devlin in [29], there can never be a God's eye view in the understanding human cognitive processes:

> "Theorist may understand the behavior of some agent by imposing an individuation scheme that seems appropriate to that agent. Imposing a scheme is itself a cognitive act carried by a certain agent in the world and it is inescapably dependent upon particular individuation capacities. The standpoint of the privileged observer is a matter of stance and not a matter of fact."

Nevertheless, it may be useful to use these biased descriptions in the design of artificial systems which are supposed to some extent to interact with human beings and thus have a computational representation of a model of the human mental structures. Moreover, the design of "intelligent" programs may benefit of these representation for a better recognition of user's intentions. This is important in order to improve the naturalness of interaction with machines by a better account of *conversational implicature* [38], that is the fact that humans adopt a strategy of the minimum effort in communication relying on the pragmatic meaning conveyed by language utterance.

The recognition of the pragmatic meaning from language utterance is a particular case of action *interpretation* where the action performed is a communicative act. In general the problem of how

[5]Obligations may impose some constraint on the structure of a situation, that is the fact that some event must occur. This can be modeled as event ramifications. In contrast, when the agent decided of not-acting in an expected manner, this can be represented by the non-occurrence of the action by explicitly adding a polarity to event representations, as for instance in Situation Theory [29] by the infon: $\ll event, agent, o_1, \ldots o_n, t, l, 0 \gg$.

reconstruct the agent's intentions and goals from the observation of her behavior is of central interest for a theory of the interaction between cognitive agents. It is fairy easy to describe and represent the occurrence of events and processes, but the framework become very complex when we are interested in what the mental structures look like when these events have been caused by a cognitive agent. In general we are able to understand what an agent does only if we are able to interpret an event or a process as an action or an activity, that is when we are able to reconstruct the presupposed intention, the goals and other possible reasons that motivated the agent in performing that action or activity. In this process of recognizing the agent's mental state prior to the execution of an action, also social and cultural conventions play a fundamental role. It highly depends on the kind of situation in which the agent finds herself, that the agent behaves in a more or less predictable way.

Intentions are easily recognizable through the observation of the agent's act by assuming that an agent performs an act following a plan, and if the observing agent knows what plan is. The recognition of the agent's goals is more complex since it requires to find an explanation of the action in terms of how the action's consequences play a role in achieving the agent's goals. This cannot be evident from the agent's behavior since the agent may have not considered all the consequences of her action and since the action may represent only a partial fulfillment of a more global goal. Again here context and social convention may be of a great help in finding the right interpretation of an agent's behavior since the agent may be co-operating with some other agents or she may have a particular (known) role within a community. In this case some preferences can be ascribed by default to the agent, which may provide, to some extent, an explanation of her actions.

1.4.1.4 Social Interaction

The path towards a theory which attempts to provide an explanation of the nature of communication between rational agents should take into account the more general aspects related to the nature of the *social interaction*. Different agents may participate in the achievement of a goal or an intention perceived as unique at a certain granularity as for instance, to lift a heavy weighted object or to play a game. The majority of human activities have social implications and human acts are often part of a social interaction.

We need to distinguish different types of interaction and also distinguish between those action which are part or not of an interaction. The presence of several agents does not necessarily implies an interaction between them. For an *interact* to take place we first require the presence of at least two agents, which are acting at the same time or at different times in the same sequence of actions. This requirement is common to most types of social interaction.

The first type of interaction we consider here is related to the *co-operative* behavior of several agents. In this case the participating agents perform an act together even if their respective acts may be distinct. In order to perform this *global act*, the involved agent should at least have the same *joint intention*, while they may share or not the same underlying goal. The set of agents participating in the realization of a global act can be perceived as a single agentive entity and thus is a matter of granularity if we can consider this set as a single *collective agent*. The fact that two or more agent have the same type of intention is not itself sufficient for co-operative behavior. For example, two persons may be fishing together and also be fishing in the same river and being aware of the presence of each other. However, the success of their intentions are not mutually dependent. Intuitively, the two persons do not co-operate. In contrast there are some other acts that require for their success both the interaction and the sharing of exactly the same intention, as for instance the act of marriage. In the simplest case, each agent may *commit* herself with some acts she knows of being mandatory for the success of the global act.

There are another type of interaction which involves co-operating agents that may perform different acts and having their own appropriate intentions. These acts are sufficient or necessary components

of a compound act, for which the participating agents share the same plan (e.g. a team that builds a house). In this type of interaction each participating agent may have a different *roles* and the acts they are performing must be *co-ordinated*. Co-ordination is a very complex issue since implies notions like *task assignment*, *supervision* and the fact that participating agents may be able of *self-organizing* or they need a *global supervision*.

Although intentions and plans may be the same for co-operating agents in order to achieve a joint intention, it is not a necessary condition that the agents share a joint goal. Vice versa, while two persons share the same goal, their intentions can be different or rather conflicting as in the case of game playing. In such a situation the players have a the same goal of have fun playing a game, but each of them have the intention to win, and of course these two intentions cannot be realized at the same time.

The form of interaction where neither the intentions and the goals are the same may lead to highly conflictive situations where the goals of only one participant are achieved that have as pre-condition the non-achievement of the other participants' goals (e.g. a war). In an intermediate degree of interaction there may be also situations in which the participants have different intentions and goals, but their activities are in a sense correlated, like for instance in some kind of *negotiations*. Sidner [62] presents a model of *collaborative negotiation* based on the idea of establishing mutual beliefs, that is, things that we hold in common. This model rests upon the absence of deception, and appears fragile in the presence of mutual misunderstanding. The work of Cohen and Levesque [23] and of Smith and Cohen [63] is very similar to Sidner's work, but relies in addition on the primitive notion of *joint goals*. Based on Searle's idea [60] that requesting something means that one is attempting to get an agent to perform an action, they define all illocutionary acts in terms of agent's mental states (illocutionary is an act performed as the result of a speaker making an utterance; the effect is called a perlocutionary act).

Some type of interaction may be very complex, but they are all based on the success of the actions which make part of the joint plan. Co-ordination require knowledge (or its assumption) of what are the goals, intentions, and capabilities of the co-ordinated agents. In the case where there is no external supervision, it is required that the agents are capable of self-motivating and self-organizing in a suitable way, and in any case there must be some a-priori social convention (or protocol) to which the participating agents have to conform.

Conventions may indicate the way a certain action should be interpreted in a given context. They might represent certain social rules which constraint the behavior of a group of agents in a given situation. For instance, a policeman will use a particular gesture to stop me rather than simply as a sign of greetings. Rules of interaction can be explicitly represented as *norms*, or implicitly assumed by the participants. Rules may imply different degrees of *obligation*. The degree of obligation can be represented by how strong is the punishment obtained by not having respected the rules. In the social interaction agents are obliged of being *responsible* of their actions. This also means that the agent's actions performed in a social context must take into account how much they interfere with the achievement of other agents' *legitimate* goals. In other words, an agent is allowed to perform only actions that have *legal* consequences. Agents are punished only for the acts for which they are responsible and have illegal consequence that are under the agent's control. Those other consequences caused by the agent's action that also depend on conditions which are not under the agent's control cannot be in principle punished unless the agent's action is a necessary condition of their realization. In this case the agent has decided to act regardless of the fact that an illegal consequence could have possibly occurred.

1.4.2 Plan-based Cooperative Dialogue Model

The previous section provide the necessary background to introduce a sophisticated dialogue model based on the Rational Agent theory. Understanding of user communication is obtained by means of *intentions or plan recognition*. The categories of the Rational Agent Theory can be formalized

within a computational model. The mental activity of generating plans aimed at achieving an agent's set of goals can be defined in computational terms as a *directed search* through a problem space with three inputs:

- A description of the current situation
- Some set of goals for the planning agent to achieve
- A set of actions which can be performed by the agent (the plan library)

A *planning* algorithm generates, as result a sequence of actions (the plan), which, when executed will achieve the given set of goals.

A simple and common formalism for representing planning problems (or scenario) is the STRIP formalism . World is described by a complete set of propositional literals. STRIPS assumes the Closed World Assumption(CWA): all atomic propositions which are true are known and all atomic propositions non explicitly listed in the description are assumed to be false. This means that the world state can be represented as a set of atomic propositions.

STRIPS can only represent conjunctive sets of attainment goals: statements which must be true only at the end of the plan. Goals which project into the future are not expressible. The STRIP formalism makes use of plan operators which are used to represent actions. A plan operator can be applied to entities of the world only if the preconditions are satisfied in the current state of world. If the precondition hold, the state of the world is updated by removing some proposition from the world state and adding some other. The following is an example of a plan operator for moving a block in the Block World scenario.

```
Name: move_block(Block,From,To)
Preconditions: on(Block,From),clear(Block_name),clear(To)
Effects:
    Add list: on(Block,To),clear(From)
    Deletion list: on(Block, From),clear(To).
```

Plan recognition is closely tied to plan generation. It can be viewed as a form of simulated or nested planning. We distinguish two type of plan recognition methods:

1. *Keyhole recognition*: the observed agent does not intend the plan to be recognized.
2. *Intended plan recognition*: the plan is intended by the planner to be recognized by the observer. The observer can usually make the assumption that the plan relies on belief which are evident to both the planner and the observer (i.e. the common ground).

While the first type makes no assumption of cooperativity, the second assumes a joint commitment in their joint activity sense making. This notion is similar to the Grice's concept of meaning-nn[6] in case of the joint activity is of communicative nature.

In [24, 3] Cohen, Allen and Perrault were the firsts to propose the interpretation of speech acts as planning operators. Plan recognition is then viewed as a form of specialized planning where recognizing a plan is a sub-goal within the planning process of understanding. Kautz in [41] considered plan generation is a form of hypothetical reasoning. Within this perspective, plan recognition appears as a more uncertain process in which events which have occurred are linked to what he calls *hypothetical explanations*. Kautz introduces the notion of *commitment* in order to overcome

[6] Grice makes a distinction between natural and non-natural meaning (meaning-nn) of utterances. The speaker S meant-nn Y by uttering U if and only if: (i) S intended U to cause some effect Y in the hearer H, and (ii) S intended (i) to be achieved simply by H recognizing that intention (i).

the limitations of previous approaches which suffer from the inability to choose among competing explanations in principled way. In fact, the arbitrary choice between competing explanation leads to an *over-commitment* to one explanation. A selection criteria is required, which is based on two assumptions:

1. the planning agent has a complete planning library (I.e. the known ways of performing an action are the only ways)
2. all actions occur for a reason and all reasons for actions are known.

1.4.3 Plan recognition and language understanding

The conceptual link between plan recognition and language understanding is obtained by background knowledge of how actions can be used to achieve goals. Representing speech acts as plan operators, the observation of a speech act allows the hearer to ascribe the preconditions of the action as conditions on the current mental state of the speaker, and the effects of the actions as the plans the speaker is currently adopting in order to achieve its goals. In [55] Pollack adopts this perspective by introduction of the notion of *recipes for actions*. A recipe for an action is the knowledge of how to do an action. Pollack further extends the original notion of plans as recipes by considering plans as *states of mind* of an agent who has the intention of performing a recipe. Plans do not simply correspond to recipe, but are complex mental attitudes comprising of a structured collection of beliefs and intentions.

In order to relax the assumption that communicating agents share the same beliefs, the theory must distinguish between plan as recipes and plan as mental states. In previous approaches plan inference depends on smaller recipes for action which are mutually known to agent and observer. This leads to problems since the application of the heuristic rule (i.e. use a subset of the recipe when preconditions are partially satisfied) by the observer does not consider the speaker novel beliefs (i.e. agent believes recipes which are not known to the observers), avoiding generating appropriate response to mistaken beliefs.

A general algorithm for plan recognition in language understanding are based on the following attitude requirements for an agent A having a plan $P = (a_1, \ldots a_n)$ for achieving S:

1. A believes that he can execute each act a_i in P (Capability).
2. A believes that executing each act a_i in P will entail the achievement of S (Sufficient).
3. A believes that each act a_i in P plays a role in his plan (Necessary).
4. A intends to execute each act a_i in P.
5. A intends to execute P as way of achieving S.
6. A intends each acts a_i in P to play a role in his plan to achieve S.

These requirements can be used to ascribe mental conditions to the speaker by the hearer. While requirements 1,2,4,5 are related to the choice made by the agent with respect to which plan will achieve their goals, the requirements 3 and 6 are related to the explanation of why the plans have been chosen (i.e. their relevance).

Invalid plans are those which fail in fulfilling the above requirements. A has an invalid plan if any of his/her beliefs in (1-3) are wrong. In particular:

- *unexecutable plans* falsify beliefs in 1 are false.

- *ill-formed plans* falsify beliefs in 2.

Consequently, false beliefs in (1-3) produce *unrealisable intentions* in their respective intentions (4-6). Unrealisable intentions produce invalid plans.

The above requirements can be used for understanding of communication also in case where the two participants in a dialogue do not share the same beliefs and beliefs as shown by the Pollack's example where two persons have the following exchange:

A: "I want to talk to Kathy, so I need to find out the phone number of St. Eligius".

S: "St. Eligius closed last month. Kathy was at Boston General, but she's already been discharged. You can call her home. Her number is 555-1238"

S believes that A has a plan by ascribing beliefs (1-3) and intentions (4-6), namely that A believes that calling St.Eligius is executable. This beliefs clashes with S who believes that the plan is unexecutable. Co-operation involves the observer in finding discrepancies between his/her own beliefs and those attributed to A, except for what the observer considers to be irrelevant. This principle forces S to act accordingly and try to reestablish the consistency between A's and S's beliefs providing the missing information which enables A to falsify one of the assumptions he/she made when choosing the plan for achieving the goal of talking with Kathy. Being cooperative, S provides also an alternative plan to A together with an explanation of the failure of the initial plan. For S to reply, S has to believe that A's act of calling St. Eligius cannot be performed as it is shut down, and even if this where not the case, it would not achieve A's goals of talking to her as Kathy. To correct A's apparent incorrect beliefs and provide the most cooperative reply, S provides A with information that will allow A to re-plan and achieve his stated goal.

In general, the observer S assumes that the speaker A shares the observer's beliefs unless no contrary evidence, and it ascribes *explanatory beliefs* during plan recognition from the observer's personal set of beliefs. In other words, the observer follows some *belief heuristics*:

- The observer believes that A has all the same beliefs on conditional generation as the observer has (e.g the agent and the observer share the same plan action library).
- The observer attributes slight variations on his/her owns beliefs (about relation between act types) to A.
- The observer believes A has confused or made a bad analogy between two similar act-types.
- The observer can explain incorrect plans by assuming that an agent has constructed a plan which includes an incorrect action choice

Plan inference involves a collection of beliefs and intentions plus ascribing higher-level beliefs and intentions, to an actor. For any act a_i in a plan P achieving a goal S, there must be a higher-level belief about what role a_i plays in P. The observer must ascribe an *explanatory plan* (e-plan) to A. In case of failure in doing this the observer may conclude that the plan is *incoherent*. Incoherent plans are different than ill-formed or un-executable plans since they appear to be of having no relevance in achieving the type of goal A is assumed to pursue. In the above example, a plan is required to call Kathy: establish a telephone channel. Having the right telephone number is a sub-goal, which enables establishing the telephone channel with Kathy, and which requires finding a suitable plan for achieving it. If A had uttered the following query:

A: "I want to talk to Kathy, so I need to find out how to stand on my head"

the observer would not have been able to find the causal link between the two plans: that for establishing the telephone channel with Kathy and that for standing on the A's head. The relevance of an act-type (i.e. the plan operator) with respect to another act-type is thus represented by the presence of conditions in the effects of the former in the preconditions of the latter.

2 Cognitive Dialogue Modeling

2.1 BDI-agent model

Following Allen the components of a conversational agent are:

- Perception
- Beliefs
- Desires/wants
- Planning/reasoning
- Commitment
- Intentions
- Acting

These categories are the fundamental components of the Belief-Desire-Intention agent model. This model defines also a computational architecture of rational agents which is represented by the diagram in figure 2.1 .

Belief is a type of *mental (propositional) attitude* which play a fundamental role in the BDI architecture. Beliefs are the main components of the agent mental state and they are propositional attitudes held by an agent to be true. Propositions can be transparent (extensional) or

opaque (intensional). Co-referential terms cannot be substituted in a opaque propositional attitude without changing its truth conditions (e.g. John may believe that the Morning star and the Evening star are different astronomic object). Opaque propositional attitudes are a sort of private knowledge[1]. Natural language may express different *degrees of certainty* in believing something:

- "John believed that David was guilty"
- "John suspected that David was guilty"
- "John had a hunch that David was guilty"

Another fundamental notion is that of *mutual belief*, which is of central importance mutual understanding in communication. A proposition P is a mutual belief (MB(A,B,P)) if shared by two agent A and B such that:

A believes P

B believes P

A believes B believes P

B believes A believes P

[1] Knowledge can be viewed as a justified true belief.

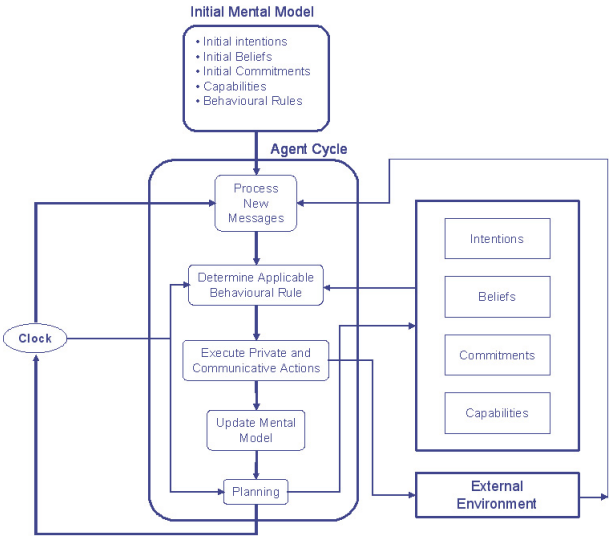

Figure 2.1: BDI-agent model

Etc., ad infinitum

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 [23], avoiding infinite recursion.

Unfortunately, this approach does not allow intermediate belief nesting: a belief become mutually believed as the result of speech acts between the dialogue participants. Other problems arise also when trying to distinguish 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/her agrees only to stop arguing). Moreover, it is psychologically too strong: real dialogues frequently feature corrections and repetitions. If mutual belief could be obtained by utterances then such dialogue control acts should not be required.

Desires are the second component of the BDI model. A desire can be intuitively defined as A state of the world the agent finds pleasant Desires seem to arise subconsciously. Different desires may conflict with each others. Desires are often identified with *goals*.

