
INFORMATION SOCIETIES TECHNOLOGY (IST) PROGRAMME**HOMEY**

“Home Monitoring through an Intelligent Dialog System”

DELIVERABLE: D11 (*external*)

WORKPACKAGE: WP6

High Level Dialogue Specification

D11 – Definition of the High Level Task Specification Language

Author: **Martin Beveridge (Cancer Research UK)**
 David Milward (Cancer Research UK)

Submission Date: 31/03/2003

Partners: Engineering Ingegneria Informatica, (I), Reitek (I), Consorzio Bioingegneria e Informatica Medica (I), Istituto Trentino di Cultura (I), Cancer Research United Kingdom (UK), Language & Computing (B)

SUMMARY

This document is part of the result of the research project HOMEY funded by the IST Programme within the 5th Framework Programme as project number IST-2001-32434.

One of the goals of the HOMEY project is to develop a technology to be used for deploying innovative tele-medicine services. These new services will be based on an Intelligent Dialog System (IDS), designed and developed to effectively manage an incremental dialog between a tele-medicine system and a patient, taking into account user needs, preferences and time course of her/his disease.

The purpose of work package 6 is to investigate the use of abstract task specifications for dialogue management to improve reusability of dialogue management components. The abstract specifications are related to existing knowledge representation schemes used in medicine, and evaluated in two main settings, firstly as the basis for VoiceXML specifications, secondly as direct input to a generic dialogue manager which allows more flexible dialogue management, allowing e.g. user initiative and corrections. The approach will be evaluated by building dialogue demonstrators in English and Italian.

In most current dialogue systems, possible interactions with the system are hand-coded in the design. This is an expensive process, especially for complex dialogues. This deliverable motivates the use of a task description language for building flexible and adaptive dialogue systems in ontologically rich domains such as medicine. It describes the components of a task specification, and proposes an architecture for dialogue systems which allows integration of domain reasoning and dialogue. A high-level dialogue specification is used to support multimodality. This can be used with the current voice-based standard, VoiceXML for spoken interaction.

CONTENTS

1. Abstract	5
1.1. PURPOSE OF THE HOMEY PROJECT	5
1.2. PURPOSE OF WORK PACKAGE 6	5
1.3. PURPOSE OF THIS DELIVERABLE	5
1.4. AUTHORSHIP OF THIS DOCUMENT	5
1.5. LIST OF ABBREVIATIONS	5
2. Introduction	6
3. Background	7
3.1. ASPECTS OF DISCOURSE STRUCTURE	7
3.1.1. <i>Linguistic Structure</i>	7
3.1.2. <i>Intentional Structure</i>	7
3.1.3. <i>Information Structure</i>	9
3.1.4. <i>Attentional State</i>	10
3.2. MULTI-LEVEL MODEL OF DISCOURSE	10
3.2.1. <i>Discourse Analysis</i>	10
3.2.2. <i>Text Generation Versus Dialogue</i>	11
3.3. DIALOGUE SPECIFICATION	12
3.3.1. <i>Dialogue Grammars</i>	13
3.3.2. <i>Plan Recognition</i>	13
3.3.3. <i>Conversational Games</i>	15
3.3.4. <i>Information State Update</i>	20
3.3.5. <i>Conversational Agents</i>	22
4. High-Level Dialogue Specification	25
4.1. OVERVIEW	25
4.2. DESCRIPTION	28
4.2.1. <i>Intentional Structure</i>	28
4.2.2. <i>Information Structure</i>	31
4.2.3. <i>Attentional State</i>	35
4.3. SPECIFICATION LANGUAGE	39
4.3.1. <i>Transactions</i>	39
4.3.2. <i>Games</i>	40
4.3.3. <i>Moves</i>	40
4.3.4. <i>Topics</i>	42
4.3.5. <i>Relations</i>	42
5. Abstract Task Specification Language	43
5.1. OVERVIEW	43
5.2. DESCRIPTION	44
5.2.1. <i>Domain Plan</i>	44
5.2.2. <i>Domain Ontology</i>	45
5.3. SPECIFICATION LANGUAGE	47
5.3.1. <i>Load</i>	47
5.3.2. <i>Get Metadata</i>	47
5.3.3. <i>Check Completion</i>	47
5.3.4. <i>Get Task Hierarchy</i>	47
5.3.5. <i>Get Task Dependencies</i>	47
5.3.6. <i>Get Task Specification</i>	48
5.3.7. <i>Set Data Item</i>	49
5.3.8. <i>Confirm Task</i>	49
5.3.9. <i>Find Associations</i>	49
5.3.10. <i>Get Category</i>	49
5.3.11. <i>Get Domain</i>	49
References	51

Appendix A: Implementing the High-Level Dialogue Specification	56
HIGH-LEVEL DIALOGUE SPECIFICATION LANGUAGE	56
<i>Transaction Element</i>	56
<i>Game Element</i>	56
<i>Move Element</i>	57
<i>Topic Element</i>	58
<i>Domain Element</i>	58
<i>Option Element</i>	59
<i>Content Element</i>	60
<i>Relation Element</i>	61
MAPPING TO LOW-LEVEL SPECIFICATION LANGUAGES	61
<i>Sentence Generation</i>	62
<i>XSL Transformation</i>	63
Appendix B: Implementing the Abstract Task Specification	66
TASK SPECIFICATION LANGUAGE	66
<i>Load</i>	66
<i>GetMetaData</i>	66
<i>IsCompleted</i>	66
<i>GetTaskHierarchy</i>	67
<i>GetTaskDependencies</i>	67
<i>GetTaskInfo</i>	67
<i>SetDataItem</i>	67
<i>ConfirmTask</i>	68
<i>FindAssociations</i>	68
<i>GetConceptCategory</i>	68
<i>GetConceptDomain</i>	68
SPECIFYING THE DOMAIN PLAN	69
<i>PROforma</i>	69
<i>Mapping to the Task Specification Language</i>	72
SPECIFYING THE DOMAIN ONTOLOGY	74
<i>L&C Ontology Browser</i>	74
<i>Mapping to the Task Specification Language</i>	74

1. Abstract

1.1. Purpose of the HOMEY project

The purpose of the HOMEY project is to carry out research and develop technology to be used for deploying innovative tele-medicine services. The new services will be based on an Intelligent Dialogue System (IDS), designed and developed to effectively manage an incremental dialogue between a tele-medicine system and a patient, taking into account user needs, preferences and the time course of her/his disease. Intelligent dialogue requires the representation of goals, intentions, and beliefs about the effectiveness of the interaction in terms of quality of health care management. The dialogue system will require dynamic adaptation in order to understand the patient's medical problems and the physician's goals, handle misunderstandings, and carry-out argumentation regarding therapy options. In order to support such adaptation, a representation of the medical domain knowledge, the evolution of the disease of a specific patient, and the history of user-system interactions need to be represented.

1.2. Purpose of Work Package 6

The purpose of this work package is to investigate the use of abstract task specifications for dialogue management to improve reusability of dialogue management components. The abstract specifications will be related to existing knowledge representation schemes used in medicine, and evaluated in two main settings, firstly as the basis for VoiceXML specifications, secondly as direct input to a generic dialogue manager which allows more flexible dialogue management, allowing e.g. user initiative and corrections. The approach will be evaluated by building dialogue demonstrators in English and Italian.

1.3. Purpose of this Deliverable

In most current dialogue systems, possible interactions with the system are hand-coded in the design. This is an expensive process, especially for complex dialogues. This deliverable motivates the use of a task description language for building flexible and adaptive dialogue systems in ontologically rich domains such as medicine. It describes the components of a task specification, and proposes an architecture for dialogue systems which allows integration of domain reasoning and dialogue. A high level dialogue specification is used to support multimodality. This can be used with the current voice-based standard, VoiceXML, for spoken interaction.

1.4. Authorship of this Document

Responsibility for authorship is divided as follows. Sections 1 and 2 were written by David Milward. Sections 3, 4, 5, and appendices A and B were written by Martin Beveridge.

1.5. List of Abbreviations

<i>CRUK</i>	Cancer Research UK	Homey partner responsible for WP6
<i>L&C</i>	Language & Computing n.v.	Homey partner responsible for WP5
<i>ITC</i>	Istituto Trentino di Cultura	Homey partner responsible for WP3

2. Introduction

In most current commercial dialogue systems, the dialogue designer specifies the exact interactions which can take place. This hand-coding allows precise control of what can occur within a dialogue. However, it is an expensive process, especially for complex dialogues where the number of states can be in the hundreds of thousands. In the medical domain, where the knowledge structures are particularly complex, the number of states could run into millions. Moreover, this kind of approach leads to inflexible dialogues, and is not very appropriate if the task is changing dynamically (for example, during the enactment of a clinical guideline).

In this deliverable we explore an alternative approach where dialogues are generated automatically from *task specifications*. We take the task specification to be the minimal amount of knowledge which has to change from one task to another i.e. anything that is not included in the generic dialogue manager. The task specification includes knowledge about the particular domain, and knowledge about tasks that the user might request or be expected to perform. In HOMEY, the domain knowledge is provided by L&C's medical ontology. The task knowledge was originally expected to be a static domain plan. However, we now use a plan execution engine, which sends tasks to the dialogue system during the enactment of a clinical guideline. Several tasks may be passed to the dialogue manager at once, including specification of dependencies (if any) between tasks. The dialogue system then performs any remaining ordering and structuring of the tasks using domain knowledge.

The second notion we introduce in this deliverable is that of a high-level dialogue specification. Once the dialogue manager has received a task it needs to decide how next to act. In conventional dialogue systems, the dialogue manager might output a single new move e.g. a question to a user. However, in a multi-modal context, the output might be a whole HTML form rather than a single question. The high-level dialogue specification provides the information required by either a graphical or spoken modality. This is then rendered appropriately. For example, HTML rendering may construct one or more forms. VoiceXML rendering may create the first question on the form, but also provide a language model enabling a user to answer any question on the form. For example, if the form asks for age and sex, the VoiceXML may include the prompt: "What is the patient's age", with an associated language model which accepts age and sex e.g. "age 55, male".

Section 3 of the document sets-out some relevant background on discourse and notions employed to account for discourse structure, and then goes on to discuss different approaches to representing dialogue and some recent approaches to building practical dialogue systems. Section 4 introduces the notion of a high-level dialogue specification, provides some motivation for this level of representation and describes what information it should capture and what form it should take. Section 5 discusses the notion of an abstract task specification, and its relation to plan execution and medical ontology components. It then outlines a task specification language that supports the required functionality.

3. Background

This section provides a brief overview of approaches to representing discourse structure and in particular dialogue.

3.1. Aspects of Discourse Structure

Three types of information have been proposed to account for the structure of discourse, namely: an *intentional structure*, an *information structure* and an *attentional state* (Grosz and Sidner, 1986; Hobbs, 1993; Moore and Paris, 1993). These are then mapped onto the *linguistic structure* (Grosz and Sidner, 1986) of a dialogue, i.e. the actual observed sequence of utterances.

3.1.1. Linguistic Structure

The linguistic structure of discourse (Grosz and Sidner, 1986) consists of the sequence of actual utterances that comprise the discourse. These are furthermore considered to naturally aggregate into units called dialogue segments which reflect the underlying intentions. The particular intention underlying a dialogue segment (DS) is referred to as the dialogue segment purpose (DSP). Each utterance in a DS serves a particular role with regard to that segment and similarly each DS fulfils certain functions with regard to the overall discourse purpose (DP). Grosz and Sidner (1986) liken this structure to the syntax of sentences in which words serve a particular role within a phrase and phrases fulfil particular functions with regard to the overall sentence. Given this model, it can therefore be seen that linguistic structure (e.g. segments and their relationships) simply reflects the underlying intentions and their relationships.

3.1.2. Intentional Structure

As described above, it has been argued that the linguistic structure of discourse simply reflects an underlying intentional structure. This intentional structure is described in (Grosz and Sidner, 1986) as consisting of a series of intentions, each of which may serve as the purpose of a dialogue segment (DSP) in the linguistic structure and which defines how the related segment contributes to the overall discourse purpose (DP). Grosz and Sidner (1986) acknowledge that there may be an infinite number of possible intentions, for example: intending that some agent intend to perform some task, intending that some agent believe some fact, intending that some agent know some property of an object etc. They claim, however, that there are only two important structural relations that can hold between them - *satisfaction-precedence* and *dominance*, defined (where I_1 and I_2 are intentions) as:

- I_1 *satisfaction-precedes* I_2 if I_1 must be satisfied before I_2
- I_1 *dominates* I_2 if I_2 provides part of the satisfaction of I_1 .

As a shorthand, satisfaction-precedence is often written simply as the relation *SP* (e.g. I_1 *SP* I_2) and dominance as the relation *DOM* (e.g. I_1 *DOM* I_2). Hovy and Maier (1993) suggest that these two relations can be seen as similar to the linear precedence and immediate dominance relations of syntactic theories, such as the ID/LP format for grammars in Generalised Phrase Structure Grammar (GPSG) (Gazdar et al., 1985). Hence they simply represent structural relations between intentions, rather than semantic relations.

An example of intentional structure is provided in (Grosz and Sidner, 1986) using a task-oriented dialogue taken from (Grosz, 1981) between an expert and an apprentice where the aim of the task is to remove a flywheel from some equipment. A fragment of this example is reproduced below.

```
DS1 E: First you have to remove the flywheel.
    A: How do I remove the flywheel?
    E: First loosen the two allen head setscrews holding it to the
        shaft, then pull it off.
    A: OK.
    DS2 I can only find one screw. Where's the other one?
    E: On the hub of the flywheel.
    A: That's the one I found. Where's the other one?
    E: About ninety degrees around the hub from the first one.
    A: I don't understand. I can only find one. Oh wait, yes I think
        I was on the wrong wheel.
    E: Show me what you are doing.
    A: I was on the wrong wheel and I can find them both now.
    ...
    DS3 The two screws are loose, but I'm having trouble getting the
        wheel off.
    ...
```

In this example DS1, DS2 and DS3 arise from intentions I1, I2 and I3 respectively which are defined by (Grosz and Sidner, 1986) as follows (where E denotes Expert and A denotes Apprentice):

```
Intentions:
    I1 = (Intend E (Intend A (Remove A flywheel)))
    I2 = (Intend A (Intend E (Tell E A (Location other setscrew))))
    I3 = (Intend A (Intend E (Tell E A (How (Getoff A wheel)))))

Dominance Relations:
    I1 DOM I2
    I1 DOM I3

Satisfaction-Precedence Relations:
    I2 SP I3
```

As can be seen the overall intention (DP) is for A to remove a flywheel and the other two intentions (to find the location of the setscrews and to get the wheel off) both contribute to this purpose hence the stated dominance relations. Furthermore the setscrews must be located before the wheel can be taken off (as stated by E: “*First* loosen ... *then* pull ...”) and hence a satisfaction-precedence relation is assumed.

An alternative approach to representing intentional structure is taken by Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) which instead tries to enumerate a set of ‘presentational relations’ which can describe how one intention relates to another, e.g. Motivation, Justification, Evidence, Concession, Antithesis etc. As (Grosz and Sidner, 1986) points out, however, the problem with this approach is determining what an appropriate set of relations might be. In principle it is possible to keep enlarging the set in order to capture increasing subtle differences in intentional relations until the theory represents no significant

generalisations about discourse at all (e.g. in the extreme, every pair of segments might be labelled with a unique relation). Furthermore, it has been argued (Moser and Moore, 1993) that many of the intentional relations in RST correspond to different types of dominating intentions in (Grosz and Sidner, 1986), e.g. “an RST EVIDENCE relation can occur only when the dominating intention is to affect another’s belief. Similarly, an RST MOTIVATION relation can occur only when the dominating intention is to affect another’s action” (p. 95). In some respects, then, RST relations might be seen as macro structures which are largely reducible to Grosz and Sidner’s (1986) approach.

3.1.3. Information Structure

In addition to the intentional approach to discourse structure described above, it has also been argued that coherent discourse can be described in terms of informational (semantic) relationships between segments. Whilst (Grosz and Sidner, 1986) does propose two relations which can be considered informational (Moore and Pollack, 1992), namely *supports* ($\text{supports}(I_1, I_2)$ iff $I_2 \text{ DOM } I_1$) and *generates* ($\text{generates}(I_1, I_2)$ iff $I_1 \text{ SP } I_2$), the main research on such relations has been carried out in the context of text generation (Hovy, 1993b). For example, in order to generate the sentence “John’s car will be admired because it is a sports car” it is necessary to know about the semantic relation of causality between the notion of ‘sports car’ and ‘being admired’ in order to generate the appropriate linking word “because” (rather than “unless”, “although” etc). The definition of information structure has therefore been dominated by the RST approach of enumerating sets of relevant relations (which RST refers to as ‘subject-matter relations’) with the usual problem of determining what set of such relations is required.

In order to address this problem, (Hovy and Maier, 1993; Maier and Hovy, 1993) carried-out a study comparing all the various relations used in the literature and classifying them into a taxonomy according to their function in the text (based on Halliday’s categorisation of linguistic phenomena into metafunctions (Halliday, 1985)). The number of researchers that had used each relation was also noted, thereby providing a rough confidence score. Their proposal was that relations could be used at whatever level of the taxonomy was appropriate to the domain (with the lower levels describing increasingly subtle distinctions). The most common top-level relations were:

- *Elaboration*: one clause presents additional detail regarding the situation described in the other clause, e.g. [I love to go to the cinema.]_{C1} [My favourite films are horror films.]_{C2}, where C_2 elaborates C_1 .
- *Cause/Result*: one clause presents the cause of the result described in the other clause, e.g. [John’s car will be admired]_{C1} [because it is a sports car]_{C2}, where C_1 is the result of C_2 .
- *Sequence*: the situations described in the clauses occur in sequence, e.g. [I had some dinner]_{C1} [then I went to the cinema]_{C2}, where C_1 occurred before C_2 .
- *Circumstance*: one clause describes the circumstances of the situation described in the other clause, e.g. [It was after midnight]_{C1} [when Bill finally got home]_{C2}, where C_1 describes the time at which C_2 occurred.

Using such relations it is then possible to describe the relationships between the propositional contents of utterances in a discourse, i.e. “an informational structure, imposed by domain

relations among the objects, states and events being discussed” (Moser and Moore, 1996, p. 416).

3.1.4. Attentional State

Another aspect of discourse structure that has been studied is the way in which the discourse unfolds over time, e.g. the way in which the participants’ focus of attention shifts and the salience of entities under discussion varies (hence affecting phenomena such as anaphora). Grosz and Sidner (1986) represent this through the notion of a dynamic attentional state which describes all the objects, properties and relations that are salient at a particular point in a discourse. In particular, they model it as a ‘focusing structure’ - a stack (or stacks) of focus spaces each relating to a discourse segment and representing the underlying intention (DSP) for that segment along with descriptions of the salient entities and relations.

Such a model is therefore clearly closely related to notions of intentional structure and the manipulation of the focus space stack is determined entirely by intentional relations. For example if intention I_1 dominates intention I_2 then a focus space FS_1 which describes the associated dialogue segment DS_1 will be added to the focus stack first and is the most salient focus space. Because I_2 forms part of the satisfaction of I_1 the next operation will add FS_2 (which describes the associated dialogue segment DS_2) to the top of the stack making it now the most salient. As a segment is completed the associated focus space is removed from the stack (hence the attentional state does not simply replicate the intentional structure) and referring expressions can be resolved by reference to the focus stack.

A key aspect of the attentional state is to coordinate the intentional and linguistic structures, allowing intentions to be mapped to utterances and vice versa. The attentional state is therefore in some respects an interface representation between the linguistic structure and essentially non-linguistic representations such as intentions (which are primarily related to task goals) and information relations (which are primarily related to domain relations between entities under discussion).

3.2. Multi-Level Model of Discourse

3.2.1. Discourse Analysis

Models of discourse such as RST suggested that only one relation should be assigned to consecutive discourse segments, effectively forcing a choice between intentional (presentational in RST) and information (subject-matter in RST) relations (Moser and Moore, 1996). Moore (1995), however, suggests that recently there is a growing consensus that all three of the structures described above are required to describe discourse structure. For example, Moore and Pollack (1992) have demonstrated that natural language interpretation and generation require parallel representations of both intentional and informational relations between discourse segments. These two structures have been characterised as representing “what is being talked about (informational)...[and] why we are talking about it (intentional)” (Moser and Moore, 1993, p.94) and can provide quite different accounts of discourse. In particular, recognising one type of relation can aid the hearer in recognising the other (Moore and Pollack, 1992; Hobbs, 1993). For example, consider the following example (adapted from Moore and Pollack, 1992).

```
[George Bush supports big business,]c1  
[so he's sure to veto House Bill 1711.]c2
```

If the hearer does not know what House Bill 1711 is, but is able to determine the intentional relation, that C_1 is intended to motivate C_2 (e.g. provide a cause of which C_2 is an effect) then they can infer that C_1 is evidence for C_2 and that House Bill 1711 must be something that a supporter of big business would oppose. Conversely, if the hearer knows that House Bill 1711 concerns something, e.g. environmental controls, that is typically opposed by big business then they are able to determine that C_1 is evidence for C_2 and hence can conclude that the speaker's intention in uttering C_1 was to increase their belief in C_2 (i.e. to motivate C_2).

As a result of this, there have recently been various attempts to combine intentional and informational approaches to discourse. For example, Moser and Moore (1996) present a synthesis of RST and the intentional relations in (Grosz and Sidner, 1986) based on parallel intentional and information structures, whilst Marcu (2000) has instead attempted to extend RST with Grosz and Sidner's relations within a single framework. In addition, Hovy (1993a, p. 37) has argued that "the inclusion of control information [e.g. attentional state] in discourse planning systems has not received the attention it deserves" and argues for an even broader notion of 'rhetorical structure' which "differs from the semantic and the intentional structure" in that it incorporates "the effects of both, as well as of other constraints on the discourse" such as attentional state (Hovy, 1993a, p.37).

