Acknowledgements

First and foremost I would like to thank my supervisor Arne Jönsson who has guided me in the work that has resulted in this thesis. He has been a very active supervisor, always available for questions and engaging discussions. His enthusiasm and optimism has helped me focus my work and overcome obstacles. He has also read and provided valuable comments on a great number of drafts of this thesis.

I would also like to thank my secondary supervisors, Lars Degerstedt and Magnus Merkel, who through insightful discussions and comments have helped me bring order to my thoughts and the material presented in this thesis. Special thanks to Lars who spent much time with me on the formal parts. A special thanks also to Nils Dahlbäck who was my secondary advisor during the first part of my graduate studies.

I have been fortunate to conduct my thesis work in a very active research environment. I would like to thank all the members of NLPLab, who have contributed to the stimulating and positive atmosphere this thesis has been created in. Thanks to, Lars Ahrenberg, Bertil Lyberg, Robert Eklund, Genevieve Gorrell, Pontus Johansson, Anders Lindström, Pernilla Qvarfordt, Sonia Sangari, Lena Höglund Santamarta, Mustapha Skhiri and Håkan Sundblad. Thanks also to Frida Andén, Sara Norberg and Mats Andrén who did much of the work on the BIRDQUEST system.

When I have had problems with uncooperative computers or questions regarding technical matters, the technical staff at IDA has helped me, many thanks for this. Thanks also to Marie Ekström Lorentzon, Lise-Lott Andersson, Britt-Inger Karlsson, and Lillemor Wallgren who have handled the administration.

Finally, I would like to thank family and friends who, although they are not quite sure of what I really do, always have been supportive and provided encouragement.

This work has been supported by the Swedish Agency for Innovation Systems, VINNOVA, and the Center for Industrial Information Technology, CENIIT.

Contents

Chapter 1

Introduction

A natural language **dialogue system** is a computer system that primarily interacts with users by utilising connected unrestricted natural language dialogue in order to achieve a specific task or tasks. Connected means that it should be possible for both the user and the system to refer back to previous interaction and to ask follow-up questions, and unrestricted means that the user's means of expressions are not limited, for example, to a predefined set of commands or words.

Dialogue systems must be able to perform a multitude of tasks, such as interpreting requests and generating responses in natural language, handling dialogue phenomena, such as referring expressions, ellipsis and clarifications, as well as access and gather information from information sources. Most of these tasks require knowledge of the world that is under discussion. However, for a feasible system, the knowledge has to be restricted to those aspects that are useful for the task at hand; this part of the world is the **domain** of the system. A domain can be defined as, "a section of the world about which we wish to express some knowledge", (Russel and Norvig 1995, p.197). A common approach to represent such knowledge in the traditional knowledge representation (KR) research area in Artificial Intelligence

(AI) (Lakemeyer and Nebel 1992) is to store facts in knowledge bases complemented with reasoning mechanisms to derive new knowledge. This means focusing on declarative and explicit knowledge rather than procedural and tacit. In the former view, "knowledge consists of propositions, whose formal structure is the source of new knowledge" (Guarino 1995, p.628). The term **knowledge** will henceforth be used to denote this sense.

The most common type of task in publicly available dialogue systems is to provide information (cf. (Goddeau et al. 1994; Bennacef et al. 1996; Aust et al. 1995)). Typically, the user wants some information that is available in one or more information sources, and the system helps the user to construct an information request in natural language, which is specific enough for the system to be able to retrieve the information. The need for domain knowledge in such systems can be illustrated by a dialogue[1] between a user and a dialogue system that provides information on birds, see Figure 1.1.

The user begins with a question about owls. Since owls is a *family* of birds and information about features are related to *bird species*, domain knowledge is accessed and the instances of bird species that belong to the family owls are presented to the user. When the user has chosen a species, the next clarification involves the choice of a feature. Once more domain knowledge is used, this time to retrieve the features that are available for species.

The user asks for an receives an answer about *distribution*. Then information about *appearance* is sought. In the database used by the dialogue system, there is however no general information on appearance, but there is some information on *wingspan*, *length*, and various types of *plumages*. Thus, the system needs to know how the features available in the database are related to more general features in order to ask for clarification. Given that the user answers the clarification request the system can then map *size* to *wingspan* and *length*, and retrieve the requested information.

[1]The dialogues with BirdQuest presented in this thesis are translations from Swedish originals. The translations are made as closely to the original wording as possible thus sometimes resulting in ungrammatical constructs.

U1: What owls are there?
Vilka ugglor finns det?

S2: Choose one of the following Owls: Eurasian Pygmy Owl, Ural Owl, Tengmalm's Owl,..., and Eurasian Eagle Owl.
Följande arter tillhör familjen ugglor: sparvuggla, slaguggla, pärluggla,..., och berguv.

U3: Eurasian Eagle Owl
berguv

S4: You can receive information about a Eurasian Eagle Owl's appearance, flight, feed, latin name, breeding location,... Please specify what you are interested in.
Du kan få information om berguvs utseende, flykt, föda, latinskt namn, häckningsplats,... Vad vill du ha information om?

U5: Does it live in Östergötland?
Lever den i Östergötland?

S6: Distribution of Eurasian Eagle Owl: Live in mountains and forests in Northern, Eastern and Southern Europe, as well as big parts of Asia and Northern Africa.
Berguvs utbredning: Förekommer i bergs- och skogstrakter, både i Nord- , Öst- och Sydeuropa samt dessutom i stora delar av Asien och i Nordafrika

U7: Appearance
utseende

S9: You can receive information about the size and plumage of a Eurasian Eagle Owl. Please specify what you are interested in.
Du kan få information om storlek och dräkt. Vad vill du ha information om?

U9: size
storlek

S10: The wingspan of an Eurasian Eagle Owl is 188 cm.
An Eurasian Eagle Owl has a length of 75 cm.
En berguvs vingbredd är omkring 188 cm.
En berguv är 75 cm läng

U11: Does it live for more than 5 years?
Lever den längre än 5 år?

Figure 1.1: A dialogue between a user and the BIRDQUEST dialogue system that provides information on Nordic birds.

The dialogue so far has focused on how domain knowledge is essential for various tasks in dialogue management and access of information sources. Using the same type of domain knowledge also solves some problems in interpretation of ambiguous requests. For example, words can have several interpretations, e.g. 'live' (swe. leva) can mean a location where a bird lives, or the time span of a bird's life. In U5 and U6 the syntactic analysis produces identical interpretations with the only difference being that "Östergötland" is interpreted as a county and "more than 5 years" as a measure. The word 'live' is interpreted as both *oldest age* and *distribution* of which only one is appropriate to each question. With the use of knowledge regarding the type of objects that the features can be applied to, disambiguation is possible and the correct interpretation can be produced.

To incorporate more domain knowledge in order to make the dialogue system capable of more efficient dialogue is a trend in dialogue system research (cf. (Milward and Beveridge 2003; Dzikovska et al. 2003; Porzel et al. 2003)). Domain knowledge representation and reasoning can be included in dialogue systems in various ways and previously this type of knowledge has often been integrated with dialogue and discourse knowledge, for example, in frame representations of the relevant objects and properties to collect for a certain information request (Bennacef et al. 1996; Seneff et al. 1998).

Furthermore, since development of a usable dialogue system requires considerable effort, an important aspect when developing a dialogue system is portability; to be able to reuse and adapt the dialogue system to new tasks and domains for new applications in the future. To facilitate portability, application-specific aspects should be separated from generic features in dialogue system architectures. *The Domain-independence Hypothesis* proposed by Allen et al. (2000, p. 1) suggests that language processing can be separated from domain knowledge representation and reasoning:

> Within the genre of practical dialogue, the bulk of the complexity in language interpretation and dialogue management is independent of the task being performed.

The hypothesis is supported by Allen et al.'s work and implies that it is possible to create frameworks[2] for dialogue systems that can be used to create new applications by the addition of application-specific task and domain knowledge.

In order to create frameworks into which new domain knowledge can be easily incorporated, a uniform approach to represent and reason about domain knowledge is crucial. One promising approach is to use ontologies. Ontologies as a means of representing and supporting reasoning about domain knowledge are becoming increasingly common, and are used in several NLP-systems today, for example, Q/A-system (cf. (Harabagiu et al. 2000; Zajac 2001)), information extraction (cf. (Gazauskas et al. 1997)) and knowledge-based machine translation (cf. (Mahesh 1996)).

Ontologies have a long history but have become more prominent in computer science areas only over the past 20 years. They are closely related to conceptual modelling in database design, to domain modelling in software engineering and expert systems in AI.

An ontology holds information about what categories exist in the domain, what properties they have, and how they are related to one another (Chandrasekaran et al. 1999). (A more detailed discussion on what constitutes an ontology is presented in Chapter 2.)

There are several advantages to using ontologies as separate domain knowledge sources. Milward and Beveridge (2003) point out that using ontologies reduces the complexity of linguistic components and also that it facilitates the reuse of existing knowledge sources, since more and more ontologies are being produced in various domains. Ad-

[2]The term framework has been defined in various ways. In this thesis, I will adhere to the following characterisation by Cunningham (2000), which incorporates both conceptual and implementational aspects, but I emphasis the first:

> A framework, "is a reusable design for all or part of a software system", (Johnson 1997), made up of, "a set of prefabricated software building blocks that programmers can use, extend, or customise for specific computing solutions." (IBM 1999).

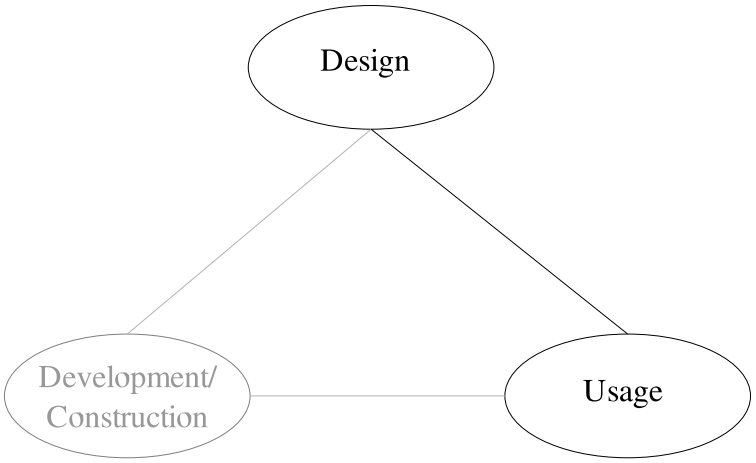

Figure 1.2: The main research areas in ontologies, design, development, and usage, and their interrelations. The solid lines and arrows mark the primary areas studied in this thesis.

ditionally, domain ontologies can facilitate system component communication in dialogue systems (Dzikovska et al. 2003).

Research on ontologies is broad, Figure 1.2 shows the main areas that are addressed and how they are related. **Design** includes issues such as what to represent, how the domain knowledge should be organised. It also includes development and use of design guidelines for ontologies. **Development and construction** deals with questions regarding capture, formalisation and implementation of domain knowledge. **Usage** looks at the functions the ontology can support and the role of the ontology in an application.

In this thesis, design and usage are in focus; there is however one aspect of development that should be stressed in the context of dialogue systems. An ontology developed based on information sources, such as texts and databases, will mirror the author's view of the

domain, which is often the expert's view. An ontology developed based on question or dialogue corpora will reflect the users' view of the domain, which may vary from novices to experts. An important issue for dialogue systems is therefore to integrate or map between the user- and information source-oriented ontology if they differ, see U5-S8 in Figure 1.1.

1.1 Issues and contributions

The goal of the work presented in this thesis is to facilitate development of information-providing dialogue systems[3] capable of domain knowledge reasoning with high portability. The approach taken is to investigate how ontologies can be used for this purpose. The research issues investigated can be divided into two main tracks related to, on the one hand, how ontologies can be used in dialogue systems, with regard to question analysis, dialogue management and domain knowledge management, and on the other hand, how they should be designed to support these tasks.

Issues

Usage of domain ontologies in dialogue systems

- What type of functionality in dialogue systems can ontologies support?
- How can ontologies be incorporated into a dialogue system framework to support high portability?

[3]Other types of dialogue systems, like problem-solving (cf. (Allen et al. 1995; Smith and Hipp 1994)), argumentation (cf.(Zukerman et al. 2000)), advisory and tutoring systems (cf. (Rangemalm 1996; Fried et al. 2003)), are not considered in this thesis.

Design of domain ontologies for dialogue systems

- How should ontologies be designed to support the domain reasoning necessary in dialogue systems?
- How can ontologies be developed to capture and integrate different views of a domain?

To investigate these issues, different approaches have been combined. Analyses of corpora and existing systems have been performed in parallel with iterative development of dialogue system applications and frameworks. The different approaches and the results from different phases are illustrated in Figure 1.3.

The right-hand side of the figure shows the two analyses that have been carried out in this thesis work. The first consisted of an analysis of various knowledge sources' functionality in existing dialogue systems and a corpus study. These were synthesised in a requirements specification of capabilities in terms of the type of knowledge sources necessary in order to achieve graceful and cooperative dialogue. The second study concerned design requirements of ontologies in dialogue systems and was based on analyses of the state of the art in ontology design, the present use of ontologies in dialogue systems, and corpora where users interacted with various systems or humans. The resulting synthesis specifies design choices and guidelines that support the design of ontologies for information-providing dialogue systems.

The left-hand side of Figure 1.3 illustrates the design and implementation work with iterative development of frameworks and applications. A framework includes generic features like models and specifications of knowledge sources to be used for, for example, interpretation and dialogue management, as well as code and design patterns (Degerstedt and Jönsson 2001). Frameworks are based on experiences from the development of applications and empirical studies, like corpora and Wizard Of Oz experiment (Dahlbäck et al. 1993). The quality and generality of a framework is evaluated through the development of applications for new domains. Shortcomings in a framework can be

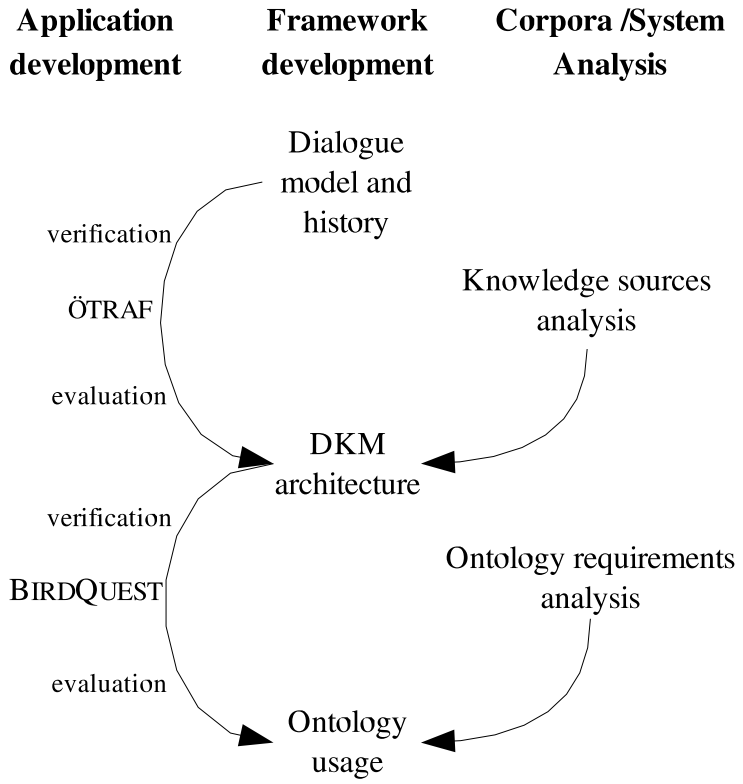

Figure 1.3: Approaches and results in the thesis research. Iterative development of frameworks is combined with analyses of corpora and other existing systems.

identified and resolved through the addition of new features in an application, which if proven useful, can be generalised and incorporated into a new framework.