Intentions derive from desires since desire represent motivation for acting intentionally. Intentions arise from rational deliberation and they are future-directed: they reflects decision an agent has made about his/her future action. Having an intention leads to intentional actions (by commitment). However, intentions are not necessarily realized.

As for Beliefs, Desires (Goals) and Intentions can be viewed as propositional attitudes. In particular the semantics of Intentions can be modeled as a plan operator with the following preconditions (or constraints):

- An agent can't intend to do two actions (or plans) that he/her believes are mutually exclusive
- An agent can't intend an action he/her believes he/her cannot perform

Since the above definition implies that the agent must have a plan for achieving a goal, having a plan is not a sufficient condition for intention. For example,

"Jack has a plan for robbing the bank".

One reading (feature-directed) may be that: Jack intends to rob the bank. Another reading (recipe): Jack has worked out a scheme by which someone could rob the bank. Having a recipe does not mean that you want to make the meal!

Agent are typically resource bounded and therefore they cannot constantly re-process their goals and beliefs to decide what to do next. At some point the agent has to commit to a particular plan of action, which might be re-evaluated if the situation drastically changes. Agents need to coordinate future actions. Once a future action is committed to, decisions are typically made as to what other actions to do along with this action.

As remarked in Bratman [16], intentions pose problems for the agent, since the agent needs to determine a way to achieve them. This implies a strong motivation in finding a plan of action to achieve the intention. Bratman proposes three different roles of intentions in the behavior of a rational agent:

- Intentions provide a *screen of admissibility* for adopting other intentions.
- Agents cannot adopt simultaneous conflicting intentions.
- Agents "track" the success of their attempt to achieve their intentions.

Agents are also motivated to build alternative plans to achieve the intended effect in case of failure of previous generated plan. What an agent intends is a subset of what an agent chooses. If the agent believes that a plan will cause a side effect, then the agent has chosen to achieve a goal and the side effect, but has only intended to achieve a goal. If the plan fails to achieve both the goal and the side effect, the agent will form a new plan to achieve the goal, but not the side effect.

2.2 The ViewFinder model

Some frameworks have been proposed so far for the representation of information assimilated through perceptive and cognitive processes into the mental state of a cognitive agent, but only few words have been spent on how this information should be organized with respect to mental attitudes such as beliefs, desires and intentions. As we have seen before, looking at natural language understanding as a cognitive activity requires at least the consideration of aspects related to intentionality, that is recognize what are the driving motivations for the production of communicative acts.

Another important issue related to mental representations is that of how acquired information is stored in appropriate data structure. We believe, in line with current trend in knowledge representation that both *local* and *global* views of information must be taken into account. Local views are those view of the perceived reality that are focused with respect to some relevant aspect of it, while global view are about the relationships that hold between local views. Moreover, global views may be used to accommodate agent's private knowledge with social or cultural conventions which may constrain the use of private knowledge.

In this section we extend the methodology for mental representations proposed by Ballim and Wilks in [11] and further extended by Ballim in [13] and by Lee in [45]. The main motivation of their works comes from some reflections about the current trends on semantics and pragmatics. The main criticism to classical approaches to semantics and pragmatics in computational linguistics are that language understanding cannot be decoupled from the representation of intentionality. Moreover, this representation must be computational if our goal is to build artificial systems that are capable of some understanding which goes beyond pure structural linguistic analysis (e.g. phonology, morphology, syntax). This does not mean that language understanding can be carried out without any account to structural linguistic analysis, but only that a meaning representation

cannot be obtained without any explicit account (i.e. representation) of mental structures (e.g. belief, knowledge).

Pragmatic theories may well account for descriptions of communication and in general of the behavior of cognitive agents, but some of them fail to provide connections with aspects of the cognitive processing of information that happen behind the scenes. In order to build mental representations of the perceived information, one can assume that there exists some general principles and general representation that are common to all the cognitive agents. This is the perspective taken by those who assume the existence of a common *language of thought* (LOT). Without entering in the philosophical discussion about the evidences for the existence of a common LOT, a discussion which can also be found in [11, pag. 45-66], we definitely agree with the perspective that it is not important what exactly is the nature of the LOT for artificial cognitive agents, whereas it is of fundamental importance the way information expressed by different LOTs 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 linguistic analysis. In this work we will partially addressed the problem of this coupling even if in the first part we considered the problem of robust linguistic analysis at different levels. There is a big gap to be filled between linguistic content representations and the way these representations are used to build the recipient's (e.g. hearer or reader) mental structures that serve as the basis for further cognitive processing.

Partitioned representations for structuring mental spaces is the main objective of recent theories in Knowledge Representation and Reasoning. Partitioned representations have been advocated as a main conceptual tool for structuring information without adding any spurious semantics to its content [30]. As in Information Retrieval, indexes leave unaltered the content of the indexed documents, partitioned representations are a great utility when one wants to improve the way information can be accessed and maintained. The main principle relies on the assumption that information typically has local coherence. Clustering information into possibly overlapping spaces and giving to these spaces a global organization definitely improves the accuracy in finding relevant information for a given task. In the case of cognitive agents, we require these tasks of being not only retrieval tasks but mainly inference tasks. Information stored in partitioned spaces can be connected together by stating explicit relationships between spaces and their content. As in semantic networks we would like to relate concepts between each others, but in case of partitioned representations, we abstract from the actual content of the spaces while retaining only the types of spaces stating what is the relation between these types and how the contained information can be amalgamated.

Partitioned representations have had a great success in programming languages, although without appealing to the same notions. First in structured and after in modular and object-oriented programming, the notion of encapsulating parts of a whole program into some coherent pieces and let them interact is some way (e.g. by means of procedure calls, parameter passing, methods invocations). Moreover, also the dynamic creation of non existing code as the by-product of static relations between logically related parts of the programs (e.g. inheritance, polymorphism) can be thought of as a successful example of the smart use of partitioned representations.

In Artificial Intelligence and in particular in Knowledge Representation and Reasoning, the usefulness of partitioned representations have been early recognized by John McCarthy [51]. A survey on this topic [15] 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. The latter, considers partitioned representations as *local theories* of the world interconnected by a network of relations. Theories of context fall in one or both of the above categories and it is easy to see that these categories are in fact two perspective of the same problem. The work of Dinsmore [30] and that of McCarthy, formalized in [21] are representative of the divide-and-conquer methodology while Local Model Semantics (LMS) [36] and ViewFinder [13] fall instead in the compose-and-conquer category. We will concentrate in this section on

the second methodology since ViewFinder is the underlying theory that will be used for mental representations. Compared to ViewFinder, LMS may seem more general since it is characterized both by a model theoretic semantics and by a proof theory, the Multi Context Systems [37]. In reality the two models are comparable since in ViewFinder provides an algebraic semantics to partitioned representations that are called environments. Environments are viewed as containers for representations of whatever kind provided that there is a clear correspondence between the symbols used in more than one environment.

As we have already discussed in previous sections, language supports only partially what is actually conveyed by utterances, that is by communicative acts. Very often only inference and the context allows the recipient of the communicative act to extract the speaker/writer intended pragmatic meaning (i.e. illocutionary and perlocutionary force). A sound and useful interpretation of communicative acts can be done only if the hearer/reader is able to reconstruct the relevant parts of the speaker/writer mental state, in particular those parts related to their goals and intentions that motivate their behavior. We could put in other words saying that understanding a communicative act is the search for the best explanation of why an agent have performed that act in a given context. There are several techniques to computationally deal with the problem of intentions recognition (or elsewhere referred as plan recognition when the agent is assumed to be rational, thus capable of generating plans of actions). Plan recognition within the framework of ViewGen has been addressed by Lee in [45]. The main point of his work is that plan recognition can be obtained by simulating the cognitive process that an agent may perform using ViewGen and a general purpose planner.

In this section we review the basic fundamental notions of ViewFinder. Details are available in [13]. The fundamental notion in ViewFinder is that of *environment*. Environments are container for information and can be *nested*. This means that an environment can contain other environment which in turn contain other environment and so on. ViewFinder is the generalization of ViewGen where environments correspond to belief spaces. A belief space contains some propositional content which is assumed to be consistent. Propositional content can be organized with respect to topics. A belief space is thus a point-of-view with respect to a particular topic, that is an attitude towards a particular content. A topic itself is a special type of environment. As in the case of environments, belief spaces can be nested, that is a belief can represent a point-of-view with respect to another belief space. Nested environments can be created extensionally or by means of operators. Environment operators project the content of an environment onto another environment. Environment operators thus provide a means for the intensional generation of nested environments. In the case of ViewGen, a particular operator is proposed: *ascription*. Ascription takes the propositional content of a belief space and projects it onto a inner belief space. Ascription can be viewed as a rule that dynamically generates or updates the content of a nested belief spaces by projecting in the content of the outer belief space, provided that there is no explicit contrary evidence. Contrary evidence means that the target belief space does not contains content that if combined with the projected information becomes inconsistent. While ViewFinder makes no claim about the nature of the content of topic environment, ViewGen uses a fragment of first-order logic. In our implementation of ViewFinder we depart from the original formulation and we propose an alternative language for representing propositional content.

ViewFinder [13] is a framework for manipulating environments. Environments (or views, or partitions, or contexts) are aimed to provide an explicit demarcation of information boundaries, providing methodological benefits (allowing on to think about different knowledge spaces), as well as processing ones (allowing for local, limited reasoning, helping to reduce combinatorial problems, etc.). The ViewFinder framework provides the foundations for 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. The above issues have been considered in greater detail in the specific case where environments represent agents' beliefs spaces and has been implemented as a PROLOG program: the View-Gen system.

The ViewGen system [66, 10, 11, 12] is intended for use in modeling autonomous interacting agents and it is a restricted implemented version of ViewFinder specifically tailored for modeling agents' mutual beliefs. A *belief environment* represents each agent belief space and it may use nested environments to represent other's agent beliefs spaces. As pointed out in [8],

> "the attribution of belief by means of ascription can be generalized to other mental attitudes providing a common theory of mental attitude attribution".

More generally, each agent in dialogue has a belief environment which contains attitudes about what other agents:

- Believe
- Want
- Intend

ViewFinder accounts for nested environments and it allows the creation of *viewpoints*, that is beliefs about other agents attitudes towards a propositional content. For instance, the expression

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

can be interpreted as the situation where John believes that Fred wants Anne to believe P. Different attitudes (e.g. beliefs, goals, intentions) correspond to different environment types. Figure 2.2 shows a graphical representation of nested environments. Viewpoints are constructed starting from the subjective point of view of the agent who is performing the cognitive act of holding a viewpoint. This agent will be referred as the *System*. Nested belief environments can also be used by a coordinating agent to reflect the agency topology and agents behavior as showed ing figure 2.3.

Two main operation with environment is ascription which can be on two type of rules:

Default ascription: Given a System belief, ascribe it to any other agent as required, unless there is no contrary evidence.

Stereotypical ascription: Given a System stereotypical belief, ascribe it to any other to which the stereotype applies.

The default ascription rule assumes that most beliefs about the world are shared by the participants to a dialogue. Thus it is not necessary to have an explicit notion of common beliefs, but rather assume that all the belief held by an agent are held by other unless they provide a contrary evidence of this fact. Evidence against default ascription is the presence of a conflicting proposition with that ascribed (e.g. opposite or inconsistent with other beliefs held by the agent). The situation where the System ascribes the belief that the world is round is showed in figure 2.4. Unlike beliefs, the assumption that other agents share similar goals or intentions cannot be made by default. However, there are conditions in which this can happen, for instance in case agents are strongly cooperative and one can ascribe them all his/her goals. Goals and intentions are better treated by stereotypical ascription and by plan recognition.

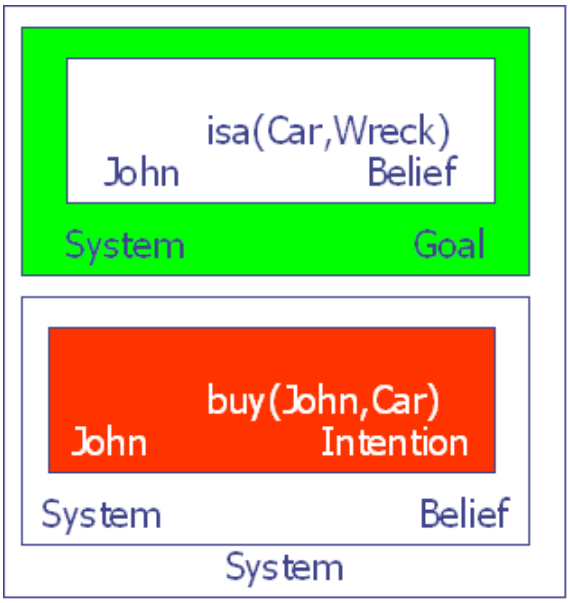

Figure 2.2: The system believes that John wants to buy a car, and it has as a goal to convince him that that car is a wreck.

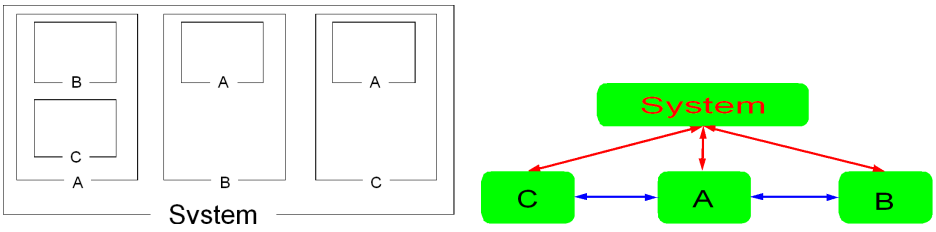

Figure 2.3: Agency Topology and Nested Environments

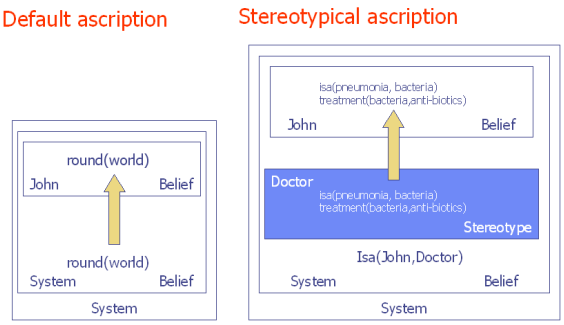

Figure 2.4: Default and Stereotypical ascription

Stereotypes represent classes of agents. The membership of an agent to a given class may denote the fact that the agent has some competence about some topic. A stereotype can be viewed as a collection of attitudes that an agent can ascribe to another agent known to be part of the corresponding class for that stereotype. This can be done unless there is explicit contrary evidence, as in the case of default ascription. Figure 2.4 shows an example of stereotypical ascription where the System has a stereotypical set of beliefs for doctors and believes that John is a doctor.

2.2.0.1 Attitudes report in dialogue with ViewGen

A significant amount of work in mental attitudes report has been done by Lee in [45], which has led to a great improvement of the original ViewGen framework. ViewGen can be used for plan recognition from speech-acts by a suitable integration of planning, ascription and inference. Lee proposes to overcome the limitations of the original implementation of ViewGen following the theoretical foundations and the generalization of ViewFinder extending the representational framework to cope with remaining mental attitudes by means of *typed environments*. The type considered by Lee are those of interest in the case of plan recognition from dialogue. Based on this extension he proposes the amalgamation of ViewGen with the Partial Order Causal Link (POCL) planner [49]. His work led to the successful treatment of a set of speech acts partially based on the Bunt's taxonomy [19] and empirically tested on a dialogue corpus.

The notion of agent stereotype is extended to those of situations types (i.e. dialogue types or protocols) and discourse types (triggered by the actual linguistic context). Finally Lee also provides an account for indirect replies [46] and implicatures [47] under the assumption that interacting agents are rational and cooperative.

Lee proposes a theory of speech acts which tries to meet the following requirements:

1. The theory should be solipsistic.

   The notion of mutual belief introduced to provide a realistic account of the effect of a speech act on a hearer, is too strong. The assumption that there exists are reality which can be "objectively" represented cannot be accounted by finitary representation. The notion of mutual beliefs requires an infinite level of nesting (defined as a fix-point operator). Although mathematically sound we look for computational efficient approximation of this idealized concepts. Default ascription seems to be a sound trade of for the modeling of mutual beliefs (elsewhere referred as mutual knowledge, common knowledge, shared knowledge, etc.).

2. The theory must provide separate interpretations for the speaker and the hearer.

   The theory must take into account the attitudes of both the speaker and hearer by allowing the separate derivation of the effects of a speech act from the speaker's and the hearer's points of view.

3. Speech acts should be minimalistic.

   The theory should made the minimal assumption on the effects of successful speech act. In general this would avoid to retract assumptions in case of evidences arising further in the dialogue. Effects of speech acts are explicitly stored in environments. However, information can be intensionally derived (i.e. ascribed) from the current mental state of the represented agent (i.e. the speaker), which might be blocked in future during the dialogue.

4. Speech acts should be extendible.

   This requirement seems to be in conflict with the minimality requirement. However, it should be possible to build a hierarchy of speech act where specializations add more applicability conditions and assumptions on the effects of the speech act.

5. The theory must provide a means to derive generalized effects from each acts conditions.

   Classification of speech acts must be based on conditions rather than on effects as argued already by Searle in . However, it is desirable to model conventional effects in a principled way of any act from its conditions. Moreover, this is necessary if we want to provide a clear distinction between an act's conventional illocutionary effect and its context-specific perlocutionary effect.