The end result of these studies has been that much of the remaining debate on discourse structure "centres around which of these three structures are primary and which are parasitic" (Moore, 1995), and the different roles they play in practical interpretation and generation tasks. The different points of view seem largely to depend on the type of discourse in question (e.g. monologue or dialogue) and the task to be achieved (e.g. interpretation or generation). The next section briefly compares approaches in text generation and dialogue systems.

3.2.2. Text Generation Versus Dialogue

Research on developing practical text generation systems (e.g. McKeown, 1985; Moore and Swartout, 1990; Paris, 1990) has tended to primarily rely on informational relations which are particularly useful for determining text structure, e.g. relations of clauses, cue words etc. This does not mean, however, that such systems do not make use of intentions, but rather that intentional relations are effectively compiled-out into schemas which are sufficient for a particular domain and task, or that they are conflated with the informational relations on the basis that it is sufficient to reason about such semantic relations between facts (Mittal and Paris, 1993). For example Hobbs (1993), whilst supporting the integration of intentional and information structures, has argued that intentions are often indirect or uninformative or, in the case of written texts and monologues, unimportant, with interpretation relying more on appreciating the information contained in the discourse.

It has been shown, however, that informational relations alone are not sufficient to describe dialogue (Mittal and Paris, 1993; Carberry et al., 1993; Traum, 1993), where handling misunderstandings or communication failures relies on knowing the original intention in

order to, for example, propose follow-up questions (Moore and Paris, 1992). This is demonstrated by the following example (adapted from Moore and Paris, 1992):

```
S: [You'll need the Phillips screwdriver.]C1  
   [It's in the top drawer of the toolbox.]C2  
   Do you have it?  
H: No
```

Using the informational relations described above, the relation between C_1 and C_2 could be described as CIRCUMSTANCE whilst the intention underlying C_1 is to make the hearer identify and find the screwdriver (Mittal and Paris, 1993). At the point where the hearer replies “no”, the speaker has to realise that this intention has not been satisfied and to employ other means to attempt to achieve the same intention. If all that was recorded was that there is a CIRCUMSTANCE relation between C_1 and C_2 , then the original intention could not be recovered and it would be unclear how to continue the dialogue. In addition more than one semantic relation may be associated with the same intention depending on the viewpoint taken, e.g. it could be argued that the location ‘top drawer’ is an attribute of the object ‘screwdriver’ (as in “the top drawer screwdriver”) and that C_1 and C_2 are in an ELABORATION relation (Mittal and Paris, 1993). Clearly, this doesn’t change the way in which the dialogue should be continued, which is instead determined by the intentions of the initiating participant.

For these reasons, dialogue systems have been primarily based on notions of intentional structure, based on underlying task plans, with relatively little use of information relations. However it has also been appreciated that information relations can contribute to dialogue, for example “they prove to be a convenient intermediary between reasoning about high level intentions and actual surface forms” and “are often convenient for inferential purposes” (Traum, 1993), they are “a useful computational tool to represent constraints we currently don’t totally understand, avoid duplicating reasoning from first principles, and provide an appropriate level of interface with the realization component” (Mittal and Paris, 1993). Furthermore, as described above, recognising information relations can assist in determining a speaker’s intentions (Carberry et al., 1993). Hence, whilst Moser and Moore (1993) considered “only monologic discourse...believing generalizations between this and multi-agent discourse to be premature” (p. 94), recently there has been some interest in applying informational relations to dialogue (e.g. Stent, 2000).

The next section describes some approaches that have been used to specify dialogue in practical dialogue systems.

3.3. Dialogue Specification

Traditional approaches to specifying dialogue in dialogue management systems have generally been divided into two main categories: dialogue grammar-based approaches and plan-based approaches. Some recent research, though, has tried to synthesise these two approaches via a model of conversational (or dialogue) games. This section briefly introduces dialogue grammars and plan recognition and then discusses Conversational Game Theory and recent information state update approaches to dialogue.

3.3.1. Dialogue Grammars

Dialogue grammars were one of the first approaches to representing dialogue and involved the development of a prescriptive grammar which described commonly-occurring sequences of utterances such as adjacency pairs (e.g. question followed by answer), and in some cases the entire dialogue. This approach therefore involves simply defining the linguistic structure of a dialogue without any reference to other non-linguistic notions such as intentions or information relations.

By employing dialogue grammars a dialogue management system can be as simple as a graph or finite state machine where each node represents a prompt to the user with a set of options, and the user's response causes a transition to a new node. Such approaches have been useful for systems where the dialogue structure closely matches the task structure. In particular, since the system always takes the initiative it can restrict the number of options presented to the user and, to an extent, induce a valid user response via the priming effect of the prompts.

In order to allow some mixed-initiative dialogue, extensions to graph systems have been developed such as frame-based systems. In these entities are defined (e.g. a journey) which have slots to be filled (e.g. departure time, departure location etc) and at each node in the graph the dialogue manager has to ensure all mandatory slots are filled. This might be achieved by the system taking the initiative and prompting the user until all information has been gathered, or the user might take the initiative and fill more than one slot at once providing all the relevant information.

The use of such approaches for the implementation of medical dialogue systems is described in deliverable D6 (Stefanelli et al., 2002).

3.3.2. Plan Recognition

In contrast to dialogue grammars, plan-based approaches to representing dialogue allow for much greater complexity in the dialogue. They take the approach that dialogue is goal-driven and so the aim of the dialogue manager is to infer these goals and respond appropriately. This approach allows for more complex phenomena such as indirect communicative acts where what is meant (illocution) is not the literal interpretation of what is said (locution), e.g. the case where a user asks a train timetable system "can you tell me when the last train to London leaves?". The correct response is for the system to inform the user of the departure time for the requested train and not to answer "yes" or "no".

In order for a dialogue system to be able to reason about goals and their connection to utterances, a model of an agent's 'mental state' is required so that speech acts can be related to these mental states in the conversational participants. The model originally proposed involved describing the configuration of beliefs, desires and intentions of an agent (Cohen and Perrault, 1979) and is often referred to as the BDI model (for a recent review of BDI, see Rao and Georgeoff, 1995). Using a BDI approach axioms can be defined which relate actions, and in particular speech acts, to mental states. For example, as described in (Pulman, 1997), a Request speech act might have a pre-condition which matches the situation where the speaker wants some action A to be done and believes that the hearer can do A, and have a post-condition such that the hearer believes that the speaker wants A – hence a change of mental state of the hearer results. Axioms of 'rational agency' can also be defined, e.g. *cooperativity*: if the hearer believes the speaker wants A then the hearer also wants A, or

desire leads to action: If X wants A and X can do A then X does A. Inference can then proceed such that the speech act leads to a change of mental state on the part of the hearer followed by action on the part of the hearer as a result of ‘rational behaviour’, e.g. (taken from Pulman, 1997)

```
Speaker requests Hearer to do A
∴ Hearer believes Speaker wants A    (Request Postcondition)
∴ Hearer wants to do A                (Cooperativity)
∴ Hearer does A                      (Desire leads to action)
```

Axioms can also be defined to capture more complex phenomena, e.g. to handle the indirect communication described at the start of this section would require an axiom such that if an agent asks if you can do something, and you can reasonably assume that they know that you can (e.g. asking a train timetable system if it knows the time of a train, or asking someone if they can pass the salt) then assume they are really asking you to do that thing (e.g. to provide a train time or to pass the salt) (Perrault and Allen, 1980).

Pulman (1997), however, also describes several problems with this speech act-based approach. Firstly the model of mental states only allows for description of beliefs, desires and intentions and yet dialogue typically contains utterances which don’t fall easily into any of these categories, e.g. not everything that is said is immediately believed by the participants, as demonstrated in the following dialogue fragment

```
[1] S: Where would you like to go?
[2] U: London.
[3] S: to London?
[4] U: Yes.
```

In this case the proposition that U wants to go to London is not properly a belief of S until utterance 4. Until that point it is merely a proposal that has not been grounded (Traum, 1996a). This has led some researchers to prefer a notion of ‘commitment’ rather than belief, where a speaker is committed to any statements they make, or agree to, or that logically follow from other commitments, whether or not they are actually believed by either of the participants (Lewin, 2000; Pulman, 1997). Commitments may therefore be retracted a later point but not denied. Such a notion is particularly important in, for example, legal dialogues where what matters is the actual statements and their ramifications rather than speculations about participants’ private beliefs. This approach also circumvents the question of what the required level of nested beliefs for dialogue is (although Carletta and Mellish (1995) suggest that no more than two levels are required).

Whilst the above problem might be handled by allowing ungrounded proposals (as in Traum and Allen, 1994) there are other dialogue phenomena which fit even less well into the BDI model: dialogue control phenomena such as acknowledgements, pause-fillers, indicating turn-taking etc which maintain the dialogue and coordinate participants. More importantly BDI doesn’t capture the notion of obligations (Traum and Allen, 1994; Kreutel and Matheson, 2000) which seem to arise from social convention and include, for example, the fact that if someone asks you a question, it is considered unreasonable not to answer. In fact speech act theory (and the BDI formulation) only deals with a single utterance and so cannot

distinguish between a response to a request or answer to a question and a standard declarative used to initiate a conversation (Pulman, 1997; Lewin, 2000). This inability to capture the local context of an utterance, and represent its function given that context, means that there is no way to capture the convention that answers follow questions or that people don't walk away in the middle of a conversation – things that, in fact, can be captured in dialogue grammars by distinguishing grammatical and ungrammatical dialogue structures. Obligations such as these are so strong that in fact participants reply to questions even when they are unwilling or unable to contribute to the goal of the question, e.g. the dialogue fragment below (Traum and Allen, 1994):

```
A: Did Pete drive here?  
B: I don't know/I don't want to talk about that.
```

In this case B refuses to assist in the satisfaction of A's intention, yet B still acts, if only to express their refusal or inability to cooperate or create some diversion – complete silence is not generally an option (or at least is a very marked case which violates social conventions). Inability to deal with uncooperative responses of this sort is an important weakness for BDI approaches as it is not clear how cooperative dialogue really is in practice (Davies, 1994). To the extent that conversation *is* cooperative, such “cooperativity in dialogue behaviour can be regarded as an instance of its social character, and can be explained as a consequence of acting according to social norms and conventions” (Bunt, 1996).

In order to handle these sorts of problems (Traum and Allen, 1994) proposed an extension to the BDI model to include ‘discourse obligations’, leading to what might be called a BDIO model (Pulman, 1997). Concomitantly the types of speech that must be described becomes much wider than those described by speech acts, leading instead to the notion of ‘dialogue act’ (Traum, 1996a; Poesio and Traum, 1998) which includes both ‘core speech acts’ (i.e. the original set of speech acts) augmented with so-called ‘argumentation acts’ (e.g. answer, signal-understanding, utterance failure etc). Pulman (1997) points out, however, that whilst this begins to capture simple conventions like question-answer it doesn't address the more general social pressure to respond, such as the hearer giving a reply to a speaker's question which does not constitute an answer but which the hearer hopes will be taken as a relevant response. Describing these more subtle phenomena within the BDI approach, however, probably requires the representation of more complex (and ill-defined) notions such as politeness (Boella et al., 1999) and other aspects of human nature and society, which in the extreme tends to AI-completeness, i.e. requires a complete model of human agency (Jurafsky and Martin, 2000).

There is another approach, however, which has developed largely in parallel with BDI approaches, which can be seen as addressing some of the problems of speech act theory (Pulman, 1997), namely Conversational Game Theory (Power, 1979; Houghton and Isard, 1987; Kowtko et al., 1993; Carletta et al., 1996).

3.3.3. Conversational Games

Conversational Game Theory is primarily a descriptive approach to dialogue rather than a theory of ‘rational agency’ as the BDI approach is intended to be. For this reason it circumvents some of the problems encountered by BDI since it starts from the premise of simply trying to describe the facts encountered in real dialogues rather than why they occur.

In order to do this it represents dialogue at two functional levels: at the plan-based level are conversational *games* which are associated with the mutual goals of the participants, and at the structural level are sequences of conversational *moves* which are intended to achieve those goals (Kowtko et al., 1993).

Moves

The notion of ‘move’ employed here extends speech acts to include acts such as reply, acknowledge, clarify etc. Moves are either initiating moves of games (i.e. rather similar to speech acts) or responding moves. Kowtko et al. (1993) and Carletta et al. (1996) propose a set of moves based on an analysis of task-oriented dialogues from the Edinburgh ‘Map Task’ corpus (Anderson et al., 1991) in which two (separated) participants are given maps, one of which has a route drawn on it. The person with the route has to give instructions to the other participant so that they can replicate the route on their own map. However, because the maps they have been given differ in certain respects (e.g. certain landmarks) misunderstandings arise which have to be resolved. The proposed set of moves is:

- Initiating moves:
 - *Instruct*: provide instruction
 - *Check*: request confirmation of known information
 - *Query-yn*: ask yes/no question for unknown information
 - *Query-w*: ask complex question (wh-question) for unknown information
 - *Explain*: provide information that wasn’t previously requested
 - *Align*: check alignment of position in task
- Response moves:
 - *Clarify*: clarify or rephrase given information
 - *Reply-y*: affirmative reply
 - *Reply-n*: negative reply
 - *Reply-w*: complex reply containing requested information
 - *Acknowledge*: acknowledge and signal continuation
 - *Ready*: indicate intention to begin a new game

The notion of move employed here is therefore similar to the notion of ‘dialogue act’ as described previously (Traum, 1996a; Poesio and Traum, 1998). In fact there has recently been an attempt by the Discourse Resource Initiative (Allen and Core, 1997; Core and Allen, 1997) to rationalise the different sets of moves/dialogue acts proposed by different researchers into a taxonomy, in much the same way that (Maier and Hovy, 1992; Hovy and Maier, 1993) have done for rhetorical relations (as described previously). The resulting ‘Dialogue Act Markup in Several Layers’ (DAMSL) classifies acts as either forward-looking or backward-looking. The forward-looking function describes acts that influence the subsequent state of the dialogue (i.e. similar to initiating moves) and the backward-looking function describes acts that refer to the previous state of the dialogue (i.e. similar to response moves). The proposed taxonomy is given below.

Forward-looking acts:

- Statement*
 - Assert*
 - Reassert*
 - Other-Statement*
- Influencing addressee future action*
 - Open-option*
 - Directive*
 - Action-Directive*
 - Info-Request*
- Committing speaker future action*
 - Offer*
 - Commit*
- Conventional*
 - Opening*
 - Closing*
- Performative*
- Other forward function*

Backward-looking acts:

- Agreement*
 - Accept*
 - Accept-part*
 - Maybe*
 - Reject*
 - Reject-part*
 - Hold*
- Understanding*
 - Signal-non-understanding*
 - Signal-understanding*
 - Acknowledge*
 - Repeat-rephrase*
 - Completion*
 - Correct-misspeaking*
- Answer*
- Information-relation*

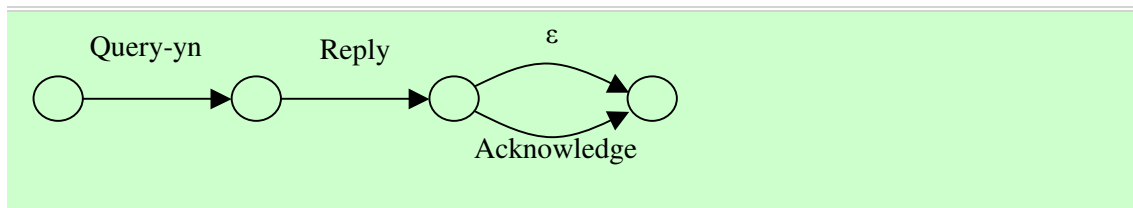
As can be seen, some of the moves proposed by (Kowtko et al., 1993; Carletta et al., 1996) can be related directly to the DAMSL acts, e.g. *instruct*=Action-directive, *explain*=Assert etc. Furthermore it is intended that DAMSL categories can be further subdivided as necessary, hence *query-yn* and *query-w* moves would both be subclasses of Info-request¹. Note also the inclusion of Information-relation as a backward-looking act. Core and Allen (1997) explain that information-relations are intended to be like the relations defined in RST (Mann and

¹ It may also be worth noting that the moves proposed by (Kowtko et al., 1993; Carletta et al., 1996) are also similar to some of the performatives suggested for agent communication languages. For example the FIPA communication language (FIPA, 1999) contains performatives such as inform (similar to *explain*), request (*instruct*), query-if (*query-yn*), query-ref (*query-w*), inform-if (*reply-y* and *reply-n*), inform-ref (*reply-w*), and agree/accept-proposal (*acknowledge*) amongst others.

Thompson, 1988) and discussed previously in Section 3.1.3, hence an utterance can be represented in DAMSL both in terms of intention, e.g. answering a question, and information relation, e.g. providing evidence for a claim in a previous utterance. A set of information relations has not yet been constructed for DAMSL however (Core and Allen, 1997). This also illustrates a further key aspect of DAMSL, namely that an utterance can fulfil more than one function simultaneously (e.g. make a request and a promise), something that isn't possible in speech act theory (Core and Allen, 1997).

Games

In Conversational Game Theory, dialogues are thought-of as being comprised of a series of games each aiming to achieve some sub-goal of the dialogue. Each game itself consists of a series of moves starting with an opening move and finishing with an end move. Importantly the definition of a game includes moves by both participants, e.g. a request game includes a request by the initiating participant and a reply by the other participant, hence conventional links such as question-answer are captured by using a unit of discourse that spans multiple utterances. The internal structure of a game is typically represented in a similar way to dialogue grammars. For example, a request game may consist of a request move from the speaker, followed by a reply move by the hearer and optionally a final acknowledgement from the speaker to indicate that the information in the reply is grounded. This can be represented as a finite-state network as shown below (the ϵ -link indicates that no move is required for the state transition).



Additionally, a game can have nested sub-games or a break. Sub-games account for phenomena such as clarifications, side sequences etc in which the sub-game contributes to the goals of the parent game. Breaks account for misunderstandings and indicate that either repair is needed in order to continue, or that the current game may have to be abandoned (Kowtko et al., 1993). This is shown in the following example (adapted from Pulman, 1997):

[1] S: Where would you like to go?	Query-w		
[2] U: Edwinstowe	Reply-w		
[3] S: Edwinstowe?	Check		
[4] U: Yes	Clarify	CHECK	
[5] S: Is that Edwinstowe in Nottingham?	Check		
[6] U: Yes	Clarify	CHECK	
[7] S: Ok	Ack		QUERYW

Pulman (1997) points-out that although there is a change in context between each utterance and three games are played, there is only one significant change to the participants' shared commitments with utterances 3-6 simply checking and grounding the information in 1 and 2. The moves in these nested games can therefore be seen as having a different status whereby "the units of planning are those represented by the acquisition of agreed propositions rather than the units that correspond to the conversational games like 'checking' or

‘acknowledgement’. It is not plausible to assume that such moves are planned: rather, they arise as an immediate response to the current state of the dialogue” (Pulman, 1997).

Whilst games can clearly capture conventional links between utterances, such as the fact that a request must be followed by a reply, Maudet and Evrard (1998) argue that more subtle phenomena can also be explained by appeal to a notion of participants’ levels of commitment to a game. The dialogues below demonstrate this.

```
[1] A: Did Pete come to the party last night? (Q)
    B: I don't want to talk about that.

[2] A: Did Pete come to the party last night?
    B: I don't know.

[3] A: Did Pete come to the party last night?
    B: Was there free beer?
    A: Yes.
    B: Then he was there.
```

In all of the above dialogues B provides a response to A’s question as required by convention, but in [1], B’s response indicates a refusal to enter the game: their commitment level is ‘no commit’ in Maudet and Evrard’s (1998) terms. This phenomenon has also led some researchers (Ginzburg, 1997; Maudet and Evrard, 1998) to view questions such as Q in [1] as having two functions: first an implicit proposal to enter a questioning game (whether B discusses Q) and secondly introducing a goal for the other participant to provide information (Q). The hearer must accept the proposal before the goal of the game can be adopted, but in [1] B rejects the proposal. In [2], B does accept the proposal to play the questioning game but Maudet and Evrard suggest that this is only a commitment to the rules of the game (which they refer to as ‘r-commit’). B doesn’t flout the rules (assuming they really don’t know the answer) but on the other hand B doesn’t adopt the goal of the game either, and consequently does not apply any strategy to achieve that goal. In [3], on the other hand, B adopts the game’s goal and initiates a strategy of using sub-games in order to reach satisfaction of that goal (i.e. a state where it is mutually known whether Pete came to the party) rather than simply asserting ignorance. Maudet and Evrard (1998) refer to this last as an example of ‘s-commit’.

The notion of viewing dialogue in terms of games and moves therefore captures the fact that most conversations to achieve a task follow standard scripts (e.g. question-answer) to achieve a limited set of goals (e.g. getting some information, instructing someone) and so generates quite specific expectations regarding a participant’s response to a conversational move (Poesio and Mikheev, 1998). At the same time the recursive structure of games and sub-games allows complex mixed-initiative dialogues to be modelled. This approach therefore combines aspects of plan-based approaches with aspects of dialogue grammars with moves providing a model of the conventional structure of dialogue, and the higher-level model of plans and goals being represented in terms of games, hence allowing more complex reasoning about the motivations of the dialogue and conversational cooperation.

The description of conversational games given above is derived primarily from their use for describing discourse structure, e.g. for dialogue mark-up (e.g. Kowtko et al., 1993; Carletta et al., 1996) and developing dialogue systems (e.g. Houghton and Isard, 1987; Williams, 1996;

Lewin, 1998). These approaches have concentrated primarily on the structural aspects of games, in particular its usefulness for predicting future dialogue moves from the dialogue history, which is important in spoken dialogue systems where it can be used to restrict the search space for the speech recogniser (Poesio and Mikheev, 1998). However, conversational games have also been used in other areas of research that are more interested in high-level or logical aspects of dialogue, such as: modelling argumentation (e.g. Maudet and Moore, 1999), modelling collaboration (e.g. Burton and Brna, 1996), developing communication protocols for multi-agent systems (e.g. McBurney and Parsons, 2002) and providing a formal semantics for programming languages (e.g. Hyland and Ong, 1995).