The approaches have complementary strengths and weaknesses. The analysis of corpora and existing dialogue systems provides a broad base and a synthesis a good ground for future work, but since it is based on written accounts of systems it risks being superficial. The development of applications provides the details but without the generality. However, through the combination of approaches both depth and breadth can be achieved.

The first phase includes a change of dialogue system architecture where a separate module for domain knowledge reasoning, a Domain Knowledge Manager, is introduced (Flycht-Eriksson 2001). In the next, which is the focus of this thesis, domain ontologies are introduced. Two areas of research, ontologies and dialogue systems, where analysed and synthesised in order perform a requirements analysis for design of ontologies to be used in dialogue systems, and to create a new framework for use of ontologies in dialogue systems.

Contributions

A requirements analysis for ontology design

A compilation of design choices and design guidelines in the ontology research area has been made for the purpose of design of ontologies for dialogue systems. These have then been analysed for various tasks in dialogue systems, more specifically for interpretation, dialogue management, and domain knowledge management, based on empirical investigations of existing dialogue systems and corpora. The result is a design specification of ontologies that are to be used in dialogue systems.

A framework for use of ontologies in dialogue systems

A framework that supports domain knowledge reasoning utilising domain ontologies in dialogue systems have been developed based on the ontology design. It includes:

- A model for ontology-based semantic analysis of questions
- A model for ontology-based dialogue management, specifically focus management and clarifications
- A model for ontology-based domain knowledge management, specifically transformation of user requests to system oriented concepts used for information retrieval

1.2 Thesis outline

The thesis comprise eight chapter, with the first being this introduction. The remaining seven chapters can be divided in two parts. The first part deals with design of ontologies. In Chapter 2 the state of the art in ontology design and development is analysed and summarised in a number of design choices and guidelines to be considered during design of ontologies. Chapter 3 presents an analysis of how ontologies have been and are currently used in dialogue systems. The result is a collection of functionalities that ontologies can support in dialogue systems. The results from these two chapters are synthesised in Chapter 4 in an analysis of design requirements for ontologies used for interpretation, dialogue management and domain knowledge management in dialogue system.

The second part deals with usage of domain knowledge and ontologies in dialogue systems, with focus on dialogue system architecture. In Chapter 5 a dialogue system application, the ÖTRAF system, and a dialogue system framework are introduced. With ÖTRAF the first step towards achieving portable dialogue systems capable of domain knowledge reasoning is taken with the introduction of a Domain Knowledge Manager. This new architecture is generalised and incorporated into the MALIN framework. Based on MALIN a new application, the BIRDQUEST system, is developed, which includes a domain ontology. This system, and an evaluation that shows the potential of ontology usage in dialogue systems, is presented in Chapter 6. The lessons learned from the development and evaluation of

the applications and frameworks are concluded in Chapter 7, where a framework for usage of ontologies in dialogue systems that is based on the ontology design from Chapter 4, is presented.

Chapter 2 **Ontologies** This chapter answers the question, "What is an ontology?" and presents current issues concerning design and development of ontologies.

Chapter 3 **Ontologies in NLP, specifically dialogue systems** In this chapter, the present use of ontologies in Natural Language Processing, with the main focus on dialogue systems, is presented.

Chapter 4 **Design requirements on ontologies** This chapter presents an investigation of the requirements various tasks in dialogue systems pose on domain ontologies. It identifies a number of design choices and guidelines that should be considered during design and development of a domain ontology to be used in intelligent information-providing dialogue systems.

Chapter 5 **Domain knowledge and ontologies in ÖTRAF and MALIN** An overview of the ÖTRAF application and the MALIN dialogue systems framework is given in this chapter with a focus on dialogue and domain knowledge management issues.

Chapter 6 **Domain knowledge and ontologies in BIRDQUEST** In this chapter, an instance of an ontology-based dialogue system is presented. An evaluation of the system illustrates the potential of ontology use in information-providing dialogue systems and give some implications for further developments of ontology-based dialogue systems.

Chapter 7 **A framework for ontology-based dialogue systems** Based on the requirements from Chapter 4 and the experiences from development of previous frameworks and dialogue system applications, chapters 5-6, new models for question analysis, dialogue management and domain knowledge management based on ontologies are developed. These are presented and discussed in this chapter.

Chapter 8 **Summary and future work** In the final chapter, the results are summarised and future work is presented.

1.3 Relation to previous work

The work presented in this thesis was conducted in a research group and many persons have been involved, especially in the development of the systems described in chapters 5 and 6. ÖTRAF was designed and developed together with Lars Degerstedt, Pernilla Qvarfordt, Nils Dahlbäck, Arne Jönsson, Håkan Sundblad and Lena Höglund Santamarta. In this system I was primarily involved in work on dialogue and domain knowledge management (Flycht-Eriksson 1999; Flycht-Eriksson and Jönsson 2000; Flycht-Eriksson 2000; Flycht-Eriksson 2001). BIRDQUEST involved Frida Andén, Sara Norberg, Lars Degerstedt, Arne Jönsson and Magnus Merkel. My own contribution to this system was the design of the ontology and work on the design of the domain knowledge manager (Flycht-Eriksson and Jönsson 2003; Flycht-Eriksson 2003). Work on a question analysis module was done together with Håkan Sundblad (Flycht-Eriksson et al. 2003).

Throughout the thesis I will use *we* to describe joint work and *I* for work I have done on my own.

Chapter 2

Ontologies

This chapter addresses the question, "What is an ontology?", and the issues of design and development of ontologies.

Two of the three ontology research areas presented in Figure 1.2, design and development, are addressed in this chapter. It aims to provide the basic terminology needed to discuss and analyse ontologies (Section2.1), and analyse the state of the art in ontology design (Section2.2) and ontology development (Section2.3) from a dialogue system point of view. The theories presented in this chapter will form the basis for the analysis of design requirements presented in Chapter 4.

2.1 What is an ontology?

Ontologies are used for various purposes in computer science, for example, knowledge sharing, artificial intelligence, natural language processing and the semantic web. This diversity is reflected in the variety of answers to the question, "What is an ontology?".

In the International Dictionary of Artificial Intelligence (Raynor 2000, p. 213) an ontology is described as, *"A particular theory or model about the nature of a domain of objects and the relationships among them"*.

The most general and commonly used definition is given by Gruber (1993a), who states that, *"An ontology is an explicit specification of a conceptualisation"*.

The keyword in this definition is **conceptualisation**. A conceptualisation is an abstract simplified view of a domain; it identifies the concepts relevant in representing the domain and their interrelationships (Genesereth and Nilsson 1987). An ontology describes this conceptualisation by making the concepts **explicit**.

This definition has been discussed, criticised and elaborated on. Since axioms used to represent an ontology can only approximate the intended models and these, in turn can only approximate a conceptualisation Gangemi et al. (1999, p. 4) suggest the following more refined definition, *"A partial specification of the intended models of a logical language"*.

Guarino (1998a) is critical of Gruber's definition since conceptualisation refers to extensional relations. He argues that it is the intensional relations that are important. He presents a new definition that differentiates between ontology and conceptualisation:

> An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualisation of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualisation) by approximating these intended models.

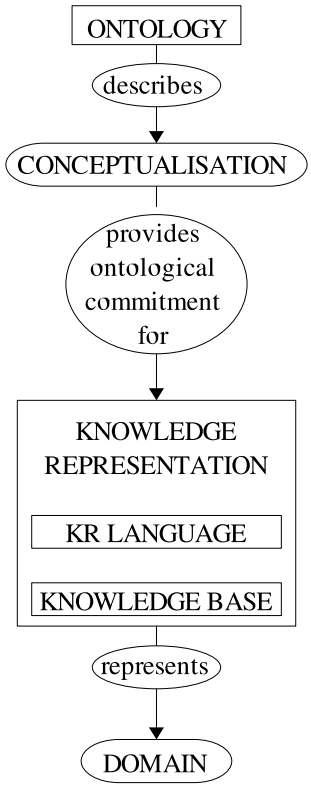

Figure 2.1: Relations between ontologies, conceptualisations, knowledge representations and domains.

Valente and Breuker (1996, p. 3) describe the relation between ontologies, conceptualisations, knowledge representations and domains as shown in Figure 2.1. Domains and conceptualisations are abstract entities while knowledge representations and ontologies are concrete artifacts that are explicit representations of the former. An ontology defines the concepts that are captured by a conceptualisation, and a knowledge representation describes the domain by a set of sentences formulated in a knowledge representation language.

The concepts captured by a conceptualisation describe a vocabulary. To use such a vocabulary is an **ontological commitment**. Thus the conceptualisation influences the knowledge representation through the ontological commitments it provides.

All the definitions presented above are rather philosophical in nature. A more pragmatic definition that stresses that an ontology to be used for practical NLP is constructed for a specific situation is given by Mahesh and Nirenburg (1995). This definition is more concerned with the artifact notion of ontologies that are constructed and used as tools.

> A situated ontology is as a world model used as a computational resource for solving a particular set of problems.

I will follow this latter view and consider an ontology to be a computational resource with a definition of concepts, and their interrelations.

2.1.1 Ontologies, knowledge bases and databases

In many cases, a distinction between the ontology and the knowledge base is not made. For example, Chandrasekaran et al. (1999) write that in AI an ontology is often considered as a representation vocabulary or/and a body of knowledge.

A way of distinguishing ontologies and knowledge bases is to examine the extension of the entities they represent (Kishore et al. 2003). Concepts in an ontology capture the intension of a discourse universe, i.e. provide symbols for the concepts in the universe. The extension of these are categorical instances, represented in a knowledge base. These categorical instances lack existence in time and space, thus they are not real instances or individuals. The real instances are not part of the ontology but the definition of them can be based on it. Facts about instances and individuals are typically stored in a database.

An example of these relations are given in Figure 2.2. Disease and bird species represent general concepts that are part of ontologies. These can be instantiated by the *categories* flu and raven, respectively. They lack temporal and spatial extension. These can in turn be instantiated as individuals that exist at a specific time and place, for example a person suffering from the flu, or Hugin and Munin.

Thus, the relation between ontology, knowledge base and database can be described as a hierarchy where the ontology provides abstract concepts, the knowledge base holds categorical instances that are extensions of the concepts, and the database contains real instances that are extensions of the categorical instances.

Ontology	Knowledge Base	Database
disease	flu	A patient suffering from the flu at a specific time and place
bird species	raven	Hugin and Munin

Figure 2.2: Relations between ontologies, knowledge bases and data bases.

2.1.2 Ontologies and natural language

Ontologies provide meaning for vocabularies. Bateman (Bateman 1991) describes three types of ontologies that differ in how they relate to language. **Conceptual ontologies** are non-linguistic language independent world knowledge representations. **Interface ontologies** serve as middleware between conceptual ontologies and language dependent resources like grammars and lexicons. **Mixed ontologies** are linguistically oriented world knowledge representations that provide semantics for grammar and lexicons.

Two kinds of architectures can be based on these, either a mixed ontology is used, or an interface ontology is used in combination with a conceptual ontology. Bateman (1991) argues for the latter approach

which has been applied in the development of the PENMAN natural language generation system that utilise the Upper Model as an interface ontology (Bateman 1990). The Upper Model is used to classify the application knowledge in general semantic categories. These are then used for a systematic mapping to surface linguistic forms.

The other approach that utilise a mixed ontology is exemplified in Ortiz et al. (2002). They propose an architecture consisting of three components: 1) Static knowledge sources include an ontology, a fact database, a lexicon and an onomasticon, i.e. a lexicon of names, 2) knowledge representation languages used for the various knowledge sources, and 3) processors for NLP tasks, like syntactic and semantic analysis, text generation, etc.

The semantics of the entities in the lexicon are described by direct or constrained mapping to concepts in the ontology. A direct mapping means that there is a concept that corresponds to the semantics of the word. When a direct mapping is not possible, i.e. there is no concept that exactly matches the semantics of the word, a constrained mapping is done. The concept that most closely captures the meaning is used and modified through changes to its properties.

2.1.3 Types of ontologies

As the previous section shows there are a number of different definitions of ontologies. Existing artifacts that are called ontologies vary between simple object type hierarchies through frame-systems to complex logic based knowledge representation systems (Smith and Welty 2001). There are a number of dimensions that can be used to characterise and categorise ontologies, for example: **Informal - Formal**, **Content - Mechanism**, and **General - Application-specific**, as further discussed below.

Ontologies can have very different characteristics, from being highly informal (expressed loosely in NL) to rigorously formal (terms with formal semantics, theorems and proofs)(Ushold and Gruninger 1996).

This distinction is also called terminological vs axiomatic ontologies (Sowa 2000). Most ontologies focus on declarative knowledge and represent content theories. Ontologies can also be used for procedural knowledge about tasks and methods, these are termed mechanism ontologies (Chandrasekaran et al. 1999). Ontologies are constructed for various purposes and with different scopes. Some attempt at general knowledge sources that capture common sense knowledge while others have a very specific application in mind (Gangemi et al. 1999).

Guarino (1998a) distinguishes four kinds of ontologies based on the last dimension. **Top-level** ontologies include general concepts like time and space, objects, events, etc. This type of ontology is domain-independent and should therefore be applicable to all problems and applications. **Domain and Task** ontologies capture, respectively, a generic domain or a generic task. Either of these types can be constructed by a specification of concepts in a top-level ontology. **Application** ontologies are both domain and task specific. These can be constructed through a specification of a set of domain and task ontologies related to the application.

This classification is expanded by Gangemi et al. (1999), who also includes: **Representation** ontologies, sometimes called meta-ontologies, specify the knowledge representation formalism used to create more specialised ontologies. **Generic** ontologies, capture the most basic aspects of an ontology, for example, part, cause, participation, representation. **Intermediate** ontologies, represent general concepts in a domain and act as an interface between generic and domain ontologies.