Dialogue Acts are represented using an extended version of the STRIPS action formalism[2], that is by specifying pre-conditions and effects. Speech acts are classified with respect to their pre-conditions, which are the mental attitudes a speaker must adopt to felicitous perform the speech act. The syntax for a speech-act definition is the following:

```
<performative>(<Speaker>,<Hearer>,<Proposition>)
Preconditions: C
```

where C is a set of conditions over the speaker mental state. For instance the representation of the Inform speech-act is given by:

```
Inform(Speaker,Hearer,Proposition)
Preconditions:
           bel(Speaker,Proposition),
           goal(Speaker,bel(Hearer,Proposition).
```

Any speech-act is treated by two separate update rules for each participant's point of view:

- Update rule from the speaker point of view:
  - For every condition C in dialogue act performed:
    - *default-ascribe(Speaker, Hearer,believe(C))*
- Update rule from the hearer point of view:
  - For every condition C in dialogue act performed:
    - *default-ascribe(Hearer, Speaker,believe(C))*

Belief attribution is made in a principled way: a default update rule, rather than specify individual rules based on each dialogue act. Figure 2.5 shows the speaker's belief attribution for an Inform speech-act. After performing an inform act, the speaker can ascribe to the hearer the belief that

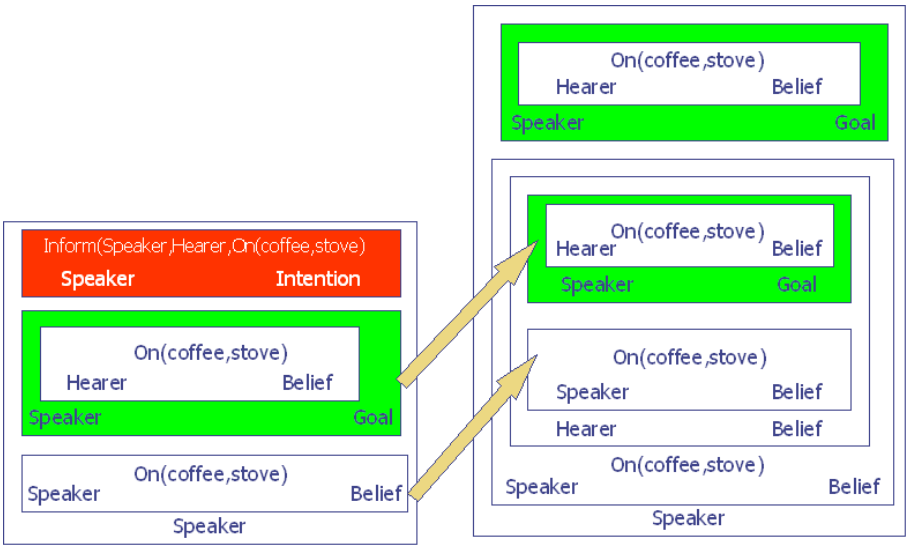

Figure 2.5: Speaker's beliefs attribution by the Inform speech-act processing

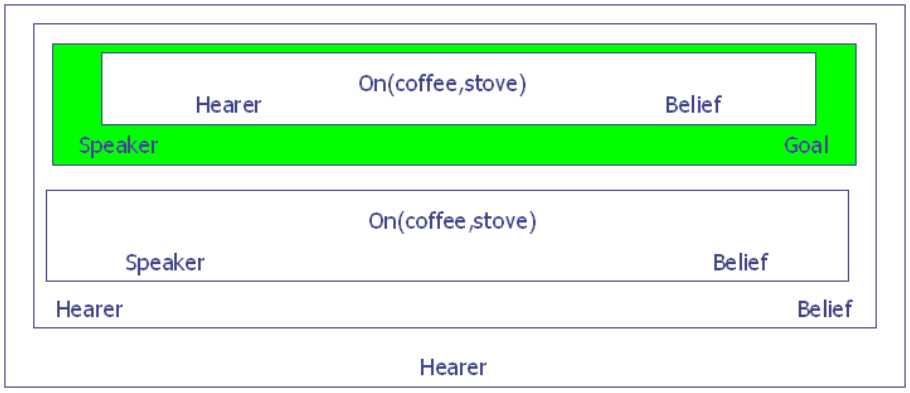

Figure 2.6: Hearer's beliefs attribution by the Inform speech-act processing

each of the precondition were met. (i.e. the speaker believes that the hearer believes the speaker believes the proposition and has the goal of getting the hearer to believe it too).

Figure 2.6 shows the hearer's attitudes after the same speech-act. Given that the speaker has performed an inform act, the hearer can ascribe to the speaker the preconditions of the inform act assuming that the speaker is being cooperative. The hearer's update rule is one level less nested: the preconditions rather than beliefs about the preconditions are ascribed.

2.2.0.2 Plan recognition in ViewGen

ViewGen is able to use a planner to *simulate* other agents planning. Since ViewGen represents the attitudes of agents in nested environments, the simulation can be applied to any depth of nesting. For instance, the system simulates John simulating Mary generating a plan to achieve a given goal considering its beliefs of Johns beliefs of Mary's beliefs, goals and intentions.

[2]See the formalization of the block worlds in Event Calculus in chapter 2.

During nested planning ViewGen has to reason about which beliefs are held to be true at that level of nesting. Belief ascription is performed when required: we cannot predict which beliefs will be relevant to a plan before the plan is constructed. Ascription must be performed as the plan is generated.

Belief ascription is modeled as a mental action. Two plan operators are required:

- Default and Stereotypical ascription:

```
Def_bel_ascr(A1,A2,Prop)
Precond.: bel(A1,Prop),
          bel(A1,not(bel(A2,not(Prop)))).
Effects:  bel(A1,bel(A2,Prop)).
```

- Belief adoption:

```
Accept_bel(A1,A2,Prop)
Precond: bel(A1,bel(A2,Prop)),
         not(bel(A1,not(Prop))),
         bel(A1,trustworthy(A2)).
Effects: bel(A1,Prop).
```

During planning simulation, plans are constructed based on the beliefs, goals and intentions at that level of nesting (corresponding to the simulating agent). If a proposition is not present in that level of nesting, then the planner must plan ascription actions to determine whether the simulated agent holds the relevant attitudes.

The planning algorithm can be extended to allow other agents plan to be recognized by inferring from agents performed actions:

- The agents set of goals he/she is trying to achieve
- The plan the agent intends to follow to achieve these goals

This can be done by collecting together the ascribable goals at the given level of nesting and attempting to find a plan which achieves at least one of the ascribable goals. Once a plan is generated, any goals achieved by the plan are ascribed.

2.3 Feature-based Topic Model

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, consider the topic `WeatherBroadcast` :

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

topic DetailedWeatherBroadcast inherits from GeneralWeatherBroadcast
```

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

The topic `DetailedWeatherBroadcast` inherits the features date and weather, redefines the feature temperature and adds information about wind and velocity. An agent could consider a specific instance of topic `DetailedWeatherBroadcast`:

```
topic DetailedWeatherBroadcast
  date: in 2 days
  weather: sunny
  temperature : >25°C
  wind: W
  velocity : 5n
```

2.3.1 Topics and Stereotypes

The notion of stereotypes discussed in chapter 7 for ViewGen is also present in ViewFinder. This is a powerful concept for classifying agents and attribute them some default behavior, knowledge or reasoning capability. We have decided to represent this notion 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 according to the agent class, modeling the notion of agent competency. The more features an agent can evaluate in a topic, the more competent she is in this topic. A same feature can have different possible values according to the agent class, modeling the notion of agent specialization. For example, consider the topic `Computer` and two classes of agents `AverageUser` and `GoodUser` :

```
topic Computer :
  class AverageUser
    hardDrive : [20GB, 30GB, 50GB]
    RAM : [128MB, 256MB]
    CPU : [1Ghz, 2Ghz]
  class GoodUser
    hardDrive : [20GB Maxtor, 30GB IBM, 50GB Hitachi]
    RAM : [128MB DRAM, 256MB SDRAM]
    CPU : [1Ghz Celeron, 2Ghz Pentium IV]
    BIOS : [Award, Phoenix]
    LAN : [Windows, Novell]
```

The features are more detailed for class `GoodUser` than for class `AverageUser`, and it has two more features `BIOS` and `LAN`. In this model, agents of class `AverageUser` are incompetent in evaluating the `BIOS` and the `LAN` type of a computer.

2.3.2 Topic definition

Let's define more formally what is a topic and give the Prolog syntax for designing topics in the system.

Consider a topic T with features $\mathcal{F}_T(A) = \{f_1, f_2, \ldots, f_n\}$ for agent $A \in \mathcal{A}$. Let $D_1^T, D_2^T, \ldots, D_n^T$ be the value domains of the features f1, $f_1, f_2, \ldots, f_n$ for the class $\mathcal{A}$. Each set D_i^T is finite discrete set. We have :

topic T
        class $\mathcal{A}$
                feature $f_1 \in D_1^T$
                feature $f_2 \in D_2^T$
                ...
                feature $f_n \in D_n^T$

A topic T can always be mapped onto a vector $t \in D_1^T \times D_2^T \times \ldots \times D_n^T$. The space $D_1^T \times D_2^T \times \ldots \times D_n^T$ of all possible instances of a topic T is called the *topic space*.

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 tuple[3]. Conversely, we say that an agent has no opinion about a topic, if she 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. The following are examples of the Prolog terms corresponding to topic definitions

```
topicDef(computer, averageUser, objectDef(hardDrive, [20GB, 30GB, 50GB])).
topicDef(computer, averageUser, objectDef(ram, [128MB, 256MB])).
topicDef(computer, averageUser, objectDef(cpu, [1Ghz, 2Ghz])).

topicDef(computer, goodUser, objectDef(hardDrive, [20GB Maxtor, 30GB IBM, 50GB Hitachi])).
topicDef(computer, goodUser, objectDef(ram, [128MB DRAM, 256MB SDRAM])).
topicDef(computer, goodUser, objectDef(cpu, [1Ghz Celeron, 2Ghz Pentium IV])).
topicDef(computer, goodUser, objectDef(bios, [Award, Phoenix])).
topicDef(computer, goodUser, objectDef(lan, [Windows, Novell])).
```

2.3.3 Feature Constraints

We have decided to model knowledge about a topic as constraints carving up the topic space. An agent builds its opinion about a topic by an elimination process. It interprets the constraints and eliminates all the incompatible vectors from the topic space. Complete knowledge is achieved when the constraints have reduced the topic space to only one possible combination of features. This method allows us to deal with open theories and disjunctive information. From a logic point of view topic features are viewed as abducible predicates (or, if we take features as complex terms, as situation theoretical issues) with integrity constraints.

We have two types of constraints: *positive* constraints and *negative* constraints. A positive constraint imposes a subset of possible values for each feature. For example, let's consider a topic T with three features $\mathcal{F}_T = \{f_1, f_2, f_3\}$, which can take values in the set $[1, 2, 3]$. A positive constraint imposes for instance that the feature f_1 must have values only in the subset $[2, 3]$, and feature f_2 in the set $[1, 2]$. This type of constraint is very useful to reduce the topic space before starting to browse into it and is semantically viewed as a commitment. For instance, one agent can commit herself to the fact that feature f_1 must have values in the set $[1, 2]$. Therefore she commits to the alternative $f_1 = 2 \vee f_1 = 3$ and blocks any further space reduction with respect to that feature. The other type of constraint is negative constraints. A negative constraint is like an integrity constraint in databases which excludes some tuples. Here, a negative constraint cuts the topic space by eliminating some combinations of feature values. Let's define formally the constraint language.

Let $\mathcal{F}_T$ be the set of features in topic T and D_i^T the value domain of the feature f_i in topic T for a given agent A.

Positive constraints are of the form: $f_i \in C_i^T$ where $C_i^T \subseteq D_i^T$.

[3] the term 'complete' has been chosen in analogy to the complete meta-predicate of the Kim and Kowalski's work [42].

Negative constraints can be of the two forms:

$$\bigwedge_{i=1}^{k}(f_i \in C_i^T) \Rightarrow f_0 \in C_0^T$$

$$\bigwedge_{i=1}^{k}(f_i \in C_i^T) \Rightarrow false$$

with $C_i^T \subseteq D_i^T$ and $f_i \neq f_j$ for each $i, j = 0, \ldots, k$ and $i \neq j$.

We will write $f = v$ instead of $f \in \{v\}$.

2.3.3.1 Examples

Consider the features f_1, f_2, f_3 and feature value domains $D_1 = D_2 = D_3 = \{a, b, c, d\}$, we can have the following positive constraint:

$$f_1 \in \{a, b, c\}$$

or the following negative constraints:

$$f_1 \in \{a, b\} \wedge f_2 = c \Rightarrow false \tag{2.1}$$

$$f_1 = a \wedge f_2 \in \{b, c\} \Rightarrow f_3 \in \{a, b\} \tag{2.2}$$

The constraint 2.1 can be interpreted as:

$$(f_1 = a \vee f_1 = b) \wedge (f_2 = c) \Rightarrow false$$

and the constraint 2.2 can be interpreted as

$$(f_1 = a \wedge (f_2 = b \vee f_2 = c) \Rightarrow (f_3 = a \vee f_3 = b).$$

This language is used in the context of nested mental attitudes and therefore it requires more information to control the interaction between viewpoints. In particular we want to represent that we have some information which we know an other agent may not have, and therefore it is not correct to ascribe it to him by default. This information is captured by *private constraints*.

Let be $\mathcal{A}$ the set of all agents defined in the system, For all $A \in \mathcal{A}$ we define $\mathcal{P}_T(A)$ as the set of private constraints for the agent A with respect to the topic T. We will see later that the constraints in this set cannot be ascribed by default from environments held by agent A to other agents.

Here some examples of constraints in PROLOG syntax:

positive constraint :

```
feature(f1, [a, b, c]).
```

negative constraints :

```
imply([feature(f1, [a, b]), feature(f2, [c])], false).
imply([feature(f1, [a]), feature(f2, [b, c])], feature(f3, [a, b])).
```

private constraint :

```
private(a1, [imply([feature(f1, [a, b]), feature(f2, [c])], false)]).
```

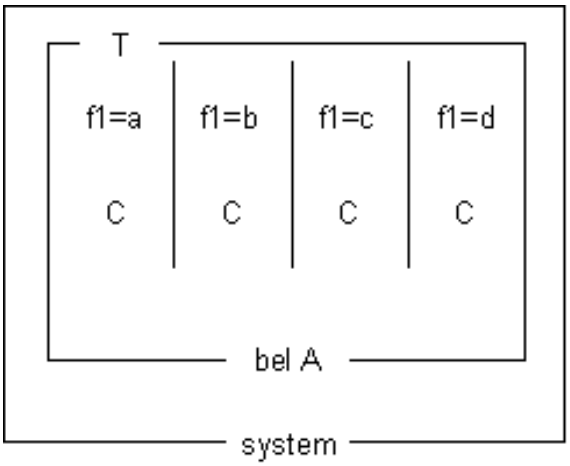

Figure 2.7: Opinions of agent A on the topic T

2.3.4 Representing opinions

So far we have represented information as topic spaces and we have called an opinion a given subspace of the whole topic space. This language may be sufficient when representing one agent's knowledge about some aspects of the world, but is not enough expressive to capture all 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 reduce each time we acquire further information, but it forces us to know everything about another agent's opinions.

Consider a topic T with features f_1, f_2, f_3, a common definition set $D = \{a, b, c\}$ and a set of constraints C. We want to represent that an agent $A \in \mathcal{A}$ knows (or more generally has an attitude towards) the value of feature f_1, but we don't know exactly what it is. We say that feature f_1 is *undefined* and we write:

$$f_1 = \lambda(A)$$

where the parameter A is the only agent capable of providing a value for the feature f_1.

What we know is that agent A has assigned a unique value to the feature f_1 in D, which can be alternatively $f_1 = a$ or $f_1 = b$ or $f_1 = c$ or $f_1 = d$. Therefore, we consider four different variants of topic T with respect to the corresponding attitude held by A, one for each possible value of feature f_1 in its value domain D.

These four models showed in figure 2.7 represent four different opinions of agent A on topic T. We know that agent A has one of these four opinion, but we don't know which one. We need to get more information on agent A to reduce the number of opinions by eliminating those which are inconsistent with integrity constraints. The final goal is to determine a unique value for the feature f_1 on topic T, which is the A's actual opinion on topic T.

2.3.5 Undefined features

Let $\mathcal{U}_T(A)$ the set of undefined features held by agent A, defined as

$$\mathcal{U}_T(A) = \{f_i \in \mathcal{F}_T(A) \mid f_i = \lambda(A), f_i \in D_{f_i}^{T(A)}\}.$$

The combination of all possible values of features $\mathcal{U}_T(A)$ defines the set of opinions held by the agent A. Each opinion is interpreted as a *model* (i.e. the set of possible tuples) and identified by

a *model signature*. Let Ms be the set of model signatures for an agent A:

$$\begin{aligned}\mathcal{M}_T(A) \quad = \quad \{f_1 = v_1 \wedge \ldots \wedge f_n = v_n \mid \\ f_i \in \mathcal{U}_T(A), \\ v_i \in D_{f_i}^{T(A)}, \\ i \in \{1, \ldots, n\}, \\ n = \mid \mathcal{U}_T(A) \mid\}.\end{aligned}$$

There are as many models as possible combinations of values for the undefined features. For example consider topic T with features f_1, f_2, f_3, common definition set $D_{f_i}^T = \{a, b, c, d\}$ for all $i \in \{1, 2, 3\}$, and undefined features $f_1 = \lambda(A), f_2 = \lambda(A)$. We have 16 models:

$$\begin{gathered}(f_1 = a \wedge f_2 = a), (f_1 = a \wedge f_2 = b), (f_1 = a \wedge f_2 = c), (f_1 = a \wedge f_2 = d) \\ \vdots \\ (f_1 = d \wedge f_2 = a), (f_1 = d \wedge f_2 = b), (f_1 = d \wedge f_2 = c), (f_1 = d \wedge f_2 = d)\end{gathered}$$

In Prolog syntax, undefined features are written as :

`LambdaValue:= feature(FeatureName, lambda(AgentName))`

`FeatureName:=` Prolog term; the feature name whose value is undefined

`AgentName:=` Prolog term; the name of the agent capable of evaluating this feature

So we would write:

```
feature(f1, lambda(A)), feature(f2, lambda(A)).
```

2.3.6 Opinion consistency

In mental state management it is important to be able to check for consistency and take appropriate actions if a conflict is detected. A topic is *consistent* if all opinions are consistent. An opinion is consistent if the associated topic space is not empty (i.e. at least one combination of features is valid). When inconsistent opinions are detected they are removed and new constraints are added to the knowledge base. Suppose an agent A with several opinions defined by $f_1 = \lambda(A)$ and $f_2 = \lambda(A)$. At one time the models identified by $(f_1 = a \wedge f_2 = b)$ and $(f_1 = a \wedge f_2 = c)$ become inconsistent. We can add the constraints:

$$\begin{aligned}f_1 = a \wedge f_2 = b \quad &\Rightarrow \quad false, \\ f_1 = a \wedge f_2 = c \quad &\Rightarrow \quad false.\end{aligned}$$

If only one opinion is consistent then it is the *actual* opinion of the agent and we can validate it. Suppose the only consistent model is $(f_1 = d \wedge f_2 = e)$, we can add the constraints:

$$\begin{aligned}f_1 \quad &= \quad d, \\ f_2 \quad &= \quad e\end{aligned}$$

and removes all other opinions.

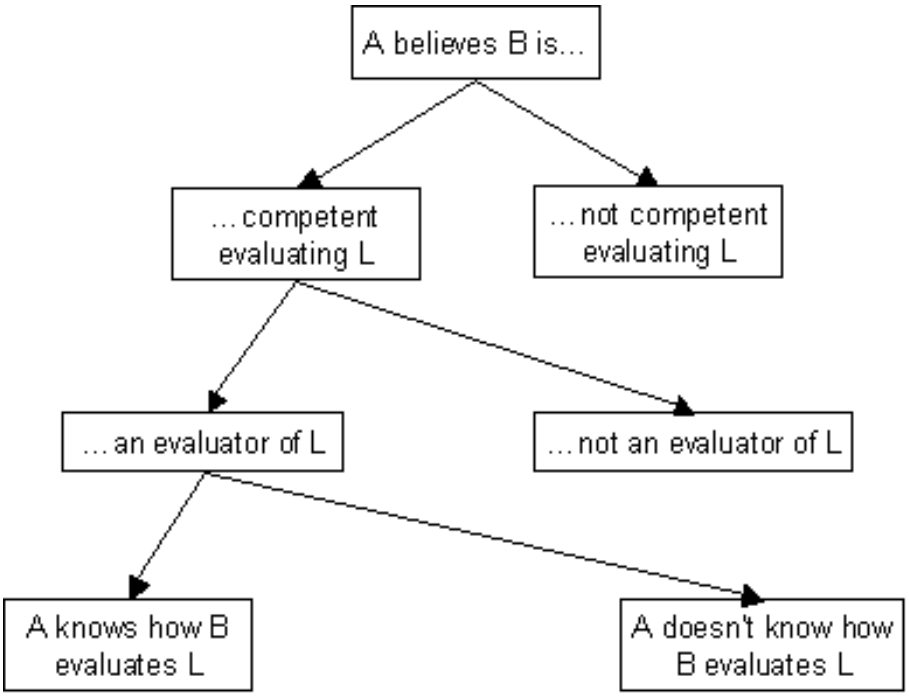

Figure 2.8: A taxonomy of Competency in Believing

2.4 Competency in believing

Ballim and Wilks in [11] have proposed a *taxonomy of competency in believing* showed in figure 2.8. They consider differently an agent who may be competent in the evaluation of a feature and an agent who actually has an evaluator for that feature. An agent can have competency to evaluate a feature, and then if he is competent, he can possibly have an evaluator for it.

In our representation we model the fact that an agent is competent to evaluate a feature if the topic definition for this class of agent actually has this feature available. Let's reconsider the example about the topic `Computer` and the two classes of agents `AverageUser` and `GoodUser` :

```
topic Computer :
  class AverageUser
    hardDrive : [20GB, 30GB, 50GB]
    RAM : [128MB, 256MB]
    CPU : [1Ghz, 2Ghz]
  class GoodUser
    hardDrive : [20GB Maxtor, 30GB IBM, 50GB Hitachi]
    RAM : [128MB DRAM, 256MB SDRAM]
    CPU : [1Ghz Celeron, 2Ghz Pentium IV]
    BIOS : [Award, Phoenix]
    LAN : [Windows, Novell]
```

Consider an agent A from any class and an agent B of class `AverageUser`. We can state that :

Agent A believes agent B is competent evaluating features `hardDrive`, `RAM` and `CPU`

Agent A believes agent B is incompetent evaluating features `BIOS` and `LAN`

	Topic	Opinion
Interpretations	*Tuples*	*Models*
Representations	*Constraints*	*Undefined features*

Table 2.1: Topic & Opinion

An agent competent to evaluate a feature, effectively has an *evaluator* if he has some constraints on this feature (i.e. common constraints or lambda value). An agent *knows* how to evaluate a feature if this feature is complete (constraints cuts all the values except one). If an agent does not know how an other agent evaluates a feature, he might consider different models for that feature (i.e. undefined feature). In conclusion we see that our knowledge representation is complete and catches all the cases of the taxonomy described in figure 2.8.

2.5 Topic and Opinion viewpoints

In our model (see table 2.1) we have two types of viewpoints about a topic of discourse: the *topic's viewpoint* and the *opinion's viewpoint*.

The *topic viewpoint* is the current attitudes we have about a topic. It is represented in the system by constraints on the topic space and interpreted as a set of tuples (i.e. combination of features) we consider valid. This way of representing and interpreting attitudes towards a topic allows us to handle *disjunction* in a natural way. Disjunction is implicitly contained in the set of tuples, so we only need to focus our efforts on eliminating the combinations we don't want. We don't need working hard to build all the possibilities we consider interesting.

The *opinion viewpoint* is the current attitude we have about topic's viewpoints of other agents. It is represented in the system by undefined features and it is interpreted as a set of models (i.e. a topic subspaces) we consider possible. This way of representing and interpreting several opinions allows us to handle *uncertainty*. If we are not sure of the opinion held by a specific agent about a topic, we consider all possible models and then eliminate those which are inconsistent.

These two viewpoints about a topic have two knowledge levels: the *representation level* and the *interpretation level*. The interpretation level transforms the attitude statements stored in the system into a mental image which enables *reasoning*, *consistency checking* and *communication*.

The representation level is concerned with the way knowledge statements are stored in the system. We have seen that attitudes about topics are stored as integrity constraints and attitudes about opinion are stored as undefined features. At this level we face the problem of granularity and how it influences the interpretation level. Suppose for example a topic T with features $\mathcal{F}_T = \{f_1, f_2, f_3\}$, common definition set $D^T_{f_i} = \{a, b, c\}$, and initial constraint:

$$f_1 = a \wedge f_2 = a \wedge f_3 = a \Rightarrow false$$

Then over time we assimilate the constraints:

$$\begin{aligned} f_1 = b \wedge f_2 = a \wedge f_3 = a &\quad \Rightarrow \quad false, \\ f_1 = c \wedge f_2 = a \wedge f_3 = a &\quad \Rightarrow \quad false. \end{aligned}$$

Now we have three constraints in our knowledge base. If we look at these constraints, we see that while feature f_1 takes all the values of definition set $D^T_{f_1}$, features f_1 and f_3 always take the same values $f_2 = a, f_3 = a$, and finally the conclusion is three times identical. Therefore, it would be correct to simplify these three constraints to one constraint more general involving only f_2 and f_3:

$$f_2 = a \wedge f_3 = a \Rightarrow false.$$

But now we have only one rule left and we have lost one level of granularity. What do we do if later we become aware that the constraint:

$$f_1 = c \wedge f_2 = a \wedge f_3 = a \Rightarrow false$$

is no more valid? We must remove the general rule and restore more specialized ones.

A natural solution to this problem is to generalize when we feel sufficient confidence for it, and specialize it again when some exceptions are found. In practice, this solution is hard to implement and requires a non negligible computational effort. So in our solution, we decided to never generalize. We store the constraints in their raw format, maintaining the maximum of granularity. This solution, however, suffers of a fast growing and redundant knowledge base.

2.6 Environments

The goal of this section is to show how our proposed model is a possible implementation of the theoretical setting proposed in ViewFinder. We start by giving an overview of some theoretical aspects of ViewFinder and outline the main ideas of our implementation. Then we present how our solution implements those ideas and bring some specific remarks.

2.6.1 The ViewFinder framework

ViewFinder is a framework for representing, projecting, and maintaining nested mental attitudes of interacting agents. A. Ballim made an intensive work on nested beliefs, but he also presented ideas to manage different *types* of nested mental attitudes. The general framework is based on nested environments and presents the basic projection operations, *ascription* and *adoption*, to control the flow of information between them.

2.6.1.1 Nested environments

An environment can be thought of as a context. It is an information container with an *owner* and a *type*. The owner is the agent who holds the information. The type is the mental attitude towards this information and it is denoted by label assigned to the environment. For instance, we could have environments whose owner is agent A and types are *Bel*, *Goals*, *Int*. These environments represent respectively the *beliefs*, *goals* and the *intentions* of the agent A.

An environment can contain other environments or some atomic content. The nesting of environments allows us to represent *viewpoints* or *nested attitudes*. For example, we can represent what believes agent A_1 believes agent A_2 about the intentions of agent A_3. This statement can be written into a term based language, for instance:

```
bel(A1, bel(A2, int(A3, Content))).
```

We will use this representation for logical reasoning and computation, but we can use the graphical representation we introduced in chapter 7 of nested attitudes based on boxes sketched in figure 2.9.

There is no limitation on the nesting level and these two type of representations capture entirely the intuitive idea behind viewpoints. At the end of the nesting chain, we find the atomic (or propositional content). This is the actual information stored in the system. It can be represented in any formal language required by a specific application. This language is the *internal mental language*, by opposition to the *external language* used for communication.

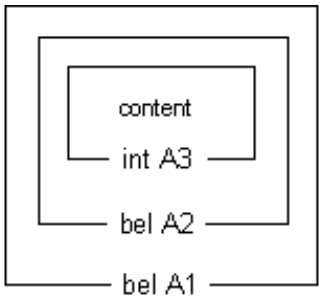

Figure 2.9: Nested environments

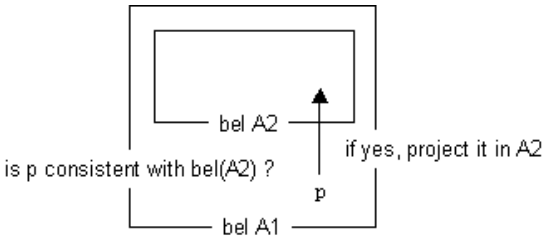

Figure 2.10: Default ascription

Storing information in a system is fundamental, but not as important as it is being able to access and manipulate it. In the two next sections we will show how the information flow between environments can be controlled. The information flow is based on two operations: *ascription* and *adoption*. The ascription operation moves information from an outer environment to an inner (nested) environment. And the adoption operation is the dual operation, it moves information from an inner environment to an outer environment.

2.6.2 Ascription

Nesting environments can be potentially infinite. For instance, we can always ask ourselves whether the agent A_1 believes agent A_n believes ... believes agent A_n. It is computationally too expensive statically generating all the viewpoints needed in the system. The natural idea is to generate nested mental attitudes *on demand*. In the case of beliefs we already discussed in chapter 7 a default rule which allows us to generate dynamically a viewpoint: *the default ascription* rule.

When we want to generate what believes agent A_1 about the beliefs of agent A_2, the default ascription algorithm projects the beliefs of A_1 to agent A_2 unless it has evidence that A_2 *cannot believe* the same things. Figure 2.10 shows the prototypical situation.

The whole theory of ViewFinder is based on the notion of *default reasoning*. An agent can assume that another agent reasons and has the same opinions as himself except if he has contrary evidence. All the complexity of the ascription algorithm relies on the computation of the set of information which can be projected into an inner environment. This algorithm is based on *consistency checking* and computes the biggest set of information consistent with the inner environment.

Ballim presented two main ascription strategies: the *conservative permissive ascription* and the *conservative restrictive ascription*.

Suppose defined the basic set relations on environments: union $\overset{\mathcal{E}}{\cup}$, intersection $\overset{\mathcal{E}}{\cap}$, strict inclusion $\overset{\mathcal{E}}{\subset}$ and non strict inclusion $\overset{\mathcal{E}}{\subseteq}$, and the consistency predicate $\top$ over an environment $\mathcal{E}$ defined as

$$\top(\mathcal{E}) \text{ if and only if } \mathcal{E} \text{ is consistent and}$$

$\top(\mathcal{E}_1 \overset{\mathcal{E}}{\cup} \mathcal{E}_2)$ if and only if the union of environment $\mathcal{E}_1 \overset{\mathcal{E}}{\cup} \mathcal{E}_2$ is consistent according to compatibility constraints between environment types (e.g. ascription of belief towards goals) and the consistency of the union of their propositional content. The problem of general attitude ascription has been discussed by Ballim in [7]. He proposes the notion of a *General Attitude Ascription* in order to accommodate the notion of ascription to mental attitudes other than beliefs, that is determining to what extent might we assume that other agents have the same desire, goals, abilities, intentions, hopes, etc. as ourselves. There are situation in which different types of attitudes can be considered compatible for default ascription, as for instance for those agent belonging to the same class. In this case it would be possible to ascribe the same goals, intentions, etc. Another problem arises when we would like to ascribe a whole nested attitude to another agent as for instance when we would like to ascribe $Bel(A, Goal(B, p))$ to the agent C obtaining $Bel(A, Bel(C, goal(B, p)))$. In such a case, we need to consider whole nested environments as propositional content and not ascribe only the propositional part of it (i.e. P). This aspect will be better discussed later in section 2.7.4.

2.6.2.1 Permissive ascription

Let $\mathcal{E}_0$ and $\mathcal{E}_1$ be two environments. Let $\mathcal{E}_2$ be the *conservative permissive projection* of $\mathcal{E}_0$ on $\mathcal{E}_1$. The permissive candidate set of $\mathcal{E}_0$ with respect to $\mathcal{E}_1$ is defined to be:

$$PC = \left\{ \mathcal{E} \;\middle|\; \begin{array}{c} \mathcal{E} \overset{\mathcal{E}}{\subseteq} \mathcal{E}_0 \\ \top(\mathcal{E} \overset{\mathcal{E}}{\cup} \mathcal{E}_1) \\ \neg\exists \mathcal{E}_i \;\middle|\; \begin{array}{c} \mathcal{E} \overset{\mathcal{E}}{\subset} \mathcal{E}_i \wedge \\ \top(\mathcal{E}_i \overset{\mathcal{E}}{\cup} \mathcal{E}_1) \end{array} \end{array} \right\}$$

The result of the projection $\mathcal{E}_2$ is defined as $\mathcal{E}_2 = (\overset{\mathcal{E}}{\cap} PC) \overset{\mathcal{E}}{\cup} \mathcal{E}_1$.

2.6.2.2 Restrictive ascription

The *restrictive ascription* acts like the permissive ascription, except that all derived proposition in the origin environment which are ascribed in the target environment, have to be ascribed with the original hypothesis. The restrictive ascription does not allow to ascribe a conclusion without ascribing the hypothesis which led to this conclusion.

We use the notation proposed by Ballim for denoting a proposition P derived in environment $\mathcal{E}$:

$$\overset{\mathcal{E}}{\circ} P \Leftrightarrow (\mathcal{E} \models P \wedge P \notin \mathcal{E})$$

.

Let $\mathcal{E}_0$ and $\mathcal{E}_1$ be two environments. Let $\mathcal{E}_2$ be the *conservative restrictive projection* of $\mathcal{E}_0$ on $\mathcal{E}_1$.

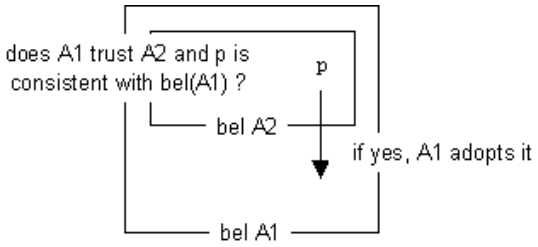

Figure 2.11: Default adoption

The restrictive candidate set of $\mathcal{E}_0$ with respect to $\mathcal{E}_1$ is defined to be:

$$RC = \left\{ \mathcal{E} \;\middle|\; \begin{array}{c} \mathcal{E} \overset{\mathcal{E}}{\subseteq} \mathcal{E}_0 \\ \top(\mathcal{E} \overset{\mathcal{E}}{\cup} \mathcal{E}_1) \\ if \overset{\mathcal{E}_0}{\circ} P \in \mathcal{E} \, then \, \mathcal{E} \models P \wedge \\ \neg\exists \mathcal{E}_i \;\middle|\; \begin{array}{c} \mathcal{E} \overset{\mathcal{E}}{\subset} \mathcal{E}_i \wedge \\ \mathcal{E}_i \, fulfills \, the \, other \, conditions \, for \, \mathcal{E} \end{array} \end{array} \right\}$$

The result of the projection $\mathcal{E}_2$ is defined as $\mathcal{E}_2 = (\overset{\mathcal{E}}{\cap} RC) \overset{\mathcal{E}}{\cup} \mathcal{E}_1$.

We can say that the ascription operation computes the set of information an agent can project into the mind of another agent. This operation can play the role of *common*, *shared*, or *mutual knowledge* as found in other theories of mental attitudes [23, 34, 42]. It assumes that each agent has similar reasoning capabilities, and because they live in the same world it is not too silly to think that the information held by one agent can be found in other agents. The ascription algorithm assumes a common way of reasoning but not a predefined common attitude about a topic. Each agent considers that when it holds any information, it can assume that another agent also holds it, except if it has explicit knowledge of the contrary.

2.6.3 Adoption

The second operation on environments is adoption, referred in [11] as *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. For example, agent A_1 is informed that agent A_2 believes the proposition P. We can then assert in the system's mental state that agent A_1 believes that agent A_2 believes P. If the agent A_1 trusts agent A_2, it will also believes P. In this case it *adopts* the belief of P from agent A_2. This situation is sketched in figure 2.11.

The adoption algorithm allows an agent to *extend* his knowledge base with information he gathers from other agents through communication and collaboration, and not only through his own observations of the real world. We use the same conservative policy as for ascription: *an agent adopts new information of another agent only if he trusts it and the new information is consistent with his own opinion.* The adoption process is triggered by the incoming of new information. Each time a message is received, there is potentially new information available. This new information is then propagated in the system using the adoption algorithm.

2.7 Feature-based Ascription and Adoption

Let's show in this section how our particular solution is a possible implementation of the theoretical framework ViewFinder. We show how we can represent nested environments and how we implement the ascription and adoption operations within the representation of topics with features and integrity constraints.

2.7.1 Representation of Nested environments

We have seen that a nested chain of environments can always be represented by an expression written in a term based language. So in Prolog we can always write :

```
Environment:= bel(AgentName, Environment) | bel(AgentName, Content) |
              des(AgentName, Environment) | des(AgentName, Content) |
              int(AgentName, Environment) | int(AgentName, Content)
```

`Content:=` Atomic content of the environment.

We can define a special predicate for each possible mental attitudes in an agent (`bel` for beliefs, `des` for desires, `int` for intentions,...). We have adopted this solution in the interactive part of the system because it is intuitive to manipulate expressions like `bel(a1, bel(a2, content))`, but it is not well suited for reasoning and computation. So we decided to represent environments by a predicate called env defined as :

`Environment:= env(EnvironmentSignature, Content)`

`EnvironmentSignature:= [MentalAttitude, ...]`

`MentalAttitude:=[EnvironmentType, AgentName]`

`EnvironmentType:= bel | des | int`

`AgentName :=` the owner of the mental attitude

`Content:=` Atomic content of the environment

For instance, the nested environment expression