In these more AI-oriented approaches to dialogue games the plan-based aspect of games is developed so that the description of a game is given by a set of logical rules. Some of these rules codify dialogue conventions of the sort described previously, such as that questions are followed by answers, but others attempt to capture non-linguistic aspects of games. These include (Maudet and Evrard, 1998):

- Rules governing the effect of a move on the cognitive context of each of the participants
- rules governing the rationality of the speaker (e.g. prevent the speaker from playing some moves based on their mental state),
- rules governing the rationality of the hearer (e.g. the principle that a rational hearer should not accept all the speaker's utterances) and
- rules governing players' high-level strategies.

In this approach, a game is more than a recipe of moves. It is instead a set of possible moves and rules of various types that affect the selection of a particular move at a particular point in the dialogue. This model obviously connects conversational games more strongly with the BDI tradition, and indeed much of this research involves BDI models of the conversational participants.

Recent research on dialogue systems has followed what in many respects seems a similar route by adopting an Information State Update view of dialogue as proposed in the TRINDI project² (Larsson and Traum, 2000).

3.3.4. Information State Update

Recent work on developing spoken dialogue systems has led to the development of a model of dialogue referred to as the Information State Update approach (Cooper and Larsson, 1999; Larsson and Traum, 2000; Lewin, 2000). In this approach, utterances are viewed as transforming a 'information state' which represents all the information available to a participant at a given point in a dialogue. Hence, it is similar to the AI-based approach to dialogue games described previously, in that dialogue is modelled as a series of moves with rules defining how moves update the information state and rules defining how new moves are selected based on the information state. For example, statements cause the information state to be updated with propositional information, whilst questions provide motivation to generate statements. The information state therefore functions rather like a blackboard and is related to

² Task Oriented Instructional Dialogue, EC Project LEA-8314, <http://www.ling.gu.se/research/projects/trindi>.

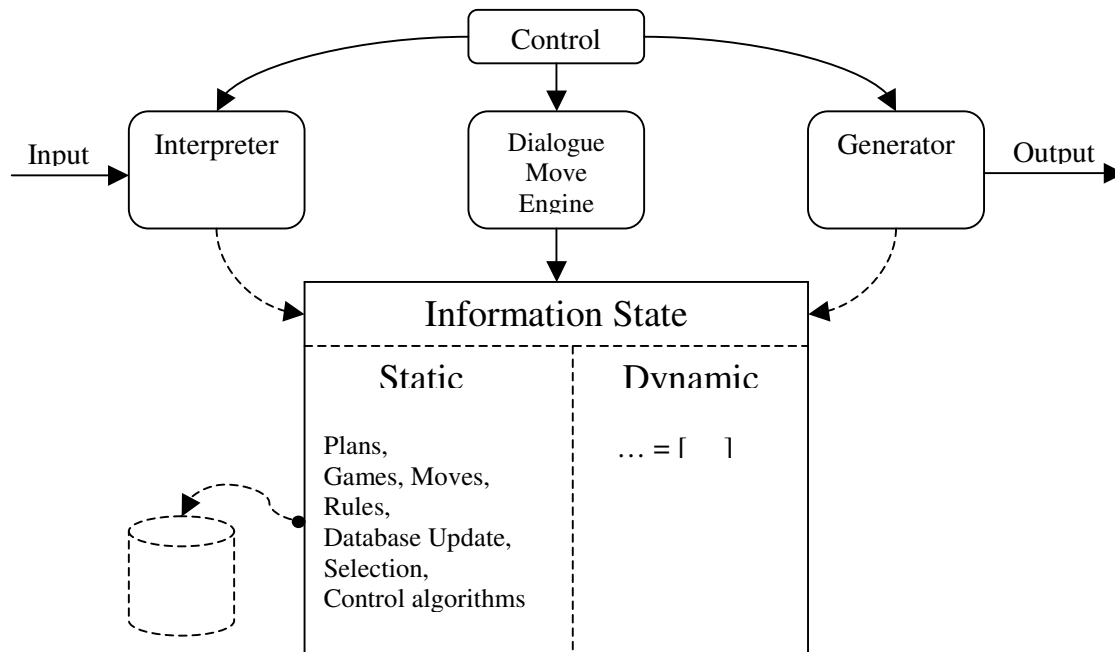


Figure 3.1: The TRINDI Architecture for Information State Update

notions of ‘conversational score’ (Traum, 1996a), ‘gameboard’ (Maudet and Evrard, 1998; Ginzburg, 1997), ‘mental state’ (Cohen and Perrault, 1979), ‘discourse context’ and so on. Furthermore, it contains both dynamic (changing over the course of a dialogue) and static components.

The set of operations that the dialogue manager must carry out include: interpreting input utterances and classifying them as instances of particular moves, updating the information state based on the identified move, selecting a move for the system to carry-out next, and updating the information state based on that move (Lewin, 2000). The architecture for the Information State Update approach proposed in the TRINDI project is shown in Figure 3.1 (based on Lewin, 2000).

Pulman (1999) argues that the attempt to relate dialogue games to information states could have several advantages from a theoretical point of view, including:

- Providing a principled way of choosing between different repertoires of games (because it would be possible to compare their definitions in terms of information state updates)
- Providing better ways of annotating corpora (because in principle it would be possible to determine the ‘correct’ annotation)
- Allowing the possibility of contributing to the handling of prosody (because the information state should contain notions of given vs. new, semantic focus etc as required for prosody).

Pulman (1999) further discusses the possibility that dialogue moves could eventually be reduced to sets of information state updates in a similar fashion to attempts to reduce speech acts to notions of belief, desire and intention, possibly allowing moves to be dispensed with

altogether. Dialogue moves would then be “a convenient macro for regularly co-occurring combinations of linguistic form and information state change” (Pulman, 1999) and games would be seen as an epiphenomenon rather than a separate level of linguistic description.

Pulman (1997) suggests that there have been various approaches which can be seen (at least implicitly) as proceeding in this manner, including the interpretation of dialogue moves within the scope of an update semantics for dialogue (Ginzburg, 1994) and attempts to provide an axiomatisation of dialogue acts in terms of their effects on different components of a participant’s information state (Poesio and Traum, 1998). These two approaches are compared in (Cooper et al., 1999; Pulman, 1999). Further attempts include (Kreutel and Matheson, 2000) who attempt to extend Poesio and Traum’s, (1998) theory of discourse obligations by showing that intentional structure (Grosz and Sidner, 1986) can be derived by inference over obligations. Since games model the structure of dialogue as a reflection of intentional structure, they argue that games should be seen as structures that emerge from participants acting according to the obligations imposed on them, i.e. as macro-structures which can be decomposed into smaller functional units which are related in terms of obligations. An alternative approach is suggested in (Pulman, 1997) which reconstructs games within a Bayesian framework.

3.3.5. Conversational Agents

Hulstijn (2000) suggests another interpretation of conversational games that might be used to derive a theoretical basis, namely a parallel with current trends towards hybrid architectures for artificial agents. Early agent architectures were essentially deliberative, relying on planning and world-modelling, but, because these turned out to be more complex problems than expected, it was realised that such an approach was inadequate when faced with an uncertain and unpredictable environment (Gat, 1998). Brooks instead proposed a different approach, Subsumption Architecture (Brooks, 1990), which involved no reasoning but instead wired together many small finite state machines in a series of layers. Complex tasks could then be accomplished reactively, e.g. by simply coupling sensors to actuators through a simple transfer function. Wooldridge (2002) points-out, however, that this architecture also has limitations: it relies only on local information and is difficult to engineer because the combined interactions between all the individual reactive components are difficult to understand.

Recently, there has been research on combining reactive and deliberative approaches in so-called hybrid architectures. This work seems to parallel the conversational game theory and information state update approaches to dialogue, which Hulstijn (2000) argues have led to the conclusion that “the perceived opposition between a plan-based and a pattern-based approach to natural language dialogue is false. ...Once the constraints of joint planning and action are taken seriously, a need for conventions to regulate the interaction arises...The smallest recipes for joint action are precisely the exchanges described by dialogue game rules. On the other hand, plans and goals may function as a semantics for dialogue game rules.” Hence, Hulstijn (2000) argues that “[t]he insight that pattern-directed approaches need to be combined with higher-level notions like plans and goals, is compatible with a general trend towards hybrid architectures for agents” where “[t]he general principle that underlies the trend seems to be that frequently occurring activities that can be ‘automated’ are often dealt with by fixed pattern-directed protocols, recipes or rules. Infrequent activities or failure and misunderstanding require higher-level deliberation. It seems that dialogue is no exception to

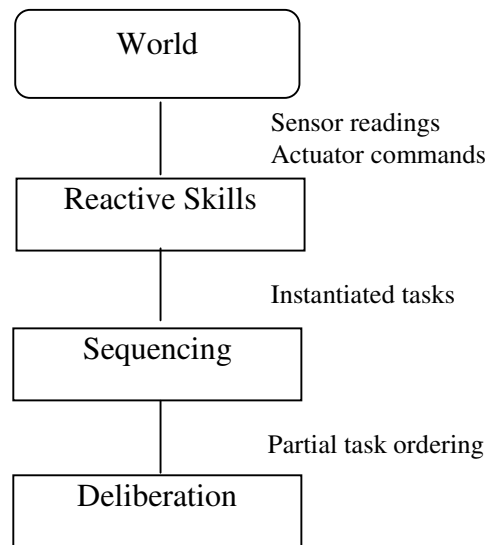


Figure 3.2: Schematic of the Three-Layer Architecture

this principle.” Hence, certain aspects of conversation, such as social conventions (e.g. question-answer), could be seen as reactive and automatic, whilst other aspects will require deliberation.

Traum (1996b) takes a similar approach in developing a reactive-deliberative agent to control the dialogue manager. Traum (1996b) states that for the dialogue they chose “a reactive approach, in which the agent is constantly making local decisions as to what to do next” partly because “timely behavior is critical: the same response can have a very different connotation if it is delayed”. When there are no local dialogue decisions to be made Traum’s (1996b) agent returns to deliberation about high-level discourse goals and domain plan negotiation. Between these two extremes he also describes a series of middle-level tasks such as planning speech acts. The balance between tasks at different levels is established by assigning priorities to different levels.

The overall architecture adopted by (Traum, 1996b) is based in the BDI tradition and speech act theory, although extended with obligations. An alternative architecture that has arisen from research in robotics is the Three-Layer Architecture (Gat, 1998). This consists of a deliberative layer, which carries-out inference in order to determine future courses of action, and a reactive layer with minimal state that can react quickly to stimuli by following predefined patterns. A key aspect of this architecture, however, is that the reactive and deliberative layers are connected by a sequencing layer. This layer defines a sequence of actions to be carried-out to achieve the goals defined by the deliberative layer. The sequencer then follows this specification in order to select which primitive behaviours the reactive layer should use at a given time. This architecture is shown in Figure 3.2. One instantiation of this architecture, namely 3T (Bonasso et al., 1995), has been used as the basis for a natural language understanding system, Dynamic Predictive Memory Architecture (DPMA) (Fitzgerald and Firby, 2000; Fitzgerald and Firby, 2001), which is situated in a robot. In DPMA goals are placed on the agenda of the middle layer both by the deliberative layer and by the speech recogniser (or other sensors) in the reactive layer. These goals are matched to reactive action packages (Firby, 1987) which attempt to provide an appropriate response (either generating an utterance or performing some action). If a goal cannot be satisfied then

control must pass to the deliberative layer to carry-out reasoning about a new course of action, which will lead to new goals being added to the middle layer's agenda.

DPMA does not employ an explicit model of dialogue (instead interpreter and generator modules are treated in the same way as sensors and actuators, with no distinction made between dialogue and other types of tasks), however, the general approach seems to fit well with models of dialogue such as conversational game theory. In this approach the deliberative layer could be seen as performing domain reasoning concerning the goals to be achieved, the sequencing layer could be seen as specifying a sequence of games required to achieve a current goal, and each of those games could be seen as determining a preset sequence of moves to be executed by the reactive layer. Hence, research on dialogue games could be combined with recent trends in agent architecture, possibly providing a basis for a more formal description of conversational game theory, in the same way that the BDI model was intended to provide a formal basis for speech acts (Pulman, 1999). Indeed much of the current work on dialogue games is within the context of multi-agent systems (e.g. Burton and Brna, 1996; McBurney and Parsons, 2002) where “game theory seems to be the predominant theoretical tool in use” and is “likely to become much more widespread in use over the coming years” (Wooldridge, 2002).

4. High-Level Dialogue Specification

4.1. Overview

As described in Section 3.3.1, finite-state network approaches to specifying dialogue, such as dialogue grammars, attempt to directly represent the linguistic structure of dialogue. The interpretation and execution of this (low-level) dialogue specification by a client system then gives rise to the actual dialogue. This is probably the most common approach taken in building dialogue systems and recently standards such as VoiceXML (W3C, 2003) have been developed to assist in the development of such systems. In addition there are various proprietary languages that support this approach, such as that developed by ITC (Falavigna et al., 1999) and described in deliverable D6 (Stefanelli et al., 2002). In generalising to multimodal dialogue, languages such as XHTML (W3C, 2000a) can be seen as providing a visual equivalent, except that spatial layout is described rather than temporal sequence. Hence, a spoken dialogue is the result of the interpretation of a VoiceXML specification by a voice-enabled browser, which controls automatic speech recognition (ASR) and text-to-speech (TTS) components to realise that specification. Similarly, a visual dialogue results from the interpretation of an XHTML specification by a visual browser, which controls visual rendering and mouse/keyboard devices to realise the interaction. In order to generate synchronised multimodal dialogues a common approach is to combine VoiceXML with XHTML to create an XML language which describes both the spoken and visual renderings of a dialogue (e.g. Niklfeld et al., 2001; Amann et al., 2001; Le Hors et al., 2001). This is also the approach taken in the HOMEY project, as described in deliverable D4 (Nardelli and Falavigna, 2002).

Whilst it is possible to author dialogues directly using low-level specification languages, it was argued in Section 3.3 that this approach is unable to handle certain dialogue phenomena, is rather inflexible, and is difficult to reconfigure for different domains. On the other hand, simply replacing the low-level description with a high-level one introduces its own problems. For example, high-level approaches such as BDI allow complex phenomena to be accounted for, but at the expense of being able to easily represent many of the phenomena that low-level approaches can handle³. The solution to this problem that has arisen recently in both research on artificial agents and dialogue systems is that both low-level (structural) and high-level (plan-based) representations are required. Crucially then, it is the “insight that pattern-directed approaches need to be *combined* with higher-level notions like plans and goals” (Hulstijn, 2000) that is important. Hence, Dahlbäck and Jönsson (1999) advocate developing approaches that find a middle ground between the conflicting demands of generality (typically provided by high-level representations) and computational efficiency (as provided by low-level representations).

The question that then arises is what information a high-level specification of dialogue should capture (Flycht-Eriksson, 1999), e.g. should it be beliefs, desires and intentions (Cohen and

³ Dahlbäck and Jönsson (1999) suggest this reflects the different approaches of the speech community on the one hand, and AI-oriented computational linguistics on the other. They suggest that the former have traditionally focused on “one-shot designs for particular systems...catering for the particular phenomena in the particular dialogue situation [rather] than on general applicability”. On the other hand, the latter have developed very general computational models, but which tend to impose a large computational overhead for practical systems, and may cater for types of dialogue that do not occur in the particular domain the system is being developed for.

Perrault, 1979), commitments (Lewin, 2000), obligations (Traum and Allen, 1994; Kreutel and Matheson, 2000) or others? The approach taken here is that the high-level specification should capture exactly those levels of description that, by general consensus (Moore, 1995), are required to account for discourse structure: namely intentional structure, information structure and attentional state. This approach is consistent with Hovy's (1993a) argument, in analogy with approaches to syntactic structure, that "the content of a discourse derives from several sources, and that a common, surface-level-ish structure is needed to house them all". In this case the high-level dialogue specification provides the common structure that he describes. A second question then arises with regard to what framework should be adopted in order to represent this common structure. The approach taken here is to adopt the framework of conversational game theory (as described in Section 3.3.3). This seems appropriate since it describes dialogue at both a plan-based level (in terms of games) and at a structural level (in terms of moves), hence bridging the gap between high- and low-level dialogue specifications. It will, however, be necessary to reconstruct the notion of games to some degree in order to accommodate all the levels of description required by the high-level specification.

The high-level dialogue specification (in terms of games) should therefore contain all the information required to generate a low-level specification (in terms of moves) for the next segment of dialogue. However, it is not intended that this specification should be authored directly. Instead, one of the aims of Work Package 6 is to investigate use of an abstract task specification, related to existing knowledge representation schemas used in medicine, as a basis for generating the high-level dialogue specification. The high-level specification is therefore related to the low-level specification on the one hand and to domain knowledge representations on the other. This is shown diagrammatically in Figure 4.1.

This distinction between dialogue specification and domain specification avoids any problems which might arise from a mismatch between representations suited to the task domain, e.g. clinical knowledge, and representations suited to language (Hovy, 1993a). For example, Dahlbäck and Jönsson (1999) distinguish two senses of the notion of 'task' as used in dialogue systems: firstly "some real-world non-linguistic activity that is directed towards achieving a particular goal" and secondly "the sequence of information that needs to be collected by the information providing system....[I]n the former case this knowledge is a separate structure, whereas in the latter it is intertwined with other aspects of the dialogue model". Dahlbäck and Jönsson (1999) therefore distinguish between underlying domain tasks (which in the approach described here are represented by the task specification) and the sequencing of dialogue-specific tasks (which are here represented by the high-level dialogue specification). Similarly, Flycht-Eriksson (2000) argues that "domain knowledge reasoning should be clearly separated from dialogue management and performed by a separate module" This is also the approach taken in the TRIPS system (Allen et al., 2001) in order to maintain portability and flexibility.

Other approaches have conflated these levels of description, e.g. (Maudet and Evrard, 1998). However, by using a single structure (a game board) to represent both the domain and the dialogue, Maudet and Evrard (1998) are forced to augment the set of dialogue moves with a series of distinct 'logical moves', which allow the player to update the game board by application of logical rules. In particular, they propose a move 'infer' which represents the application of *modus ponens* to the player's board. In their framework, such logical moves do not change the turn of the game and the same player continues to make subsequent moves until a dialogue move is made, at which point it becomes the other player's turn. Hence, these

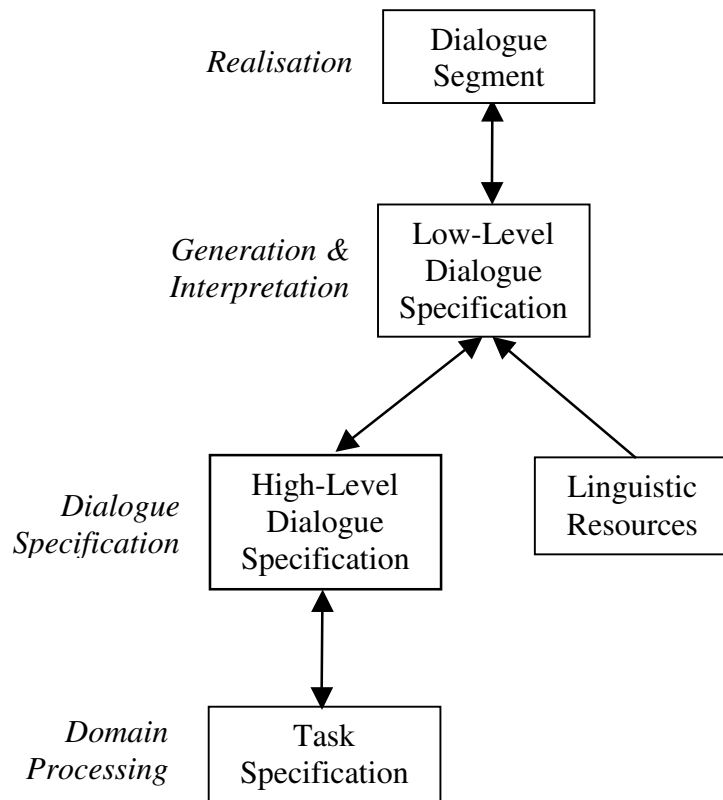


Figure 4.1: Relation of the High-Level Dialogue Specification to other Knowledge Representations

moves have a different status to the dialogue moves. Although the high-level dialogue specification described here is in some respects similar to the notion of game board in (Maudet and Evrard, 1998) the high-level specification only describes dialogue moves. Inference (e.g. applying rules that govern the system’s cognitive context, rules of rationality, or rules defining high-level strategies (Maudet and Evrard, 1998)) is instead carried-out at the domain level by the task specification. The dialogue specification then changes indirectly as a reflection of changes in the task specification and the obligations it imposes.

The notion of high-level dialogue specification adopted here is also similar in some respects to that of information state in the information state update model of dialogue (Larsson and Traum, 2000). In this approach, however, the level of description is games rather than moves (with moves instead represented by the low-level specification) and domain reasoning is delegated to a task specification layer. This architecture also has a lot in common with the TRIPS system (Allen et al., 2001) which has recently been applied to the domain of providing home-based advice for patients on following drug regimens (Ferguson et al., 2002). The abstract task specification here is similar to the TRIPS ‘Task Manager’, which controls panning and scheduling, and the high-level dialogue specification is similar to what TRIPS refers to as the ‘Behavioural Agent’. This is responsible for the overall behaviour of the system, based on the goals and obligations of the system (e.g. task execution requests from the Task Manager), interpretation of user utterances, and any exogenous events (e.g. from monitors). TRIPS also has an ‘Interpretation Manager’ and ‘Generation Manager’ (where the Generation Manager can produce speech or graphics), but these interpret and generate individual utterances, whereas the approach described here is to generate low-level specifications to be realised by a separate client.