2.1.4 Ontologies and Ontology

The term **Ontology** has a long history. Originally it was used in philosophy to denote the study of what exists in the world. Webster's Third New International Dictionary (Gove and Merriam-Webster 2000) gives the following definition:

> 1. A science or study of being: specifically, a branch of metaphysics relating to the nature and relations of being; a particular system according to which problems of the nature of being are investigated; first philosophy.
> 2. A theory concerning the kinds of entities and specifically the kinds of abstract entities that are to be admitted to a language system.

The first definition pertains to the traditional philosophical meaning of the term ontology. Since the 1980s the term ontology has been used in computer science with a somewhat different sense (Smith and Welty 2001), illustrated by the second entry above and the discussion above.

Two differences in philosophical and computational ontologies are the goal and scope of the ontology. Philosophers strive to discover the nature of reality and represent it in one ontology that is valid for all reality (cf. Lenat 1995; Sowa 2000.). Researchers in computer science construct several ontologies that can be used as tools by computers and humans for specific tasks. These ontologies only capture a limited part of reality (Kishore et al. 2003).

In computer science, ontologies deal primarily with meaning rather than existence. An ontology can be considered to be a representation vocabulary and a set of facts expressed within this (Chandrasekaran et al. 1999). This distinction is also made by Guarino (1998a):

> In the philosophical sense, we may refer to an ontology as a particular system of categories for a certain vision of the world. As such, this system does not depend on a particular *language* /../ in AI, an ontology refers to an *engineering artifact*, constituted by a specific *vocabulary* used to describe a certain reality, plus a set of explicit assumptions regarding the *intended meaning* of the vocabulary word.

In this thesis, ontologies refer to the computer science notion, i.e. the AI part of the definition by Guarino, and not the philosophical meaning of ontology.

2.2 Design of ontologies

The design of ontologies concerns the type of knowledge to represent and how this knowledge is organised. A good design requires careful consideration of several design issues, and that choices are made explicitly so that their consequences can be seen. In this section I raise a number of such issues and compile a list of design choices that can serve as a foundation for design of ontologies in dialogue systems. I also present an analysis and compilation of design guidelines that address the issues and choices proposed by various researchers. The result is thus a subset of general ontology design that is applicable for dialogue systems.

2.2.1 Design choices

In their framework for comparison of ontologies Noy and Hafner (1997) distinguish three levels for the content of an ontology: the taxonomy organisation, the internal structure of and relations between concepts, and axioms. Knowledge of instances and facts are usually not considered part of an ontology but rather a knowledge base. They are however closely linked to the ontology and in practice it is useful to consider the design of the knowledge base at the same time as the ontology. These four issues will be discussed and a number of design choices from the frameworks presented by Noy and Hafner (1997) and Corcho and Gómez-Pérez (2000) will be discussed and synthesised.

Entity types and properties

Looking at the anatomy of ontologies one finds that there are some general issues that are addressed in ontologies (Chandrasekaran et al. 1999):

- There are *objects* in the world
- Objects can have *attributes* that can take *values*
- Objects can exist in various *relations* with each other
- Attribute and relations can change over *time*
- There are *events* that occur at different *time instances*
- There are *processes* in which objects participate and that occur over time
- The world and its objects can be in different *states*
- Events can *cause* other events or states as *effects*
- Objects can have *parts*

There are thus four types of entities to represent in ontologies, objects, events and processes, attributes, and relations. A distinction between the first two and the latter are often made. Mahesh (1996) classifies objects and events[1] as free-standing entities that are defined through their attributes and relations. For the representation of these in an ontology I will use the term **entity type**. Attributes relate an entity with a numerical or literal constant while relations hold between two entities. I will use the term **property** as a common type

[1]The distinction between objects and events is also called continuant and occurant or endurant and perdurant (Gangemi et al. 2003). Of course the question of what constitutes an object or an event/process depends on the time perspective used. For example a tree is usually considered an object but can in some cases be viewed as a process.

for attributes and relations. Altogether, I will use the term **concept** to denote entity types and properties.

Guarino (1998b) makes a distinction between particulars and universals. Particulars are entities of the world, i.e. entities that can have instances, and universals are the attributes and relations used to describe the particulars, which cannot have instances. Thus particulars corresponds to entity types and universals to properties.

Concepts in ontologies can vary from nearly *atomic* to highly *structured*. The internal structure of concepts can be used to represent various attributes of and relations between entity types, for example, the participation of objects in events and processes, and part-whole relations between objects (Noy and Hafner 1997). Concepts can also be atomic either because there are no properties related to the objects and events, or because properties are represented in a separate taxonomy and related to the objects and events through restriction on the possible domain and range. The latter approach is taken, for example, in the semantic web (W3C 2004c). Mahesh and Nirenburg (1995) advocate a hybrid approach where objects have internal structure but properties also are represented separately from them in the ontology. They mean that it is the rich *inner structure* that allows for a sophisticated use of the ontology, for example, to perform disambiguation.

The most basic type of properties is *local*, i.e. they belong to a specific entity type, for example age to person. *Polymorphic* properties can have the same name but be applied to different types of objects and values, for example author of a book is a person while author of a thesis is a student. Properties can also be *instance* oriented, i.e. they allow for different values for each instance of an entity type, or *class* oriented, i.e. they have the same value for all instances of an entity type (Corcho and Gómez-Pérez 2000).

The last distinction is similar to *own* and *inherited*, where own means that a property is not inherited. This can be useful if a property is used for meta-information about a entity type or to represent synonyms (Noy et al. 2000).

Default values for attributes are values which are used if no other explicit value is present. *Type constraints* indicate the allowed type of values for an attribute. Restrictions on the number of values an attribute can have is called *cardinality constraints. Operational definitions* of values allow formulas or rules that are applied during runtime to calculate the value of an attribute (Corcho and Gómez-Pérez 2000).

Most existing ontologies use only binary relations. Since it is possible to transform *n-ary relations* to binary, only binary relations are required. However, n-ary relations might be useful since they allow representations that are more expressive and more straightforward to use. *Domain and Range restrictions* are used to constrain the type of objects relations are applicable to. If properties are represented explicitly and separately from entity types these types of restrictions are necessary to link the properties to the appropriate object types.

How and if time and space are treated in ontologies varies greatly. Some ontologies include time points and time periods as temporal objects (Noy and Hafner 1997). These temporal and spatial objects typically have an internal structure where temporal or spatial aspects are properties or roles. Usually a frame-type representation with slots and fillers are used (Gennari et al. 2002). Guarino (1998b) introduces the quality location as a type that includes regions of space and intervals of time. These specify the extension of a concrete object in time and/or space. Santos and Staab (2003) propose that temporal aspects should be factored out and represented in a separate ontology that can be re-used. The temporal ontology is linked to a time-less domain ontology through an operator that assembles an object and properties with temporal restrictions. Similar proposals for the representation of space exist based on, for example, geographical locations and Region Connection Calculus (RCC) (Grenon 2003).

The use of *part-whole* relations to define ontologies varies greatly. In some ontologies this type of relation is not present at all, and in others there are several subtypes. Winston et al. (1987) differentiate between six types of part-whole relations: component-object, stuff-object, portion-mass, member-collection, place-area, feature-activity

and phase-process. The classification is based on whether the part is functional, homeomerous, or separable from the whole, i.e. if it is spatially and/or temporally restricted to contribute to the function of the whole, if it is the same kind of thing as the whole, or if it can be separated from the whole. Another suggestion of categorisation based on compositional structure is given by Gerstl and Pribbenow (1995), who differentiates between masses that have quantities, collections that have members and complexes that have components. They also include two other part-whole relations based on external criteria, segment and portions, which are constructed through the application of a schema or a property dimension.

Taxonomy organisation

Most ontologies have an explicit hierarchy of the objects, a taxonomy. Noy and Hafner (1997) describe three different approaches. The most common is the use of one tree-like taxonomy where IsA links relate categories of objects with disjoint sub-categories. Multiple inheritance allows a category to have several super-categories. Another approach is to use parallel dimensions on the top-level and create sub-categories through combinations of these.

Figure 2.3 shows two different approaches. A traditional tree is used in WordNet (Miller et al. 1993), where concepts are divided in sub-concepts, for example, distinguishing between living and non-living. Parallel dimensions are used in Dahlgren's ontology (Dahlgren 1995), which characterise concepts along several axis, for example, individual/collective and abstract/real. A third approach is to have several small taxonomy islands that can be related to each other. There are also other ways of organising ontologies, for example, using atomic concepts that are used to construct more complex concepts in a bottom-up approach (van der Vet and Mars 1998).

The IsA relation is the most common way of forming a taxonomy, but there are a number of other possible relations that can be utilised to organise the concepts. *Partitions* can be used to define disjoint

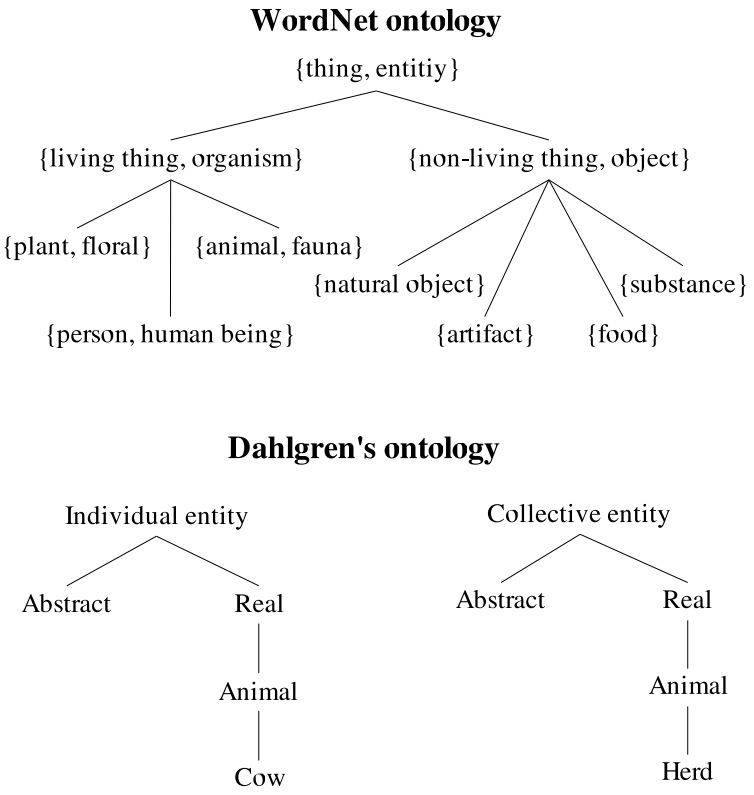

Figure 2.3: Two types of taxonomies, the tree in WordNet and parallel dimensions in Dahlgren's ontology.

classes. *Disjoint decompositions* describe disjoint sub-classes and *exhaustive sub-class decomposition* disjoint sub-classes that make up all possible sub-classes of a class. There is also a negation of SubclassOf, *NotSubclassOf* that can be useful to state that a class not is a specialisation of another class.

DOLCE	MICROKOSMOS
Endurant	Object
Physical	Physical-object
Non-physical	Mental-object
	Social-object
Perdurant	Event
Eventive	Physical-event
Stative	Mental-event
Social-event	
Quality	Property
Attribute	
Relation	
Abstract	

Figure 2.4: The top-level divisions of the DOLCE and Microkosmos ontologies.

For most ontologies *monotonic* and *simple* inheritance are the basis, i.e. values can be inherited from one parent with no contradicting information in sub-classes. *Multiple* inheritance must primarily be dealt with in ontologies whose taxonomy is organised based on multiple dimensions and categories are created through combinations. This means that several values can be inherited from different parents and that conflicts must be resolved. *Non-monotonic* inheritance is more often present when default values are used, which can be overridden by specific values in sub-classes.

The taxonomies can differ in the top-level categories they use. Some categories are very common, for example, abstract/concrete and object/process. Sowa (2000) who presents a top-level that is based on philosophical foundations includes these aspects, and categories are

constructed through combinations. Other examples for top-level distinctions are used in DOLCE (Gangemi et al. 2001; Gangemi et al. 2003), which also is inspired by philosophical principles but with a clear orientation toward language and cognition, and Microkosmos (Mahesh 1996), which is a large ontology developed for machine translation, see Figure 2.4. Although the categories have different names they are quite similar with endurant corresponding to object, perdurant to event and quality to property.

Gangemi et al. (2001) outline a methodology for the design of top-level categories based on formal ontology. Formal relations such as instantiation and membership, parthood, connection, location and extension, and dependence are used as a basis to form formal properties. The formal properties include the common concrete versus abstract, individual versus collection, but also dependence versus independence, and extensionality. They propose that based on combinations of the formal properties a base set of categories can be constructed and serve as a back-bone for domain analyses.

Axioms

Sometimes all relevant information in an ontology is captured by concepts and the taxonomy, or additional information might be included in the application code. However, some ontologies use explicit representation of axioms, for example, to represent constraints on property and role values. Other kinds of information are defaults, contexts, modal or uncertain information. For this purpose, an extension of first order predicate logic is often used (Noy and Hafner 1997).

Staab and Maedche (2000) list a number of categories for axioms that can be used to specify most types of axioms in ontologies. Some of them refer to taxonomic relations of concepts, *(Exhaustive) partitions* and *Non-monotonicity*. The latter refers to a situation in which inheritance is overridden when a local value contradicts the value of a superclass. Many of the other types of axioms deal with relations, for example, axioms for *Relational algebra* define properties of relations,

such as reflexitivity, symmetry, transitivity and inverse. Axioms for the *Composition of relations* allow for constructions like GrandFather is composed of FatherOf and ParentOf. *Sub-relation relationships* can also be used to describe relations among relations.

The two remaining types of axioms deal with *Part-whole reasoning* and *Temporal and modal contexts*. In the CYC ontology (Lenat 1995), one can find examples of other types of axioms, *Certainty*, which states the likelihood of a certain assertion being true, *Reification*, which turns a predicate into an object and allows statements about categories, *Contexts*, which constrain the contexts in which an assertion can be true.

Staab and Maedche (2000) propose that the categories be used as a basis for language-independent representation of axioms in ontologies. These are then realised in the underlying representation language and inference machinery. This separation would make it possible to more easily design and understand the meaning of axioms.

Many features represented by axioms can also be represented by slots in frames, thus axioms concerning taxonomic and property information have often been linked to concepts (Noy and Hafner 1997). For other types of axioms, independent axioms are usually used.

Instances/Facts

Individuals are instances of categories, facts are relations that hold between elements, and claims are facts asserted by an individual. The first two types should be considered as universal expressions of truths, but a claim made by one individual might be inconsistent with claims made by others. While individuals and facts are common in knowledge bases today, claims have more recently become more important with the development of the semantic web (Corcho and Gómez-Pérez 2000).

The design choices for ontologies presented in this section are summarised as 14 questions (DC1-D14) in table 2.5.