```
bel(a1, bel(a2, int(a3, content)))
```

is transformed in

```
env([[int, a3], [bel, a2], [bel, a1]], content).
```

Notice that the most nested environment appears first. This implementation detail which makes it much easier to do computation in the most inner environment and then go recursively in outer ones.

We still have to represent the atomic (propositional) content of an environment. The atomic content can be any representation decided for a specific application, but in our solution the atomic content is based on topics and features as described in section 2.3. The content of an environment is defined by the language:

`Content:= topic(TopicName, Elements)`

`Elements:= [Constraints, LambdaValues, PrivateConstraints]`

`Constraints:= constraints([Constraint, ...])`

`LambdaValues:= lambdaValues([LambdaValue, ...])`

`PrivateConstraints:= privateConstraints([PrivateConstraint, ...])`

`TopicName:=` Prolog term; the name of the topic contained in the environment

`Constraint:=` A constraint as defined in section 2.3

`LambdaValue:=` An undefined feature as defined in section 2.3

`PrivateConstraint:=` A private constraint as defined in section 2.3.

It is easy to see that our implementation solution allows us to work with any types of environments and nested mental attitudes without any limitations in depth.

2.7.2 Ascription

A topic is represented in an environment as a set of constraints. The ascription algorithm computes the *maximal subset of constraints consistent with the destination environment.* A subset of constraints is consistent with the destination environment if added to the environment, the topic space does not become empty (i.e at least one combination of features is valid). In all the possible consistent subset of constraints, the algorithm selects the biggest. In our implementation the algorithm uses a depth-first search to find the biggest consistent set in the space of all possible subsets. If no consistent set exists, then no constraints are ascribed. Semantically, it means that the opinion of the agent owning the environment is incompatible with the point of view generated. The two agents have a totally different opinion about the same topic. In the case where a consistent subset is found, the generated opinion is a merge between the default opinion of the agent owning the environment and the known particularities of the second agent. We distinguish two cases: ordinary ascription and disjunctive ascription.

In case of *only one opinion* in the target environment (i.e. non disjunctions) we can formalize our ascription algorithm as follows. Suppose a source environment $\mathcal{E}_0$ with propositional content represented as a set of constraints C_0 and a target environment $\mathcal{E}_1$ with a set of constraints C_1. We want to compute the set of constraints C_2 from the destination environment $\mathcal{E}_2$ following the conservative permissive strategy.

We define C_0^{pub} the set of *public initial constraints*[4] for an agent A as:

$$C_0^{pub} = C_0 \setminus \mathcal{P}_T^A(\mathcal{E}_0, \mathcal{E}_1)$$

where $\mathcal{P}_T^A(\mathcal{E}_0, \mathcal{E}_1)$ denotes the set of A's private constraints restricted to environments $\mathcal{E}_0, \mathcal{E}_1$.

The permissive candidate set of constraints is defined as:

$$C_{max} = \left\{ C \;\middle|\; \begin{array}{c} C \subseteq C_0^{pub} \\ \top_{\mathcal{E}_1}(C \cup C_1) \\ \neg\exists C_i \;\middle|\; \begin{array}{c} C \subset C_i \wedge \\ \top_{\mathcal{E}_1}(C_i \cup C_1) \end{array} \end{array} \right\}$$

with $\top_{\mathcal{E}}(C \cup C')$ if and only if $C \cup C'$ is a consistent with respect to the integrity constraints present in the environment $\mathcal{E}$. The result of the projection C_2 is defined as:

$$C_2 = (\bigcap C_{max}) \cup C_1.$$

We see that the sets of constraints which can be ascribed to the target environment are defined exactly in the same manner as the sets of possible environments defined by Ballim in the case of

[4] Remember that an agent can also have private constraints.

the conservative permissive strategy. In the case of a single opinion in the target environment, our implementation is equivalent to the original ViewFinder's ascription algorithm.

Let's consider now the case of a *disjunction of models* in the target environment. In this case the ascription algorithm projects the *common biggest consistent set of constraints to all the possible models* of a topic T:

$$C_2 = \bigcap_{m_T \in \mathcal{M}_T(A)} \{\cap C_{max}(m_T)\} \cup C_1$$

where

$$C_{max}(m_T) = \left\{ C \;\middle|\; \begin{array}{c} C \subseteq C_0^{pub} \\ \top_{\mathcal{E}_1}(C \cup C_1 \cup m_T) \\ \neg\exists C_i \;\middle|\; \begin{array}{c} C \subset C_i \wedge \\ \top_{\mathcal{E}_1}(C_i \cup C_1 \cup m_T) \end{array} \end{array} \right\}$$

The index m_T ranges over the possible model signatures for the topic T. In this case the algorithm is even more conservative because the final set to be ascribed is common to all possible opinions defined in the target environment.

In summary, we can say that our implementation of the ascription algorithm corresponds to the definition given in ViewFinder of the conservative permissive ascription strategy. In fact, our algorithm follows also a restrictive strategy, because the derived constraints are always stronger than the initial ones. For instance, suppose that from a constraint

$$C_1 : f_1 \in D_1 \wedge \ldots \wedge f_n \in D_n \Rightarrow f_{n+1} \in D_{n+1}$$

we can deduce later the constraint

$$C_2 : f_1 \in D_1 \wedge \ldots \wedge f_n \in D_n \Rightarrow false.$$

The constraint C_2 is more restrictive than the constraint C_1. So if we ascribe by default C_2, then $C_1$is automatically ascribed too.

2.7.3 Adoption

The adoption algorithm is triggered when new information becomes available in an environment, for instance after the processing of a speech act. The set of new information is *propagated* to other environments using adoption. We follow again a conservative policy in our implementation in the sense that an agent can adopt either the whole set of new information, or nothing. An agent can not adopt from another agent only a subset of the new information. This policy is based on the idea that the statements contained in the set of new information are not logically or semantically independent (with high probability). So it would be incorrect to adopt only a subset. In our case the content is based on topics, so the sets of information which can be adopted are sets of constraints.

2.7.4 Policies

Adoption and ascription are two operations based on default behavior, but for many applications we need a finer control of the flow of information. We decided to provide *local rules* to control more specifically their action, but we stay compatible with the default reasoning schema.

2.7.4.1 Ascription policies

We can block the ascription of specific information to a specific agent by the use of *private rules* defined in section 2.3.3. An agent can decide that a set of constraints is private for a specific agent and can not be ascribed to it. For example, agent A_1 believes that a constraint C_1 holds, but it explicitly knows that agent A_2 can not be aware of this information; it can state that constraint C_1 is private for agent A_2 and therefore the ascription algorithm does not project this constraint to A_2 (even if it is consistent).

The default ascription rule which projects information from any environment to any other environment regardless of their types, might not be semantically correct for a lot of normal cases.

Along the lines of the discussion in [7] we have decided that there is no *a-priori* default ascription rule. In the system definition we have to explicitly create rules which allows one agent to ascribe his mental attitudes to another agent. Consider an agent A which believes P, desires to achieve P or intends to do P, and which ascribes this information to the beliefs, desires and intentions of an agent B. We write in PROLOG syntax :

`ascribes(AgentName1, EnvType1, AgentName2, EnvType2)`

`AgentName1, AgentName2` := Prolog term; the agent names

`EnvType1, EnvType2` := Prolog term; the environment types available in the system

In conclusion we can say that we are able to *control the ascription process* at the environment type level, at the agent level and finally at the content level.

2.7.4.2 Adoption policies

The adoption process is concerned with the problem of *trust*. An agent adopts information from another agent only if it trusts it. We decided that by default an agent *does not trust* any other agents. To allow adoption, we have to add explicitly adoption rules in the system definition. An adoption rule allows an agent to adopt information from another agent for a given topic. In PROLOG we write:

`adopts(AgentName1, AgentName2, TopicName)`

`AgentName1, AgentName2` := Prolog terms; the agent names

`TopicName` := Prolog term; the topic name

The `TopicName` parameter allows the representation of competencies of specialists. For example, an agent A adopts information about topic `Pneumonia` from agent `Doctor`, but prefers to adopt information about the `Stock Exchange` from agent `Trader`.

2.7.5 Interaction between Adoption and Ascription

The combined use of ascription and adoption brings up an interesting problem. In a simple case, imagine that agent A_1 adopts the proposition P from agent A_2 after receiving a message from A_2. Then agent A_1 tries to figure out what is the opinion of agent A_3. He ascribes to A_3 the proposition P (we assume now that P is consistent with A_3). Semantically, it means that agent A_1 assumes that agent A_3 is also aware of proposition P since it might happen that A_3 has also heard the message from A_2, which, except the case of a broadcast, it is rarely true.

One simple solution to this problem is to block ascription to all adopted information. In our example agent A_1 does not ascribe to A_3 the proposition P. In consequence the only way for A_1 to believe that A_3 believes P is to explicitly receive a message from A_3 or to receive a message from A_2 about the fact that A_2 has also informed A_3 about P. Blocking ascription to adopted

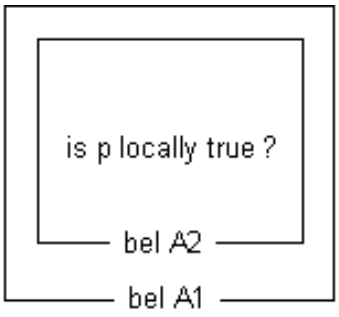

Figure 2.12: Answering a query

information means at the implementation level that any information stored in an environment has to keep track of its origin. In order to make things properly we should attach to every statement in the knowledge base an trace of where it comes from. In our proposal we make it simpler and we follow a permissive approach allowing the ascription of adopted information.

2.8 Querying the knowledge base

The access to information stored in the knowledge base is done through queries. A query will try to prove a goal in an environment. We describe in this chapter the general mechanism for answering queries and then explain more precisely their nature in the case of topic spaces.

A query is composed of two parts: the *environment access string* and the *goal* to prove. The environment access string is a nested chain of mental attitudes describing the environment containing the goal. For example, suppose we want to answer the query: "Does agent A_1 believe agent A_2 believes proposition P?", we submit to the system the string :

```
bel(A1, bel(A2, p)) ?
```

To answer the query the system executes two steps:

1. it builds the nested environment concerned by the query using the environment chain `bel(A1, bel(A2)).`
2. it tries to prove the goal P in the built environment.

To build the environment, the system recursively uses the ascription algorithm to project knowledge from the most outer environment to the most inner one. Once the context is set up, the system can try to prove the goal in this local context. This operation is done by a *meta-interpreter* capable of proving goals in the internal language of this environment.

It is important to notice that the meta-interpreter is not attached to the environment mechanism and the projection operations, but depends on the content stored in the environment. We can imagine environments with different types of content and a general query algorithm which selects the appropriate meta-interpreter according to the local internal language using polymorphism.

2.8.1 Query on topic spaces

In our solution based on topic spaces, a query tries to prove that the topic space provided as an argument matches the topic space stored in the environment (in all opinions). The environment access string is the same as in the general mechanism, but the goal has the form :

```
feature([f1,...,fn], [[v11, ..., v1n], [v1m, ..., vnm]]).
```

We submit a projection of the whole topic space on the features $f_1, \ldots, f_n$ as a set of m tuples:

$$\{[v_{11}, \ldots, v_{1n}], \ldots, [v_{m1}, \ldots, v_{mn}]\}.$$

For example, suppose we want to answer the query: "Does agent A_1 believe agent A_2 believes features f_1, f_2 can only be the pairs $(a,a),(a,b),(b,b)$?", we submit to the system the string :

```
bel(A1, bel(A2, feature([f1, f2], [[a, a], [a, b], [b, b]]))) ?
```

The system projects the whole topic space on the features f_1, f_2 and compares it to the set of tuples $[a,a],[a,b],[b,b]$. If it matches then the proof is correct.

Perhaps the system has several opinions about this topic and the associated models are not all identical. In that case the system can not prove that one specific model is correct, so he answers by a question mark :

```
feature([f1,...,fn], ?)
```

There is one interesting remark to make at this point. In this solution, the proof is *local to the context*, the meta-interpreter uses only contextual information and a specific reasoning algorithm. But we can imagine a more sophisticated system which tries to prove it locally and if it fails, gathers information from other environments using adoption process and then proves it. In natural reasoning, when we want to solve a problem, we first try to solve it with the information we have in our mind. If we don't succeed we try to get more information from other people in order to solve it. This is the same idea, but the biggest difficulty is how to get useful information from other agents. This is the problem of goal oriented search. In our system we use only local information to answer queries. But each time new information gets into the system through messages, a big amount of information is derived using a *forward chaining* process and then propagated to other agents by adoption. In this solution we generate a lot of information which will, perhaps, never be used.

2.9 Assimilating mental attitudes from communication

It is interesting to store information and manage a mental state, but if the mental state is not able to evolve over time it is not very useful. 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 [45] and Dragoni's works [31].

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 and
3. transfer the knowledge contained in the message into the mental state.

These three problems have different issues and constraints. We describe in this section the problems of communication language and assimilation process. The knowledge representation problem has already been treated in the previous sections.

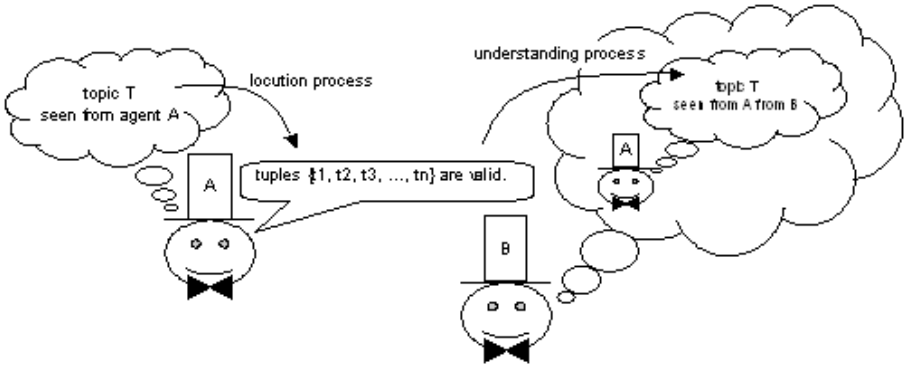

Figure 2.13: Inter-agent communication process

2.9.1 Communication content language

In agent communication there are two levels: *speech act* level and *content* level. The speech act level defines the type of messages exchanged by the agents based on speech act theory. The content level is concerned with the content of the message exchanged. Choosing a communication content language for agents is not an easy problem. The main couple of constraints are *expressiveness* and *complexity*. The more expressive is the language, the more complex it is to process it.

In our proposal 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, interpreted as a set of valid tuples. When two agent want to exchange information about a topic, they exchange their current interpretation of that topic. In a normal communication process the sender generates an interpretation from his internal representation of the topic. This interpretation is sent to the receiver who transforms it into his internal representation and finally derives new information. The communication process is sketched in figure 2.13.

The interpretation of a topic is always a set of tuples defined as the projection of the whole topic space on a subset of features. So we define the content language by the following Prolog syntax:

`Content:= topic(TopicName, feature(FeatureNames, Tuples))`

`FeatureNames:= [FeatureName, ...]`

`Tuples:= [Tuple, ...]`

`Tuple:= [Value, ...]`

`TopicName:=` Prolog term; the name of the topic

`FeatureName:=` Prolog term; the name of the selected features

`Value:=` Prolog term; the possible values of the selected features

For instance, suppose a topic T with features f_1, f_2, f_3 a common definition set $D = \{a, b, c, d\}$, and that the projection of the whole topic space on f_1, f_2 gives the following subset of valid couples:

$$\{(a, a), (a, b), (b, b)\}$$

we will communicate a message with content:

```
topic(T, feature([f1, f2], [[a, a], [a, b], [b, b]]))
```

Even if this content language allows us to exchange information on the interaction of several features, our current implementation can only treat messages with only one feature. This limitation is due to complexity in the understanding process. It becomes quite complex to transform a subset of tuples into constraints, but in the limited case of one feature it is straightforward. In conclusion we can say that actually the only messages which can be exchanged have content of the form:

`Content:= topic(TopicName, feature([FeatureName], Values))`

`Values:= [[Value], ...]`

`TopicName:=` Prolog term; the name of the topic

`FeatureName:=` Prolog term; the name of the selected feature

`Value:=` Prolog term; the possible values of the selected feature

For instance we can only communicate: `topic(T, feature([f1], [[a], [b]]))`

We will see later that this restriction is not as strong as it seems and can be relaxed in future versions of the system.

2.9.2 Knowledge assimilation

From the time a message arrives in the system to the time 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).

This process is showed in figure 2.14.

2.9.2.1 Message Delivery

The translation of the message content to internal representation is done for simple languages by an interpreter (e.g. formal languages by a compiler and natural language by a translator). This stage is concerned with linguistic problems and essentially understanding. The transformation of a linguistic statement into a mental statement is the base of understanding. In our demonstrator the communication language is simple enough to use an interpreter to transform it into internal representation. The integration of ViewFinder in a Dialogue Management system will require the extraction of dialogue acts from natural language utterances and their assimilation into our proposed mental state representation.

The set of tuples contained in the message is transformed into a set of constraints which can then be added to the actual mental state. Formally we use the rule :