This approach, therefore, probably has most in common with the three-layer architecture for artificial agents described earlier (Gat, 1998). For example, the low-level specification describes a finite-state machine (expressed in a mark-up language) that can handle some small unit of interaction (the sequence of utterances associated with a segment of dialogue) without need for any reasoning or planning. It is therefore a fast reactive layer which simply executes the dialogue component created by the generator on the basis of the high-level dialogue specification. The task specification, on the other hand, carries-out domain-level deliberation based on representations of the overall process to be accomplished, and specifies tasks that need to be completed at various points, e.g. acquiring values for a set of data items, negotiating a proposal. The function of the high-level dialogue specification is to bridge the gap between the reactive and deliberative components by mapping the tasks proposed by the task specification into a sequence of components that can be executed by the reactive layer. In fact, in the high-level dialogue specification these components are games to be played and the generator maps these into a VoiceXML/HTML specification of the moves to be performed by the client browser.

Note that, since the high-level dialogue specification is both modality-independent and language-neutral, some linguistic resources are required to map it to the low-level specification. In particular, a lexical ontology, such as WordNet (Felbaum, 1998), can be used to provide flexible sentence generation and interpretation. Although such lexical resources have been much used in other fields of natural language processing, such as machine translation (e.g. Mahesh and Nirenburg, 1996), text summarisation (e.g. Chaves, 2001), or information extraction (e.g. Burke et al., 1995), they do not seem to have so far been utilised to any great extent in spoken dialogue systems (but see Dahlbäck and Jönsson, 1997; Ferguson et al., 2002). Deliverable D9 (Ceusters et al., 2002) describes in detail how a lexical ontology might assist in lexical processing in dialogue systems.

4.2. Description

The previous discussion has attempted to motivate a division of the dialogue model into high- and low-level representations. The following sections describe in detail the aspects of dialogue structure (intentional, informational and attentional) which are represented in the high-level dialogue specification.

4.2.1. Intentional Structure

Intentions can easily be captured in a conversational game framework by relating them directly to the goals associated with games (Kreutel and Matheson, 2000). In this way intentions are implicitly captured by the structure of games (Maudet and Evrard, 1998), and, if such an isomorphism between intentions and games is accepted, dominance and satisfaction relations between intentions can be treated simply as relations between games.

The set of game types proposed here for describing dialogue are based on those described by (Kowtko et al., 1993; Carletta et al., 1996) and includes games whose characteristic (initiating) moves are *explain*, *instruct*, *query-yn* or *query-w* respectively. The implicit intentions associated with the games they initiate are described below. Here ICP denotes the initiating conversational partner, OCP denotes the other conversational partner, and ϕ is a variable over concepts (which may denote propositions, actions, entities etc).

```
Explain : Intend(ICP, Believe(OCP,  $\varphi$ ))  
Instruct: Intend(ICP, Intend(OCP, Done( $\varphi$ )))  
Query-yn: Intend(ICP, Intend(OCP, Know-if(ICP,  $\varphi$ )))  
Query-w : Intend(ICP, Intend(OCP, Know-value(ICP,  $\varphi$ )))
```

A game whose initial move is an *explain* move implicitly represents an intention on the part of the ICP that the OCP should update their beliefs with the proposition denoted by φ . An *instruct* game captures an intention on the part of the ICP that the OCP should carry-out the action denoted by φ . A *query-yn* game represents an intention on the part of the ICP that the OCP should intend that the ICP know if there is an instance of the entity denoted by φ , and a *query-w* game captures an ICP's intention that the OCP intend that the ICP know the value associated with the entity denoted by φ . An *explain* move would typically be realized as a statement, an *instruct* move as a command, a *query-yn* move as a yes/no question, and a *query-w* move as a wh-question. As an illustration, consider the examples given below.

```
Explain(patient-has-measles):  
  Intend(ICP, Believe(OCP, patient-has-measles))  
  
Instruct(blood-test):  
  Intend(ICP, Intend(OCP, Done(blood-test)))  
  
Query-yn(breast-nodule):  
  Intend(ICP, Intend(OCP, Know-if(ICP, breast-nodule)))  
  
Query-w(age):  
  Intend(ICP, Intend(OCP, Know-value(ICP, age)))
```

These therefore reflect forward-looking dialogue acts (Allen and Core, 1997; Core and Allen, 1997). In addition to these initiating moves, the games they initiate will also have response (backward-looking) moves. These constitute moves towards satisfying the intention underlying the game, as described below. In addition to these moves, either participant may also respond with the initiating move of a new sub-game whose intention subserves that of the parent game.

```
Acknowledge: acknowledge and signal continuation  
  
Reply-yn : yes/no reply (in response to a query-yn)  
  
Reply-w : reply supplying a value (in response to a query-w)
```

Note that, whilst obligations imposed by the task specification will provide some of these dialogue intentions – “those represented by the acquisition of agreed propositions” (Pulman, 1997) – others will arise as a result of the dialogue itself in terms of obligations imposed by the user, e.g. to reply to a clarification request. These “communicative subgoals may also arise locally in the dialogue because of unanticipated responses and because of the complexity of the perceptual, understanding, evaluation, and other cognitive processes involved in interpreting and generating communicative behaviour” (Bunt, 1996). Moreover

“it is not plausible to assume that such moves are planned: rather, they arise as an immediate response to the current state of the dialogue” (Pulman, 1997). These obligations imposed by the user will also lead to particular intentions in the dialogue system, which must then be balanced with those deriving from obligations imposed by the task specification (this approach of deriving intentions from obligations is similar to (Kreutel and Matheson, 2000)).

In balancing the demands from the task specification and user, it is generally assumed that obligations imposed by user moves should be processed before those derived from the task specification (Traum and Allen, 1994; Traum, 1996a; Traum, 1996b). Hence, the system must respond to clarification questions or meta-level questions by the user before it can continue to pursue domain goals. These clarifications or side sequences are handled in the conversational game framework by nested sub-games, where the clarification is seen as being embedded in the original game since it contributes to achieving the parent goal. An example of this is given below.

S: Does the patient have any skin changes?	<i>Query-yn</i>]
U: What do you mean?	<i>Query-w</i>	
S: I mean, do they have any of the following: breast cyst, nipple eczema or abscess?	<i>Reply-w</i>	
U: Oh right... they have a breast cyst.	<i>Ack</i>	
S: Ok.	<i>Reply-yn/Reply-w</i>	
	<i>Ack</i>	

In this case, the task specification initially imposes an obligation on the dialogue system to determine whether there are any skin changes. The dialogue system therefore has the *intention* to ask the user whether there are any skin changes, which is represented by proposing a *query-yn* game on this topic. This sets-up the expectation for a *reply-yn* move by the user, but instead the user initiates a new *query-w* game whose aim is to elicit a clarification of the original query. Since the goal of this game is to assist in the satisfaction of the original system goal, it is seen as a nested game. This move by the user then imposes an obligation on the dialogue system to provide a reply, and since this takes precedence over any outstanding obligations from the task specification the system derives the intention to explain the set of possible replies to the user, hence obeying the nested structure. Once the clarification game is completed, the user then makes a reply that satisfies the intention underlying the original *query-yn* game (as well as providing some extra more specific information, namely the type of skin change) hence, the original obligation imposed by the task specification has been met.

Clarifications may also be raised by the system itself. For example, if the user’s reply to a query was not sufficient for the system to meet the obligations defined by the task specification. For instance, consider the dialogue given below.

S: What skin changes does the patient have? [expected reply is: cyst, nodule etc...]	<i>Query-w</i>]
U: They have a lump	<i>Reply-w</i>	
S: What type of lump?	<i>Query-w</i>	
U: It’s a cyst	<i>Reply-w</i>	
S: Ok.	<i>Ack</i>	

In this case the task specification has imposed an obligation to acquire information on the skin changes the patient has, and requires the returned value to be one or more of cyst, abscess or nodule. The dialogue system therefore has the *intention* to ask the user what skin changes the patient has. The user responds with the expected *reply-w* move but the value specified is not in the range required. On the other hand, the supplied value is related to the items in the required range (it is just not specific enough) so it is probably the case that the user is making a reply move and not initiating a sub-game. In order to meet the obligations set by the task specification, the dialogue system must initiate a clarification sub-game in order to try to obtain an appropriate value. Since the user's reply was too general, the system derives the intention to ask the user to give a more specific reply, which is then satisfied, hence fulfilling the system's obligations.

This last example shows that nested sub-games may arise not only as a result of obligations imposed directly by the user in the form of questions, but also as a result of the user not providing sufficient information for the dialogue system to fulfil the obligations defined by the task specification. In fact, these may compound each other, leading to deeper levels of nesting as in the example below.

S: What skin changes does the patient have?	Query-w	}
U: They have a lump	Reply-w	
S: What type of lump?	Query-w	
U: What do you mean?	Query-w	
S: I mean, is it an abscess, cyst or nodule.	Reply-w	
U: Oh right...	Ack	
It's a cyst	Reply-w	
S: Ok.	Ack	

Examples such as this suggest that the dialogue system needs to keep a stack of intentions, separate from those derived from the task specification, to handle nested sub-games. In this case, the system should process the intentions on the stack one-by-one until empty, before processing intentions derived from the task specification.

Recent dialogue systems such as the GoDiS system (Larsson, Ljunglof, Cooper, Engdahl and Ericsson, 2000) also make a similar distinction to that employed here. They distinguish a PLAN structure, which specifies pre-planned dialogue actions, and an AGENDA structure, which contains short-term goals or obligations that might arise, for instance, from the user raising a question.

4.2.2. Information Structure

The information structure of discourse was described in Section 3.1.3 as being “imposed by domain relations among the objects, states and events being discussed” (Moser and Moore, 1996, p. 416). Dahlbäck and Jönsson (1997) similarly argue that task-specific knowledge must be augmented with a conceptual model that describes general information concerning the relationships between objects in a domain. For example, in a library system they suggest a conceptual model in which ‘book is-a publication’, ‘author is-aspect-of publication’ etc. It is assumed here that such information should be derivable from the domain concepts and relations specified in the task specification, and should complement the intentional structure. It is clear, however, that such domain-specific relations will be different from the domain-

independent relations required for language, hence motivating a separate level of representation.

One of the problems associated with applying information relations to dialogue is determining the appropriate units that such relations should apply to. In text generation, they are applied to successive utterances, but in dialogue successive utterances may be made by different participants, i.e. they may be different turns in the conversation. Furthermore these turns are linked to each other in terms of their functions, e.g. question-answer, explain-acknowledge, as represented by conversational games. It might therefore seem, as Stent (2000) points out, that the simplest approach would be to only describe informational relations within turns and use conversational games to describe relations between turns. Stent goes on, however, to point out that this is not sufficient, as there are in fact many cases of information relations spanning turns. For example, consider the two dialogues given below (adapted from Stent, 2000).

[1]	(a) A: So that takes care of the ill guy.	<i>Explain</i>	}	<i>EXPLAIN</i>
	(b) B: Okay...	<i>Ack</i>		
	(c) And that takes two hours.	<i>Explain</i>	}	<i>EXPLAIN</i>
	(d) A: Right.	<i>Ack</i>		
[2]	(a) A: First they can take out the power.	<i>Explain</i>	}	<i>EXPLAIN</i>
	(b) B: Right...	<i>Ack</i>		
	(c) And then we have to wait.	<i>Explain</i>	}	<i>EXPLAIN</i>
	(d) A: Yup.	<i>Ack</i>		

Example [1] above illustrates a cross-speaker elaboration relation in which (a) and (b) form an Explain game initiated by participant A, and (c) and (d) form a second Explain game initiated by participant B, and the propositional content of (c) is in an elaboration relation with that of (a). Similarly, example [2] above illustrates a cross-speaker sequence relation in which (a) and (b) again form one game and (c) and (d) form another and (c) is in a sequence relation to (a). Both the above examples therefore involve relations between statements made by different speakers and in different games. Stent (2000) explains that “in a DAMSL-tagged set of 8 dialogs in our corpus, 40% of the utterances were statements, and many of these appeared in sequences of statements. The relationships between many of these statements are unclear without a model of rhetorical structure”. On the basis of such examples, it might appear that information relations should be defined between the initiating moves of games. However, Stent (2000) also describes relations involving question-answer pairs, such as that given below (adapted).

(a) A: We have to send buses to the lake.	<i>Explain</i>	}	<i>EXPLAIN</i>
(b) B: Ok...	<i>Ack</i>		
(c) How many are we sending?	<i>Query-w</i>	}	<i>QUERYW</i>
(d) A: Two.	<i>Reply-w</i>		

In this example the topic of both utterances (c) and (d) (i.e. the whole QUERYW game) is in an object-attribute elaboration relation with utterance (a) (which introduces the topic of the EXPLAIN game). This suggests that the appropriate level for defining these information relations is that of conversational games, as indicated in Figure 4.2. That is to say that informational relations arise as a result of domain relations between the topics of games, and

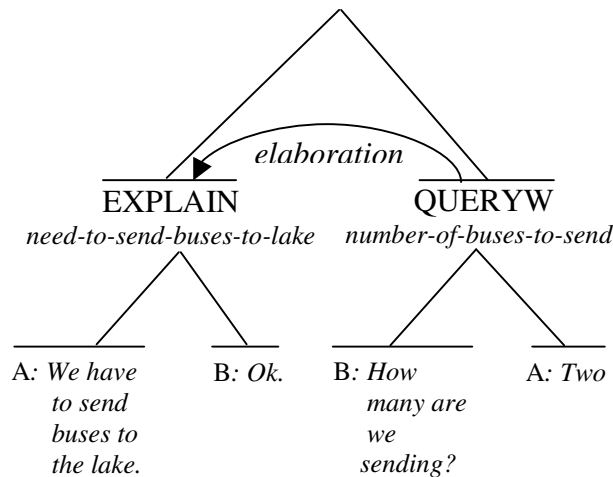


Figure 4.2: Sample analysis of the informational structure of a dialogue.

it is this that leads to cross-speaker relations because subsequent games may be initiated by different speakers.

A further example is given below in the context of a medical dialogue system which is trying to determine whether a patient with suspected breast cancer should be referred to a specialist or not.

(a) S: Is there any nipple discharge?	Query-yn] OUEYYN
(b) U: Yes	Reply-y	
(c) S: Ok...	Ack] OUEYYN
(d) And is it bloodstained?	Query-yn	
(e) U: No	Reply-n	
(f) S: Ok.	Ack	

In this example, the second QUERYYN game (utterances d, e and f) seeks to elaborate the information provided in the first game. This relation arises because the topic of the second game (bloodstained nipple discharge) elaborates the topic of the first game (nipple discharge). This relation is important for the purposes of dialogue management because it is the basis for the selection of the cue word “and” in utterance (d), and also licenses the use of an anaphor to refer to the topic being elaborated. As will be described in the next section, it is also important because it can also be used to determine the order in which games are played, in order to ensure the semantic coherence of the dialogue.

A similar aspect of the high-level dialogue specification, which, however, is not directly linked to the notion of information structure, is an ‘upper model’ classification of concepts under discussion (Bateman, 1990). The notion of an upper model is a set of linguistically-motivated domain-independent concepts that are appropriate for language generation, hence an interlingua between domain concepts and their grammatical expression. It is possible to “achieve the necessary link between particular domain knowledge and the upper model by having an application *classify* its knowledge organization in terms of the general semantic categories that the upper model provides” and “the text generation system is then responsible for realizing the semantic types of the level of meaning with appropriate grammatical forms” (Bateman, 1990). The crucial point about such a model is that the level of abstraction should

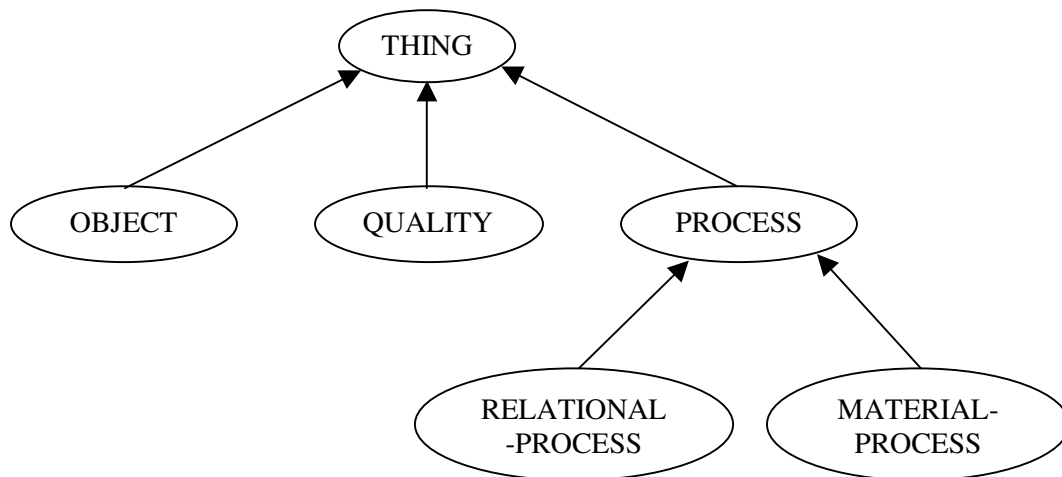


Figure 4.3: An Example Upper Model Organisation

be “sufficiently high that it generalizes across syntactic alternations, without being so high that the mapping between it and surface form is impossible to state” (ibid).

An example upper model from (Bateman, 1990) is shown in Figure 4.3: An Example Upper Model Organisation. It can be seen that the concepts in this figure are able to subsume concepts from various domains, e.g.

```

OBJECT = {system, ship, heart, ...}
QUALITY = {faulty, new, inoperable, ...}
RELATIONAL-PROCESS = {connects, coordinates, ...}
MATERIAL-PROCESS = {sail, discharge, ...}
  
```

Each upper model concept also licences a particular set of realisations in the text generator, although there is not a one-to-one mapping from upper model concepts to realisations. For example, the following specification, using only upper model terms, specifies that there is a directed action involving ‘discharge’ which acts on the substance ‘electricity’, and a non-directed action involving ‘breakdown’ performed by the object ‘system’, and these two processes are in a cause-effect relation, i.e. the discharge action causes the breakdown action.

```

((c0 / cause-effect
  :domain discharge
  :range breakdown)
 (discharge / directed-action
  :actee (electricity / substance))
 (breakdown / nondirected-action
  :actor (system / object)))
  
```

By assigning concepts to differently ranked heads in the sentence grammar, a variety of surface realisations is possible, such as those given below (as well as other variations such as questions, elliptical forms etc.)

```
Electricity being discharged resulted in the system breaking down.  
Because electricity was discharged, the system broke down.  
Because of electricity being discharged the system broke down.  
Electricity was discharged causing the system to break down.  
...
```

This approach of classifying domain concepts in such a way as to support language generation seems very useful for a dialogue system as well as text generation systems. Hence such upper model classifications of the topics of games are included as part of the high-level dialogue specification.

4.2.3. Attentional State

The high-level dialogue specification needs to provide a description of attentional state in addition to intentional and information structure. In particular, since there are likely to be several playable moves at any point in the dialogue, the dialogue specification should indicate which is the preferred move. Maudet and Evrard (1998) describe this “choice of the move that should be played within possible moves” as the “strategy problem”. They suggest that strategy is game-independent, depending instead on agent behaviour and goal-dependent heuristics, and so they define a function f_{strat} , which yields a totally ordered set δ_{strat} of preferred moves, from which the highest element m is chosen as the next move. Maudet and Evrard (1998) do not specify the function f_{strat} but clearly it must take into account the constraints imposed by the intentional structure (in order to conform to the underlying task) and possibly relations in the information structure (in order to ensure semantic coherence in the sequence of moves). For example if the intentional structure specifies that $I_1 SP I_2$ then I_2 should not be chosen until I_1 is satisfied. Similarly, if $I_1 DOM I_2$ and $I_1 DOM I_3$ then it makes sense to address I_2 and I_3 together as they satisfy a common goal. This can be achieved in the spoken modality either by presenting them in succession, or by aggregating them, e.g. “what are the patient’s age and sex?”⁴. Similarly, in the visual modality, they might appear grouped together. In the approach taken here, the choice of next move is made by marking a particular game as *focused* in the high-level dialogue specification. The move associated with this focused game then corresponds to m in (Maudet and Evrard, 1998).

It is clear, however, that the generation component will need to know more than just what the next move is, it will need to know the whole set of games that are playable by either participant – in Maudet and Evrard’s (1998) terms the set of conventionally and rationally acceptable dialogue moves. The problem is that, given the notion of conversational game as a fairly small unit, namely a predefined sequence of moves, as described in Section 3.3.3, then a conversation may clearly involve the playing of several such games simultaneously. For example, in mixed-initiative dialogue, the system may aggregate several initiating moves, as described above, and, similarly, the user may aggregate several reply moves in a single utterance, e.g. “the patient is thirty-five, female with a breast cyst”. This can be supported by using the full set of playable games to define an appropriate language model for the speech recogniser⁵ or to generate a visual form containing all possible replies.

⁴ Although this is not generally done in spoken dialogue systems since, in general, the more complex the prompt, the more complex the user’s reply, which means developing the language model for the speech recogniser becomes much more difficult.

⁵ An untrained speech recogniser is assumed here as the focus of the HOMEY project is on telephone call centre-based services. If the service is only to be available to a small group of users then another possibility is to

The notion of attentional state required here is therefore different to the approach in (Grosz and Sidner, 1986). They describe a stack of focus spaces, each of which maps an intention onto a dialogue segment, with the topmost element representing the current dialogue segment, and with focus spaces only added to the attentional state when needed and removed when the associated dialogue segment is complete. The approach taken here is instead to have an attentional state in which all possible games are represented simultaneously so that the participants may make moves in several games in parallel and move between games (this approach is also taken in (Burton and Brna, 1996)).