Design choices for ontologies	
Entity types and Properties	
DC1	Do the concepts have internal structure or are they atomic?
DC2	What type of properties are allowed? Local, polymorphic, instance, class, own, inherited?
DC3	What facets of attributes are allowed? Default values, type constraints on values, cardinality, operational definitions?
DC4	What type of relations are allowed? Binary or n-ary, domain restrictions, range restriction?
DC5	How are time and space treated?
DC6	How is part-whole relations treated?
Taxonomy organisation	
DC7	What is the general organisation of categories? Tree, multiple dimensions, distributed?
DC8	What type of relations between categories are allowed?
DC9	What kind of inheritance can be dealt with? Monotonic, non-monotonic, simple, multiple?
DC10	What top-level divisions are there?
Axioms	
DC11	What type of axioms are allowed?
DC12	Are axioms independent or linked to entity types or properties?
DC13	How are axioms expressed?
Instances	
DC14	What types of instances are allowed? Individuals, facts, claims?

Figure 2.5: A compilation of design choices to consider for design of ontologies for dialogue systems.

2.2.2 Design guidelines

Design issues raise a number of questions to be answered during design. There are a number of collections of guidelines (Gruber 1993a; Guarino 1998b; Valente and Breuker 1996; Gómez-Pérez 1999; Noy and McGuinness 2001; Mahesh 1996) that partially try to answer these questions. These guidelines vary from very general to specific and are developed for different types of ontologies. Here a compilation of the ones applicable to the design of domain ontologies are presented and discussed.

Concepts

Most of the design guidelines concern the definition of entity types and properties. Gruber (1993a) proposes that definitions be objective and complete with both necessary and sufficient conditions. Guarino (1998b) also advocates the use of identity criteria that represent necessary and/or sufficient conditions as essential for making ontological distinctions. Since it is hard to define both necessary and sufficient conditions he says that sufficient conditions are enough in most cases.

Minimal ontological commitment means that the ontology should represent the weakest theory by modelling only the essential terms, thus allowing various instantiations and specialisations of the ontology (Gruber 1993a). Noy and McGuinness's guideline of complete class hierarchy agrees that not all possible properties and distinctions should be included, only those salient for the application. The number of levels of concepts should be based on the application, one level more specific and more general can be useful but not more. The latter guideline is not as strict on minimising ontological commitment but it is restrictive.

Coherence in definitions means that they should be logically consistent and sanction inferences that are consistent with the definitions (Gruber 1993a). The basic categories should not only be com-

plete and consistent, they should also contribute to a framework that makes sense by itself (Valente and Breuker 1996).

Guarino (1998b) states that it is important to be clear about the domain, i.e. of what type the different entities are. He differentiates between particulars, universals and linguistic entities, e.g. nouns, verbs or adjectives. Particulars and universals should be represented in two different ontologies.

Some examples of criteria for the definition of a new category for a concept are given by Mahesh (1996) and Noy and McGuinness (2001). For example, do not decompose concepts if not necessary for the application. Try to introduce a distinction/property before a new class for similar concepts (Mahesh 1996).

A new category should be introduced if additional properties or restrictions are needed to describe a concept, if it is a concept in the domain experts characterisation of a domain, if a distinction is important and is regarded as a different object or if it forms a natural hierarchy together with other concepts Noy and McGuinness (2001).

Since roles and attributes are non-rigid they cannot serve as identity criteria and should therefore be separated from the taxonomic categories, i.e. two separate ontologies for entity types and properties should be used (Guarino 1998b).

To achieve generality and facilitate reasoning, properties should be placed at the highest possible level. If a property is applicable to most of the siblings it should be placed at the parent level (Mahesh 1996).

An inverse relation should be included if a new relation is defined (Mahesh 1996; Noy and McGuinness 2001). Using inverse relations means storing redundant information since one relation can be deduced from the other. However, for knowledge acquisition purposes it can be useful since it allows more flexibility for the user to provide the information.

Entity types should be separated from instances, where the latter belong to a knowledge base rather than to the ontology. Some rules of thumb to decide if something is a concept or an instance are: If an entity can have its own instances it is a concept. If it has a fixed time and space it is an instance (Mahesh 1996).

Names should be standardised, and Mahesh (1996) and Noy and McGuinness (2001) give examples of some detailed criteria for naming concepts that should be considered. For example (Noy and McGuinness 2001), polysemy in concept names, use of cases to distinguish entity types and properties, use of delimiters, use of singular or plural in class names, use of abbreviations, use of superclass names in a subclass name. Mahesh (1996) gives a similar selection of suggestions for naming concepts that also include, use of words whose sense is similar to the concepts, use of the most frequent sense as the baseline concept and extensions of the name for the other, for example BANK and RIVER_BANK, differentiate entity types and properties, for example, EMPLOYEE and EMPLOYED_BY, EMPLOYER_OF, use of scientific terms rather than lay terms.

Taxonomy

Some guidelines relate to the organisation of the concepts in taxonomies. Guarino (1998b) proposes that a tree with mutually disjoint categories with different identity criteria should serve as a basic backbone for structuring taxonomies. Noy and McGuinness (2001) propose that hierarchical relations should be transitive and that class cycles should be avoided. They also state that disjoint sub-classes allow better ways for validating an ontology.

A diverse upper level facilitates introduction of new concepts through multiple inheritance since new concepts can easily be defined using pre-existing ones (Gómez-Pérez 1999). Noy and McGuinness (2001) agree that multiple inheritance should be supported to facilitate the addition of new concepts.

Similar concepts should be grouped together as sibling sub-classes while dissimilar concepts should be further apart (Gómez-Pérez 1999). There should be an appropriate number of siblings, only one sub-class indicates a modelling problem or an incomplete ontology, and too many (more than twelve) siblings indicates the need for intermediate levels (Noy and McGuinness 2001).

Expressitivity and maintenance

There are a small number of guidelines that deal with expressitivity and maintenance of ontologies. Gruber (1993a) advocates that it should be possible to extend and specialise the ontology monotonically, i.e. define and add new categories without revising existing definitions. This guideline is however contradicted by Noy and McGuinness (2001), who says that the evolution of an ontology might require a reexamination of classification criteria. Valente and Breuker (1996) also point at the problem with parsimony, being parsimonious with data supports processing and total economy, it may however make maintenance and modification more difficult.

Reuse should be facilitated through modularisation of ontologies if possible (Gómez-Pérez 1999). To further facilitate reuse and maintenance representation choices based on underlying notation or implementation should be avoided (Gruber 1993a).

The guidelines (DG1-DG16) are summarised in Figure 2.6[2]. Together with Figure 2.5, this cover the aspects to consider for design of ontologies in dialogue systems. In the next section we turn to development and construction of ontologies.

[2]I have reformulated them in a uniform format and present them starting with the most general, moving to the more specific.

Guidelines for design of ontologies	
Concepts	
DG1	Use identity criteria for definitions/ontological distinctions
DG2	Minimise ontological commitment
DG3	Strive for coherence in definitions
DG4	Differentiate various types of entities in the domain
DG5	Use well-defined criteria for definition/introduction of entity types
DG6	Make role and attributions explicit
DG7	Place properties at a general level
DG8	Define inverse relations when a new relation is introduced
DG9	Separate entity types from instances
DG10	Standardise names of entity types and properties
Taxonomy	
DG11	Organise entity types and properties in trees with (mutually) disjoint categories
DG12	Allow multiple inheritance
DG13	Use an appropriate number of sibling sub-classes
Expressitivity and maintenance	
DG14	Support extendibility
DG15	Facilitate reuse through modularisation
DG16	Avoid encoding bias

Figure 2.6: A compilation of design guidelines to consider for design of ontologies for dialogue systems.

2.3 Development of ontologies

There are no standard methods for the development of ontologies. "Building ontologies is still a matter of craft rather than an understood engineering process" (Jones et al. 1998, p.13). There have however been a number of development methodologies proposed for ontologies of various types and sizes and a selection of these will be presented and discussed in this section, together with some examples of tools and editors.

Depending on available resources and the purpose of an ontology there are several approaches to the development of ontologies (Holsapple and Joshi 2002). The **inspirational** approach means that the developer starts from his or her personal views of the domain and use imagination and creativity to create an ontology. In the **inductive** approach specific cases of the domain are analysed as a basis for ontology development. In the **deductive** approach general principles are used to filter and distill a general source and filling in gaps so that the result reflects a specific domain subset. In the **synthetic** approach the developer unifies a number of specific domain/application ontologies to create a more general domain ontology. The **collaborative** approach involves a number of persons or organisations that collaborate to create an ontology that incorporates multiple views of a domain.

A presentation of the trends in the methods that are presently used is given by Jones et al. (1998). Methods tend to be **Task oriented**. Taking a task as a starting point helps focus the acquisition and evaluation but also restricts the potential reuse. The methods can be categorised as **Stage-based or Evolving prototype**. The suitability depends on whether the purpose and requirements are clearly defined from the beginning. Methods also include both **Informal and Formal descriptions**. The different descriptions are produced at different stages in the development process. The transition between these bridges the gap between an application and the world it models. Another trend is the creation and use of **Ontology libraries**. A collection of ontologies could help the development of future ontologies. Finally, **Guidelines** are being introduced to guide the choices made at various stages during development.

2.3.1 Methodologies and tools

Some of the most mature development methods have been the result of large projects that develop a specific type of ontology or a tool. These methods are briefly outlined and compared in this section.

Enterprise

Ushold and Gruninger (1996) focus on ontologies that can be used for shared understanding in an area, thus a collaborative approach is used. The method and techniques they propose are based on their experiences with the development of the Enterprise ontology (Uschold et al. 1995). The skeletal development method consists of five steps.

1. Identify purpose and scope.
2. Build the ontology
    - ontology capture
    - ontology coding
    - integration of existing ontologies
3. Evaluation
4. Documentation
5. Guidelines

The first step includes defining the reason for constructing the ontology and characterise the intended usage. The main part of the work is done in step two where terms are identified, defined, coded and integrated with other existing ontologies. The first part, ontology capture, can be performed using an informal method consisting of four phases. Phase one, scoping, involves brainstorming followed by grouping. Phase two, producing definitions, is the most important phase. Some guidelines advocate starting with the trickiest terms that are closely related and where ambiguities might arise, and to use a middle-out approach where basic terms are defined first and generalisations and specialisations are treated last. Phase three, review, means to iterate over the definitions and revise them if necessary. Phase four, meta-ontology, involves the creation of a meta-ontology based on the definitions of central terms.

After ontology capture, the next part in building an ontology is coding. The basis for the implementation work is that a meta-ontology and representation language are chosen. If there are existing ontologies that complement the terms produced during ontology capture they can be integrated, however, the question of ontology integration is complex since the underlying assumptions of ontologies and representation formalisms vary. Once the ontology is built the next step is to evaluate it. The evaluation can be performed at different levels: the definition of the ontology, the representation and implementation of the ontology or the documentation of the ontology. Evaluations of the definition may include the structure and architecture, if the ontology adheres to design principles, the syntax, the content, consistency, completeness and conciseness. The evaluation can also focus on whether or not the ontology provides sufficient support for the tasks performed (Gómez-Pérez 1995). When an ontology has been evaluated it should be documented, making explicit underlying assumptions and the meta-ontology that has been used. Finally, a methodology should include guidelines for all of the preceding steps, see Section2.2.2.

TOVE

A similar type of ontology is TOVE (TOronto Virtual Enterprise). The method, developed based on experiences with this ontology, relies on the use of formal specifications (Grüninger 1996; Grüninger and Fox 1995). It focuses on steps 1-3 of the Enterprise method. Purpose and scope are captured in *motivating scenarios* that are formulated based on examples and problems from the domain and *informal competency questions* that pose requirements on the ontology. These are based on the motivating scenarios. *Terminology specification* of concepts, usually in a formal language like first order logic, corresponds to ontology capture. Coding is done using *formal competency questions* that are formalisations of the informal competency questions expressed in the same representation language as the ontology, and *axiom specifications* that define and constrain the interpretation of the entities in the ontology so that it expresses the formal compe-

tency questions. *Completeness axioms* are used for the evaluation of the ontology in terms of the situations in which the formal competency questions can be answered.

METHONTOLOGY

The METHONTOLOGY framework widens the perspective and looks not only at development but also at the entire life cycle of an ontology (Blazquez et al. 1998).

The ontology development process includes the activities performed during the development of an ontology. Three categories are considered: project management, development-oriented and support. The development-oriented activities are specification, conceptualisation, formalisation and implementation. Specification corresponds to the identification of scope and purpose in the Enterprise model, conceptualisation corresponds to identification and the definition of terms, implementation corresponds to the coding phase. Formalisation is a step between conceptualisation and implementation not found in the Enterprise development model but is similar to axiom specification in TOVE. Support activities include knowledge acquisition, evaluation, integration and documentation.

The ontology life cycle identifies the stages the ontology goes through and how these are related to the activities. Before ontology development begins *Planning* is performed. The crucial stage is *Conceptualisation*, in which knowledge acquisition and evaluation are performed. Integration of existing ontologies are done throughout the development process, not only during implementation (Gómez-Pérez et al. 1996).

WebODE is an ontological engineering workbench that covers and gives support to most of the activities involved in METHONTOLOGY (Ontological Engineering Group 2003; Arpiréz et al. 2003).

Protégé

Protégé is an open-source Java tool with many third party plug-ins that supports a variety of tasks for the development of knowledge-based applications (Protege-2000 2004; Gennari et al. 2002) Based on experience with this tool and others, a guide with some general ontology development guidelines has been developed (Noy and McGuinness 2001). The tool uses a frame representation of concepts and this design of ontologies influences the development method, which to a great extent overlaps the previously presented methods, but has a number of more detailed steps for the internal structure of concepts:

1. Determine domain and scope
2. Reuse of existing ontologies
3. Enumerate important terms
4. Define classes and taxonomy
5. Define properties of classes:
    - intrinsic properties, properties an object has in itself.
    - extrinsic properties, properties in relation to others.
    - parts, physical or abstract.
    - relationships, with other concepts.
6. Define facets of properties:
    - Cardinality
    - Value type, for example, string, number, boolean, enumerations of possible values and other types of objects present in the ontology.
    - Domain and Range, the type of object a property is connected to and the allowed classes of values, respectively.
7. Create instances

The first three steps correspond well to other methods, but the taxonomy organisation is not explicit elsewhere. The definition of the taxonomy can be done top-down, middle-out or bottom-up. Which approach to use depends on the developers' knowledge of the domain, but middle-out tends to be the easiest. Properties are defined as internal structures of concepts and can be of various types and have different restrictions, such as cardinality and value type. The definition of instances is also included in the method, though instances in most cases are not considered part of ontologies.

Maturity of methodologies and tools

Fernández (1999) evaluates a number of ontologies against the IEEE standard for *Developing Software Life Cycle Process*. The evaluation criteria focus on general points and the maturity of the methodologies. Most methodologies are found to be highly immature with regard to the standard. However, METHONTOLOGY only lacks a few aspects. The method needs to be extended with more detailed techniques and recommendations for pre-development processes.

Although the evaluation shows that no method has yet reached maturity, it shows promise in the fact that there are a number of existing methodologies that can be used as a starting point for the development of one or several standardised methodologies, which can be useful for different ontology types and settings.