```
topic(T, feature([f], [[v1], ..., [vn]]))
```

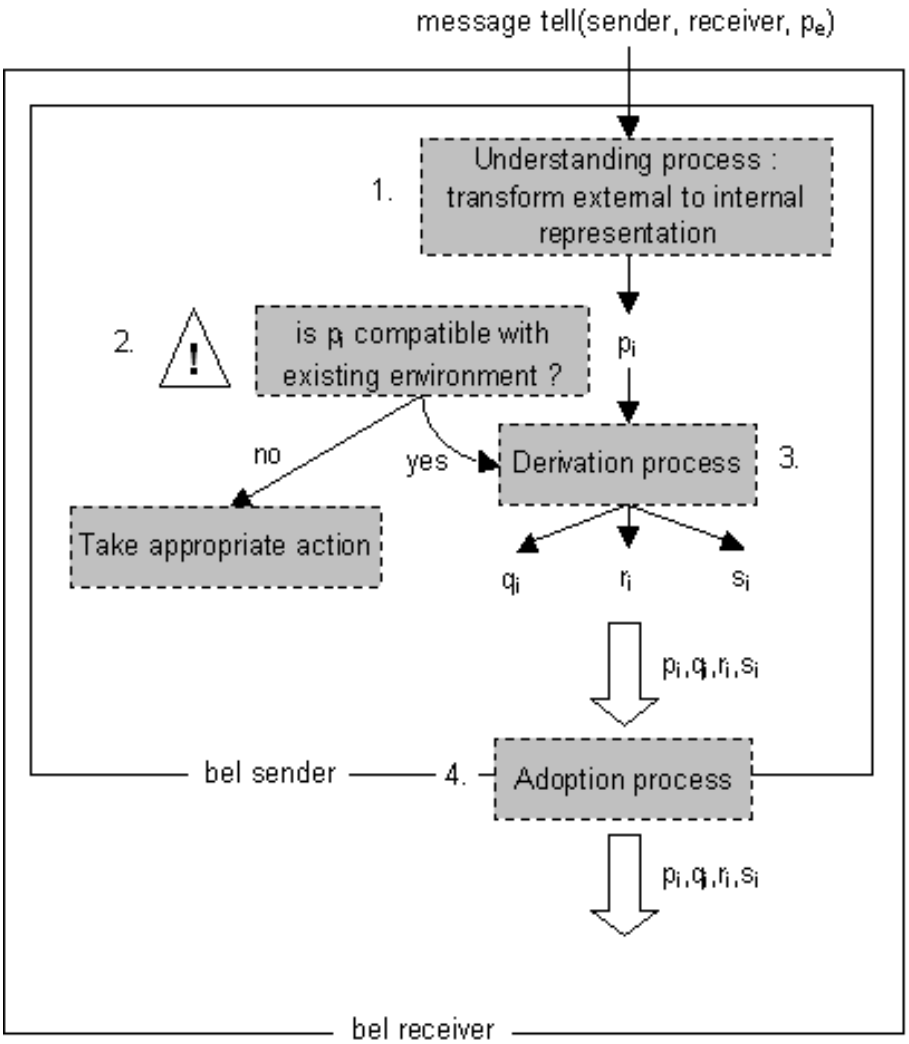

Figure 2.14: The Knowledge Assimilation Process

which add the constraint $f \in \{v_1, \ldots, v_n\}$ to topic T.

With this rule we are able to detect if an agent makes up her mind over time. Suppose we receive a message concerning feature f_1 and we add the constraint $f_1 \in D$. Later, we receive a second message from the same agent concerning again f_1, but this time we add the constraint $f_1 \in D'$ with $D \neq D'$. Since the two sets of available values for f_1 are different, we detect that the sender has made up her mind. We can then take the appropriate action (e.g. ask for more information, reject the message).

2.9.2.2 Consistency check

Once the receiver has understood the message, she can use it and evolve 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 contradiction she can take the appropriate action, for instance inform the sender of the contradiction, ask her for more information or reject the message. If the message is compatible, she assimilates it.

There are two main possibilities for the receiver:

1. she has already received this message and the whole information is already known;
2. it is a new message and she assimilates the new information.

If it is a new message, the receiver can ever maintain a mental representation of the sender which has exactly the same amount of information of the message; or she can maintain a mental representation which has less information (more general opinion). In the first case the message acts as a *grounding*. In the second case, the receiver has to complete her mental representation of the sender. In both the two cases, she commits with the information contained in the message in order to detect further changes of opinions from the sender. Then she tries to derive new information using her actual mental state and the information received.

2.9.2.3 Knowledge integration

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 of the sender and derives constraints that should be true in the sender's mental state. Constraints are only added, there is no simplification. A simplification step would reorganize constraints, but loose granularity.

Let's show now how we can derive new constraints from an existing content. Suppose a topic T with features $f_1, \ldots, f_N$ and associated definition sets $D_1^T, \ldots, D_N^T$. The negative constraints stored in an environment are of two forms:

1. $f_i \in D_i \wedge \ldots \wedge f_j \in D_j \Rightarrow false$
2. $f_i \in D_i \wedge \ldots \wedge f_j \in D_j \Rightarrow f_j \in D_k$

with $D_i \subseteq D_i^T, D_j \subseteq D_j^T, D_k \subseteq D_k^T$.

The effect of a constraint is either false or a positive constraint on a feature. We can always transform the first form in the second form for a specific feature f_k:

$$f_i \in D_i \wedge \ldots \wedge f_j \in D_k \wedge \ldots \wedge f_j \in D_j \Rightarrow false \quad \Leftrightarrow \tag{2.3}$$

$$f_i \in D_i \wedge \ldots \wedge f_j \in D_j \Rightarrow f_j \in (D_k^T \setminus D_k) \qquad \forall k.i \leq k \leq j \tag{2.4}$$

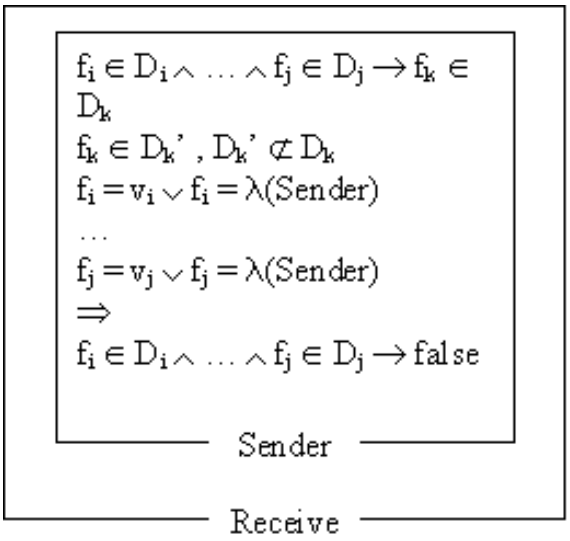

Figure 2.15: Propagation Rule

We say that a feature f_k of a topic T held by an agent A is *complete* if we know that agent A evaluates this feature to only one possible value:

$$complete(f_k) \textbf{ iff } (f_k = v_k \vee f_k = \lambda(A))$$

Let $f_k \in D'_k$ be the constraint associated to a feature f_k received in a message, and let

$$f_i \in D_i \wedge \ldots \wedge f_j \in D_j \Rightarrow f_j \in D_k$$

be a negative constraint on f_k of the second form, then we have the following *propagation rule*:

$$\frac{\begin{array}{c} f_i \in D_i \wedge \ldots \wedge f_j \in D_j \Rightarrow f_j \in D_k \\ D'_k \not\subset D_k, complete(f_i) \wedge \ldots \wedge complete(f_j) \end{array}}{f_i \in D_i \wedge \ldots \wedge f_j \in D_j \Rightarrow false.} \quad (2.5)$$

Positive constraints are of the form: $f_i \in C_i^T$ where $C_i^T \subseteq D_i^T$.

Negative constraints can be of the two forms:

$$\bigwedge_{i=1}^{k} (f_i \in C_i^T) \Rightarrow f_0 \in C_0^T$$

$$\bigwedge_{i=1}^{k} (f_i \in C_i^T) \Rightarrow false$$

with $C_i^T \subseteq D_i^T$ and $f_i \neq f_j$ for each $i, j = 0, \ldots, k$ and $i \neq j$.

Figure 2.15 shows a situation where the above rule is used by the receiver to derive a new constraint from current speaker state. The derivation rule **??** propagates new constraints in the system. This rule means that if we receive in a message the possible values of a feature f_k and we have a negative constraint on this feature, and the possible values of this feature are not included in the effect of the constraint, and we know that the sender has complete knowledge of the cause, we can conclude that the cause is false. We now to show that this rule is sound.

Theorem 2.9.1 (Soundness of the derivation rule) *Let $f_k \in D'_k$ be the constraint associated to a feature f_k, and let*

$$f_i \in D_i \wedge \ldots \wedge f_j \in D_j \Rightarrow f_j \in D_k$$

be a negative constraint on f_k of the second form, if $D'_k \not\subset D_k$ and $complete(f_i) \wedge \ldots \wedge complete(f_j)$ then

$$f_i \in D_i \wedge \ldots \wedge f_j \in D_j \Rightarrow false.$$

PROOF.

We now introduce a useful notation we borrowed from [11]. We define the *unknown* operator for a proposition P, denoted by $\mathbf{U}(P)$ by the following equivalence:

$$complete(f_k) \Leftrightarrow \forall D_k \subseteq D_k^T : \neg\mathbf{U}(f_k \in D_k)$$

Intuitively, if the sender has complete knowledge of a feature, she can always evaluate the truth value of the proposition $f_k \in D_k$ for all sets $D_k \subseteq D_k^T$. Therefore, for all $D_i \subseteq D_i^T, \ldots, D_j \subseteq D_j^T$

$$\begin{aligned} complete(f_i) \wedge \ldots \wedge complete(f_j) &\Rightarrow \neg\mathbf{U}(f_i \in D_i) \wedge \ldots \wedge \neg\mathbf{U}(f_j \in D_j) \\ &\Leftrightarrow \neg\mathbf{U}(f_i \in D_i \wedge \ldots \wedge f_j \in D_j) \end{aligned}$$

Let's call p the constraint's preconditions $f_i \in D_i \wedge \ldots \wedge f_j \in D_j$. We want to prove:

$$(p \Rightarrow f_k \in D_k) \wedge f_k \in D'_k \wedge D'_k \not\subset D_k \wedge \neg\mathbf{U}(p) \Rightarrow \neg p$$

which is equivalent to prove:

$$(\neg p \vee f_k \in D_k) \wedge f_k \in D'_k \wedge D'_k \not\subset D_k \wedge \neg\mathbf{U}(p) \Rightarrow \neg p$$

We need to consider two cases: $D'_k \cap D_k = \emptyset$ and $D'_k \cap D_k \neq \emptyset$

The first case is trivial since $f_k \in D'_k \wedge D'_k \not\subset D_k \wedge D'_k \cap D_k = \emptyset$ implies that $f_k \notin D_k$ and thus $\neg p$.

The second case is much more complicated since we have two sub-cases: the case when $D_k \not\subset D'_k$ and the case when $D_k \subset D'_k$. Only in the second case we can say that $f \in D_k$, whereas in the first case we can only say $f_k \in D_k \vee f_k \notin D_k$, that is

$$\mathbf{U}(f_k \in D_k) \tag{2.6}$$

.

To complete the proof we need to prove a Lemma that shows a general statement on the unknown operator. Intuitively, if we believe that an agent A considers the disjunction $(b \vee a)$ as true, and that she doesn't know anything about a (i.e. it can be true or false) and that she knows the truth value of b (whatever it is), then b must be true. Ballim and Wilks provided this rule in [11] together with other rules for a logical calculus of ignorance using the unknown operator $\mathbf{U}(P)$.

Lemma 2.9.1.1

$$(b \vee a) \wedge \mathbf{U}(a) \wedge \neg\mathbf{U}(b) \Rightarrow p \tag{2.7}$$

PROOF.

We consider three possible truth values for a and b: false, true and unknown. If b were false then a must be true contradicting $\mathbf{U}(a)$. If b were unknown it would contradict $\neg\mathbf{U}(b)$. Therefore the only remaining value for b is true.

□

We now have that:

$$(\neg p \vee f_k \in D_k) \wedge f_k \in D'_k \wedge D'_k \not\subset D_k \wedge \neg \mathbf{U}(p)$$

implies by 2.6 that

$$(\neg p \vee f_k \in D_k) \wedge \mathbf{U}(f_k \in D_k) \wedge \neg \mathbf{U}(p)$$

which in turn, by Lemma 2.7, implies $\neg p$, concluding our proof.

□

With the rewriting rule 2.3 and the propagation rule **??** we can derive new constraints each time information is added to the system. This reasoning schema goes exactly in the direction of assimilating knowledge about a topic by eliminating incompatible combinations of features. The derived constraints are stronger than the original one and therefore more informative.

Intuitively, if we believe the sender believes rules of the form "*Preconditions implies Specific interpretation of a topic*", and we receive an incompatible interpretation with these *Preconditions*, we can believe that the sender doesn't hold these *Preconditions*. We can then add this information to our vision of his topic space.

2.9.2.4 Adoption

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 propagated to the receiver using adoption. If the receiver trusts the sender, and if the new information is compatible with his opinion, he adopts it for himself. This new information can be separated in two sets, the *raw information received* (message content) and the *derived information*.

At this point we have to distinguish two cases: the case where the message received is only a confirmation, and the case where the message really brings new information. In the first case, the sender's mental state seen by the receiver has exactly the same amount of information as in the message. The message does not bring any new information. In this case, the receiver only adopts derived information, because the raw information is already contained in his mental state. In the second case the message really brings new information, therefore the receiver adopts the whole amount of information (the raw information and the derived information).

In conclusion we can say that knowledge assimilation is a fundamental process for every agent. In the case of natural language it is extremely complicated, but the four steps outlined in this section are fundamental. Transforming a message received into a mental representation (interpreting the message), consistency checking (analyzing the content of the message), deriving new information (using the message) and finally adopting this information (integrating knowledge) are the fundamental steps leading to *understanding*.

2.10 Revising knowledge

It is very useful to have an extended mental state expressive enough to represent other agents' viewpoints, but we also need to accept changes in their opinions over time. In this section we discuss the problem of attitude revision or how to manage make up of mind.

The problem of belief revision can be simplified as follows: we receive a message from an agent A and we detect that it is in contradiction with our current point of view on agent A. What do we think? We think that there is a contradiction, but since in her last message, agent A has probably made up her mind, and for her it might not be a contradiction at all. The problem now is to update our representation of her mental state accordingly. In a dialogue we could we solve this problem by asking an explanation. We ask agent A what are the reason of these changes and then

we update our representation. If the changes are important, the explanation may be complex and agent A needs a lot of effort to explain us everything. In most cases we don't understand all the reasons of the changes and we only trust the results. We thus remove our old opinion about agent A and store the new one.

What does it mean erasing our old opinion about agent A? Does it mean erasing everything or only a subset of their beliefs? In our mind these boundaries are not crisp: we can replace a part of an opinion and don't worry too much about remaining contradictions. In contrast in a computational representation everything is connected through causal links and the remaining a subset may cause other contradictions.

2.10.1 The "Untell" message

One practical solution to belief revision is to use the *untell* speech act. When an agent A changes its mind she informs the other agents that she doesn't hold a particular set of propositions anymore. She uses the "untell" speech act and tells the other agents that the communicated set of propositions has to be removed from their knowledge base. Each agent has to remove this set of propositions from his viewpoint of the sender. The assumption that an agent will inform other agents that she made up her mind is far from being realistic. We don't usually spend time informing all known people that we have made up our mind! When we detect that our mental representation of another agent is no more accurate, we ask for explanations if we are curious or if we have interest; but in many cases we only adopt the new point of view. The *untell* message goes perhaps in the good direction for attitude revision, but it is too strong for human-based communication.

If we go further in the implementation of processing an *untell* message we rapidly face lot of problems. It is easy to remove from a knowledge base a set of propositions as specified in the *untell* speech act semantics, but in the case of multiple viewpoints, the derived information and the adopted information becomes rapidly unmanageable.

Suppose we communicate to an agent A the proposition P. The agent A assimilates this new information, derives perhaps new information and finally adopts it for herself in case she trusts us. Later we make up our mind and don't consider P as true anymore. We inform the agent A with an *untell* message that P has to be removed. What will the agent A do with the derived information she has obtained and adopted? If the logical links are still valid, it is correct to remove all the derived information. But if these links were just assumed or have changed over time, it is not so clear how to remove the derived information. If we decide to remove all the derived information, we rapidly face with computational problems. It means that we need to keep for every information a derivation path (explanation) and maintain this over time.

There is still the problem of adopted information. In our example agent A has adopted proposition P, and the derived information because she trusts us. When we say that we don't consider P as valid anymore, what does she think? Does she still trust us and also rejects P and the derived information, or she asks for more information to take freely a decision, or she already committed with P? This is again not a trivial problem.

In our demonstrator we implemented the *untell* message as follows: the receiver of an *untell* message removes from her viewpoint of the sender the raw information (content of the message) but not the derived one. Then if she previously adopted the information, she rejects it too. However, the derived information which was adopted, still remains in the receiver's opinion.

As much as the knowledge assimilation process is semantically clear, and the actual problems are more concerned with computational complexity, the attitude revision process is still very complicated from the semantic point of view, and of course at the computational level. It would be interesting to go further in the study of this process and the way we deal naturally with changes in opinions.

2.11 Three Wise Men Puzzle revisited

The three wise man puzzle has attracted many researchers in knowledge representation since it is a prototypical problem of how difficult is to deal with partial information and communication. We started to be interested at this problem when we tried to implement the logical calculus proposed in [11] as its solution. Although the solution is correct, we realized that this logical calculus is difficult to turn into a computational proof procedure. We considered then a new way to go and we decided to look at a different representation of ignorance and in general of open predicates. We first looked at abduction and from this we considered the merging of the feature logic with the notion of *integrity constraints*. The result we have discussed in the previous section is a system which has several flavors: *features*, *situations*, *abduction*, *constraint satisfaction*, *model-based reasoning*, etc.

The *Three Wise Man Problem* (TWMP) was formalized by McCarthy in [50]. We consider one of the early versions by Attardi and Simi in [5] who state the TWMP as follows:

> *A king wishing to know which of his men is the wisest puts a white hat on each of their heads, tells them that at least one hat is white, and asks the first to tell the color of his hat. The man answers that he does not know. The second man gives the same answer to the same question. The third man answers that his hat is white. How did the third man know that his hat is white ?*

In this section we will show a complete simulation of this puzzle made in ViewFinder. We have three agents with reasoning capabilities and a King which is an interactive agent with the user. We provide then a description of the problem (initial state) to the three agents, and then interactively we ask for the color of the agents hats. Each time an agent answers to a request, her answer is broadcasted to the other agents. In that way we are simulating the fact that each agent can hear the answer of an other one. The interesting point in this version is the interactive part. Each agent has autonomous reasoning capabilities, and each time we do a request on one agent, she will do his best to answer. This approach allows to simulate the above stated problem but also study what happens, with more than three agents, or with more than two colors (white, black, red, ...).

2.11.1 Intuitive solution

Let's simulate intuitively what happens. Initially each agent can see the hats from the other agents but not his own. In the original version, each agent has a white hat. Initially each agent knows that at least one hat is white and that the two other agents he sees have a white hat.

Then the King asks to the first agent the color of his hat. What does the first agent know? He knows that at least one hat is white and he sees two white hats. Therefore he can not say anything about the color of his hat. He answers that he does not know. Every agent hears the answer.

Then the King asks the second agent the color of his hat. What does the second agent know? He knows that at least one hat is white, he sees two white hats and he knows that first agent does not know the color of his hat. He can deduce that if first agent does not know the color of his hat, then there is at least one hat white between second and third agent. But, even with this new information he can not figure out the color of his hat, because he still sees the third hat which is white. Therefore, he answers that he does not know. Every agent hears the answer.

Finally the King asks to the third agent the color of his hat. What does the third agent know? He knows that at least one hat is white, that the first agent does not know the color of his hat, that the second agent is aware that the first agent does not know the color of his hat and finally that the second agent does not know the color of his hat. If the second agent is aware that the first agent does not know the color of his hat, then the second agent knows that there is at least one hat white between second agent and third agent. Then if second agent still does not know the color of his hat, then there should be at least one hat white on the third agent. Therefore the third agent knows that his hat is white, and answers "white" to the King.

2.11.2 Our approach

Let's discuss now our solution and show how it closely corresponds the intuitive solution. We have three agents which each have the reasoning capabilities found in our implementation of ViewFinder. Each agent is capable of maintaining viewpoints on other agents and updating his mental state from communication as we discussed in the previous sections. We use the topic model to describe the TWMP. We model the problem as following: we have three agents `a1`, `a2` and `a3` of class `Wise`. We have a topic `HatColour` which has three features `a1`, `a2` and `a3` to represent the color of the hat from agents `a1`, `a2` and `a3`. These features take values in the set `[b, w]`, for black and white. We then have:

```
instanceOf(a1, Wise),
instanceOf(a2, Wise),
instanceOf(a3, Wise),
topic HatColour:
  class Wise:
    feature a1: [b, w]
    feature a2: [b, w]
    feature a3: [b, w]
```

This is the description of the context, then we have to give to each agent its initial knowledge. Each agent has the same type of knowledge, there is only a permutation in the names of the different agents. It is thus sufficient to show the initial knowledge from agent a1 :

Agent `a1` knows that at least one hat is white. This knowledge is translated into a constraint of the form

```
a1=b and a2=b and a3=b -> false.
```

Agent `a1` also sees the color of the hat from agent `a2` and agent `a3`. We can translate this information into constraints of the form:

```
a2=w and a3=w.
```

Then agent `a1` knows that agent `a2` and agent `a3` are capable of evaluating the color of his hat but himself is not able of doing so. This information can be represented with undefined values. Finally agent `a1` knows that agent `a2` and agent `a3` do not see their own hat, so he's not allowed to ascribe them the fact that he sees their color. The constraints `a2=w and a3=w` are private. Formally we obtain :

- agent `a1` believes for topic HatColour that:
  - `feature a2=w,`
  - `feature a3=w`,
  - `feature a1=b and feature a2=b and feature a3=b -> false`,
  - he can not ascribe to agent `a2` : `feature a2=w`,
  - he can not ascribe to agent `a3` : `feature a3=w`,
  - agent `a2` believes that
    - feature `a1` can be evaluated by agent `a2`, but agent a1 does not know its value: `feature a1=lambda(agent a2)`,
    - he can not ascribe to agent `a3` : `feature a3=w`,

- agent `a3` believes that
  - feature `a1=lambda(agent a3)`,
  - he can not ascribe to agent `a2 : feature a2=w`.

We obtain the initial knowledge of agent `a2` and agent `a3` by permuting the agent names. This is the whole initial information provided to each agent, which is represented by the following PROLOG code:

```
% Loads agent a1
:- processAction(loadEnv(bel(a1, topic(hatColour,
[
 constraints([
   % a1 sees hat from a2 which is white
   feature(a2, [w]),
   % a1 sees hat from a3 which is white
   feature(a3, [w]),
   % At least one hat is white
   imply([feature(a1, [b]), feature(a2, [b]), feature(a3, [b])], false)
]),
privateConstraints([
   % a1 can not ascribe to a2 the fact that his hat is white
   private(a2, [feature(a2, [w])]),
   % a1 can not ascribe to a3 the fact that his hat is white
   private(a3, [feature(a3, [w])])
 ])
])))).

:- processAction(loadEnv(bel(a1, bel(a2, topic(hatColour,
[
 constraints([
 ]),
 lambdaValues([
    % a2 sees the hat from a1 but a1 does not know its colour
    feature(a1, lambda(a2))
 ]),
 privateConstraints([
    % a2 can not ascribe to a3 the fact that his hat is white
    private(a3, [feature(a3, [w])]) ]) ])))))).

:- processAction(loadEnv(bel(a1, bel(a3, topic(hatColour,
[
 constraints([
 ]),
 lambdaValues([
    % a3 sees the hat from a1 but a1 does not know its colour
    feature(a1, lambda(a3))
 ]),
 privateConstraints([
    % a3 can not ascribe to a2 the fact that his hat is white
    private(a2, [feature(a2, [w])])
 ])
])))))).
```

It is interesting to notice that the information provided is only a translation of the problem definition and there are no additional reasoning tips or built in rules. Then each agent using this initial information and updating his mental state with communication will do his best to find the colour of his hat!

Lets go deeper now in the reasoning process of each agent. The King asks to agent `a1` the colour of his hat. Agent `a1` queries his mental state to find the value of the feature `a1` in the topic `HatColour`. The entire space for topic `HatColour` is interpreted as the tuples:

```
(b, b, b),
(b, b, w),
(b, w, b),
(b, w, w),
(w, b, b),
(w, b, w),
(w, w, b),
(w, w, w).
```

The constraints `feature a2=w`, `feature a3=w,` and `feature a1=b and feature a2=b and feature a3=b -> false` will cut this space to:

```
(b, w, w),
(w, w, w).
```

But we still see that feature `a1` can either be black or white. So agent `a1` can not conclude on the color of his hat.

He answers the King and communicates to the other agents that his hat, feature `a1` of topic `HatColour`, can be either black or white. He sends a message of the form `feature a1=[b, w]`, saying that he actually considers that he has no way of distinguishing if `a1=b` or `a1=w`. Agents `a2` and `a3` receive the message, process it and update their mental state. Let's see what happens in the mental state of agent `a2` when he receives the message.

Agent `a2` receives a message from agent `a1`. First he builds his point of view of agent `a1`. He uses the ascription algorithm and projects to `a1` all the information consistent with his point of view. Formally we have :

- agent `a2` believes for topic `HatColour` that
  - `feature a1=w`,
  - `feature a3=w`,
  - `feature a1=b and feature a2=b and feature a3=b -> false`,
  - he can not ascribe to agent `a1 :  feature a1=w`,
  - he can not ascribe to agent `a3 :  feature a3=w`,
  - agent `a1` believes that
    - `feature a2=lambda(agent a1)`,
    - he can not ascribe to agent `a3 :  feature a3=w.`

Agent `a2` ascribes to agent `a1` the constraints

```
feature a1=b and feature a2=b and feature a3=b -> false
```

and `feature a3=w`. He cannot ascribe the constraint `feature a1=w` because it is private. We obtain the constraints :

- agent `a2` believes for topic `HatColour` that
  - agent `a1` believes that
    - `feature a2=lambda(agent a1)`,
    - `feature a3=w`,
    - `feature a1=b and feature a2=b and feature a3=b -> false`,
    - he can not ascribe to agent `a3 : feature a3=w`,

Or seen as a topic space :

- agent `a2` believes for topic `HatColour` that
  - agent `a1` believes that
    - if `feature a2=b`
      - `(b, b, w)`,
      - `(w, b, w)`,
    - if `feature a2=w`
      - `(b, w, w)`,
      - `(w, w, w)`

Agent `a2` can explains that agent `a1` is not able to evaluate the color of his hat, by the fact that in every model agent `a1` believes that feature `a1` can be black or white. Once agent `a2` has built the viewpoint of agent `a1` on the topic `HatColour`, he notices that the message received is compatible with his mental model. The message says that feature `a1` has to be black or white, and agent `a2` believes that in all models of agent `a1`, feature `a1` has to be black or white. Therefore, agent `a2` is able to assimilate this message and derive new information. Using the constraints

```
feature a1=[b, w], feature a1=b and feature a2=b and feature a3=b -> false,
```

the information that agent `a1` has complete knowledge about the hats of agent `a2` and `a3`, and the derivation process, he concludes that agent `a1` may hold the constraint

```
feature a2=b and feature a3=b -> false.
```

The derivation has the following steps :

1. `feature a1=[b, w], complete(a2), complete(a3)`
   `feature a1=b and feature a2=b and feature a3=b -> false`
2. `feature a1=[b, w], complete(a2), complete(a3)`
   `feature a2=b and feature a3=b -> feature a1=w`
3. `feature a2=b and feature a3=b -> false.`

Agent `a2` commits this new information in agent `a1`'s viewpoint, and because he trusts agent `a1`, he adopts this new information. We have seen that agent `a2` has a mental model of agent `a1` that matches exactly the received message. In that case the message sent from agent `a1`, behaves like a confirmation for agent `a2`. Agent `a2` therefore commits his mental model from agent `a1` and only adopts derived information. At the end of the assimilating process agent `a2` stores the new constraint feature

```
a2=b and feature a3=b -> false.
```

Semantically, it means that agent a2 believes that at least one hat is white between agent a2 and agent a3. We see here the great similarity with the intuitive solution. The same reasoning process occurs in agent a3 so we do not detail it.

Then the King asks to agent a2 the color of his hat. Agent a2 queries his mental state to compute the value of feature a2 in topic HatColour. The constraints

```
feature a1=b and feature a2=b and feature a3=b -> false,
feature a1=w, feature a3=w and feature a2=b,
feature a3=b -> false
```

will cut the topic space to :

```
(w, b, w),(w, w, w).
```

Even with the new constraint adopted from agent a1, agent a2 is not able to determine the color of his hat. So he answers to the King that he does not know his color. The two other agents are informed of this answer and update their mental state. Let's look at the updating process in agent a3. When agent a3 receives the answer from agent a2 he first builds his viewpoint of agent a2. As usual he uses the ascription process and projects all the possible constraints. Formally we have:

- agent a3 believes for topic HatColour that
  - feature a1=w,
  - feature a2=w,
  - feature a1=b and feature a2=b and feature a3=b -> false,
  - feature a2=b and feature a3=b -> false,
  - he can not ascribe to agent a1 : feature a1=w,
  - he can not ascribe to agent a2 : feature a2=w,
  - agent a2 believes that
    - feature a3=lambda(agent a2),
    - he can not ascribe to agent a1 : feature a1=w

Here we have two cases:

1. Agent a3 is implicitly aware that agent a2 has received the same messages as him, and therefore he can project to a2 information he has adopted from communication. This is a shortcut which can be used if messages are always broadcasted.
2. Agent a3 has explicitly received a forward message which informs him that agent a2 has received the same message from agent a1. In this case, agent a3 has updated his viewpoint of agent a2 and derived the constraint feature a2=b and feature a3=b -> false.

In both cases agent a3 believes that agent a2 believes the constraints