In addition, it will be useful to distinguish those games in the attentional state that are currently playable by the system, and those which the system is not currently planning to play but that the user might. In Maudet and Evrard's (1998) terms, the former is a similar notion to the subset δ_{strat} of preferred moves for the system, whilst the latter would relate to the set of moves playable by either participant that are not in the set of preferred moves for the system. In the approach taken here, this distinction is captured by marking games as *foreground* if they are playable by the system and as *background* otherwise. Note that this distinction also captures constraints imposed by satisfaction-precedence relations in the intentional structure. For example, if intention I_1 satisfaction-precedes intention I_2 , then the associated game G_1 that (implicitly) represents the intention I_1 will be marked as foreground and the game G_2 that represents I_2 will be marked as background, hence G_1 is playable by the system but G_2 is not. This means that satisfaction-precedence relations need not be represented explicitly (for the interpreter/generator), but rather can be captured indirectly by virtue of their effect on attentional state. Games may also be marked as foreground/background for other reasons than intentional relations, e.g. if a game has already been played then it will no longer be in the set of playable games for the system (although it will be for the user as they may wish to re-play it at a later point to correct errors).

As mentioned earlier, information relations also play a part in the strategy for choosing the next move. This is necessary in order to preserve dialogue coherence by ensuring that the next move made by the system is as semantically relevant as possible to previous moves in the dialogue history. For example, suppose the high-level dialogue specification describes an intentional structure such as that below:

```
I1 = Intend(S, Intend(H, Know(S, CLINICAL DETAILS)))  
I2 = Intend(S, Intend(H, Know-if(S, NIPPLE DISCHARGE)))  
I3 = Intend(S, Intend(H, Know-if(S, BLOODSTAINED NIPPLE DISCHARGE)))  
I1 DOM I2  
I1 DOM I3
```

From this it can be inferred that I_2 and I_3 must both be satisfied in order for I_1 to be satisfied, but it does not specify which of I_2 and I_3 should receive focus first in the attentional state. Suppose however that the high-level dialogue specification also describes an information structure with a relation such as that below:

train the speech recogniser on each user, hence avoiding the need for a language model. The problem noted above will still hold for the visual modality, however.

I_3 ELABORATE I_2

In this case, it can be seen that I_2 is the nucleus of an elaboration relation of which I_3 is the satellite, and hence I_2 should receive focus first in order for the dialogue to be coherent. Furthermore, if at any point in the dialogue the user takes the initiative and addresses I_2 then the next application of the move selection strategy should yield an attentional state such that the game representing I_3 is focused (so that its associated move is the next one selected). This would allow the system to shift to the topic introduced by the user, and hence preserve dialogue coherence. As an example, suppose the foreground games in the attentional state at some point in the dialogue are represented by the following totally-ordered set of initiating moves and associated topics: $\langle \text{query-}w(\text{age}), \text{query-}w(\text{cyst}), \text{query-}w(\text{discharge}), \text{query-}w(\text{bloodstained-discharge}) \rangle$, and assume also an elaboration relation between $\text{query-}w(\text{bloodstained-discharge})$ and $\text{query-}w(\text{discharge})$. The default focused game would be $\text{query-}w(\text{age})$, hence the following initial prompt by the system:

S: What is the patient's age?

Suppose that the user then takes the initiative and not only answers the question posed by the system but also provides additional information that the patient has nipple discharge. This could arise because, unlike the system plan, the user's plan was to introduce this topic as early as possible in the conversation. The dialogue history might then appear as:

S: What is the patient's age?
U: They're thirty and they have severe nipple discharge
S: Ok.

Since the user has made reply moves to both the $\text{query-}w(\text{age})$ and $\text{query-}w(\text{discharge})$ moves, their associated games are completed and hence marked as background (meaning that the system will not re-initiate them). This process of matching user moves to games in the attentional state is referred to here as move accommodation, following (Larsson, Cooper and Engdahl, 2000; Larsson, Ljunglof, Cooper, Engdahl and Ericsson, 2000).

The set of foreground games at this point would now be represented by the following set of initiating moves: $\langle \text{query-}w(\text{cyst}), \text{query-}w(\text{bloodstained-discharge}) \rangle$, which, by default, would suggest playing the game initiated by the $\text{query-}w(\text{cyst})$ move next. However, the game strategy should take account of the fact that there is an elaboration relation between the foreground game initiated by $\text{query-}w(\text{bloodstained-discharge})$ and the background game whose initiating move was $\text{query-}w(\text{discharge})$. Hence, in order to maintain dialogue coherence, the next move should be $\text{query-}w(\text{bloodstained-discharge})$, causing the following prompt by the system:

S: What is the patient's age?
U: They're thirty and they have severe nipple discharge
S: Ok...
And is the nipple discharge bloodstained?

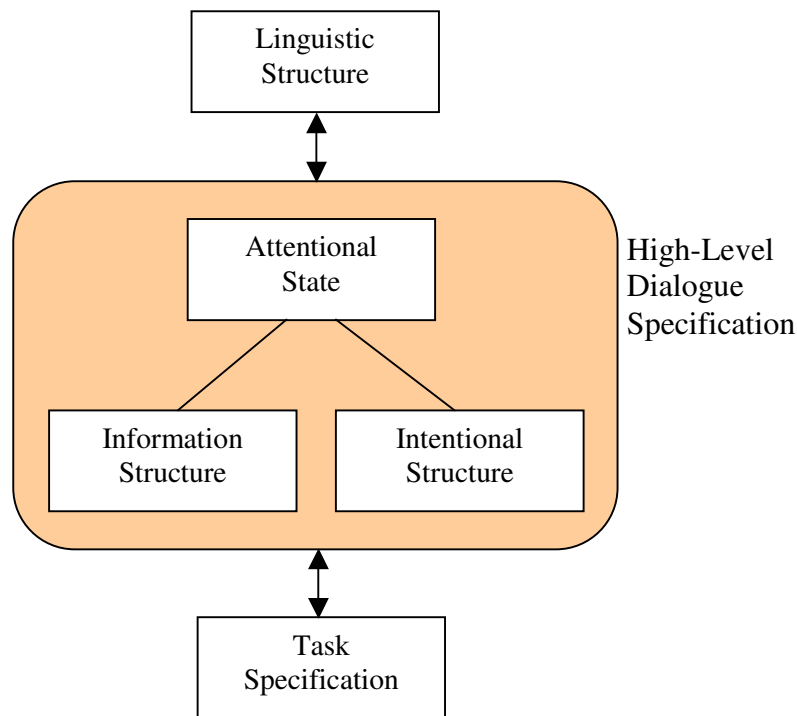


Figure 4.4: Relation between the three types of information in the high-level dialogue specification

Once the user replies to the question the only remaining game for the system to play is the one whose initiating move is *query-w(cyst)*, hence the full dialogue history might look like the example given below.

```
S: What is the patient's age?
U: They're thirty and they have severe nipple discharge
S: Ok...
   And is the nipple discharge bloodstained?
U: No.
S: Ok...
   Does the patient have a cyst?
...
```

Note that if information relations were not taken into account by the game strategy in this way then the above dialogue would have appeared as in the example below. In this case the user's introduction of the topic of nipple discharge is ignored by the system until questions concerning that topic arise in its own plan (and, in fact, that could be much later in the conversation than is shown in this example):

```
S: What is the patient's age?
U: They're thirty and they have severe nipple discharge
S: Ok...
   Does the patient have a cyst?
U: No.
S: Ok...
   Does the patient have bloodstained nipple discharge?
...
```

As can be seen from the previous discussion, the attentional state draws on both intentional and informational relations in order to determine which games can be initiated by the system (foreground) or by the user only (background) and which of the foreground games should be chosen next (focused). It is clear, however, that the strategy employed in the attentional state may be influenced more by intentional or information structure at different times. In particular, it is assumed here that intentional structure overrides information structure so that if the underlying obligations dictated by the domain tasks give rise to a satisfaction-precedence relation e.g. $I_1 SP I_2$, then I_1 should always be satisfied before I_2 even if that conflicts with information relations. On the other hand if intentional relations do not specify a strict ordering then information relations will need to be relied on in order to create a coherent ordering.

These two approaches to game strategy (predominantly intentional and predominantly informational) are often associated with different types of dialogues (Dahlbäck and Jönsson, 1999). For task-oriented dialogue, it is clear that the system must have a representation of the shared non-linguistic tasks to be accomplished and the order in which they should be performed, and so the dialogue will be driven primarily by intentions derived from these underlying tasks. For information-seeking dialogues, however, the system need not have a complex task structure but merely a definition of the set of information that needs to be collected at a particular point. The sequence in which the different items of information are collected (and hence the dialogue structure) will be derived primarily from information relations between those items. The model proposed here for the high-level dialogue specification can therefore support both types of dialogue, possibly interleaved.

4.3. Specification Language

The previous sections have described the various types of information that the high-level dialogue specification should represent (as shown in Figure 4.4) and how each might be represented within a conversational game framework. This section summarises the requirements for a game-based representation of the high-level dialogue specification. An XML language that supports the creation of such specifications is briefly described in Appendix A, along with a discussion of how it can be mapped onto low-level specification languages.

4.3.1. Transactions

The notion of conversational games typically employed in dialogue can only represent local coherence of dialogue – typically restricted to one initiative-response unit. In order to account for the structure of a whole dialogue, Maudet and Evrard (1998) note that one approach has been to add more structure to the game board in order to track certain aspects of the dialogue structure. For example, Ginzburg (1997) makes use of a partially ordered set of ‘questions under discussion’. This approach also seems similar to that taken in the Information State Update approach to dialogue, where the notion of game board is replaced by an information state that can have complex internal structure (e.g. Larsson, Cooper and Engdahl, 2000).

Maudet and Evrard (1998) argue, however, that an alternative approach is to treat games as primitives that can then be manipulated at a higher level. This is the approach adopted here, where games are primarily derived from a task specification. The structure of the dialogue at any point is therefore defined by a game structure which contains all the possible games that can be played (based on the task specification). This allows multiple games to be played in

parallel, and allows the user to interrupt the system plan and move to a different game (in which case the system should choose its next move to maximise relevance with the user's, as described previously). Moreover, the need to represent intentional dominance relations between games suggests that this specification should be a tree structure, similar to the approach taken in (Burton and Brna, 1996) which also represents dialogue as a tree of parallel games.

This tree of games will be referred to as a transaction, based on Carletta et al.'s (1996) proposal of transaction as a level of dialogue organisation above that of games. In their analysis a transaction was a sequence of games which accomplished some substantial part of the dialogue – substantial enough, in fact, that conversational participants never recursively nested them (presumably because the cognitive load of nesting such large structures was prohibitive). They found instead that if problems arose during a transaction then participants would start the transaction again from the beginning, unlike games where they would typically initiate sub-games to repair the problem. In the context of a medical domain, the care pathway for a patient may involve combining several guidelines (e.g. patient referral, chemotherapy guidelines, psychological support etc.) each of which may involve separate dialogues. Importantly, these dialogues would generally be fairly substantial and so would occur in sequence (it is unlikely that one would want to start a conversation regarding patient referral then switch to a discussion on chemotherapy administration and then return to the original position in the patient referral conversation). Each of these dialogues might therefore be seen as transactions within an overall conversation regarding provision of care for a patient.

4.3.2. Games

As described above, games implicitly represent intentions in the high-level dialogue specification, and the hierarchical relations between games in a transaction capture intentional dominance relations. As discussed in Section 4.2.3, satisfaction-precedence relations need not be explicitly represented. Instead a game must allow for the representation of attentional information such as the scope (foreground or background) of the game. This foreground/background distinction can be useful for the generation component so that it can determine whether different games should be displayed differently in the visual form. For example, games that are background (and have not already been played) should perhaps be disabled on a visual form, until other information has been gathered which causes them to become foreground. Similarly, dominance relations might assist the generation component in grouping games so that games that have a common parent goal tend to be played in succession (or aggregated). Additionally it must be possible to specify whether a game is focused or not so that the system can determine which game to initiate. Obviously, only one game can be focused in any one turn of the conversation.

Whilst a notion of 'game' alone is sufficient to represent attentional information, in order to capture other aspects of games it is necessary to for there to be some internal structure. The most obvious aspect of this internal structure is the move associated with a game.

4.3.3. Moves

Every game must contain a move. The initiating move of a game is referred to as its characteristic move, e.g. a game that starts with a request move is a request game. The set of

initiating moves assumed here is as follows (based on Kowtko et al., 1993; Carletta et al., 1996):

- Initiating (Forward-Looking)
 - *Open*: open the dialogue (greeting)
 - *Close*: close the dialogue (farewell)
 - *Explain*: provide some information to the user
 - *Instruct*: provide instruction
 - *Query-yn*: ask a yes/no question
 - *Query-w*: ask a complex question
- Response (Backward-Looking)
 - *Open*: acknowledge opening of dialogue
 - *Close*: acknowledge closing of dialogue
 - *Acknowledge*: acknowledge explanation/instruction and signal continuation
 - *Reply-yn*: reply to a yes/no question
 - *Reply-w*: reply to a complex question
 - *Reply-part*: partially reply to a question (e.g. an overly general response)

The only move used here that is not used in (Kowtko et al., 1993; Carletta et al., 1996) is ‘*reply-part*’. This is used to indicate that the user made a move that could be interpreted as addressing the associated game, but the reply was not precise enough to be sure (e.g. the notion of an under-specified reply discussed in deliverable D9 (Ceusters et al., 2002)). This imposes an obligation on the system to request clarification of what the user meant.

In addition, a *query-w* move on the part of the system requires some additional information to be represented regarding the move: namely, what value-type the associated reply should have. Whilst this is clear for a *query-yn* move (for which the reply type is Boolean) the *query-w* move sets-up an expectation for a complex reply that could be of two types: a concrete domain such as ‘number’, ‘date’ etc (Nardi and Btchman, 2002), or an enumerated type such as {male, female}. Hence, a *query-w* move must have a domain associated with it, where the domain can be either a reference to a concrete domain name (e.g. ‘number’) or a set that enumerates all possible values. In the case of an enumeration, the role that relates the values in the enumerated set to the topic of the game should be specified. This can help in subsequent clarifications where it is useful to know what relation to the topic the system expected the user’s response to have. Furthermore the definition of an enumeration should indicate whether the values specified are mutually exclusive or not in order to determine whether or not the user can supply multiple values in their reply.

Similarly, a *reply-yn* or *reply-w* move must have a value associated with it that constitutes the actual content of the reply. For these two moves, this is simply a value that belongs to the domain specified by the original *query-yn* or *query-w*. In the case of a *reply-part*, however, the reply is under-specified and so a representation of the actual concept referred to by the user reply should be included in the response (to assist in subsequent clarification by the system). For example:

```
[1] S: What skin changes does the patient have?  
      [expected reply is cyst, nodule,...]  
[2] U: They have a lump.  
[3] S: What type of lump?
```

In this case the system expected a reply such as ‘cyst’, ‘nodule’ etc. but the user replied with the more general term ‘lump’. Whilst this is clearly intended as a reply to the question, the representation of this reply should specify that the user indicated only that there was a lump, (which cannot be taken as implying that there is, for example, a cyst or nodule). This information can then be used in the subsequent clarification: “what type of lump?”

4.3.4. Topics

The topic of a game specifies the concept that all moves in the game relate to, e.g. ‘organism age state’, ‘nipple discharge’ etc., and therefore provides a way of capturing aspects of information structure. Individual moves in the game constitute requests, replies, explanations etc. in the discussion of the topic. In addition to the topic concept itself, it is also useful, for the purposes of sentence generation, to include an upper model classification, or category, of the topic (as described in Section 4.2.2). This can be represented as an attribute of the topic.

4.3.5. Relations

As described in Section 4.2.2, information relations can contribute to determining the order in which games should be played, e.g. to ensure that the next move is semantically related to the previous move or moves (within the constraints set down by the intentional structure). Information relations are also useful for sentence generation, e.g. if there is an information relation defined between two games then cue words (e.g. “and”, “then”, “although” etc) can be chosen to indicate that relation to the user in speech. Similarly, visual layout (such as indentation) can be used to represent such relations on a form. The game structure should therefore contain a representation of relations between games. This can be achieved by adding a specification to the definition of the subordinate game, or ‘satellite’ of the relation (Mann and Thompson, 1988). This specification should describe the type of the relation (e.g. elaboration), and indicate which game is the head of the relation, or ‘nucleus’ (Mann and Thompson, 1988).

5. Abstract Task Specification Language

5.1. Overview

An important aspect of the approach described in this deliverable is that dialogue games are treated as primitives that are manipulated by higher-level knowledge representations. Maudet and Evrard (1998) have distinguished this approach from models that instead add more structure into to game board itself – similar in some respects to recent information state update models of dialogue (e.g. Larsson and Traum, 2000). In this case the higher-level knowledge representation is supplied by an abstract task specification for a particular domain, and the dialogue specification described in the previous section is then derived from this task specification. The abstract task specification must therefore provide information of two distinct functional types: information on what task is to be accomplished (*domain plan*), and information on the concepts associated with that task and their relations (*domain ontology*). This approach, shown diagrammatically in Figure 5.1, is consistent with that taken by (Dahlbäck and Jönsson, 1999; Flycht-Eriksson, 2000) who suggest a modular architecture with separate modules for the dialogue model, the task model and domain knowledge.

The abstract task specification is derived from the domain plan and ontology through the mediation of an abstract task specification language, which implements an interface or façade for the underlying knowledge representations. The module that constructs the high-level dialogue description, here referred to as the *dialogue specification engine*, can retrieve the task specification at any point in the dialogue via the messages supported by the task specification language.

Note that the two functional components, domain plan and ontology, need not be actually implemented by separate physical components – they could be provided by a single system – however, it is useful to view them as separate at the logical level. Furthermore, although the notions of domain plan and ontology adopted here are based on specific technologies available within the HOMEY project (CRUK's process specification language *PROforma* and L&C's Ontology Browser, see Appendix B), one of the aims of providing a task specification language is to allow different specific technologies to be used to implement the functionality required by the dialogue specification engine. Hence the implemented system should be re-configurable in two senses:

1. It should be easy to change the domain of the dialogue by providing a new plan and, possibly, ontology (depending on the extent to which it is situated in a particular domain).
2. It should be possible to change the underlying technologies (*PROforma*, L&C ontology etc.) by providing an implementation of the task specification language described here for the new components.

Note also that the specification language itself can be implemented by any physical mechanism desired, e.g. messages passed on a socket connection, SOAP messages across HTTP, method calls on objects or components, and so on. An example implementation using instance methods on a Java class is outlined in Appendix B, along with a description of how the domain plan and ontology might be implemented using *PROforma* and the L&C Ontology Browser respectively.

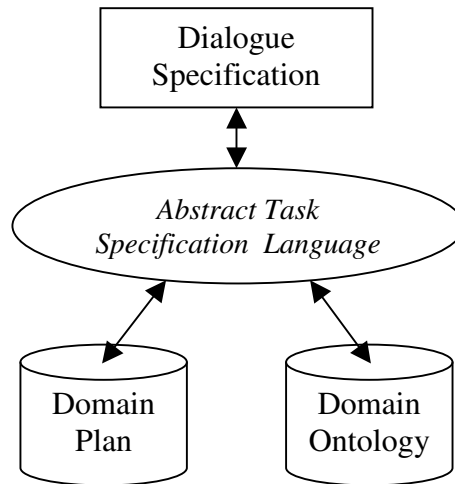


Figure 5.1: Components of the abstract task specification

5.2. Description

5.2.1. Domain Plan

Since the domain plan determines the overall process to be followed and the individual tasks required, it can quite naturally be seen as the basis for deriving the intentional structure of a dialogue concerning that domain. In fact, the task specification may be seen as imposing certain obligations on the dialogue system in order that the plan execution system can achieve successful completion of the plan. These obligations will then give rise to intentions on the part of the dialogue system to engage in particular dialogues with the user in order to meet those obligations. Such an approach is consistent with claims that dialogue structure is largely determined by task structure (Grosz and Sidner, 1986), or to put it another way: “engaging in a dialogue is typically not a goal in itself, but is motivated by some underlying task or goal one wants to achieve, and for which the dialogue is instrumental” (Bunt, 1996).

Furthermore, intentional relations such as those described in (Grosz and Sidner, 1986) can be derived directly from relations between tasks in the domain plan (Young and Moore, 1994). For example, task preconditions in the plan can be considered to give rise to satisfaction-precedence relations in the intentional structure. Hence if task T_2 has preconditions such that it cannot be started until task T_1 has completed then a satisfaction-precedence relation can be inferred between I_1 (the dialogue intention that derives from the obligation imposed by T_1) and I_2 (the dialogue intention that derives from the obligation imposed by T_2) such that $I_1 SP I_2$. Such dependencies can also arise more indirectly through the data flow rather than control flow of the plan. For example, if task T_2 has preconditions such that it cannot be started until a data item has a (particular) value and the goal of task T_1 is to acquire a value for that data item, then a satisfaction-precedence relation, $I_1 SP I_2$, can again be inferred (where I_1 and I_2 are defined as before).

Dominance relations can similarly be inferred from task decomposition. For example, if task T_1 is decomposed into tasks T_2 and T_3 (i.e. T_1 is an abstract task whose implementation is provided by T_2 and T_3) then the satisfactions of I_2 and I_3 (the dialogue intentions that derive from obligations imposed by T_2 and T_3 respectively) each form part of the satisfaction of I_1 (the dialogue intention that derives from the obligation imposed by T_1). Hence, a dominance

relation can be inferred between I_1 and I_2 & I_3 , such that $I_1 \text{ DOM } I_2$ and $I_1 \text{ DOM } I_3$. More precisely, task decomposition yields *immediate* dominance, which defines a total ordering over games, rather than the partial ordering described in (Grosz and Sidner, 1986). Immediate dominance, notated as DOM_1 , is defined as follows: $I_i \text{ DOM}_1 I_k$ iff $I_i \text{ DOM } I_k$ and there is no I_j such that $I_i \text{ DOM } I_j$ and $I_j \text{ DOM } I_k$. Hence, in the example above, the relevant relations are $I_1 \text{ DOM}_1 I_2$ and $I_1 \text{ DOM}_1 I_3$.