Denny (2002) presents an extensive survey of ontology editing tools. About fifty tools of various levels of sophistication were available as of 2002. Some problems brought out in the survey regarded interoperability and usability. Many tools are based on the use of specific syntactic and semantic forms, which obstruct compatibility and the exchange of ontologies. The only support for interoperability is the support by some tools to export and import serialisations of various representation languages. However, such translations are often only partial and expressitivity is lost.

Usability issues concern interfacing other information components, graphical support for organising and managing an emerging ontology, and inferencing support.

There are only a few tools that support these interoperability and usability issues. These tools typically have evolved over a long time and are linked to different development methods. Besides the tool already mentioned above, KAON and Ontolingua fulfil these requirements. KAON has its roots in the semantic web community and includes a comprehensive tool suite for ontology creation and management and provides a framework for building ontology-based applications (FZI and AIFB 2004; Oberle et al. 2004). Ontolingua is described in Section2.3.2.

2.3.2 Representation languages

As stated in Section2.1.3, ontologies can have different degrees of formalisation ranging from informal natural language description to fully axiomatised logic based representations. The choice of representation language depends on the purpose and intended use of the ontology, but typically ontologies are used to build applications that require some sort of formalisation of the ontology. In this section, I will list some requirements on representation languages and briefly present and discuss the strengths and weaknesses of some of the most commonly used representation languages.The choice of representation language is important since it reflects a commitment to a certain viewpoint, for example logic views the world in terms of individuals and relations while frames capture prototypical objects in a hierarchy (Davis et al. 1993).

On the implementation level, a representation language should provide efficient storage, inferencing capabilities and support consistent encoding of ontology entities. On the logical level, it is desirable that a language is precise and expressive, has compositional semantics, and supports sound inference procedures. On the epistemological level, a language should see to it that the representations can be

constructed and organised in the most natural way in the domain, that representations are modular so that change and evolution are facilitated, that flexibility of granularity is supported. Finally, on the conceptual level a representation should provide a concise and precise manner of representing the real world entities, relations, constraints and axioms (Kishore et al. 2003).

Logic-based languages

Logic has a long tradition in AI as a knowledge representation language. One of the strengths of logic as a language for ontology formalisation is that it is straightforward to state facts about categories, and states and processes can be described with situation calculus. There are also other types of logics with different extensions, which make them suitable for ontologies. Modal logics make it possible to represent beliefs, tense and temporal expressions. Higher order logics view relations as objects, and thus support reasoning about properties as well as entity types (Noy and Hafner 1997).

Description logic, evolved from semantic networks, and uses the same type of taxonomic structure. It is based on first-order logic with some second-order features. The core is the notion of class that defines what properties an object belonging to a class must satisfy. Modelling concepts and attributes allow for boolean operators, quantification over roles, type restrictions and cardinality constraints. Inferences supported are subsumption, classification and consistency, i.e. to check if a category is a subset of another category, to check if an object belongs to a category and to check if membership criteria are verifiable. Description logic is a monotonic logic but many versions support some form of non-monotonic reasoning (Smith forthcoming).

FLogic integrates frame-based and first-order predicate calculus. Features of frame-based languages, like inheritance, polymorphic types and complex objects are accounted for in a declarative fashion. It has a sound and complete resolution-based proof theory (Kifer et al. 1990).

KIF is intended to be a general representation format for knowledge interchange. It is based on first order predicate calculus with prefix notation and extensions to handle meta-knowledge and non-monotonic reasoning rules (Genesereth and Fikes 1992). On top of KIF, the Frame Ontology, FO, that serves as a meta-ontology, has been developed. It contains the axiomatisation of class, instance, attribute, attribute constraints, subclass-of and instance-of relations, etc. Ontolingua (Gruber 1993b) combines KIF and FO, where the FO is used as a basis, but KIF statements can be included to represent axioms. Ontolingua provides a suite of ontology authoring tools and a library of modular, reusable ontologies. The tools focus on the development of ontologies through assembling and extending ontologies from the library (Fikes et al. 1997). The environment is available as a World Wide Web service (Ontolingua 2004).

Semantic web languages

The semantic web rests on a layered architecture, where various mark-up languages are used to add structure and precise meaning to web documents. XML is the basis on which the other mark-up languages are built. It provides a standard machine-readable syntax in which sets of tags to be used for specific purposes can be defined. With XML, data can be structured in terms of labelled trees. Document Type Definitions, DTDs, and XML Schema, XMLS, can be used to define the structure of XML documents (W3C 2004a).

Resource Definition Framework, RDF, is the next layer, which provides means for representing meta-data. It is a language in which factual statements about web resources and their properties can be represented. RDF schemas, RDFS, then focus on how meta-data in the form of RDF should be interpreted. With RDFS, vocabularies for expressing meta-data can be defined; relations between classes and properties are represented in type hierarchies, and constraints are used, for example, to restrict the domain and range of properties. The vocabularies can thus be seen as simple ontologies (W3C 2004c).

Finally a logical layer can be built on top of RDFS. OWL is a knowledge representation language, which can be used for ontological modelling and reasoning. It provides three sub-languages with increasing expressitivity, OWL Lite, OWL DL and OWL Full (W3C 2004b). The first is useful for taxonomies and thesauri and contains simple constraints. OWL DL is compatible with Description Logic and provides all constraints (although some can only be used under some restrictions) and is computationally complete and decidable. While OWL Full has no restrictions, it gives no computational guarantees.

OWL can be used to define classes (entity types), properties that can be either *ObjectProperty* (relation) or *DatatypeProperty* (attribute) and individuals. The taxonomic relations *subClassOf* and *subPropertyOf* are used to construct class and property taxonomies, and domain and range restrictions are used to relate classes to properties. Other types of restrictions, such as cardinality constraints and value type restrictions, can also be used. The language also supports statements about equality, boolean class expressions, annotation and versioning information.

2.4 Implications for design of ontologies for dialogue systems

In this chapter, I have discussed what constitutes an ontology and have presented the state of the art in ontology design and development of domain ontologies.

The question, "What is an ontology" has no simple answer, but for a more applied purpose an ontology can be seen as a computational resource with a definition of concepts which includes entity types and properties. Entity types include objects and events/processes, and properties attributes and relations. The definitions restrict the vocabulary used for the knowledge representation of a domain.

The type of ontologies in focus are content theories rather than mechanism theories, restricted to specific domains or applications, which are (semi-)formal.

Design focuses on two important issues, definitions of concepts and the organisation of taxonomies. For these a number of design choices and guidelines have been compiled and discussed. Other aspects like axioms, inference mechanisms and instances are also mentioned with regard to design choices, and design guidelines include maintenance and expressitivity.

A survey of development methods for domain ontologies shows that inductive or deductive approaches are used, and that methods are mostly task-oriented and stage-based. Both formal and informal approaches are used. Many of the most evolved methods are also accompanied by tools.

The design of ontologies for information-providing dialogue systems will be further discussed in Chapter 4. In that chapter, a requirements specification will be presented based on the analyses in this chapter and the next chapter, which presents ontology usage in dialogue systems.

Chapter 3

Ontologies in NLP, Specifically in Dialogue Systems

This chapter analyses past and present use of domain knowledge sources and ontologies in natural language processing, specifically in dialogue systems.

The previous chapter presented the design and development of ontologies. In this chapter, we turn to the third area of ontology research (see Figure 1.2), namely usage. Dialogue systems that presently use ontologies are analysed and the functionality the ontologies support is mapped out. This analysis will, together with the previous chapter, form the basis for the analysis of design requirements on ontologies in information-providing dialogue systems, presented in Chapter 4.

3.1 Ontologies in NLP

Ontologies have only very recently been introduced in dialogue systems. They, have however been around a little longer in other areas related to dialogue systems, such as question answering systems, machine translation and information extraction. To illustrate the potential of ontology usage in NLP, these areas are briefly outlined in this section.

3.1.1 Question Answering

Question Answering (Q&A) systems aim at providing answers to natural language questions in open domains. Typically, a factual question is posed and a text snippet collected from a corpora or the web is returned as an answer. To create successful system capabilities for the interpretation of questions, retrieval of texts, extraction of answers, and presentation of answers are needed.

For the first phase, question analysis, many Q&A systems utilise ontologies or taxonomies to identify the expected answer type. For example, in the Q/A-system FALCON (Harabagiu et al. 2003) the question, "What is the wingspan of a condor?" results in the answer-type dimension since wingspan is a hypernym of dimension. FALCON utilises the WordNet (Miller et al. 1993) ontology and maps its categories to categories in a hierarchy of answer types.

WordNet is one of the most commonly used ontologies in NLP. It was not initially developed to be an ontology, rather as lexical resource, but has been used as an ontology by many research groups. It is organised as a semantic net with hyponym, meronym and other relations between synsets. Synsets are sets of synonyms that express a concept (Miller 1995). Figure 2.3 shows some examples of synsets, e.g. {animal, fauna}, and the hypernym relations between them.

During the retrieval phase, ontologies can be used to alter and expand the search terms from the questions, for example, synonyms, hyponyms and hypernyms can be retrieved from an ontology. In some cases, it may even be possible to use part-of relations and replace a part or member with the whole or a set. To be certain that a retrieved text contains the answer, a semantic unification based on ontological knowledge can be performed (Zajac 2001). In FALCON abductive backchaining is used to prove that an answer follows from the question and for this semantic relations from the ontology are used (Harabagiu et al. 2003).

3.1.2 Machine Translation

In knowledge based machine translation (MT), texts are translated by mapping meanings (Mahesh 1996). To represent this meaning, an ontology can be a useful tool. In the Microkosmos project, the ontology serves as a central resource for the modules in the translation system. It has three main purposes, it provides semantics for the words in the different lexicons, it supports semantic analysis of ambiguous words and non-literal meanings through the relations of concepts, and it serves as a basis for Text Meaning Representations (TMR).

Frame-based representations of concepts with selectional restrictions on the values of properties are used for disambiguation. For example, the Spanish word 'adquirir' can mean both acquire and learn, but in the context of, "El grupo Roche adquirio Docteur Andreu" the constraints on learn to have an argument of type ABSTRACT-OBJECT and Docteur Andreu refers to the name of a corporation, which is of type SOCIAL-OBJECT, the meaning aquire is preferred.

Text meaning representations are usually instantiated subgraphs of the ontology that describes a more complex entity or event. Knowledge from the ontology can thus be used to fill in gaps, for example, by the use of default values of attributes.

3.1.3 Information Extraction

Information extraction usually means collecting relevant information from unstructured text and filling in a form or template. Ontologies can be used for various tasks in IE systems: designing templates, constructing extraction patterns, coreference resolution, and merging partial results (Appelt and Israel 1999).

Most IE systems use extraction templates based on events that specify entities and their relations in terms of their participation in the event. An example of this can be found in the LaSIE system (Gazauskas et al. 1997) where the domain of business meetings is captured in an ontology that contains objects, such as persons, places and organisations, and events involving persons attending meetings. Attributes express the properties of objects and events, as well as the relation between them. In LaSIE the text is interpreted into quasi-logical forms (QLF) that are conjunctions of first order logical terms. With the use of the ontology the QLFs can be interpreted and merged into a discourse model that describes the semantics of a text. The ontological knowledge is used to provide presuppositions and support for coreference resolution and inferences, which can be used for the construction of a template from the extracted information pieces represented by the discourse model.

An ontology can contain procedural attachments for the objects, properties and relations that can be used for information extraction. For example, in the information extraction system presented by Wee et al. (1999) demon attributes of three different types, TO-FILL, NORMALIZE and WHEN-FILLED, are used during different stages of template filling. Ontological knowledge of domain and range restrictions on properties and relations, and their cardinality constraints can also be used to help the information extraction populate a database (Embley et al. 1998).

Knowledge about hyponym relations can also be used for creating extraction patterns. For example, in Bagga et al. (1996) it is shown how specific extraction rules generated from training data can be

generalised by replacing concepts with their super-ordinates using the hyponym/hypernym relations in WordNet. For example, IBM Corporation can be replaced by {business, concern} which in turn can be generalised to {enterprise} and so on.

3.2 Dialogue systems

In early dialogue systems the distinction between domain, task and dialogue knowledge have not been distinguished. To be able to respond properly to a user request in an information providing system, the system has to have knowledge about both the domain and the task. Thus, domain and task knowledge have often been integrated in the dialogue model.

A typical example of the integration of domain and task knowledge is the use of semantic frames which are part of many information-providing dialogue systems (cf. (Bennacef et al. 1996; Carlson and Hunnicut 1996; Seneff et al. 1996)). They are used to guide the dialogue, represent the discourse history and to represent the relevant domain concepts. They are sometimes coupled to rules for the interpretation of domain concepts and rules to fill in default values in a frame.

In the SUNDIAL project (Speech UNderstanding in DIALogue) several dialogue systems for information exchange over the telephone have been created for different languages (Heisterkamp et al. 1992). The systems provide information about air and rail traffic.

In SUNDIAL, a *belief model* represented as a semantic net is utilised for interpreting user input and for dialogue management. The model consists of a semantic class hierarchy with subsumption relations and roles (binary relations) between the classes. It also has the potential to model constraints on the values of roles. Thus, the belief model has many features in common with ontologies.

A representation of a user utterance is constructed compositionally of the concepts that can be matched to words or phrases in the input. The resulting semantic representation is sent to the Dialogue Manager for further contextual interpretation. New information is anchored in a discourse model if it is of the right type expressed by the value constraints. For example, "I want to fly to Paris" results in a representation with the concepts, Journey, Location, Paris, Arrive related through Journey → the_arrival → Arrive → the_place → Location:Paris. New information in, "From London" is interpreted as Depart → the_place → Location:London which can be linked to the previous through Journey → the_departure → Depart.

The VERBMOBIL system is not a traditional dialogue system as the system is not an active participant in the dialogue. Instead, it listens to two persons having a conversation in a language that is not their mother tongue, e.g. a German and a Japanese trying to book a meeting by speaking English. The system's task is to monitor the dialogue and to give the users translations when required (Alexandersson et al. 1994).

The domain model in VERBMOBIL consists of four parts: a hierarchy of speech-event types, a conceptual hierarchy, thematic relations and a model of calendar information (Quantz et al. 1994). The model is implemented in description logic and thus supports the definition of concepts and roles. The domain model supports several functions: subsumption/disjointness checking, retrieval of sub- and super- concepts, retrieval of role information, and default-based reasoning. The model's actual use in VERBMOBIL is for interpretation, disambiguation, determination of speech-event types and for building context representation.

During interpretation, the speech recogniser produces sentences that are syntactically correct but semantically incomplete. These sentences can be detected with the use of sortal information from the conceptual hierarchy in the grammar rules. For example, through subsumption tests, arguments for verbs, adjectives and other lexical entries with argument structure can be judged as appropriate or not.