```
feature a2=b and feature a3=b -> false.
```

Then as usual agent a3 projects to agent a2 the other compatible constraints, to obtain:

- agent a3 believes for topic HatColour that
  - agent a2 believes that
    * feature a1=w,
    * feature a3=lambda(agent a2),
    * feature a1=b and feature a2=b and feature a3=b -> false,
    * feature a2=b and feature a3=b -> false,
    * he can not ascribe to agent a1 : feature a1=w.

Or seen as a topic space :

- agent a3 believes for topic HatColour that
  - agent a2 believes that
    * if (feature a3=b)
      · (w, w, b),
    * if (feature a3=w)
      · (w, b, w),
      · (w, w, w)

It is fundamental to notice that agent a3 considers two disjunctive models of agent a2. If agent a2 has evaluated feature a3 to black then feature a2 is white, but if he has evaluated it to white, then feature a2 can be black or white. In this statement we see appear the solution to the Three Wise Men problem...

When agent a3 has built his point of view of agent a2, he compares it to the received message. The message claims that it is impossible to have only feature a2=b or feature a2=w. In that case, the only model compatible is the model with feature a3=w. So when agent a3 receives the message, he discards the model feature a3=b and he is left with only one model feature a3=w. In this case, he can do a further simplification and eliminate all undefined values. He commits that agent a2 believes that feature a3 is white. Because he trusts agent a2, he adopts the new information feature a3=w. Finally agent a3 is able to answer to the king that his hat is white.

In conclusion we can say that our solution is very close to the intuitive solution and that the ViewFinder reasoning capabilities (ascription, adoption and undefined values) are powerful enough to solve the Three Wise Men puzzle.

This simulation can be done with an arbitrary number of agents and hat colors, each time the last agent finds the color of his hat. The scaling is possible because the reasoning process always derives that, if there is at least one hat white in the current agent set and the current agent is not able to evaluate the color of his hat, then there must be at least one white hat in the remaining set of agents. Formally the process rewrites the fundamental rule:

feature a1=b and ... and feature an=b -> false

to

feature a2=b and ... and feature an=b -> false.

Then by recursion, the last agent is able to deduce the color of his hat.

2.11.3 Related works

In this section we compare our solution with two other solutions developed in these recent years. We look in turn at Ballim and Wilks's solution based on viewpoints [11] and Kim and Kowalski's solution based on meta-logic [42].

Ballim and Wilks give a theoretical example of the power of the ViewGen framework by showing how it is possible to solve the Three Wise Men puzzle with nested viewpoints. Kim and Kowalski use a meta-interpreter and a representation of common knowledge to show the reasoning process of the third agent. Although the reasoning process is similar, this solution suffers of some unnatural concepts. Our solution extends the approach from Ballim and Wilks and proposes a computational system supporting online reasoning for a real simulation.

2.11.3.1 Ballim and Wilks: Viewpoints

In the solution proposed in [11, pag. 208], the proof of how the third agent knows the color of his hat is based on the inference rule we have proved as a lemma in section 2.9.2:

$$(b \vee a) \wedge \mathbf{U}(a) \wedge \neg\mathbf{U}(b) \Rightarrow p$$

Using this rule a first time, the third agent concludes that the first agent believes that at least one hat is white between the second and the third agent; then using it a second time, the third agent concludes that the second agent believes the third hat is white. Using *percolation* (i.e. adoption)to exchange information between viewpoints, the third agent concludes his hat is white.

The reasoning steps are equivalent to our solution, but it is much more difficult to implement a real simulation. In this solution the third agent simulates how the second agent reasons about the first agent. Because the third agent has the idea to simulate the reasoning of the second agent on the first, he is able to find the color of his hat. In the proof it is a fact, it is shown that in order to solve this problem the third agent should do so, but in a real simulation agents should be free to answer to a query as they want. The main problem with this solution is the gap between online reasoning and an off-line proof. If we try to do a simulation using the existing tool ViewGen and plug a speech act processing module, it turned to be hard to implement. In this sense we can say that our solution is a deep revision of ViewGen which integrates consistency checking, undefined features and private knowledge, and which enables a real simulation close to natural reasoning. As we proved in section 2.9.2 our derivation rule is equivalent to the inference system used by Ballim and Wilks.

2.11.3.2 Kim and Kowalski: Meta Logic

Kim and Kowalski proposed a solution to the Three Wise Men puzzle using meta-logic. They use a predicate `demo(A, P)`to express that an agent `A` can prove a proposition `P`. He extends a simple meta-interpreter based on Horn clause logic to deal with undefined values. In belief management we must be capable of representing an agent who has *complete knowledge* of a proposition `P` (i.e. it can prove either `P` or `not(P)`) or, at the opposite, that it does not know the truth value of `P`. They introduced a predicate `complete(A, P)` which is equivalent to $\neg\mathbf{U}(P)$ of Ballim and Wilks:

`complete(A, P)` $\Leftrightarrow$ `demo(A, P)` $\vee$ `demo(A, ?P)`

Then they provide a reasoning rule for proving a proposition `P` when undefined values are possible:

`demo(A, P ? Q)` $\wedge$`not demo(A, Q)` $\wedge$ `complete(A, P)` $\Rightarrow$ `demo(A, P)`

It is easy to see that this reasoning rule is equivalent to the inference rule proposed by Ballim and Wilks:

$$(b \vee a) \wedge \mathbf{U}(a) \wedge \neg \mathbf{U}(b) \Rightarrow p$$

So the reasoning process to solve the Three Wise Men puzzle is essential the same in the three solution we considered, but Kim and Kowalski have no environments, ascription or adoption operations. The solution they adopt for the problem of exchanging information through viewpoints is that of *common knowledge*. They have a pool of information shared by each agents. In this pool they puts both the problem description and then the result of the reasoning process of each agent. The first agent can conclude using the reasoning rule that the second hat or the third hat is white. He puts this result in the common knowledge pool. This is correct because each agent can simulate the reasoning of the first agent based on the initial common knowledge and therefore each agent can conclude the same result. The second agent uses the reasoning rule a second time and concludes that the third hat is white and puts this result in the common knowledge pool.

Kim and Kowalski's solution is correct and uses the same reasoning process as that of Ballim and Wilks, and our. However their solution suffers of two unnatural concepts. The first problem is similar to that of Ballim and Wilks's solution: the gap between the off-line proof and online reasoning. The proof they provide is correct, but it is hard to imagine an autonomous agents answering a query from the King by searching for such a complex proof without any a priori guidelines (see [42, pag. 243] for details). The second problem is the use of common knowledge. Common knowledge can be justified in some cases, but the notion of relative viewpoints is closer to natural reasoning and natural knowledge representation. In this case, adding results of reasoning to the common knowledge pool, is a weak implementation of the adoption operation. We believe it is much more natural for an agent to freely adopt what he wants from other agents, than to put it in a common pool think everyone can access, and then assume that everyone automatically adopts it.

2.11.4 Common Knowledge: is it really necessary?

In many approaches to cognitive processing of information, elsewhere referred also as knowledge representation, private knowledge is distinguished from *common*, or *mutual*, or *shared* knowledge. These terms should denote the information that is shared by two or more agent in order to engage a successful dialogue, that is succeed in understanding each others.

Sidner [62] presents a model of *collaborative negotiation* based on the idea of establishing mutual beliefs, that is, things that we hold in common. This model rests upon the absence of deception, and appears fragile in the presence of mutual misunderstanding. The work of Cohen and Levesque [23] and of Smith and Cohen [63] is very similar to Sidner's work, but relies in addition on the primitive notion of *joint goals*. Based on Searle's idea [60] that requesting something means that one is attempting to get an agent to perform an action, they define all illocutionary acts in terms of agent's mental states (illocutionary is an act performed as the result of a speaker making an utterance; the effect is called a perlocutionary act).

We believe that the notion of Common Knowledge can be replaced by the notion of ascription and adoption. An agent may assume that another agents share part of its knowledge if they belong to the same agent class unless they provide contrary evidence of it. Conversely if an agent acquires some information from another agent the former can adopt this information in its knowledge base provided that it trusts the informing agent and unless it is in contradiction with the knowledge already present. Often the use of Common Knowledge has been advocated as the means to establish individual knowledge from knowledge about group members' knowledge. As pointed out in [53]:

> "Knowledge possessed in common by a group is nothing more than appropriately coordinated individual knowledge, viz. knowledge that each member of the group has and that each member knows they have",

we also believe that Common Knowledge can be dynamically built in terms of communicative and mental operations performed by some interacting agents. Using a default mechanism we avoid the need of constructing infinite chains of (mutual) nested modal knowledge operators whose least upper bound is what we are trying to attain: the Common Knowledge. Common Knowledge is then made intensional and defeasible since it appears as the result of a mental operation on the actual content of the agent's mental state and it persist until extensional knowledge is assimilated that conflicts with it.

The awareness that an agent shares another agent's knowledge is relevant only in the case of adoption and it can be modeled by distinguishing between private and adopted knowledge. In the case of ascription, the ascribing agent is making hypotheses on the other agent's knowledge that are used for its local reasoning. In general the latter agent may not be aware of this fact unless it is adopting the same knowledge from the former agent. Within this view, Common Knowledge arises from interaction and it is not stipulated *a-priori*.

Sharing a Common Knowledge implies that the sharing agent must be aware of the fact that the knowledge is shared and this fact is also shared, thus starting an ad-infinitum regression. This is one of the most common definition of Common Knowledge.

In [53] Situation Theory is used to represent viewpoints about communicating agents' belief states. The agent's mental state is modeled by a situation in which some local reasoning can be performed. We will see that this approach is similar to that of ViewFinder.

When faced to the problem of representing knowledge, we are immediately faced to the problem of finding a formalization of the real world which best suits our needs. This is rarely an easy problem and many solutions are often possible. In this project we need to manipulate and store structured information, so we have adopted an object model of the real world.

2.12 Intensional objects and concept hierarchies

The work of Ballim on ViewFinder addressed an important aspect which we did consider here. The problem of intensionality and in particular of the correspondence between different linguistic expression which denote the same object has been treated with particular care in his thesis. The notion of viewpoint allows us the maintain different "versions" of the same referential object. These version can be linked or not, and it depends on the degree of awareness of the cognitive agent the ability of determining that two different name have the same referent. When communicating with another agent who refers an object using a different name than the one we use, we should be able to trigger the link to the same object and selecting the right topic. The topic selection is then crucial to the success of the communication. We could keep talking about different things without realizing that we are talking about the same thing.

Another important issue is related to ontologies and concept hierarchies. In real dialogues we should be able to individuate references to object by communication. As we have seen in chapter 8, semantic interpretation based on lexical decomposition of words' semantic features, allows us to extract representations of evoked (and linguistically referred) individuals from the discourse. We would like now trigger a topic space from it and be able to incrementally instantiate the features values in the way we discussed in this section. The need of ontologies and in particular of conceptual hierarchies is apparent.

We are not the only one in advocating the need of conceptual hierarchies in dialogue. For this purpose see . However, we believe that again computational logic could be of great help. The work of Ballim et al. on multiple inheritance [9] and the work of Erbach on multi-dimensional inheritance with feature structures [33] could be a good starting point to enhance our model of mental states along this direction.

2.13 Mental State Recognition from Communication

Natural language dialogue involving more than one agent needs the modeling of connections between participants mental states. If we consider the information state approach, updates triggered by an utterance uttered, say, by agent A, cannot change only the hearer's mental state. The fact that A has uttered something denotes a change (or a perturbation of stability) in A's mental state which caused the utterance action.

This must be at least reflected in the representation that the hearer has of the speaker's mental state. Moreover, the speaker A may only make minimal assumption that the hearer has recognized the intended meaning of A's utterance and thus only partially full intentions and goals. In other words, recognition of mental state from communication might not be complete and a theory should be based on partial representations. As pointed out also by Cooper in [25],

> "as two agents A and B are involved in a conversation together their information states are aligned in some way. ... Now we suppose that B say something to A. It is not only the case that A gains the information that B wishes to convey, but B also now has information about what B herself said and what consequences that might have for what has been established as agreed in the conversation, or at least accepted for the sake of argument at this point in the dialogue (i.e. the common ground)."

This observation is crucial and the account of the information state approach to dialogue modeling for mental state recognition fails in capturing the above aspects, since it makes only distinction between private and shared belief, where shared beliefs are those propositions that have been "grounded" by communication. This model is adequate in cases where the system is only concerned with reconstructing a pre-defined belief space (e.g. a slot-filler structure). It is surely inadequate for the modeling of complex interactions such as that described in the "Three wise men problem".

2.14 A BDI view of a Dialogue Move Engine

We propose the integration of Information and Mental State in Dialogue Management. We propose to move towards a general distributed solution for Language Engineering in which software agents encapsulate flexible analysis modules:

- Specify at an abstract level how modules interact and the data-flow;
- Map the abstract specification to a concrete architecture;
- No initial commitment on the architecture topology that may also vary during the processing;
- Consider the analysis process as composed of different overlapping sub-tasks where robust natural language processors are the building blocks.

The diagram in figure 2.16 provides an overall picture of our proposed Cognitive Dialogue Architecture applied to Dialogue Systems.

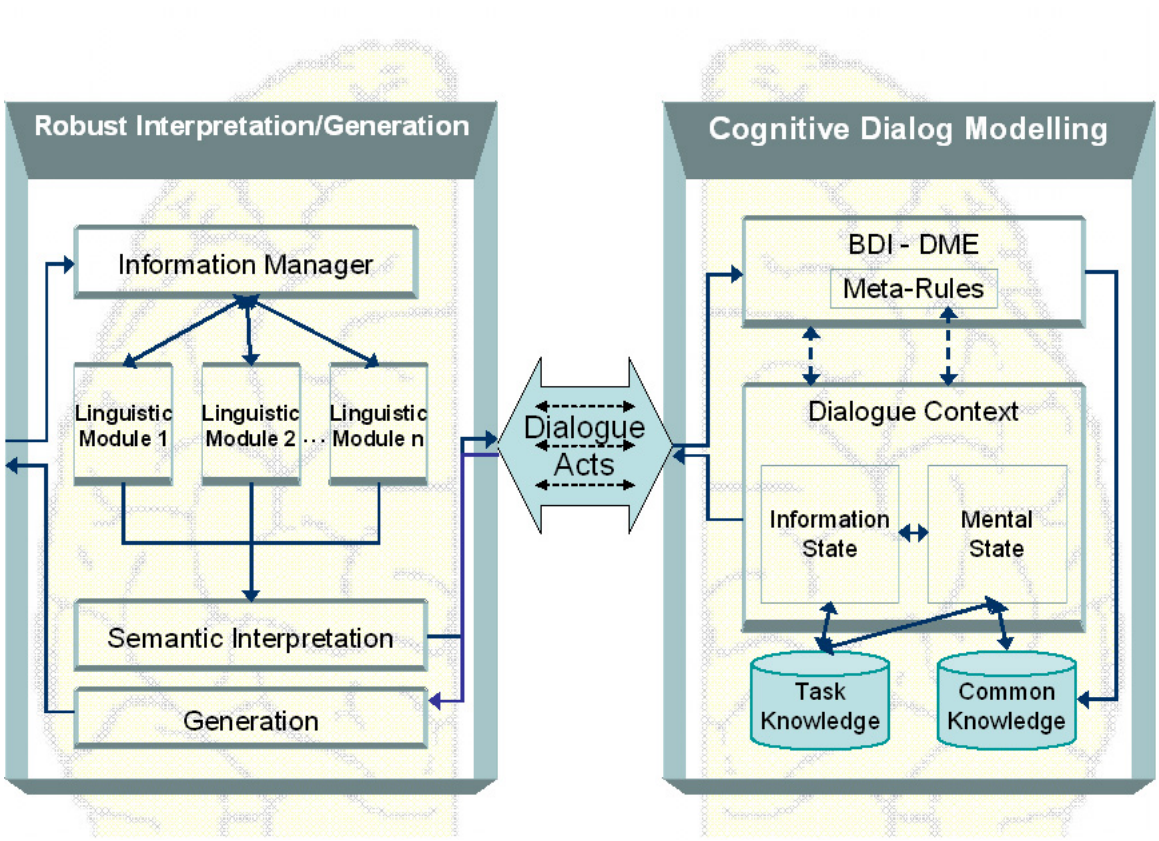

Figure 2.16: Cognitive Language Architecture for Dialogue Systems

Bibliography

[1] J. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. Towards a generic dialogue shell. *Natural Language Engineering*, 6(3):1–16, 2000.

[2] J. Allen, G. Ferguson, and A. Stent. An architecture for more realistic conversational systems. In *Proceedings of Intelligent User Interfaces 2001 (IUI-01)*, Santa Fe, NM, 2001.

[3] J.F. Allen and C.R. Perrault. Analyzing intention in dialogues. *Artificial Intellicence*, 15(3):143–178, 1980.

[4] Jens Alwood. An activity-based approach to pragmatics. In Harry Bunt and William Black, editors, *Abduction, Belief and Context in Dialogue*, Studies in Computational Pragmatics, pages 47–80. John Benjamins, 2000.

[5] G. Attardi and M Simi. Metalanguage and reasoning across viewpoints. In *Proceedings of ECAI-84*, pages 315–324, 1984.

[6] J. Austin. *How to do things with words.* Oxford University Press, London, UK, 1962.

[7] A. Ballim. Macro and micro attribution of mental attitudes to agents. In J. Horty and Y. Shoham, editors, *Reasoning about Mental States: Formal Theories & Applications (Papers from the 1993 AAAI Spring Symposium).* AAAI press, 1993.

[8] A Ballim. Propositional attitude framework requirements. *Journal for Experimental and Theoretical Artificial Intelligence(JETAI)*, 5:89–100, 1993.

[9] A. Ballim, S. Candelaria de Ram, and D. Fass. Reasoning using inheritance from a mixture of knowledge and beliefs. In S.R. Ramani and K. Anjaneylu, editors, *Proceedings of the KBCS'89 conference on Knowledge Based Computer Systems*, pages 387–396, Delhi, 1989. Narosa Publishing House.

[10] A. Ballim and Y. Wilks. Relevant beliefs. In *Proceedings of ECAI-90*, pages 65–70, Stockholm, 1990.

[11] A. Ballim and Y. Wilks. *Artificial Believers.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1991.

[12] A. Ballim, Y. Wilks, and J. Barnden. Belief, metaphor, and intensional identification. *Cognitive Science*, 15(1):133–171, 1991.

[13] Afzal Ballim. *ViewFinder: A Framework for Representing, Ascribing and Maintaining Nested Beliefs of Interacting Agents.* Ph.d., University of Geneva, 1992.

[14] P. Bohlin and S. Larsson. Godis and the dialogue move engine. "http://www.ling.gu.se/ sl/DME/", 2002.

[15] 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".

[16] M Bratman. *Intentions, Plans and Practical Reason.* Harward University Press, Cambridge, MA, 1987.

[17] K. Bühler. *Iena, 1934: sur les trois fonctions de la communication*, chapter chap. 2, sur l'acte et l'action. Bühler-Studien, Francfort-sur-le-Main, 1984.

[18] H. Bunt. Conversationa principles in question-answer dialogus. In D. Krallmann, editor, *Zur Theory der Frage*, pages 119–141. Narr Verlag, 1979.

[19] H. Bunt. Information dialogues as communicative action in relation to partner modelling and information processing. In M.M. Taylor, F. Neel, and D.G. Bouwhuis, editors, *The structure of Multimodal Dialogue*. Elsevier Publishers, 1989.

[20] Harry Bunt. Dialogue pragmatics and context specification. In Harry Bunt and William Black, editors, *Abduction, Belief and Context in Dialogue*, Studies in Computational Pragmatics, pages 81–150. John Benjamins, 2000.

[21] Saša Buvač, Vanja Buvač, and Ian A. Mason. Metamathematics of contexts. *Fundamenta Informaticae*, 23(3), 1995.

[22] Gavin E. Churcher, Eric S. Atwell, and Clive Souter. Dialogue management systems: a survey and overview. School of Computer Science studies research report series 97.06, University of Leeds, Leeds, February 1997.

[23] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[24] P.R. Cohen and C.R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.

[25] R. Cooper. Information states, attituds and dialogue. In *Proceedings of the Second Tblisi Symposium on Language, Logic and Computation*, Tblisi, September 1997.

[26] D. Cristea, N. Ide, and L. Romary. Veins theory. an approach to global cohesion and coherence. In Christian Boitet Whitelock and Pete, editors, *Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 281–285, Montreal, 1998. Morgan Kaufmann Publishers.

[27] D. Davidson. The logical form of action sentences. In Rescher, editor, *The logic of decision and action*, pages 81–120. Pittsburgh University Press, 1967.

[28] H. Decker. Some notes on knowledge assimilation in deductive databases. In Burkhard Freitag, Hendrik Decker, Michael Kifer, and Andrei Voronkov, editors, *Transactions and Change in Logic Databases, International Seminar on Logic Databases and the Meaning of Change and ILPS 97 Post-Conference Workshop on (Trans)Actions and Change in Logic Programming and Deductive Databases, (DYNAMICS'97)*, volume 1472 of *Lecture Notes in Computer Science*, pages 249–286. Springer, 1998.

[29] Keith Devlin. *Logic and Information.* Cambridge University Press, 1991. American Association of Publishers award as Most Outstanding Book in Computer Science and Data Processing of 1991.

[30] John Dinsmore. *Partitioned Representations: A study in mental representation, language understanding and linguistic structure*, volume 8 of *Studies in cognitive systems.* Kluwer, 1991.

[31] A.F. Dragoni, P. Giorgini, and L. Serafini. Mental states recognition from communication. *Journal of Logic and Computation*, 12(1):119–136, February 2002.

[32] Fred Irving Dretske. *Knowledge and the flow of information.* Basil Blackwell, Oxford, 1981.

[33] G. Erbach. Multi-dimensional inheritance. In H. Trost, editor, *Proceedings of KONVENS94*, pages 102–111, Vienna, 1994. Springer.

[34] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about knowledge.* MIT Press, 1996.

[35] J. Fry, S. Ginzton, M. Peters, B. Clark, and H. Pon-Barry. Automated tutoring dialogues for training in shipboard damage control. In Laila Dybkjae, David Traum, Julia Hirschberg, Ronnie Smith, and Jan van Kuppevelt, editors, *2nd SIGdial Workshop on Discourse and Dialogue*, Aalborg, September 1-2 2001. ACL. "http://www.sigdial.org/sigdialworkshop01/".

[36] C. Ghidini and F. Giunchiglia. Local model semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.

[37] F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics (or: How we can do without modal logics). *Artificial Intelligence*, 65:29–70, 1994.

[38] Paul Grice. Logic and conversation. In P. Cole and J.L. Morgan, editors, *Syntax and Semantics*, volume 3, chapter Speech Acts. Academic Press, New York, 1975.

[39] Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225, 1995.

[40] S. Jekat, A. Klein, E. Maier, I. Maleck, M. Mast, and J.J. Quantz. Dialogue acts in vermobil. Verbmobil Report 65, DFKI, 1995.

[41] 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.

[42] 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.

[43] S. Larsson, R. Cooper, and S. Ericsson. System description of godis. In *Presented at Third Workshop in Human-Computer Coversation*, Bellagio, Italy, July 2000.

[44] 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.

[45] M. Lee. *Belief, Rationality and Inference: A general theory of Computational Pragmatics.* Ph.d., University of Sheffield, 1998.

[46] M. Lee. Implicit goals in indirect replies. In *Proceedings of the ESCA tutorial and research workshop on Interactive Dialogue in Multi-Modal Systems*, Kloster Irsee, Germany, 1999.

[47] M. Lee and Y. Wilks. Eliminating deceptions and mistaken belief to infer conversational implicature. In *Proceedings of theIJCAI-97 workshop on Conflict, Cooperation and Collaboration in Dialogue Systems*, 1997.

[48] D.K. Lewis. Scorekeeping in a language game. *Journal of Philosophical Logic*, 8:339–359, 1979.

[49] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In Thomas Dean and Kathy McKeown, editors, *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 634–539. MIT Press, 1991.

[50] John McCarthy. Formalization of two puzzles involving knowledge. http://www-formal.stanford.edu/jmc/puzzles/puzzles.html, 1987.

[51] John 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.

[52] C.W Morris. *Foundations of the Theory of Signs*. Chicago University Press, Chigago, 1938.

[53] H. Nakashima, S. Peters, and H. Schütze. Communication and inference through situations. In *Proc. of the 12th IJCAI*, pages 76–81, Sidney, Australia, 1991.

[54] V. Pallotta. *Cognitive Language Engineering: Towards Robust Human-Computer Interaction*. Ph.d. thesis, Swiss Federal Institute of Technology Lausanne (EPFL), August 2002.

[55] Martha Pollack. Plans as complex mental attitudes. In Philip R. Cohen, Jerry Morgan, and Martha Pollack, editors, *Intentions in Communication*, pages 77–103. MIT Press, Cambridge, Massachusetts, 1990.

[56] D. Sadek. Design considerations on dialogue systems: from theory to technology - the case of ARTIMIS. In *Proceedings of the ESCA Workshop on Interactive Dialogue in Multimodal Systems*, pages 173–187, Kloster Irsee, Germany, 22-25 June 1999.

[57] D. Sadek and R. De Mori. Dialogue systems. In R. De Mori, editor, *Spoken Dialogues with Computers*. Academic Press, 1998.

[58] E.A. Schegloff and H. Sacks. Opening up closings. *Semiotica*, 7(4):289–327, 1973.

[59] J. Searle. *Indirect Speech Acts*. Academic Press, New York, 1975.

[60] J.R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, UK, 1969.

[61] S. Seneff and J. Polifroni. Dialogue management in the mercury flight reservation system. In *Proceedings of the Satellite Dialogue Workshop, ANLP-NAACL*, Seattle, April 2000.

[62] C. Sidner. An artificial discourse language for collaborative negotiation. In *Proceedings of AAAI-94*, pages 814–819. MIT Press, 1994.

[63] I.A. Smith and P.R. Cohen. Toward a semantics for an agent communication language based on speech-acts. In *Proceedings of AAAI-96*, pages 24–31. MIT Press, 1996.

[64] T. A. van Dijk. *Text and Context. Explorations in the semantics and pragmatics of discourse*. Longman Group Ltd, London, 1977.

[65] J. Widom and S. Ceri, editors. *Active Database Systems - Triggers and Rules For Advanced Database Processing*. Morgan Kaufman Publishers Inc., 1996.

[66] Y. Wilks and A. Ballim. Multiple agents and the heuristic ascription of belief. In *Proceedings of the 10th International Joint Conference on Artificial Inteligence*, pages 118–124. Morgan Kaufmann, 1987.