Note that, consistent with the desire to find a middle ground between the generality of AI-oriented approaches and computational efficiency (Dahlbäck and Jönsson, 1999), only a plan execution system is assumed here rather than a full-blown AI planner (e.g. Young and Moore, 1994; Freedman, 2000 and other). This is also motivated by the medical domain that this approach is based in, where the domain plans are representations of best-practice guidelines determined by committees of experts. Hence the plan is expected to be followed in a deterministic fashion. On the other hand approaches such as simple task hierarchies (e.g. Gorin et al., 1997) seem insufficient as they only capture dominance relations between tasks and not satisfaction-precedence relations, yet the latter are important in task-oriented dialogue (e.g. *first* administer drug X *then* take the patient's blood pressure).

In summary, the domain plan must provide information to the dialogue specification engine regarding the tasks that currently need to be accomplished and the relations between them. In particular, the specification engine must know if one task is part of the decomposition of another or is dependent on another as a result of task preconditions so that it can determine what intentional dominance and satisfaction-precedence relations there should be in the high-level specification. In addition, it must be possible for the specification engine to update the state of the domain plan, e.g. to set the value of a data item that has been requested or to confirm the completion of some action.

5.2.2. Domain Ontology

It has been suggested previously that information relations in the high-level dialogue specification can be derived from underlying domain relations. The informational relations useful for language are, however, generally at a more abstract level than such domain ontological relations. For example an 'elaboration' relation between two concepts, C_1 and C_2 , in the information structure, such that $C_1 \text{ elaborate } C_2$, might arise from various ontological relations between C_1 and C_2 such as those below (Mann and Thompson, 1988):

- C_1 denotes a specific instance of the generalisation denoted by C_2
- C_1 denotes an instance of the class denoted by C_2
- C_1 denotes an attribute of the object denoted by C_2
- C_1 denotes a member of the set denoted by C_2
- C_1 denotes a part of the whole denoted by C_2
- C_1 denotes a step in the process denoted by C_2

Stent (2000), in trying to apply information relations to the annotation of dialogues, points out that the elaboration relation was found to be too broad a category. In fact "in some sections of our dialogs almost every utterance is an elaboration of the first one" (ibid). Stent (2000) therefore proposes breaking the elaboration relation down into more specific subtypes, such as *set-member*, *process-step*, *object-attribute*, *part-of* etc. Even these subtypes such as 'part-of', however, are too abstract to have a direct correlate in the domain ontology. For

example, a medical ontology might typically require more fine-grained notions of ‘part-of’, themselves arranged in a hierarchy, such as the fragment shown below (Ceusters et al., 2002):

```
IS-ANTERIOR-OF
IS-POSTERIOR-OF
IS-MATERIAL-PART-OF
  IS-TANGENTIAL-MATERIAL-PART-OF
  IS-NON-TANGENTIAL-MATERIAL-PART-OF
  ...
IS-SPATIAL-PART-OF
  IS-PROPER-SPATIAL-PART-OF
    IS-LINEAR-DIVISION-OF
    IS-LAYER-OF
    ...
  IS-TANGENTIAL-SPATIAL-PART-OF
  IS-NON-TANGENTIAL-SPATIAL-PART-OF
  ...
...
```

Hence it is necessary to define the ‘part-of’ relation that is relevant for language in terms of these domain-specific relations, as in the description logic⁶ expression given below which defines the set of things that are ‘part-of’ C_2 to be the disjunction of the sets of things that ‘isAnteriorOf’ C_2 , ‘isPosteriorOf’ C_2 , ‘isMaterialPartOf’ C_2 , ‘isSpatialPartOf’ C_2 etc.

$$\exists \text{isPartOf}.C_2 \equiv \exists \text{isAnteriorOf}.C_2 \sqcup \exists \text{isPosteriorOf}.C_2 \sqcup \\ \exists \text{isMaterialPartOf}.C_2 \sqcup \exists \text{isSpatialPartOf}.C_2 \sqcup \dots$$

It is then possible to define the more abstract informational relations (such as elaboration) in similar terms. For example, the definition given below specifies that an elaboration relation can be inferred between C_1 and C_2 if C_1 is subsumed by C_2 , or C_1 is a member of the set of individuals that are part of C_2 , or C_1 is a member of the set of individuals that are attributes of C_2 , etc.

```
C1 ELABORATE C2 if:
  C1  $\sqsubseteq$  C2                (C1 is a specific instance of C2)
  C1  $\in$   $\exists \text{isPartOf}.C_2$     (C1 is a part of C2)
  C1  $\in$   $\exists \text{isAttributeOf}.C_2$  (C1 is an attribute of C2)
  ...
```

The domain ontology must therefore provide information to the dialogue specification engine regarding the possible relations that can be assumed between concepts, both those that are directly specified in the ontology and those which can be inferred by reasoning over the ontology (e.g. inheritance of relations down the subsumption hierarchy). Furthermore the ontology should be able to provide an ‘upper model’ (Bateman, 1990) classification of concepts, as described in Section 4.2.2, in order that appropriate concept categories can be determined for the high-level dialogue specification.

⁶ For an overview of description logic see (Nardi and Brachman, 2002).

5.3. Specification Language

The knowledge represented by the domain plan and domain ontology is accessed through a common abstract task specification language. The full set of requests that the task specification language must support is described below.

5.3.1. Load

The simplest functionality that must be supported is for the dialogue specification engine to request that a domain (plan and ontology) be loaded. This typically would occur at the start of a dialogue when a user initiates a conversation on a particular subject, but if the conversation is substantial it may consist of several transactions in sequence (as described earlier) where the start of each new transaction may require the loading of a new domain. For example, the dialogue may start by discussing whether or not a patient should be referred to a specialist, then move on to determine specific aspects of treatment such as chemotherapy and then on to other topics, where each area of discussion involves a new domain representing the required clinical knowledge in that area, and involves a new transaction in the dialogue (with details of the previous transaction no longer needed).

5.3.2. Get Metadata

The dialogue specification engine must also be able to request meta-level information regarding the current domain, such as the id of the domain plan, its name, and description. This is useful for determining aspects of the high-level dialogue specification, e.g. transaction name, as well as for parts of the dialogue, e.g. the description could be used as part of an opening move to welcome the user to the system.

5.3.3. Check Completion

The dialogue specification engine will also need to know whether the process defined by the domain plan has been successfully completed or not. This will allow it to determine whether a closing game should be initiated or whether the dialogue should be continued.

5.3.4. Get Task Hierarchy

This functionality allows the dialogue specification engine to retrieve a tree of tasks from the domain plan. The tree structure should reflect the way in which higher-level tasks are related to lower-level tasks in the plan (i.e. the task decomposition relations). Hence if task T_1 is decomposed into tasks T_2 and T_3 then in the returned tree structure T_1 should dominate both T_2 and T_3 .

5.3.5. Get Task Dependencies

The task hierarchy described above is sufficient to determine intentional dominance relations, however it will also be necessary for the dialogue specification engine to know which tasks a particular task is dependent on in the domain plan. For a given task T_0 , this will be the set of tasks $\Sigma = \{T_1, \dots, T_n\}$, such that T_i is an element of Σ if T_i is an antecedent of T_0 in the plan and either

- T_0 has a precondition that specifies that T_i must be in a certain state, or
- T_0 has a precondition that specifies that a data item d_i should have a particular value and the goal of T_i is to acquire a value for d_i .

In either of these cases the task T_0 can be seen as dependent on T_i . A request for task dependencies by the specification engine should therefore return this set Σ , which can then be used to determine satisfaction-precedence relations in the high-level dialogue specification. Note that this information is requested separately from the task hierarchy since it is only relevant for those tasks whose state is dormant (since for active or completed tasks these dependencies must already have been satisfied).

5.3.6. Get Task Specification

This functionality allows the dialogue specification engine to retrieve full details regarding a specific task. In response to this request the information described below should be supplied⁷.

- *Id*: the unique id of the task within the domain plan so that it can be referred to.
- *Name*: a string specifying the name of the task (as defined by the author of the plan).
- *Caption*: a string caption associated with the task (as defined by the author of the plan).
- *Start Time*: the time at which the task was activated, for the purposes of creating a fixed linear (temporal) ordering (e.g. to construct a visual form). This need not be real time; it could be some counter internal to the domain plan execution engine.
- *Type*: the type of the task, assumed to be one of the following:
 - *Plan*: an abstract task that is decomposed into a series of other tasks
 - *Action*: a task whose goal is to change the outside world (e.g. to make the user adopt some belief or perform some action)
 - *Enquiry*: a task whose goal is to change the system's state (e.g. by motivating the user to provide values for system data items)

In addition to the attributes described above, other more specific attributes should also be supplied depending on the type of task being described. For each task of type 'Plan', the following attribute should also be specified:

- *State*: the execution state of the current task. Assumed to be one of: dormant, active or completed.

For each task of type 'Action', the following attributes should also be specified:

- *State*: the execution state of the current task, assumed to be one of dormant, active or completed.
- *Act*: the type of action to be performed, either explain or instruct
- *Topic*: the concept that the action relates to, e.g. 'patient to be referred to specialist'

For each task of class 'Enquiry', the following attributes should also be specified:

- *State*: the request state of the associated data item, either active or completed.
- *Topic*: the concept that the data item relates to, e.g. 'breast cyst'

⁷ Note that the information specified here is based on the PROforma task ontology (Fox and Das, 2000). However, it should be general enough to be supported by other plan execution systems also.

- *Default*: the default value for the data item (if any)
- *Value*: the value currently associated with the data item (if any)
- *SubType*: the type of the data item: either yes/no or complex
- *Enumeration*: an enumerated set of values that the acquired value should be a member of (if any)
- *Role*: the relation between the elements of the enumeration (if specified) and the topic.
- *Multivalue*: whether the elements in the enumeration (if specified) are mutually exclusive or not.

5.3.7. Set Data Item

As a result of interacting with the user the dialogue specification engine will need to update the domain with any information obtained. The specification engine should therefore be able to request that a particular value be associated with a specified data item in the task specification.

5.3.8. Confirm Task

As a result of interacting with the user the specification engine may wish to indicate that an action requested by the domain plan has been successfully completed, e.g. that some information has been given to the user and they have acknowledged it, or that the user has confirmed that they have carried-out some action successfully. The task specification language should therefore support requests to confirm specific tasks of type 'Action'.

5.3.9. Find Associations

This functionality allows the dialogue specification engine to obtain a list of associations between two concepts. This can be provided by the domain ontology in the task specification. For example

- *IsA*: this association can be derived from subsumption in the ontology.
- *PartOf*: this association can be derived from the existence in the ontology of related more specific relations such as IS-ANTERIOR-OF, IS-SPATIAL-PART-OF, IS-TANGENTIAL-PART-OF etc (as described earlier).

5.3.10. Get Category

This allows the dialogue specification engine to determine a category for a concept, where 'category' is an upper model classification of a concept (as described earlier). This can be obtained from the task specification by searching for concepts in the domain upper ontology that subsume the concept to be classified. The main problem is determining the relevant set of upper ontology concepts to use. In the work described here, this will probably need to be determined empirically through the construction of systems for the medical domain, although the set of concepts proposed by (Bateman, 1990) (see Section 4.2.2 above) may form a good starting point.

5.3.11. Get Domain

When the dialogue specification engine needs to make a request for a complex data value (i.e. not just a yes/no question) and no enumeration of acceptable replies is specified in the task description, it will be necessary to determine what concrete domain might be appropriate given the topic of the request. The task specification should therefore support requests to find

a domain type for a concept. This can be done by finding a more general concept that subsumes the one specified and for which a domain type is known. For example, a request for an AGE STATE is not a simple yes/no request, nor can an enumeration be supplied because the relevant set is the concrete domain of (natural) numbers. The ontology should therefore be searched for some subsuming concept that allows the relevant domain ('number') to be identified, e.g. KIND-OF-QUANTITY STATE (assuming that quantities are described by numbers).

References

- Allen, J., Ferguson, G., and Stent, A. (2001). An Architecture for More Realistic Conversational Systems. *Proc. Intelligent User Interfaces 2001 (IUI-01)*, Santa Fe, NM, Jan 14th – 17th.
- Allen, J., and Core, M. (1997). *DAMSL: Dialogue Act Markup in Several Layers*. Draft contribution for the Discourse Resource Initiative (www.georgetown.edu/luperfoy/Discourse-Treebank/dri-home.html).
- Amann, N., Hue, L., Lukas, K. (2001). *Position Statement for Multi-Modal Access*. Submission by Siemens to the W3C Working Group on Multimodal Interaction, 26th November 2001.
- Anderson, A. H., Bader, M., Bard, E. G., Boyle, E., Doherty, G.M., Garrod, S., Isard, S. D., Kowtko, J. C., McAllister, J., Miller, J., Sotillo, C. F., Thompson, H. S., and Weinert, R. (1991). The HCRC Map Task Corpus. *Language and Speech*, 34(4):351-366.
- Bateman, J.A. (1990). Upper Modeling: A General Organization of Knowledge for Natural Language Processing. *Proc. Workshop on Standards for Knowledge Representation Systems*, Santa Barbara, March 1990.
- Boella, G., Damiano, R., Lesmo, L., and Ardissono, L. (1999). Conversational Cooperation: the Leading Role of Intentions. In *Proc. Amstelogue 99, the 3rd Workshop on the Semantics and Pragmatics of Dialogue*. University of Amsterdam, May 1999.
- Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, D. P., and Slack, M. G. (1995). Experiences with an Architecture for Intelligent, Reactive Agents. *Proc. International Joint Conference on AI*.
- Brooks, R. A. (1990). Elephants Don't Play Chess. *Robotics and Autonomous Systems* 6:3-15.
- Bunt, H. (1996). Dynamic Interpretation and Dialogue Theory. In M. Taylor, F. Neel and D. Bouwhuis (eds) *The Structure of Multimodal Dialogue*, vol. 2, John Benjamins, Amsterdam.
- Burke, R. D., Hammond, K. J., and Kozlovsky J. (1995). Knowledge-based information retrieval from semi-structured text. *AI Applications in Knowledge Navigation and Retrieval. Papers from the 1995 AAAI Fall Symposium (Tech. Report FS-95-03)*, AAAI Press, Menlo Park, pp. 15 - 19
- Burton, M., and Brna, P. (1996). Clarissa: an exploration of collaboration through agent-based dialogue games. In *Proceedings of the EuroAIED*, Lisbon.
- Carberry, S., Chu, J., and Green, N. (1993). Rhetorical Relations: Necessary but Not Sufficient. *Proc. Workshop on Intentionality and Structure in Discourse Relations*. ACL-93, Columbus, OH.
- Carletta, J., and Mellish, C. (1995). *Requirements for Belief Models in Cooperative Dialogue*. Technical Report HCRC/RP-66, Human Communication Research Centre, University of Edinburgh, UK.
- Carletta, J., Isard, A., Isard, S., Kowtko, J., and Doherty-Sneddon, G. (1996). *HCRC Dialogue Structure Coding Manual*. Technical Report HCRC/TR-82, Human Communication Research Centre, University of Edinburgh, UK.
- Ceusters, W., Beveridge, M. A., Milward, D., and Falavigna, D. (2002). *Specification for Semantic Dictionary Integration*, Deliverable D9, HOMEY Project, IST-2001-32434.
- Ceusters, W., Martens, P., Dhaen, C., and Terzic, B. (2001). LinkFactory: an Advanced Formal Ontology Management System. *Proc. Interactive Tools for Knowledge Capture Workshop, KCAP-2001*, Victoria B.C., Canada.
- Ceusters, W., Waagmeester, A., De Moor, G. (1997). Syntactic-semantic tagging conventions for a medical treebank: the CASSANDRA approach. In J van der Lei, WPA Beckers, W Ceusters, JJ van Overbeeke (eds.): *Proceedings of MIC'97*, Veldhoven, The Netherlands, pp. 183-193.
- Chaves, R. P. (2001). WordNet and Automated Text Summarization. *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium, NLP RS*, Tokyo, Japan.
- Cohen, P., and Perrault, C. R. (1979). Elements of a Plan-Based Theory of Speech Acts. *Cognitive Science* 3(3):177-212.

- Cooper, R., and Larsson, S. (1999). Dialogue Moves and Information States. In *Proc. of the Third International Workshop on Computational Semantics (IWCS)*.
- Cooper, R., Larsson, S., Matheson, C., Poesio, M., and Traum, D. (1999). *Coding Instructional Manual for Information States. Technical Report*, Gothenberg University.
- Core, M. G., and Allen, J. F. (1997). Coding Dialogs with the DAMSL Annotation Scheme. In *Working Notes of the AAAI Fall Symposium on Communicative Action in Humans and Machines*, AAAI, Boston, MA.
- Dahlbäck, N., and Jönsson, A. (1997). Integrating Domain Specific Focusing in Dialogue Models. In *Proceedings of EuroSpeech '97*, Rhodos, Greece.
- Dahlbäck, N. and Jönsson, A. (1999). Knowledge sources in spoken dialogue systems. In *Proceedings of Eurospeech '99*, Budapest, Hungary.
- Das, S., Fox, J., Hammond, P., and Elsdon, D. (1997). Decision Making and Plan Management By Autonomous Agents: Theory, Implementation and Applications. *Proc. 1st International Conference on Autonomous Agents*, California, pp. 276-283.
- Davies, B. (1994). To Cooperate or not to Cooperate – is that the Question?. In *Proc. of the Edinburgh Linguistics Department Conference '94*, University of Edinburgh, pp. 17-32.
- Falavigna, D. and Gretter, R. (1999). Flexible Mixed Initiative Dialogue over the Telephone Network. *Proc. of ASRU'99*, 12th - 15th December, Colorado.
- Fellbaum, C. (Ed.) (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.
- Ferguson, G., Allen, J., Blaylock, N., Byron, D., Chambers, N., Dzikovska, M., Galescu, L., Shen, X., Swier, R. and Swift, M. (2002). *The Medication Advisor Project: Preliminary Report*. Technical Report 776, Computer Science Department, University of Rochester, New York.
- FIPA (1999). *Specification part 2 – Agent Communication Language*. 16th April 1999. (www.fipa.org).
- Firby, R. J. (1987). An Investigation into Reactive Planning in Complex Domains. *Proc. 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, Milan, Italy, pp. 202-206.
- Fitzgerald, W., and Firby, R. J. (2000). Dialog is Task Execution; Task Execution is Best Done Reactively; Therefore Dialog Systems Call for a Reactive Task Execution Architecture. *Proc. AAAI 2000 Spring Symposium*.
- Fitzgerald, W., and Firby, R. J. (2001). *Dynamic Predictive Memory Architecture Overview*. I/Net, Inc. Technical Report. (www.inetmi.com).
- Flycht-Eriksson, A. (1999). A Survey of Knowledge Sources in Dialogue Systems. *Proc. IJCAI-99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pp 41-48.
- Flycht-Eriksson, A. (2000). A Domain Knowledge Manager for Dialogue Systems. *Proceedings of the 14th European Conference on Artificial Intelligence, ECAI 2000*. IOS Press, Amsterdam.
- Fox, J., and Das, S. (2000). *Safe and Sound: Artificial Intelligence in Hazardous Applications*, AAAI Press, Menlo Park, CA, and MIT Press, Cambridge, Mass.
- Freedman, R. (2000). Using a Reactive Planner as the Basis for a Dialogue Agent. *Proc. Thirteenth Florida Artificial Intelligence Research Symposium (FLAIRS '00)*, Orlando, Florida. AAAI Press.
- Gat, E. (1998). On Three-Layer Architectures. In *Artificial Intelligence and Mobile Robots*, D. Kortenkamp, R.P. Bonasso and R. Murphy (eds), AAAI Press, Menlo Park, CA.
- Gazdar, G., Klein, E., Pulman, G. K. and Sag, I. A. (1985). *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge.
- Ginzburg, J. (1994). An Update Semantics for Dialogue. In *Proc. of the International Workshop on Computational Semantics (IWCS)*, H. Bunt (ed), pp. 111-120. ITK, Tilburg.
- Ginzburg, J. (1997). On Some Semantic Consequences of Turn Taking. In *Proc. of the MunDial97 Workshop on Formal Semantics and Pragmatics of Dialogue*, University of Munich.
- Gorin, A. L., Riccardi, G., and Wright, J. H. (1997). How May I Help You? *Speech Communication*, 23(1):113-127.