The subsumption relations between the concepts in the conceptual model makes it suitable for decisions about possible references. It also allows for inheritance; if a temporal object is available for scheduling, the super-ordinate object is regarded as free too, and similarly, if an object is not available, none of the subordinated objects are either. When new information is given it can be seen as a new proposal or a refinement of a previous one, depending on the subsumption relations between the concepts in the new information and the context. For example, if the question, "Can we meet on Thursday" is answered with, "after lunch" it is seen as an agreement and a clarification since the new information is part of the specification of a meeting time, however if it is followed by, "on Friday" it is a new proposal since this new information is not unifiable with the old.

In the following sections, more detailed descriptions will be given of some recent dialogue systems that use explicit domain ontologies as domain knowledge sources for many tasks, such as interpretation, clarifications and contextual interpretation.

3.2.1 UK CANCER REFERRALS and HOME CONTROL

Milward and Beveridge (2003) describe two applications of ontology-based dialogue systems, one for decision support for cancer referrals and one for home control. These systems are based on a three-level architecture that utilises two separate knowledge sources for task knowledge and domain knowledge.

A Domain Manager in the deliberation layer constructs an abstract task specification (ATS) (Beveridge and Millward 2003) based on informational and intentional information provided by the Ontology Engine and Plan Execution Engine. Next, in the sequencing layer, the ATS is processed by a Game Engine that decides on the proper conversational game to be played to accomplish the task described by the ATS. The game is realised through a Move Engine that produces a sequence of communicative acts that are appropriate to the current dialogue. Finally, the acts are realised by a Client browser in the last

S1: Hello, this is the Cancer Research UK cancer referrals advice system...
U2: Hi
S3: What is the patient's age?
U4: She is thirty five, female, with nipple retraction, a breast abscess, and an ulcer.
S5: What kind of ulcer does the patient have?
U6: A cutaneous ulcer
S7: Does the patient have an acquired deformity of nipple?
U8: No, but she has some nipple discharge
S9: Does the patient have bilateral nipple discharge?
U10: No, she has bloodstained nipple discharge
S11: Does the patient have cloth-staining discharge?
U12: No
S13: Does the patient have a breast nodule?
U14: She has breast induration and she has a cyst
S15: What kind of cyst does the patient have?
U16: What do you mean?
S17: I mean: do they have a breast cyst, retention cyst or sebaceous cyst?
U18: She has a breast cyst
S19: Does the patient have any intractable pain?
U20: No
S21: Does the patient have any asymmetrical breast nodularity?
U22: Yes, she has a breast lump
S23: Does the patient have nipple eczena?
U24: No
S25: This patient is a woman with possible breast cancer who should be referred to a specialist.
U26: Right, thanks...
S27: Ok. Thank you for calling. Goodbye.
U28: Bye.

Figure 3.1: A dialogue between a user and the UK CANCER REFERRALS dialogue system that provides decision support for cancer referrals.

layer which provides reactive skills for communication management. It is based on a finite state network represented in VoiceXML.

The ontology used in the US Cancer referral system is an existing medical domain ontology that contains medical terms with IsA relations. This information is used in several ways to facilitate dialogue interaction. An example dialogue with the system shown in Figure 3.1 exemplifies many of these. Four specific usages of the ontologies will be further discussed.

Over-specified answers

In utterance S13 in Figure 3.1 the system asks if the patient has a breast nodule and receives the answer (U14) that she has a breast induration and a cyst. Since breast induration IsA breast nodule, according to the ontology, the system can infer that the patient does have a breast nodule and the answer can be correctly interpreted.

Under-specified answers

Sometimes users give answers with too general concepts, like cyst in U14. In this case, a clarification is initiated where the user is required to provide more specific information. The hypernym relation of the ontology is used to provide alternatives, as seen in S15.

Dialogue coherence

The structure of the dialogue contains a partial ordering of topics derived from the task structure provided by the intentional information. For example, satisfaction-precedence and dominance relations describe sequential and hierarchical relations of intentions. This can however be complemented with information from the ontology regarding the relation between topics, thus achieving a total ordering.

words	concepts	instances
w1: turn_off(i1)	a1: turn_off	
w2: back	c2: back	
w3: bedroom	c3: bedroom	b1 b2
w4: light	c4: light	l1 l2 l3 l4

Figure 3.2: The stepwise interpretation of the utterance, "turn off the back ... bedroom ... light", moving from words to concepts to instances from the left to right.

If a topic is subsumed by (a hyponym of) another topic it is an elaboration of this topic and should receive focus after the other topic. If a user shifts focus to a new topic, the system should consider asking follow-up questions to this before returning to the previous dialogue focus again. An example of this is given in S5 in the dialogue in Figure 3.1, where the provided information on an ulcer is followed up.

Semantic interpretation

A commonly used robust approach to the interpretation of user utterances in dialogues is partial parsing where task relevant terms are identified and the rest is ignored. Milward (2002) proposes a semantic-based composition strategy for interpretation where domain knowledge in the form of an ontology is used. For the home domain the ontology consists of an IsA hierarchy that describes different locations, devices and positions. A second hierarchy captures which devices are In what locations and how the locations are related. This allows for the interpretation of utterances, such as, "Turn off the back ... bedroom ... light".

The initial analysis identifies words and phrases, the first column in Figure 3.2. Next these words are linked to ontological concepts (column two). After that, instances of the concepts are introduced (column three). The relations captured in the ontology hierarchies

are used to relate the instances, for example there is an IN relation that links l1 to b1 and l2 to b2. b1 is also related to c2 and therefore the interpretation turn_off(l1) is chosen.

An evaluation indicated high performance but showed problems with conjunctions. This was expected since syntactic information is needed to group the words in the conjuncts.

3.2.2 SmartKom

SmartKom is a multi-modal multi-domain dialogue system platform. It consists of several modules and knowledge sources that interact. The processing modules can be grouped as belonging to input processing, media analysis, interaction management, application management, media design and output rendering. It utilises explicit user models, discourse models, domain models, task models and media models (Wahlster et al. 2001).

In SmartKom ontologies are chosen as the unifying knowledge representation used by the various components (Gurevych et al. 2003). The ontology is language independent and provides a general top-level and several domain-specific ontologies. Two important distinctions made are role vs type and object vs process. Types are basic ontological entities that are generic and roles are specific to a certain situation, event or process. The domain-specific ontologies therefore deal only with roles. Processes are represented as frames with a set of slots (properties) that relate different objects to them through the roles they play in the process, for example, agent, theme, experiencer, location, etc.

The ontology in SmartKom is used for many NLP tasks, such as speech recognition and dialogue management, and for information exchange. These are described below (Porzel et al. 2003).

Speech recognition

In automatic speech recognition systems (ASR) several hypotheses are usually produced and presented in an n-best list. To decide on the most semantically plausible of these, semantic coherence can be computed based on ontological knowledge (R.Porzel et al. 2003). This is done by converting the hypothesis to concept representations by looking up word concept relations in a lexicon, and then computing the average path-length of the connection between all concepts in the concept representations.

Natural language understanding

SmartKom utilises a template based semantic parser. A template has a condition part and an action part. The action is triggered when the conditions are met, in terms of the presence of certain words or instances, and typically produces new instances. The ontology is used for two purposes, to create templates that are ontologically consistent and as general as possible, and to resolve referring expressions. More general classes can be utilised in the specification of a template since the ontology can be used to map more specific user utterances to the template class. Similarly, more general instances, as often provided in referring expressions, can be mapped to the more specific in the template, thus deducing the type of the referring object based on the context (template) it appears in.

Contextual interpretation

One of the tasks of the Dialogue Manager is to enrich and validate user input based on the previous discourse (Pfleger et al. 2003). For this purpose, an OVERLAY operation is utilised. The operation computes a score for the current input and structures from the discourse context, based on conflicting values and types. The most similar and recent structure from the discourse is chosen based on this score and

U: What's on TV tonight?
S: [Displays a list of films] Here you see a list of films.
U: Show me the cinema program.

Figure 3.3: A dialogue excerpt from interaction with SMARTKOM.

information is inherited to the new structure based on a common superclass of the structures. For example, in the dialogue excerpt in Figure 3.3 the common superclass for TV program and Cinema program is AvEntertainment and through this, the property begin-time can be inherited from TV program to Cinema program via AvEntertainment, which all share this property.

Referring expressions

The Dialogue Manager also resolves referring expressions using the discourse model and domain knowledge from the ontology (Pfleger et al. 2003). A referent is searched in the local and global focus of the discourse model and the ontology is used for type checks.

Interface specification

The modules in SMARTKOM exchange information in a uniform format based on the ontology. Since the ontology is implemented in OIL it can be used to generate XMLS and DTD (see Section2.3.2) that can be used for interface specifications.

3.2.3 TRIPS

TRIPS is a framework for multi-domain problem-solving dialogue systems (Allen et al. 2001).

```
(define-type LF_FOOD
 :parent LF_Physical-object
 :sem (phys-obj (origin natural)
                (object-function comestible)
                (mobility movable)))

(define-type LF_FILLING
 :parent LF_MOTION
 :sem (situation (cause agentive))
 :arguments (THEME (phys-obj (mobility moveable)))
            (AGENT (phys-obj (intentional +)))
            (GOAL (phys-obj (container +))))
```

Figure 3.4: Examples of definitions from the language ontology of the concepts for food objects and filling events, in the TRIPS dialogue system.

The architecture is based on several agents where each agent performs linguistic or planning processing that contributes to the overall dialog interaction. There are four main agents responsible for interpretation, generation, and behavioral (application-specific) aspects (Blaylock et al. 2002). Here the focus is on interpretation, which has been based on the use of ontologies.

The interpretation of user utterances can be customised for new domains through the addition of application-specific domain knowledge in the form of domain ontologies. A generic language ontology is used for the lexicon and by the parser to produce meaning representations. These can then be converted to domain representations by mapping the generic concepts in the language ontology to domain specific concepts in the domain ontology (Dzikovska 2004).

The ontologies are based on different design criteria. The language ontology should be as general as possible, have a flat structure and linguistically motivated sense distinctions and have representations with argument structure close to the linguistic form. The domain ontology should cover domain-specific concepts and be organised ac-

```
(SPEECHACT sa1 SA_REQUEST :content e123)
(F e123 LF_FILLING*load :agent pro1 :theme v1 :goal v2)
(IMPRO pro1 LF_PERSON :context-rel +YOU+)
(THE v1 LF_FOOD*orange)
(THE v2 LF_VEHICLE*truck)

(define-lf-to-kr-transform load-transform-trips
 :pattern ((?spec ?ev LF_FILLING :agent ?a :theme ?t
                                 :goal ?g)
           -> ((TYPE ?ev LOAD) (ACTOR ?ev ?a)
               (CARGO ?ev ?t) (CONTAINER ?ev ?g))))

(AND (TYPE e1 LOAD) (ACTOR e1 YOU123)
     (CARGO e1 oranges2) (VEHICLE e1 TRUCK))
```

Figure 3.5: The stepwise interpretation of the utterance, "Load the oranges into the truck", with examples of definitions from the language ontology and mapping rules for transformation to domain knowledge representation, in the TRIPS dialogue system.

cordingly, have a deeper structure with more fine-grained distinction and have a representation suitable for reasoning.

Entities in the ontologies have a type, and linked to this type is a set of features, such as form, origin, and function for physical objects. The entities also include argument structures. In the language ontology these are linguistically oriented thematic structures with, for example, agent, goal and theme roles, see Figure 3.4 for some examples.

The transformation to domain knowledge representation includes mapping of syntax and concepts. The structures in Figure 3.5 show the semantic representation produced by the parser, the mapping rule used, and the resulting domain knowledge representation for the utterance, "Load the oranges into the truck".

The transformation involves replacing the definite descriptions, for example variables ?v1 ?v2, with a constant, in this case ORANGES2 and TRUCK3. The mapping of arguments in the parser output corresponds directly to the slots of the knowledge representation format and the transformation is therefore straightforward. More complex cases where several frames are produced from one output representations can also be made through hierarchical transformation rules.

3.3 Implications for design of ontologies for dialogue systems

The analysis presented in this chapter shows many potential uses of ontologies in dialogue systems. Question answering and machine translation illustrated how ontologies can be used to interpret user utterances, to decide on the expected answer type and disambiguate the content. From an information extraction point of view, the use of ontologies for coreference resolution and discourse representation can be applied to dialogue systems.

The ontologies currently used in dialogue systems are mixed, i.e. they are linguistically oriented world knowledge representations that provide semantic information, but in TRIPS a two level approach is used where linguistically oriented ontological knowledge is transformed to domain specific knowledge representations. The ontologies are used for the interpretation of questions and requests, where ontological knowledge is primarily used for disambiguation. We also see how ontological knowledge can be used for dialogue management, reference resolution, clarifications and contextual interpretations.

Chapter 4

Design of Ontologies for Dialogue Systems

This chapter investigates the functions ontologies used in dialogue systems should support and what implications these have on the design of such ontologies.

In this chapter the analysis of design and usage of ontologies, presented in Chapters 2 and 3 are synthesised in a requirement specification on ontologies for information-providing dialogue systems. The result is a recommendation of design decisions to follow in order to create an ontology such that it can support interpretation, dialogue management and domain knowledge management in information-providing dialogue systems.

The importance of the design choices (DC1-DC14) and guidelines (DG1-DG16) presented in Chapter 2 for the design of ontologies to be used in information-providing dialogue systems is discussed in light of three tasks: interpretation, dialogue management and domain knowledge management. Question analysis means the interpretation of user utterances in a dialogue, the result being a representation of the se-

mantic content of the utterance. Dialogue management involves tasks like handling anaphora and ellipsis, clarifications, and focus management. Domain knowledge management are tasks related to domain reasoning and retrieval of requested information.

The discussion is based on the use of ontologies in existing dialogue systems, see Chapter 3, and analysis of question and dialogue corpora from various domains; bus time-table information, nordic birds and teletext information provided by the Swedish Public Service Television Company (SVT Text). The last is an example of a multi-domain corpus including economy, television programs, sports and news. The former corpora has been collected in order to develop the dialogue systems presented in chapters 5 and 6, and will be described in more detail in those chapters. The latter corpus consists of 219 questions collected from potential users of a dialogue system for that domain.

The selection of systems is small since there are not many dialogue systems available that utilise ontologies. However, the systems seem fairly representative as they cover several domains and tasks. The corpora are also of prototypical types for information-providing dialogue system, the bird domain deals with fairly simple objects and properties, while travel information includes more complex objects that need to be specified through dialogue. The teletext is an example of a multi-domain where the news part in itself is almost an open domain that includes events and complex relations. Thus, taken together, the systems and corpora should capture the core features of information-providing dialogue systems.

The different design choices that were derived in Chapter 4 are elaborated in terms of what are *necessary* requirements, i.e. core functionality that most information-providing dialogue systems require, what are *optional* requirements, i.e. features that are optional but can be useful in several dialogue systems, and what are *domain* dependent requirements, i.e. functionality that is required in some domains but can be left out in others.

4.1 Entity types and properties

For entity types and properties, the design choices to consider, as discussed in Chapter 2 are:

DC1 Should the concepts have internal structure or be atomic?

DC2 What type of properties should be allowed? Local, polymorphic instance, class, own, inherited?

DC3 What facets of attributes should be allowed? Default values, type constraints on values, cardinality, operational definitions?

DC4 What type of relations should be allowed? Binary or n-ary, domain restrictions, range restriction?

DC5 How should time and space be treated?

DC6 How should part-whole relations be treated?

4.1.1 Internal structure

The question of the internal structure of concepts has no simple answer. In the studied systems, both atomic and structured concepts were used. SUNDIAL and UK CANCER REFERRALS use atomic approaches in which words are mapped to concepts and relations that are later related, for example, through the argument structure of verbs/relations. SMARTKOM uses a frame-like representation with slots for properties, for example, TV Program has begin_time. The type that is preferred depends on the type of syntactic parsing that precedes the semantic interpretation. Partial or phrase spotting approaches are more suitable for atomic concepts, while full parsing can use internal structure of concepts. A problem with representations that use internal structure to represent properties is that these are not given the same status and cannot be represented separately.

Since partial information about properties is given in elliptic utterances in dialogues, they should be possible to represent separated from objects.

For dialogue and domain knowledge management, the choice between atomic concepts or concepts with internal structure depends on the type of dialogue model used and the domain. With internal structure, properties can be inherited between objects/events, like in SmartKom with the overlay operator. With atomic concepts, properties can also easily inherit objects. In the bus time-table domain, problems with frame-like representations were encountered when entities were seen as both values in frames and as objects. For example, a bus stop can denote a value for the departure slot in a trip specification but it can also be an object, as in the question, "Which buses pass the bus stop Travel Centre". In contrast, concepts with internal structure can support the Dialogue Manager's specification of a request, since it can serve as a task model where an empty slot indicates the need for a clarification.

Thus, the choice between atomic concepts and concepts with internal structure must be considered in light of the domain and the type of syntactic parsing and dialogue model used. Atomic concepts are simpler and provide the necessary functionality in most cases. They are sufficient for question analysis, and for focus management, but often need some complement, like a system task model (see Section5.3.2), to be used for clarification requests by the Dialogue Manager.

4.1.2 Type of properties

Several types of properties are present in the systems or indicated as useful by the corpora. All properties are local, either they are slots of frames (concepts) or they are linked to concepts through domain restrictions. Since dialogue systems deal with natural language that is inherently polymorphic it is common that properties are polymorphic. For example, in SVT Text, the attribute result that corresponds to the word, 'do' (swe. gå) in, "How did the stock market *do*?" and,

"How did Björklöven *do*?" (swe. "Hur *gick* börsen?" and, "Hur *gick* det för Björklöven?"). Examples are also found in the BirdQuest corpus where the attribute appearance can be applied to both birds, eggs and nests.

Instance attributes that can take different values for instances are found in most of the studied systems and corpora. Typically, many questions concern values of an attribute for instances, like, "Is the bedroom light on?" and, "How old does a common seagull get?". Class attributes that have the same value for all instances of a concept are not present. There are also no examples of own properties, all properties are inherited by sub-classes.

Consequently, the type of properties needed are local, polymorphic, instance and inherited. Class and own properties are optional and can be useful in some domains, primarily to define and describe concepts.

4.1.3 Facets of attributes

Attributes can have various types of restrictions. Several of them are indicated as necessary or optional by the analysis of systems and corpora. Type constraints that restrict the type of concept an attribute can be applied to are used both for interpretation and dialogue management. For example, in TRIPS they are used for interpretation. "The boy smiled" is considered to be a valid utterance, but "The idea smiled" is not, since smiled has a restriction that only applies to humans. In SMARTKOM the OVERLAY operator used for contextual interpretations relies on type restriction (Alexandersson and Becker 2003)

Cardinality constraints are not utilised in any of the systems, but can be useful for several tasks in dialogue systems. For example, if partial parsing is used, the combination of identified units can be based on cardinality constraints. For focus management, the number of possible values can guide inheritance and combination of old values with new values. For example, in the BirdQuest domain where a bird

is described, some properties can take several values and for those new values should be combined with old values, while for properties that can have only one value, a new value should replace the old value.

Default values are primarily useful for dialogue and domain knowledge management, for example, in the bus time-table domain date and time can be left out and assumed to be today and now in requests for bus times, "When does the bus for the city centre leave from the University?". Defaults are also used in TRIPS to facilitate lexicon construction. Operational definitions are not present but could be used for example to compute the value of NOW and TODAY.

Thus, type constraints are often a necessity, and default values and cardinality are optional. Operational definitions can be used in some domains.

4.1.4 Type of relations

Binary relations are more common than n-ary, in general, and the same is true of the systems. For example, the ontology in SMARTKOM is represented using semantic web languages that use binary relations only. However, in some cases the former might be more natural and efficient to use, for example to express temporal aspects, such as a TV-program airing at a channel on a certain time.

When atomic concepts are used, the domain and range restriction are crucial in order to link relations to the type of concepts they apply to. The domain and range restriction are used for various tasks in dialogue systems. In question analysis they are utilised for disambiguation, and in dialogue management and domain knowledge management for contextual interpretation and inheritance.

Consequently, binary relations are sufficient, and domain and range restrictions are a necessity.

4.1.5 Treatment of time and space

The requirements on treatment of time and space are, of course, dependent on the temporal and spatial complexity of the domain(s) a dialogue system is used for. In the UK CANCER REFERRALS and HOME CONTROL systems, time is not treated. A rather simple solution for spatial knowledge used in the home control domain is to use the relation In to represent spatial relations between objects, for example that devices are in locations. In more complex domains, such as those in SMARTKOM, temporal and spatial aspects are represented as attributes of events in the ontology, for example, begin_time of a TV-program. A further step is to follow Santos and Staab's (2003) approach and represent the temporal and spatial aspects separately. In TRIPS times is a special class of objects that can be used in temporal relations (Dzikovska 2004). In the SVT Text and Ötraf corpora spatial and temporal objects are common, for example, weather reports and bus trips have temporal and spatial aspects like dates and towns that can be represented in separate ontologies. These temporal and spatial objects can then be linked to other objects and events through relations.

Using a separate representation of time and space facilitate the reuse of these aspects across domains. However, it may require more from the domain knowledge management that has to link temporal and spatial objects to other objects and events, and transform values to suitable formats. For example, vague temporal or spatial descriptions like *tomorrow* and *near* must be transformed to precise dates and distances.

4.1.6 Part-whole treatment

The BirdQuest corpus indicates that PartOf relations are needed in that domain. Birds are described in terms of colours and sizes of body parts and therefore PartOf relations are needed. These could be used for domain knowledge management and clarifications, for example,

when a user asks, "What colour is a crow"? the database needs to be accessed and the colour collected for all the body parts of the birds.

4.2 Taxonomy organisation

The organisation of concepts and taxonomies requires a reflection of the following issues:

DC7 How should categories be organised? Tree, multiple dimensions, distributed?

DC8 What type of relations between categories should be allowed?

DC9 What kind of inheritance should be dealt with? Monotonic, non-monotonic, simple, multiple?

DC10 What top-level divisions should there be?

4.2.1 Organisation of categories

Categories can be organised as trees or through combinations of various dimensions. They can either be collected in one taxonomy or in several distributed structures. In all the systems, IsA trees were used to represent the taxonomy rather than parallel dimensions. In the multi-domain systems, TRIPS and SMARTKOM, there are several distributed taxonomies. In SMARTKOM these are linked together through a common upper-level ontology.

The use of a simple tree structure allows straightforward inheritance of properties between concepts. This forms the basis for the OVERLAY operator used in SMARTKOM, and facilitates specification of the language ontology used for interpretation in TRIPS, since many features can be inherited.

Thus, tree structures seem to be the natural choice for dialogue systems. This is consistent with guideline DG11 that says that entity types and properties should be organised in trees. To use one tree or several distributed trees depends on the character of the domain.

4.2.2 Relations between categories

Most systems use only IsA links to organise the taxonomy. Since SmartKom is based on OIL it also allows for partitions, but it is not clear if and how these are used. Thus, IsA relations are sufficient for most dialogue systems. However, guideline DG11 indicates that partitions in the form of (exhaustive) disjoint categories should be used.

4.2.3 Type of inheritance

Since most of the taxonomies are untangled trees, the kind of inheritance used in them is simple, and in most cases also monotonic. However, since default values are used in some systems, non-monotonic inheritance is also used, for example in TRIPS. Guideline DG12 suggests that multiple inheritance should be allowed.

Consequently, single monotonic inheritance is the baseline but multiple and non-monotonic are useful in some cases.

4.2.4 Top-level divisions

Most top-level divisions in the systems are very domain specific. SmartKom uses a more general top-level categorisation that separates Roles and Types, which are further divided in Abstract and Instance, but with the focus on processes that describe the type of services provided in the various applications.

From a pragmatic point of view, it is possible to build application ontologies for dialogue systems without general distinctions. But if existing ontologies are to be integrated with new ontologies, a common top-level plays a crucial part in integrating and mapping concepts.

Thus, when designing an ontology to be used in a dialogue system one should consider the importance of reusability and extendibility as proposed by guidelines DG14 and DG15. To be able to integrate new domains or tasks, the top-level distinctions should be general to a certain extent.

4.3 Axioms

If axioms are used in an ontology, the following questions should be considered:

DC11 What type of axioms should be allowed?

DC12 Should axioms be independent or linked to concepts?

DC13 How should axioms be expressed?

In the analysis of systems and corpora, most aspects are captured by the taxonomy or definition of concepts, for example, partitions, non-monotonicity and part-whole. Thus, axioms are not used to any great extent.

Guideline DG8 proposes that inverse relations should be defined whenever a new relation is introduced. If one would like to make this relationship between relations explicit, axioms for relational algebra could be used.

Guidelines DG4 and DG6 indicate that properties should be represented separately from entity types and this means that sub-relation

relationships need to be expressed, this can however also be done in a taxonomy, rather than with axioms.

Subsequently, axioms are not necessary but could be used in some domains to represent relationships between properties. In this case, they should be independent of concepts.

4.4 Instances

If instances are included in the ontology, the following question should be answered:

DC14 What types of instances should be allowed? Individuals, facts, claims?

In most of the systems, individuals as instances of entity types are prominent. Only UK CANCER REFERRALS lacks individuals. In HOME CONTROL, the individuals are part of the ontology as leaf nodes in the object hierarchies, for example, l3 is an instance of the concept lamp. In TRIPS, the individuals are introduced with the mapping of ontological objects to instances.

Facts that represent instances of relations can also be useful. For example, in BIRDQUESTinformation about relations between locations, such as the county Östergötland is PartOf the country Sweden, needs to be represented as facts.

Thus, individuals and facts should be allowed, but represented separately from the ontology in a knowledge base. Since information-providing dialogue systems deals with objective information, claims are not present at all.

4.5 Summary

The analysis is summarised in Figure 4.1 where the requirements on various design decisions are indicated. **N** denotes that the feature is necessary, **O** that the feature is optional but can be useful, and **D** that the feature is applicable in some domains. For some decisions there is a choice between two alternatives, for example, in DC1. Then one alternative can be marked as necessary and the other as optional or domain-dependent. Since only one of the alternatives are needed this means that if the latter are used the former marked necessary can be ignored.

The resulting analysis thus indicates the minimal requirements (those marked with an N) that are needed for an ontology to support interpretation, dialogue management and domain knowledge management in information-providing dialogue systems. Depending on the domain the design might need to be extended with a few more features (DC3, DC5, DC6, DC10, DC13).

The design is intended for information-providing dialogue systems, but could be used as a basis for ontologies in other types of dialogue systems. However, the design will probably have to be extended with other types of features, for example, to handle causality in dialogue systems that reason on event chains (cf. (Grosz 1977; Smith and Hipp 1994)).

Design decision	Req
Entity types and Properties	
DC1 Atomic concepts or Internal structure	N/O
DC2 Local attributes	N
DC2 Instance attributes	N
DC2 Class attributes	O
DC2 Polymorphic attributes	N
DC2 Inherited attributes	N
DC2 Own attributes	O
DC3 Default values for properties	O
DC3 Cardinality restrictions on properties	O
DC3 Procedural attachments for properties	D
DC3 Domain restrictions on properties	N
DC3 Range restrictions on properties	N
DC4 Binary relations or arbitrary n-ary relations	D
DC5 Treatment of time and space	D
DC6 Part-whole treatment	D
Taxonomy organisation	
DC7 Tree or parallel dimensions	N/D
DC7 Single or distributes taxonomies	D/D
DC8 SubclassOf	N
DC8 Partitions	O
DC9 Monotonic or non-monotonic inheritance	N/O
DC9 Simple or multiple inheritance	N/O
DC10 Generic top-level divisions	D
Axioms	
DC11 Type of axioms	D
DC12 Independent axioms	D
DC13 Expression of axioms	D
Instances	
DC14 Instances of categories	N
DC14 Facts (instances of relations)	N
DC14 Claims (assertion of fact by instance)	D

Figure 4.1: Requirements on design of ontology for dialogue systems. **N** means necessary, **O** means optional, **D** means necessary in some domains.

Chapter 5

Domain Knowledge Management in ÖTRAF and MALIN

In this chapter, a dialogue system application, ÖTRAF*, and a framework,* MALIN*, are presented and discussed with a focus on the representation and use of domain knowledge and reasoning.*

The following chapters turn from design to usage of ontologies in dialogue systems. They present various dialogue applications and frameworks, which serve as the basis for the creation of the framework for ontology use in information-providing dialogue systems presented in Chapter 7. A first step towards such a framework, is a new architecture for information-providing dialogue systems that incorporate a separation of dialogue management and domain knowledge management. This architecture was introduced in an application ÖTRAF and was later generalised to a dialogue systems framework, MALIN, both of which are presented in this chapter.

5.1 Dialogue system capabilities and knowledge sources

In (Flycht-Eriksson 2001) a number of capabilities were identified, see Figure 5.1, regarding tasks, mixed-initiative, focus and discourse, and domain reasoning. These capabilities are considered necessary to achieve graceful and cooperative interaction and were relates to the type of knowledge they rely on. The capabilities formed a basis for a corpus analysis aimed at discovering the set of phenomena an information-providing dialogue system must deal with, and the knowledge sources necessary to handle these.

5.1.1 The ÖTraf corpus

The domain for the corpus study was information about local public transportation. This domain is similar to the much studied domains of air and rail journeys but differs in the complexity of the geographical domain. Users' natural way of expressing departure and arrival locations rarely coincide with the official names of the bus-stops. These locations are instead described using street or area names, locative expressions like, "close to the library", or by pointing and clicking in a map. The temporal domain is, however, very similar with expressions like, "tomorrow morning" and, "around eight".