- Grosz, B. (1981). Focusing and Description in Natural Language Dialogues. In *Elements of Discourse Understanding*, A. Joshi, B. Webber and I. Sag (Eds). Cambridge University Press, Cambridge, pp. 213-235.
- Grosz, B., and Sidner, C. (1986). Attention, Intention and the Structure of Discourse. *Computational Linguistics* 12(3):175-204.
- Halliday, M. A. K. (1985). *An Introduction to Functional Grammar*. Edward Arnold Press, Baltimore.
- Hobbs, J. R. (1993). Intention, Information, and Structure in Discourse: A First Draft. *Burning Issues in Discourse, NATO Advanced Research Workshop*, Maratea, Italy, pp. 41-66.
- Hobbs, J. R. (1996). On the Relation between the Informational and Intentional Perspectives on Discourse. In *Burning Issues in Discourse: an Interdisciplinary Account*, E. Hovy and D. Scott (eds), Springer-Verlag, Berlin.
- Houghton, G., and Isard, S. D. (1987). Why to Speak, What to Say and How to Say it: Modelling Language Production in Discourse. In *Modelling Cognition*, P. Morris (ed), John Wiley & Sons, pp. 249-267.
- Hovy, E. H. (1993a). In Defense of Syntax: Informational, Intentional, and Rhetorical Structures in Discourse. *Proc. Workshop on Intentionality and Structure in Discourse Relations*. ACL-93, Columbus, OH.
- Hovy, E. H. (1993b). Automated Discourse Generation Using Discourse Structure Relations. In *Artificial Intelligence 63*, Special Issue on Natural Language Processing.
- Hovy, E., and Maier, E. (1993). *Parsimonious or Profligate: How Many and Which Discourse Structure relations?* Unpublished manuscript. (www.isi.edu/natural-language/people/hovy/papers/93discproc.pdf).
- Hulstijn, J. (2000). Dialogue Games are Recipes for Joint Action. *Proceedings of Gotalog 00, 4th Workshop on the Semantics and Pragmatics of Dialogues*. Gothenburg.
- Hyland, J. M. E., and Ong, C.-H. L. (1995) Pi-Calculus, Dialogue games and PCF In *Proc. 7th ACM Conference on Functional Programming Languages and Computer Architecture*, 26th-28th June, La Jolla, California. pp 96-107, ACM Press.
- Jurafsky, D., and Martin, J. H. (2000). *Speech and Language Processing*. Prentice-Hall, New Jersey.
- Kowtko, J. C. and Isard, S. D. (1993). *Conversational Games Within Dialogue*, Research Paper 31, Human Communication Research Centre, Edinburgh.
- Kreutel, J., and Matheson, C. (2000). Obligations, Intentions, and the Notion of Conversational Games. In *Proceedings of Gotalog 00, 4th Workshop on the Semantics and Pragmatics of Dialogues*. Gothenburg.
- Larsson, S., Cooper, R., and Engdahl E. (2000). Question Accommodation and Information States in dialogue. *Proc. Third Workshop in Human-Computer Conversation*, Bellagio, July 2000.
- Larsson, S., Ljunglof, P., Cooper, R., Engdahl, E., and Ericsson, S. (2000). GoDiS – An Accommodating Dialogue System, *Proc. ANLP/NAACL-2000 Workshop on Conversational Systems*, Seattle, May 2000.
- Larsson, S., and Traum, D. (2000). Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6:323—340, Special Issue on Spoken Language Dialogue System Engineering.
- Le Hors, A., Powers, C., and Wium Lie, H. (2001). *XHTML+Voice Profile*. Submission by IBM, Motorola and Opera Software to the W3C Working Group on Multimodal Interaction, 30th November 2001.
- Lewin, I. (1998). *The Autoroute Dialogue Demonstrator*. Technical Report CRC-073. SRI Cambridge Computer Science Research Centre.
- Lewin, I. (2000). A Formal Model of Conversational Game Theory. *Proceedings of Gotalog 00, 4th Workshop on the Semantics and Pragmatics of Dialogues*. Gothenburg.
- Mahesh, K. and Nirenburg, S. (1996). Meaning Representation for Knowledge Sharing in Practical Machine Translation. *Proc. Florida Artificial Intelligence Research Symposium (FLAIRS-96)*, May 19-22, Key West, FL.
- Maier, E. A., and Hovy, E. H. (1993). Organizing Discourse Structure Relations using Meta-Functions. In *New Concepts in Natural Language Processing: Planning, Realization, and Systems*, H. Horacek and M. Zock (eds). Pinter Publisher, London, pp. 69-86.

- Mann, W. D., and Thompson S. A. (1988). Rhetorical Structure Theory: Towards a functional theory of text organization. *Text*, 8(3):243-281.
- Marcu, D. (2000). Extending a Formal and Computational Model of Rhetorical Structure Theory with Intentional Structures a la Grosz and Sidner. In *Proc. COLING 2000*, pp. 523-529.
- McKeown, K. R. (1985). *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, Cambridge.
- Maudet, N. and Evrard, F. (1998). A generic framework for dialogue game implementation. In J. Hulstijn and A. Nijholt (eds) *Proceedings of the second workshop on Formal Semantics and Pragmatics of Dialogue*, May 13-15, University of Twente, Enschede, Netherlands.
- Maudet, N. and Moore, D. (1999). Dialogue Games for Computer Supported Collaborative Argumentation. *Proc. Computer-Supported Collaborative Argumentation for Learning Communities, CSCL'99 Workshop*, 11th - 12th Dec., 1999, Stanford University.
- McBurney, P. and Parsons, S. (2002). Games that Agents Play: A Formal Framework for Dialogues Between Autonomous Agents. *Journal of Logic, Language and Information*, Special Issue on Logic and Games, 11(3):315-334.
- Mittal, V. O., and Paris, C. L. (1993). On the Necessity of Intentions and (at Least) the Usefulness of Rhetorical Relations: A Position Paper. *Proc. Workshop on Intentionality and Structure in Discourse Relations*. ACL-93, Columbus, OH.
- Moore, J. (1995). The Role of Plans in Discourse Generation, In *Discourse: Linguistic, Computational, and Philosophical Perspectives*, Daniel Everett and Sarah G. Thomason (Eds.).
- Moore, J. D., and Paris, C. L. (1992). *Planning Text for Advisory Dialogues: Capturing Intentional, Rhetorical and Attentional Information*. Technical Report from the University of Pittsburgh, Department of Computer Science (Number 92-22) and from USC/ISI, #RS 93-330.
- Moore, J. D., and Pollack, M. E. (1992). A Problem for RST: The Need for Multi-Level Discourse Analysis. *Computational Linguistics* 18(4).
- Moore, J. D., and Swartout, W. R. (1990). Dialogue-Based Explanation. In *Natural Language in Artificial Intelligence and Computational Linguistics*, C.L. Paris, W.R. Swartout & W.C. Mann (eds). Kluwer, Boston, pp. 3-48.
- Moser M., and Moore, J. D. (1993). Investigating Discourse Relations. *Proc. Workshop on Intentionality and Structure in Discourse Relations*. ACL-93, Columbus, OH.
- Moser, M., and Moore, J. D. (1996). Toward a Synthesis of Two Accounts of Discourse Structure. *Computational Linguistics* 22(3):409-419.
- Nardelli, L. and Falavigna, D. (2002). *Multi-Modal Mark-Up Language Specification*. Deliverable D4, HOMEY Project, IST-2001-32434.
- Nardi, D., and Brachman, R. J. (2002). An Introduction to Description Logics. *The Description Logic Handbook*, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, pp 5-44.
- Niklfeld, G., Finan, R., Pucher, M. (2001). Architecture for Adaptive Multimodal Dialog Systems Based on VoiceXML. In *Proceedings of EuroSpeech 2001*.
- Paris, C. L. (1990). Generation and Explanation: Building an Explanation Facility for the Explainable Expert Systems Framework. In *Natural Language in Artificial Intelligence and Computational Linguistics*, C.L. Paris, W.R. Swartout & W.C. Mann (eds). Kluwer, Boston, pp. 49-82.
- Perrault, C. R., and Allen, J. F. (1980). A Plan-Based Analysis of Indirect Speech Acts. *American Journal of Computational Linguistics*, 6(3-4):167-182.
- Poesio, M. and Mikheev, A. (1998). The Predictive Power of Game Structure in Dialogue Act Recognition: Experimental Results Using Maximum Entropy Estimation. In *Proc. ICSLP '98*.
- Poesio, M., and Traum, D. (1998). Towards an axiomatization of dialogue acts. In J. Hulstijn and A. Nijholt, editors, *Proceedings of the Twente Workshop on the Formal Semantics and Pragmatics of Dialogues*, pages 207-222, Enschede, May 1998.

- Power, R. (1979). The Organization of Purposeful Dialogues. *Linguistics*, 17:107-152.
- Pulman, S. G. (1997), Conversational Games, Belief Revision and Bayesian Networks, *CLIN VII: Proc. 7th Computational Linguistics in the Netherlands meeting*, Nov 1996, ed. J. Landsbergen et al., 1-25.
- Pulman, S. G. (1999), Relating Dialogue Games to Information States, *Proc. European Speech Communication Association workshop on Dialogue and Prosody* (ed. J. Terken and M. Swerts), De Koningshof, The Netherlands, 17-24.
- Rao, A. S., and Georgeff, M. P. (1995). BDI Agents: from theory to practice. *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, San Francisco, CA.
- Stefanelli, M., Rognoni, C., Quaglini, S., Giorgino, T., and Azzini, I. (2002). *Algorithms and Tools for Dialogue Adaptation*, Deliverable D6, HOMEY Project, IST-2001-32434.
- Stent A. (2000). Rhetorical Structure in Dialog. *Proc. 2nd International Natural Language Generation Conference (INLG'2000)*.
- Traum, D. R. (1993). Rhetorical Relations, Action and Intentionality in Conversation. In *Proc. ACL SIG Workshop on Intentionality and Structure in Discourse Relations*, ACL-93, Columbus, OH, pp. 132-135.
- Traum, D. R. (1996a). Conversational Agency: The TRAINS-93 Dialogue Manager. *Proc. Twente Workshop on Language Technology: Dialogue Management in natural Language Systems (TWLT 11)*, pp 1-11.
- Traum, D. R. (1996b). A Reactive-Deliberative Model of Dialogue Agency. In J. P. Muller, M. J. Wooldridge, and N. R. Jennings, editors, *Intelligent Agents III - Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages (ATAL-96)*, Lecture Notes in Artificial Intelligence, Springer-Verlag, Heidelberg.
- Traum, D. R., and Allen J. F. (1994). Discourse Obligations in Dialogue Processing. *Proc. 32nd ACL, Las Cruces, New Mexico*, pp. 1-8.
- Verspoor, C., and Uceda, V. (1999). The Phrasal Lexicon in Multilingual Generation. *Submitted to 37th Annual Meeting of the ACL*. (<http://citeseer.nj.nec.com/54436.html>).
- W3C (1999). *XSL Transformations (XSLT) Version 1.0*. W3C Recommendation 16th November 1999. (<http://www.w3.org/TR/xslt>).
- W3C (2000a). *XHTML 1.0, The Extensible HyperText Markup Language (Second Edition)*. A Reformulation of HTML 4 in XML 1.0. W3C Recommendation, 26 January 2000. (www.w3.org/TR/xhtml1/).
- W3C (2000b). *Extensible Markup Language (XML) 1.0 (Second Edition)*. W3C Recommendation 6th October 2000. (<http://www.w3.org/TR/REC-xml>).
- W3C (2003). *Voice Extensible Markup Language (VoiceXML) Version 2.0*. W3C Candidate Recommendation 28th January 2003. (www.w3.org/TR/voicexml20/).
- Williams, S. (1996). Dialogue Management in a Mixed-Initiative, Cooperative, Spoken Language System. In S. LuperFoy, A. Nijholt, and G.V. van Zanten (eds) *Dialogue Management in Natural Language Systems. Proc. 11th Twente Workshop on Language Technology*. Enschede, NL: Universiteit Twente.
- Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley & Sons Ltd, Chichester, UK.
- Young, M. R., and Moore, J. D. (1994). Does Discourse Planning Require a Special Purpose Planner? *Proc. Annual Meeting of the American Association for Artificial Intelligence*, Seattle.

Appendix A: Implementing the High-Level Dialogue Specification

This appendix briefly outlines an approach to implementing the high-level dialogue specification language, and mapping between it and various low-level specification languages.

High-Level Dialogue Specification Language

This section describes an XML (W3C, 2000b) implementation of the high-level dialogue specification language described in Section 4.3.

Transaction Element

The root element of a specification document is a `Transaction` element, as defined below, which contains a tree of games representing possible move sequences that may be made during the current state of the dialogue. It is defined as:

```
<Transaction
  name = string
  <!-- Content: (Game+) -->
</Transaction>
```

The `Transaction` element may optionally be assigned a string that represents the transaction name, and may contain a set of one or more `Game` elements (note that there must be at least one such top-level element).

Game Element

The `Game` element defines a conversational game that either the system or the user may address during the dialogue. It is defined as:

```
<Game
  id = string
  name = string
  scope = (Foreground | Background)
  focus = boolean
  <!-- Content: ((Game+) | (Topic, Move, Relation*)) -->
</Game>
```

A `Game` element has a mandatory `id` attribute that uniquely defines the corresponding game within the transaction. A `Game` element may optionally also have a name assigned to it via the `name` attribute. The attributes `scope` and `focus` are described below. A `Game` element can contain either one or more other `Game` elements, or a `Topic`, `Move` and zero or more `Relation` elements. Note that if it contains other games then it must not contain `Move`, `Topic` or `Relation` elements. Otherwise, it must contain exactly one `Move` element, one `Topic` element, and optionally a series of `Relation` elements. The `Move`, `Topic` and `Relation` elements are described below.

As described above, `Game` elements in the dialogue specification language can be nested in order to represent dominance relations between their associated underlying intentions. For example, consider the structure shown below (ignoring details of the definition of the individual games):

```
<Transaction>
  <Game id="12" name="Clinical Details" ...>
    <Game id="26" name="Cyst Enquiry" ...>
      ...
    </Game>
  </Game>
</Transaction>
```

Here the ‘Cyst Enquiry’ game is nested inside the ‘Clinical Details’ game because in the intentional structure the intention to know about the patient’s clinical details dominates the intention to know whether they have a cyst.

The scope attribute of the Game element allows the specification of whether a game is foreground (playable by the system) or background (a game that the user might engage-in as a result of taking the initiative in the dialogue). For example, consider the following game structure in which only the first game is available to the system to initiate (although the user may address the second game in their response by making use of mixed-initiative):

```
<Transaction>
  <Game id="12" name="Drug Administration" scope="Foreground">
    ...
  </Game>
  <Game id="26" name="Blood Pressure Enquiry" scope="Background">
    ...
  </Game>
</Transaction>
```

The focus attribute is assigned a True or False value, which indicates whether the associated game is to be played in the next system turn or not. Note that only games with foreground scope can be assigned focus. For example, in the following structure it is the ‘drug administration’ game (*id=12*) that will be selected in the current system turn:

```
<Transaction>
  <Game id="12" name="Drug Administration" scope="Foreground"
    focus="True">
    ...
  </Game>
  <Game id="26" name="Blood Pressure Enquiry" scope="Background">
    ...
  </Game>
</Transaction>
```

Move Element

The Move element determines the current move within a specified game. It is defined as:

```
<Move
  type = string>
  <!-- Content: (Domain | Content)? -->
</Move>
```

The `Move` element has a mandatory `type` attribute that specifies the particular type of move to be made in the current state of the associated game. It can contain a `Domain` element or a `Content` element (both described below). An example of a game whose initiating move is a *query-w* move is shown below.

```
<Transaction>
  <Game id="26" name="Age Enquiry" ...>
    <Move type="Query-w">
      ...
    </Move>
  </Game>
</Transaction>
```

Topic Element

The `Topic` element defines the topic of a game, and hence of all moves relating to that game. It is defined as:

```
<Topic
  concept = string
  category = string
  value = string>
  <!-- Content: (none) -->
</Topic>
```

A `Topic` element has a mandatory `concept` attribute that specifies the ontological concept that forms the topic of discussion in the game. The `category` attribute allows the specification of an upper model classification for the topic concept. The `value` attribute allows the specification of a current value associated with the specified topic concept (e.g. as specified in a reply move).

For example, the following structure defines two games, both requests, one (*id=26*) regarding the ontological concept of AGE STATE and the other (*id=27*) regarding the ontological concept of CYST. The former is of category ‘Property’ and the latter is of category ‘Material Entity’. Furthermore, a value has been specified for the topic of the first game (AGE STATE) by a previous reply move in this game (hence its scope is background).

```
<Transaction>
  <Game id="26" name="Age Enquiry" scope="background">
    <Topic concept="AGE STATE" category="Property" value="35"/>concept="CYST" category="Material Entity"/>
```

Domain Element

If the move associated with a game is a complex query then a domain of acceptable responses to the request can be specified by a `Domain` element. This element is defined as:


```
<Domain
  name = string
  role = string
  multivalue = boolean>
  <!-- Content: (Option+)? -->
</Domain>
```

If the name attribute is defined then the domain is a concrete (named) domain and the Domain element cannot contain any Option elements. An example of a concrete domain is 'Number', as shown in the age enquiry game below:

```
<Transaction>
  <Game id="26" name="Age Enquiry" ...>
    <Topic concept="ORGANISM AGE STATE" .../>
    <Move type="Query-w">
      <Domain name="number"/>
    </Move>
  </Game>
</Transaction>
```

If no name is specified then an enumeration is assumed and at least one Option element must be present (the Option element is described below). The optional role attribute specifies the relation of the domain to the topic of the game and is useful when using enumerations to define the relation between the set of options and the topic of the request, e.g. "Is-a", "Part-of" etc. The optional multivalue attribute defines whether the values in the enumeration are mutually exclusive or not. An example of an enumeration is the enquiry for a person's sex as shown below:

```
<Transaction>
  <Game id="27" name="Sex Enquiry" ...>
    <Topic concept="ORGANISM SEX STATE" .../>
    <Move type="Query-w">
      <Domain role="Is-a" multivalue="False">
        ...
      </Domain>
    </Move>
  </Game>
</Transaction>
```

Option Element

The Option element specifies a single set element in an enumeration. It is defined as:

```
<Option
  concept = string>
  <!-- Content: (none) -->
</Option>
```

The concept attribute is mandatory and specifies an ontological concept. The example of an enumeration given above can therefore be fully defined as shown below:

```
<Transaction>
  <Game id="27" name="Sex Enquiry" ...>
    <Topic concept="ORGANISM SEX STATE" .../>
    <Move type="Query-w">
      <Domain role="Is-a">
        <Option Concept="MALE SEX"/>
        <Option Concept="FEMALE SEX"/>
      </Domain>
    </Move>
  </Game>
</Transaction>
```

Content Element

Responses by the user or system often need to specify the content to be communicated. This is specified by the Content element. This element is defined as:

```
<Content
  value = string
  concept = string>
  <!-- Content: (none) -->
</Content>
```

The Content element can specify a value that should be associated with the topic of the game, as in the example below:

```
<Transaction>
  <Game id="11" name="Patient Age Enquiry" ...>
    <Topic concept="ORGANISM AGE STATE" .../>
    <Move type="Reply-w">
      <Content value="35"/>
    </Move>
  </Game>
  <Game id="12" name="Patient Sex Enquiry" ...>
    <Topic concept="ORGANISM SEX STATE" .../>
    <Move type="Reply-w">
      <Content value="FEMALE SEX"/>
    </Move>
  </Game>
  <Game id="13" name="Cyst Enquiry" ...>
    <Topic concept="CYST" .../>
    <Move type="Reply-yn">
      <Content value="True"/>
    </Move>
  </Game>
</Transaction>
```

The first game (*id=11*) specifies a reply to a query regarding a patient's age and specifies that their age is 35. The second game (*id=12*) specifies a reply to a query regarding a patient's sex and specifies that their sex is represented by the concept FEMALE SEX. The third game (*id=13*) specifies a reply to a yes/no query regarding whether a patient has a cyst, and specifies that they do.

It can also be the case, however, that a user's response does not address the topic of the original question directly, but instead refers to a related more general concept (a *Reply-part*

move). In this case, the `Content` element should stipulate the concept that the reply value refers to. This can be done by assigning a value to the `concept` attribute of the `Content` element. This is shown in the example below in which the user is making a reply move to a yes/no question regarding whether or not a patient has a cyst, but has responded with the more general reply that they have a lesion.

```
<Transaction>
  <Game id="13" name="Cyst Enquiry" ...>
    <Topic concept="CYST" .../>
    <Move type="Reply-yn">
      <Content concept="LESION" value="True"/>
    </Move>
  </Game>
</Transaction>
```

Relation Element

Information relations between games are represented by a `Relation` element in the dependent (satellite) game of the relation. This element is defined as:

```
<Relation
  type = string
  nucleus = string>
  <!-- Content: (none) -->
</Relation>
```

The `type` attribute specifies the type of the relation (e.g. elaboration) and the `nucleus` attribute specifies the head of the relation. For example, two games related in an elaboration relation would be defined as follows, where the second game (*id=13*) elaborates the first game (*id=8*):

```
<Transaction>
  <Game id="8" name="Discharge Enquiry" ...>
    <Topic concept="NIPPLE DISCHARGE" .../>
    <Move type="Query-yn"/>
  </Game>
  <Game id="13" name="Bilateral Discharge Enquiry" ...>
    <Topic concept="BILATERAL NIPPLE DISCHARGE" .../>
    <Move type="Query-yn"/>
    <Relation type="Elaboration" nucleus="8"/>
  </Game>
</Transaction>
```

Mapping to Low-Level Specification Languages

Since the high-level dialogue specification is both language-neutral and modality-independent, in order to map it to a low-level specification it is necessary to:

- Generate the actual utterances to be made by the system (in the appropriate language)
- Map the game specifications into moves encoded in an appropriate mark-up language for the client that is interacting with the system (VoiceXML, XHTML or a multimodal XML language).

An approach to implementing these two steps is described next.

Sentence Generation

As described in deliverable D9 (Ceusters et al., 2002), a domain-specific lexical ontology (semantic dictionary) is provided by the L&C ontology by associating terms with domain concepts. Hence, lexical relations, such as hyperonymy, between terms can be derived from domain links, such as subsumption, between the associated concepts. Furthermore, because the L&C domain ontology expresses collections of attributes as complex concepts (e.g. a DISCHARGE that is BLOODSTAINED is captured by the complex concept BLOODSTAINED DISCHARGE) the associated terms form a phrasal lexicon (Verspoor and Uceda, 1999). This means that in most cases the utterances generated by the system can be created using a fairly simple template and phrases from the lexicon, chosen on the basis of the topic of the current game. For example, consider the high-level specification shown below:

```
<Transaction>
  <Game id="1" name="Urgent Referral">
    <Topic concept="PATIENT-IN-NEED-OF-URGENT-REFERRAL"/>
    <Move Type="Explain"/>
  </Game>
</Transaction>
```

In this case the domain ontology contains the complex concept PATIENT-IN-NEED-OF-URGENT-REFERRAL and hence the lexicon contains the associated phrase “patient in need of urgent referral”. This can be combined with a template for *explain* moves to derive a rendering such as that below (where the contribution of the template is underlined):

```
This is a patient in need of urgent referral.
```

However, more complex renderings, e.g. “this patient is in need of an urgent referral”, “this is a patient who is in need of an urgent referral” etc, would require decomposition of the concept which represents the topic of the game, and development of more complex templates which capture the required grammatical structures.

In addition, selection of the appropriate template can be guided by the category (upper model classification) of the current game topic. For example, consider the sentences below:

```
[1] What is the patient's age/sex/... ? [category="Property"]
[2] Is there a cyst/nodule/... ? [category="Material Entity"]
[3] Is there any eczema/retraction/... ? [category="Material Process"]
```

Because the category of the topic (e.g. age, sex etc.) in [1] is ‘property’, a template has been chosen which phrases the question so as to refer to an attribute of some object (in this case a patient), rather than asking, for example, “is there a sex?” or “is there any age?” (as would be generated for a material entity or material process). Similarly, in [2] the fact that the topic (cyst, nodule etc) is classified as a material entity leads to a realisation such as “is there a ...” rather than, for example “what is the patient’s cyst?” or “Is there any nodule?”. In [3] the classification of eczema, retraction etc as material processes motivates a sentence template such as “is there any ...” rather than , for example, “what is the patient’s eczema?” or “Is there an eczema?”. The set of appropriate upper model concepts will probably need to be

determined empirically as part of the process of implementing the approach described in this deliverable.

XSL Transformation

The result of the generation process, the low-level specification, is encoded as a VoiceXML, XHTML or multimodal XML document depending on the type of the client browser. A visual browser then interprets the low-level specification in order to determine how to control visual rendering and interaction through keyboard and/or mouse. A VoiceXML browser similarly uses the specification to determine how to control speech recognition and synthesis components, as well as the telephony platform.

Since the high-level dialogue specification and the low-level specification are both expressed as XML languages the mapping between them can be performed by defining XSLT (W3C, 1999) stylesheet templates that transform the modality-independent high-level specification into the target mark-up language. For example, consider the high-level specification shown below:

```
<Transaction>
  <Game id="1" name="Urgent Referral">
    <Topic concept="PATIENT-IN-NEED-OF-URGENT-REFERRAL"/>
    <Move Type="Explain"/>
  </Game>
</Transaction>
```

If the client is an XHTML browser then one way of realising this specification would be to transform it into an XHTML document such as that below:

```
<HTML>
  <body>
    <form method="post" action="...">
      <p>This patient is in need of urgent referral</p>
      <input type="hidden" name="acknowledge" value="true"/>
      <input type="submit" value="ok"/>
    </form>
  </body>
</HTML>
```

In this case the *explain* move is realised as a text paragraph which has been generated using terms derived from the lexicon. An *explain* move sets-up an expectation for an *acknowledgement* by the other participant and this is realised through a submit button whose label is “ok”. Clicking this button submits the form which returns control to the dialogue engine and passes back a single attribute-value pair “acknowledge=true” (as a result of the hidden input field).

The same specification could similarly be realised as a VoiceXML document such as the following:

```
<VXML>
  <form>
    <prompt>This patient is in need of urgent referral</prompt>
    <field name="acknowledge">
      <grammar root="rule1">
        <rule id="rule1">
          <one-of>
            <item>ok</item>
            <item>right</item>
          </one-of>
          <tag>true</tag>
        </rule>
      </grammar>
    </field>
    <filled>
      <submit next="..." />
    </filled>
  </form>
</VXML>
```

This realises the *explain* move as a prompt and specifies a language model (grammar) for replies that can be mapped to an *acknowledge* move. This grammar can then be used by the speech recogniser to help interpret the speech signal. If either of the specified valid replies are recognised then the attribute-value pair “acknowledge=true” (derived from the field name and the semantic tag assigned to the associated grammar) will be returned to the dialogue system. As a further illustration, consider the specification of a *query-w* game given below:

```
<Transaction>
  <Game id="27" name="Sex Enquiry">
    <Topic concept="ORGANISM SEX STATE"/>
    <Move type="Query-w">
      <Domain role="Is-a" Multivalue="False">
        <Option Concept="MALE SEX"/>
        <Option Concept="FEMALE SEX"/>
      </Domain>
    </Move>
  </Game>
</Transaction>
```

This could be realised by an XHTML document such as that given below:

```
<HTML>
  <body>
    <form method="post" action="...">
      <p>What is the patient's sex?</p>
      <select name="ORGANISM SEX STATE">
        <option value="MALE SEX">Male</option>
        <option value="FEMALE SEX">Female</option>
      </select>
      <input type="submit"/>
    </form>
  </body>
</HTML>
```


Here the initial *query-w* move is realised as a text question and the expected *reply-w* move by the user is realised as a drop-down list containing the range of (mutually exclusive) values that the user may supply. If the user has selected “Female” from the drop-down list then, on clicking the “submit” button, an attribute-value pair “ORGANISM SEX STATE=FEMALE SEX” is returned to the dialogue system. Similarly, a VoiceXML realisation might be similar to that shown below:

```
<VoiceXML>
  <form>
    <prompt>What is the patient's sex?</prompt>
    <field name="ORGANISM SEX STATE">
      <grammar root="rule1">
        <rule id="rule1">
          <one-of>
            <item>male<tag>MALE SEX</tag></item>
            <item>female<tag>FEMALE SEX</tag></item>
          </one-of>
        </rule>
      </grammar>
    </field>
    <filled>
      <submit next="..." />
    </filled>
  </form>
</VoiceXML>
```

This again realises the *query-w* move as a prompt and the expected *reply-w* move is caught by the <filled> element causing control to return to the dialogue system. The range of possible replies is defined by the speech grammar defined in the <grammar> element and each possible reply is assigned a semantic interpretation via a <tag> element. Hence, if the user replies “female” then an attribute-value pair “ORGANISM SEX STATE=FEMALE SEX” is submitted.

Normally a high-level specification will specify more than one playable game. In the visual modality this might lead to several games being rendered simultaneously to create a form. In the speech modality it would lead to an extended language model which allowed the user to give other replies than just the answer to the question asked. Once the user's response has been passed back to the interpreter it can be matched against the games in the high-level specification to determine how it should be interpreted, i.e. what move the user intended to make and in which game – a process referred to as move accommodation (Larsson, Ljunglof, Cooper, Engdahl and Ericsson, 2000) as described previously.

Appendix B: Implementing the Abstract Task Specification

This appendix briefly outlines an approach to implementing the abstract task specification, in particular the task specification language, domain plan and domain ontology.

Task Specification Language

The task specification language described in Section 5 can be implemented as a Java class with instance methods that implement the specific messages described. Example method definitions are listed and briefly described below.

Load

This method loads the task specification (domain plan and ontology) for the current domain of discourse.

```
public boolean load(String domain)
```

Parameters:
domain — Specification of the discourse domain

Returns:
Success or failure of the load operation

GetMetaData

This method retrieves meta information regarding the currently loaded domain.

```
public Map getMetaData()
```

Parameters:
None

Returns:
Map structure containing attribute-value pairs for the relevant attributes (currently the id of the root plan, its name and the description provided by the plan's author).

IsCompleted

This method determines whether all the tasks in the currently loaded domain plan have been completed or not.

```
public boolean isCompleted()
```

Parameters:
None

Returns:
True or False.

GetTaskHierarchy

This method retrieves a tree of tasks where the tree structure reflects task decomposition relations in the domain plan.

```
public List getTaskHierarchy()
```

Parameters:

None

Returns:

A list of elements where each element either specifies the id of a task, or is itself a list containing further elements (hence recursive structure).

GetTaskDependencies

This method retrieves a list of tasks that the specified task is dependent on.

```
public List getTaskDependencies(String taskId)
```

Parameters:

`taskId` — The unique identifier of the task for which dependencies are to be found.

Returns:

A List structure where each element is the unique id of a task that the specified task (parameter) is dependent on.

GetTaskInfo

This method retrieves a task description for a specified task.

```
public Map getTaskInfo(String taskId)
```

Parameters:

`taskId` — The unique identifier of the task to be described.

Returns:

A Map structure containing a description (in attribute-value pairs) of a task.

SetDataItem

This method updates a specified data item in the domain in order to associate it with the specified value.

```
public void setDataItem(String dataId, String value)
```

Parameters:

`dataId` — The unique identifier of the data item to be updated.

`value` — The value that should be associated with the specified data item. This value will be coerced to the data type required by the specified data item.

Returns:

None.

ConfirmTask

This method sets the state of the specified task of class 'Action' to 'completed'.

```
public void confirmTask(String taskId)
```

Parameters:

`taskId` – The unique identifier of the task (which must be of class 'Action') whose completion is to be confirmed.

Returns:

None.

FindAssociations

This method attempts to find semantic associations between the specified concepts. This is done by checking for subsumption in the domain ontology (fromConcept *Is-A* toConcept) and for relations such that fromConcept is the source of the link and toConcept is the target.

```
public Set findAssociations(String fromConcept, String toConcept)
```

Parameters:

`fromConcept` – The source concept for the association to be found.

`toConcept` – The target concept for the association to be found.

Returns:

A Set structure where each element is a String description of an association, e.g. "IsA" or "PartOf".

GetConceptCategory

This method tries to determine an upper model classification (category) for the specified concept. This is done by searching the upper domain ontology for any relevant concepts that subsume the specified concept. If more than one is found then one is chosen according to a hierarchy.

```
public String getConceptCategory(String concept)
```

Parameters:

`concept` – The knowledge name of the concept to be classified

Returns:

The knowledge name of the relevant upper ontology concept that subsumes the specified concept.

GetConceptDomain

This method tries to determine if instances of the specified concept class belong to a concrete domain, and if so returns a description of the domain.

```
public String getConceptDomain(String concept)
```

Parameters:

concept – The knowledge name of the concept for which a named domain is to be found

Returns:

The String name of the associated concrete domain if found.

Specifying the Domain Plan





PROforma

One approach to implementing the domain plan is to use the *PROforma* toolset provided by CRUK (Das and Fox, 2000). This is a collection of tools for authoring, publishing and enacting clinical guidelines written in the *PROforma* process specification language. This section briefly introduces *PROforma* and then discusses how it could be used to represent the required information.

The *PROforma* process specification language supports the definition of clinical guidelines and protocols in terms of:

- A well-defined set of tasks that can be composed into networks representing plans or procedures carried out over time. These enable the high level structure of a guideline to be represented.
- Logical constructs (such as situations, events, constraints, pre- and post-conditions, and inference rules) which allow the details of each task and inter-relationships between tasks to be defined using templates.

The *PROforma* language therefore uses four types of task, each of which is also associated with a graphical representation (icon), which is used in the graphical authoring tools for creating guidelines (described later). These tasks are described in the table below:

Icon	Task	Description
	Plan	Sets of tasks to be carried out to achieve a clinical goal. Plans are the basic building blocks of a guideline, and may contain any number of tasks of any type.
	Decision	Tasks which involve choices of some kind, such as a choice of investigation, diagnosis or treatment.
	Action	Typically clinical procedures (such as the administration of an injection) which need to be carried out.
	Enquiry	Actions returning required information; typically requests for information or data from the user.

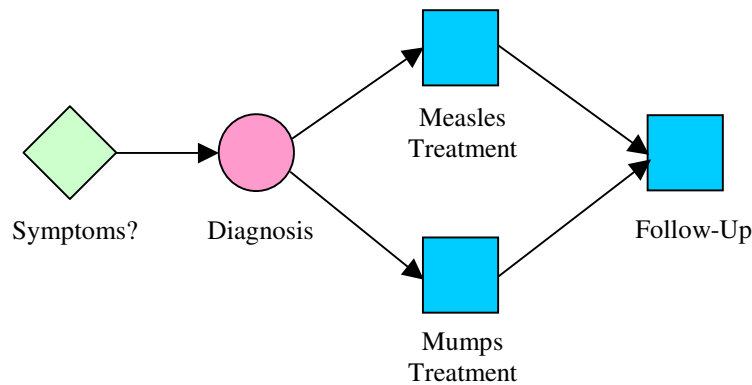


Figure B.1: An example guideline as represented in the authoring tool

Each task type is implemented as a sub-class of an abstract ‘task’ super-class. The attributes of each sub-class determine the behaviour of its members during enactment of a guideline. All sub-classes inherit some generic attributes from the super-class which define: a goal that the task is to achieve, a trigger that causes a task to be considered for enactment (asynchronously), pre-conditions for enacting the task, post-conditions that should hold after enactment, a cycling schema to control task iteration, and whether or not the task must be authorised by another agent before enactment. In addition to these generic attributes, each sub-class defines some class-specific attributes.

The plan class has additional attributes which define: the tasks that compose the plan (note that these may be of any type, including plan itself, hence allowing recursion to be represented), constraints on the scheduling order of tasks, conditions for successful termination, and conditions for unsuccessful termination. The decision class has attributes defining: decision candidates to be considered, arguments for and against candidates, whether one or many candidates can be chosen and the scheme for combining arguments. Actions have an attribute to define a procedure to be carried-out by another agent (either human or machine) and enquiries have attributes to define the set of data items for which values are to be obtained from another agent.

In order to assist in the creation of *PROforma* guidelines, the *PROforma* toolset contains a graphical authoring tool, which allows guidelines to be specified by drawing a high-level diagram depicting the tasks involved (using the icons shown earlier) and the relationships, e.g. scheduling constraints, between them (represented by arrows). The authoring tool also supports the definition of generic attributes (e.g. pre-condition, goal etc) and task-specific attributes for tasks (e.g. the data sources for an enquiry). An example guideline is shown in Figure B.1.

Once the guideline has been authored it can be submitted to the *PROforma* engine for enactment, at which point the individual tasks and attributes are used to generate procedures to carry-out. In the case of the example guideline given above, the enactment engine will request data regarding a patient’s symptoms, makes a decision based on that data as to whether the patient has measles or mumps, then, depending on the decision, provide instructions on treating the disease followed by some instructions on follow-up treatment.

Although the graphical representation of *PROforma* plans shown above is useful for the purposes of authoring, the actual guideline specification is represented using a derivative of the R^2L language (Das, 1997). As an illustration, a (simplified) guideline definition based on Figure B.1 above is shown below:

```
plan :: Mumps_Measles_Treatment;
  component :: Enquiry_1;
  component :: Decision_1;
    schedule_constraint :: completed(Enquiry_1);
  component :: Action_1;
    schedule_constraint :: completed(Decision_1);
  component :: Action_2;
    schedule_constraint :: completed(Decision_1);
  component :: Action_3;
    schedule_constraint :: completed(Action_1);
    schedule_constraint :: completed(Action_2);
end plan.

decision :: Decision_1;
  candidate :: Mumps;
    argument :: confirming, swollen_glands = 'yes'
    attributes
      argument_name :: 'swollen_glands = "yes"';
    end attributes;
    recommendation :: netsupport(Decision_1, Mumps) > 1;
  candidate :: Measles;
    argument :: confirming, swollen_glands = 'no'
    attributes
      argument_name :: 'swollen_glands = "no"';
    end attributes;
    recommendation :: netsupport(Decision_1, Measles) > 1;
end decision.

enquiry :: Enquiry_1;
  source :: swollen_glands;
end enquiry.

action :: Action_1;
  precondition :: result_of(Decision_1)="Measles";
  procedure :: 'Measles treatment';
end action.

action :: Action_2;
  precondition :: result_of(Decision_1)="Mumps";
  procedure :: 'Mumps treatment';
end action.

action :: Action_3;
  procedure :: 'Follow up...';
end action.

data :: swollen_glands;
  type :: boolean;
  caption :: 'Are your glands swollen?';
end data
```

Mapping to the Task Specification Language

The task types defined in the task specification language can be derived fairly directly from *PROforma* task classes. For example, a ‘Plan’ task in the task specification language (i.e. a task which is decomposed into sub-tasks) can be derived from the specification of a component of class ‘Plan’ in *PROforma* as shown below:

```
plan :: T;  
  ...  
end plan.
```

Similarly, an ‘Action’ task whose ‘Act’ attribute is ‘Explain’ in the task specification language can be derived from *PROforma* specifications such as that shown below.

```
action :: T;  
  procedure :: Explain;  
  context :: c;  
end action.
```

In this case, the ‘Topic’ of the task is the ontological concept whose knowledge name is given by *c*. An ‘Action’ task whose ‘Act’ attribute is ‘Instruct’ can be similarly derived:

```
action :: T;  
  procedure :: Instruct;  
  context :: c;  
end action.
```

Here an Instruct task can be inferred whose topic is again defined by the concept *c*. An ‘Enquiry’ task whose ‘SubType’ attribute is yes/no can be derived from specifications such as the following:

```
enquiry :: T;          data :: c;  
  source :: c;         type :: boolean;  
end enquiry.          end data.
```

Here the subtype of the task is derived from the *PROforma* data item specification associated with the enquiry – in this case an item of type Boolean, hence a yes/no query is inferred. More complex queries similarly derive from enquiries in *PROforma*, with the particular subtype derived from the definition of the associated data item. For example, an ‘Enquiry’ task whose ‘SubType’ attribute is complex, but is not specified by an enumeration, can be derived from a *PROforma* specification such as the following:

```
enquiry :: T;          data :: c;  
  source :: c;         type :: text;  
end enquiry.          end data.
```

Here, the data item is a ‘text’ type but with no range of valid values specified, hence an appropriate named domain must be derived based on the associated concept in the domain

ontology. An ‘Enquiry’ task whose ‘SubType’ attribute is complex, and is associated with an enumerated type, can be inferred from a PROforma specification such as the following:

```
enquiry  :: T;  
    source :: c;  
end enquiry.  
data    :: c;  
    type  :: text;  
    range :: <comma delimited list>;  
end data.
```

Here, the data type is again a complex type (text) but a range of valid values is defined as a comma delimited list and so this can be used to define the domain of valid replies to the enquiry by the user. If the data type is ‘setof_text’ then the individual items specified in the range of valid values are not mutually exclusive, otherwise they are.

In addition to deriving the actual types of tasks involved, the hierarchical relations required for the task specification can similarly be derived from the PROforma task decomposition. This decomposition is represented by specifications of the form shown below.

```
plan :: T1;  
    component :: T2;  
    component :: T3;  
end plan.  
  
action :: T2;  
    ...;  
end action.  
  
action :: T3;  
    ...;  
end action.
```

In this case T_1 is a task which is decomposed into two other tasks T_2 and T_3 , hence a tree structure in which T_1 dominates T_2 and T_3 can be inferred and supplied as the result of a request for a task hierarchy. Furthermore, a component can itself be a plan, hence allowing recursive structure to be represented. Similarly, dependencies between tasks can arise from task preconditions in two ways. First, a task may have a precondition that relies on a particular state being achieved by another task (i.e. dependency through control flow). This case would arise from a PROforma specification such as that given below:

```
enquiry :: T1;  
    ...;  
end enquiry.  
  
action :: T2;  
    precondition :: completed(T1);  
end action.
```

In this case, T_2 is a task that depends on task T_1 being in a completed state. A second form of dependency arises through data flow when a task is dependent on a data item being associated with a particular value and another task has responsibility for acquiring its value. This is represented in PROforma by a structure such as the following:

```
enquiry :: T1;  
  source :: d1;  
end enquiry.  
  
action :: T2;  
  precondition :: d1 = ...;  
end action.
```

Here, the task T_2 is dependent on T_1 because its precondition requires a value for the data item d_1 , which is the source for T_1 .

Specifying the Domain Ontology

L&C Ontology Browser

The domain ontology can be specified using the Ontology Browser provided by L&C and described in deliverable D9 (Ceusters et al., 2002). Using this system the information required by the task specification language, such as associations between concepts, concept classification and determining domains, can be derived as described below.

Mapping to the Task Specification Language

Associations between concepts can arise either from the fact that one concept subsumes the other or from the definition of some other (non-subsumption) relation between concepts. Hence a concept C_1 has an “IsA” association with a concept C_2 if it is inferable from the domain ontology that C_2 subsumes C_1 . This can arise directly from a specification such as that below in the ontology, expressed in L&C’s Cassandra notation (Ceusters et al., 1997):

```
! ((C1) {[IS_A] (C2) })
```

Alternatively, it may be inferred by the Ontology Browser from a chain of subsumption relations, e.g.

```
! ((C1) {[IS_A] (C0) })  
! ((C0) {[IS_A] (C2) })
```

Similarly, C_1 could be seen as being in a “PartOf” association with C_2 in the case that certain more specific mereological/topological relations are specified (or inferable) between C_1 and C_2 , e.g.

```
! ((C1) {[IS-PROPER-MATERIAL-PART-OF] (C2) })  
! ((C2) {[HAS-SPATIAL-POINT-REFERENCE] (C1) })  
! ((C1) {[IS-LINEAR-DIVISION-OF] (C2) })  
! ((C1) {[IS-ANTERIOR-OF] (C2) })  
! ((C2) {[HAS-PROPER-SPATIAL-PART] (C1) })  
...
```

Determining concept categories and domains could similarly be implemented by using the Ontology Browser to classify concepts. For example, it is possible to determine whether a concept can be classified by some upper model category such as ‘Material Entity’ by

searching the ontology to see if a given concept is subsumed by the concept MATERIAL ENTITY. For example, the concept CYST can be classified as a material entity by virtue of the following inference chain:

```
! ((CYST) {[IS_A] (SPACE OCCUPYING LESION)})  
! ((SPACE OCCUPYING LESION) {[IS_A] (ACQUIRED PATHOLOGICAL BODY  
STRUCTURE)}))  
! ((ACQUIRED PATHOLOGICAL BODY STRUCTURE) {[IS_A] (PATHOLOGICAL  
STRUCTURE)}))  
! ((PATHOLOGICAL STRUCTURE) {[IS_A] (HUMAN BODY STRUCTURE)}))  
! ((HUMAN BODY STRUCTURE) {[IS_A] (ORGANIC SOLID STRUCTURE)}))  
! ((ORGANIC SOLID STRUCTURE) {[IS_A] (SOLID STRUCTURE)}))  
! ((SOLID STRUCTURE) {[IS_A] (MATERIAL ENTITY)}))
```

Similarly, the domain of values that a concept can be associated with can be determined by using the Ontology Browser to try to classify a concept as a subclass of some more general concept for which the domain is known (as described previously).