As a foundation for the study, a corpus of 43 dialogues between bus travellers and personnel at the local bus traffic company was collected. Of these, 5 dialogues were removed from the corpus since they were outside the scope of the system-to-be. The remaining 38 dialogues included requests for information about bus and rail journeys, bus routes, bus-stops, price and the lost and found.

A problem when using human-human dialogues for dialogue-system development is that the person who provides the services in the dialogues does not have the same performance characteristics as a com-

	Desirable capabilities
Handling tasks and request	
A1	To identify the task.
A2	To identify sub-tasks and know how they are related to a task.
A3	To reason about how much of a task has been achieved so far.
A4	To decide what action to take in order to achieve a task.
A5	To deal with situations in which no answer can be retrieved.
A6	To deal with situations in which the answer includes too much information.
A7	To detect and deal with hypothetical questions.
A8	To explicitly communicate a commitment made by the user.
Achieving mixed-initiative dialogue	
A9	To allow the user to over-answer questions.
A10	To allow the user to initiate clarification sub-dialogues.
A11	To allow the user to abandon the current request and pose a new request instead.
Handling focus and discourse	
A12	To follow shifts in focus.
A13	To resolve anaphora and ellipsis.
A14	To answer questions on what has been said and done during the conversation.
A15	To answer questions about the reason why an action was performed.
Domain reasoning	
A16	To map a description to an entity.
A17	To detect ambiguous descriptions and deal with them.
A18	To detect erroneous descriptions and deal with them.
A19	To know the type and structure of the entities in the domain.
A20	To reason about and derive new information from the information provided by the user.
A21	To deal with a user's erroneous inferences or false presuppositions.
A22	To have domain-related default information.
A23	To adapt to the user's domain expertise.
A24	To know what the system can and cannot do.

Figure 5.1: Capabilities for achieving graceful and cooperative interaction.

puter, and that communication between humans differs from communication between a user and a dialogue system (Jönsson and Dahlbäck 1988). A method called *distillation* (Larsson et al. 2000) was used to solve this problem. The purpose of distillation is to rewrite human-human dialogues to more resemble human-computer interaction. The distillation is based on a set of guidelines developed with respect to the linguistic, functional and ethical properties of the desired system. The result of the distillation was a corpus of dialogues, more resembling human-computer interaction[1].

Each exchange between the 'user' and the 'system' was analysed with a focus on the capabilities the system needs to interpret and respond to user utterances.

5.1.2 Capabilities

The corpus study indicated that most of the suggested capabilities were necessary, but not all. The required capabilities are presented in Figure 5.2.

The capabilities in the table should, however, not be seen as a complete list as some useful capabilities might not have been present in the dialogues in the corpus. It seems, for example, reasonable to believe that whenever capabilities A16 and A17 are required, capability A18 will also be needed. The same holds for capabilities A10 and A11, which are closely related to capability A9. The capabilities discovered in the corpus serve, however, as a minimal set of capabilities that should be supported by information-providing dialogue systems.

[1]The method of distillation has been developed and refined over the past years. The distillation of the dialogues in the corpus used here was based on a first draft of the guidelines, which were very general (Dahlbäck and Jönsson 1999). More precise guidelines have been developed since then. However, since later developments have mostly focused on the linguistic and ethical properties but not as much on the functional properties, it should not affect the quality of the distilled dialogues used in this study since it only considers the functional properties of the dialogues.

	Desirable capabilities
Handling tasks and request	
A1	To identify the task.
A2	To identify sub-tasks and know how they are related to a task.
A3	To reason about how much of a task has been achieved so far.
A4	To decide what action to take in order to achieve a task.
A5	To deal with situations in which no answer can be retrieved.
A6	To deal with situations in which the answer includes too much information.
Achieving mixed-initiative dialogue	
A9	To allow the user to over-answer questions.
Handling focus and discourse	
A12	To follow shifts in focus.
A13	To resolve anaphora and ellipsis.
Domain reasoning	
A16	To map a description to an entity.
A17	To detect ambiguous descriptions and deal with them.
A20	To reason about and derive new information from the information provided by the user.
A22	To have domain-related default information.
A24	To know what the system can and cannot do.

Figure 5.2: Capabilities required by an information-providing dialogue system as found in the corpus study.

5.1.3 Knowledge sources

The capabilities rely on different types of knowledge sources. Besides domain knowledge, dialogue and task knowledge are crucial.

In dialogue systems two aspects of the dialogue need to be modelled; a generic description of the structure of dialogues that can be used to form a coherent dialogue with the user, and a dynamic representation of the current dialogue. I will refer to the first as the **dialogue model** and the second as the **dialogue history**. The dialogue model is often closely connected to and dependent on the information represented in the dialogue history.

The terms *task* and **task model** are often used when describing dialogue systems, but they can refer to very different phenomena. In an information-providing system, a task model can assist the system in judging if all the required parameters are present and in cases where they are not, determine what type of information to collect from the user. The use of an explicit task model makes the system more flexible since it is simpler to modify or add new tasks.

In a analysis of existing dialogue systems, dialogue models, dialogue histories and task models were used for capabilities A1-A4 and A9. A10 needs both a dialogue model and a dialogue history. For A5-A6 and A11 a dialogue model is required and for A12-A13 a dialogue history. For A5-A6 and A12-A13 domain knowledge would also be useful. For A16-18, A20, A22 and A24 domain knowledge is essential (Flycht-Eriksson 2001). The relations between capabilities and knowledge sources and models are summarised in the table in Figure 5.3. Knowledge sources that are Required to achieve a specific capability are marked an **R** and **L** means at Least one of the knowledge sources is required.

5.2 The LINLIN framework

The development of the ÖTRAF system was based on an existing dialogue systems framework, LINLIN (Jönsson 1997). LINLIN has primarily been used to create dialogue systems that utilise databases as information sources. In these systems, users can pose requests for sets of objects that match certain criteria, or different properties for an object or a set of objects. The dialogue is connected, the user can refer back to the set presented earlier. Users can also shift the focus during the interaction.

The LINLIN framework has a modular architecture with an interpreter, a Dialogue Manager and a generator (see Figure 5.4). The role of the *Interpreter* is to analyse user input and deliver a meaning representation that can be used by the Dialogue Manager. It does this

	Knowledge source/ Capability	Dial Mod	Dial Hist	Task Mod	Dom Mod
A1	To identify the task.	R		R	
A2	To identify sub-tasks and know how they are related to a task.	L		L	
A3	To reason about how much of a task has been achieved so far.		L	L	
A4	To decide what action to take in order to achieve a task.	R	L	L	
A5	To deal with situations in which no answer can be retrieved.	R		L	L
A6	To deal with situations in which the answer includes to much information.	R		R	
A9	To allow the user to over-answer questions.	R	L	L	
A10	To allow the user to initiate clarification sub-dialogues.	R	R		
A11	To allow the user to abandon the current request and pose a new request instead.	R			
A12	To follow shifts in focus.	R	R		
A13	To resolve anaphora and ellipsis.		R		
A16	To map a description to an entity.				R
A17	To detect ambiguous descriptions and deal with them.				R
A18	To detect erroneous descriptions and deal with them.				R
A20	To reason about and derive new information from the information provided by the user.				R
A22	To have domain-related default information.			L	L
A24	To know what the system can and cannot do.				R

Figure 5.3: Capabilities and knowledge sources that support them. Knowledge sources that are Required are marked with an **R** and **L** means at Least one of the knowledge sources is required.

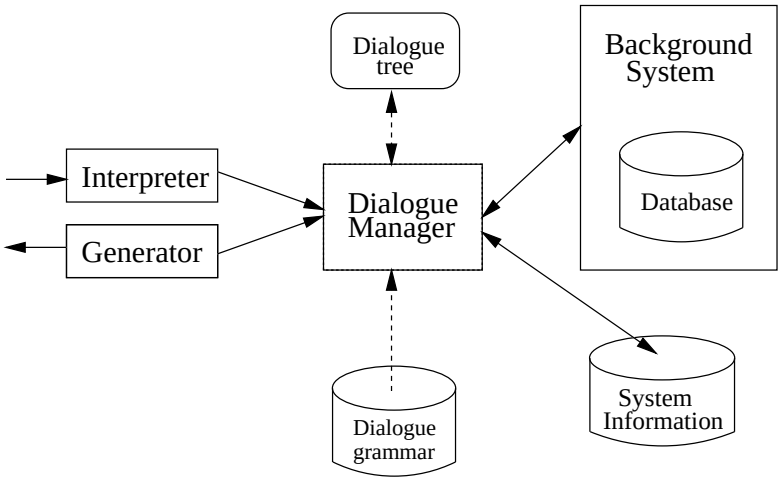

Figure 5.4: Architecture of the LINLIN framework for information-providing dialogue systems.

utilising a unification-based parser (Strömbäck and Jönsson 1998). The *Generator* is responsible for the realisation and presentation of questions and answers specified by the Dialogue Manager. The *Dialogue Manager*'s primary task is to control the flow of the dialogue by deciding how the system should respond to user utterances. As a basis for this, the Dialogue Manager utilises a dialogue model and a dialogue history (Jönsson 1993).

The dialogue model is structured in terms of discourse segments, and these are structured in terms of moves and embedded segments. An initiative-response (IR) structure determines the compound discourse segments, where an initiative opens the IR-segment and the response closes the IR-segment (Dahlbäck 1991). The discourse segments are classified by general speech act categories, such as question (Q) and answer (A) (Jönsson 1997), rather than specialised (cf. (Hagen 1999)), or domain-related (Alexandersson and Reithinger 1995) ones. The action for the Dialogue Manager to carry out, as modelled

$$\begin{bmatrix} Object: & \begin{bmatrix} Manufacturer: & Volvo \\ Model: & 850 \\ Year: & \end{bmatrix} \\ Property: & \begin{bmatrix} Aspect: & speed \\ Value: & \end{bmatrix} \end{bmatrix}$$

Figure 5.5: An OPM for the question, "How fast is a Volvo 850?"

in a dialogue grammar, depends on how domain entities are specified and their relations to other entities in the domain and the dialogue history.

The domain entities are modelled by *Objects* and *Properties*. Objects model the set of objects in the database and Properties denotes a complex predicate ascribed to this set. The parameters in Objects and Properties are application-dependent. *Markers* are also utilised for various purposes (Jönsson and Strömbäck 1998), for example to mark yes/no questions. Structures that represent information about objects, properties and markers are termed OPMs. Figure 5.5 shows a sample OPM which represents the request, "How fast is a Volvo 850?".

There is only one dialogue history, which is maintained by the Dialogue Manager. Thus, the other modules in the system have no memory of the previous interaction since this could cause conflicts. The dialogue history records focal information, that is, what has been talked about and what is being talked about at the moment. It is represented as a dialogue tree and the nodes in the dialogue tree record information in OPMs.

5.3 The ÖTRAF system

This section describes the ÖTRAF system, a multi-modal dialogue system that provides information about local public transportation.

Figure 5.6: Graphical interface of the multi-modal dialogue system ÖTRAF.

Multi-modal means that it is possible to use natural language combined with pointing and clicking in the map, menus, buttons and tables, to interact with the system. The graphical user interface (Qvarfordt 2003) is shown in Figure 5.6.

5.3.1 System architecture

Figure 5.7 shows the architecture of the ÖTRAF system. The solid arrows in the figure indicate how information flows between the modules. The Dialogue Manager should only be concerned with phenomena related to the dialogue with the user. It should not be involved in the process of accessing the information sources or performing domain reasoning. A separate module, a Domain Knowledge Manager, devoted to access of information sources and domain knowledge

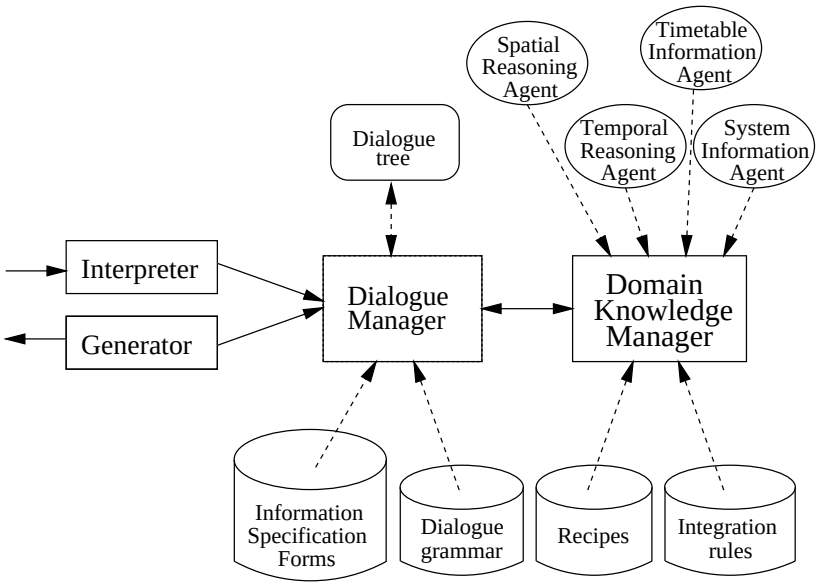

Figure 5.7: Architecture of the ÖTRAF dialogue system.

sources, was therefore introduced (Flycht-Eriksson 2000). The Dialogue Manager is still the central controller of the interaction but it cooperates with the Domain Knowledge Manager with the aim of achieving a natural and intuitive dialogue.

A user utterance is parsed by the *Interpreter* and a feature structure, containing the relevant information in the utterance, is passed on to the *Dialogue Manager*. The Dialogue Manager then takes the given information and processes it into the dialogue context. This means that the new information may be incorporated in a previous partially specified request or that it results in a new request. If a request is fully specified, the Dialogue Manager sends it to the *Domain Knowledge Manager*. The Domain Knowledge Manager decides how the requested information can be retrieved and the result is passed back to the Dialogue Manager in a specified response format, i.e. a feature

structure containing specific fields like status, result and error. The response is then used by the Dialogue Manager to communicate to the *Generator* what response it should present to the user.

5.3.2 Dialogue management

LINLIN only supports some of the required capabilities identified from the corpus. There is only one type of rather simple task-related request, but in many domains the system must deal with more complex requests. Complex requests are concerned with the specification of objects with complex relations. The specification of such an object requires that the user provides information on a specific set of parameters, which often involves several dialogue turns. The specification is used to construct a matching object by retrieving, and sometimes integrating, knowledge from one or several domain knowledge sources and information sources. For example, to answer requests on a trip, the system needs to have a number of parameters specified, such as departure and arrival time and place, before it is able to access the time-tables.

A complex task-related request introduces sub-tasks and the capabilities related to these (A2 and A3). With the introduction of the complex task-related requests the risk of problematic responses to requests increases. Capability A5 should thus be added and capability A6 has to be elaborated upon.

To achieve the capabilities related to complex requests, the dialogue model and dialogue history were complemented with system task models. Explicit system task models were chosen because they can support several tasks and facilitate the modification and addition of new tasks.

The system task model holds the parameters that have to be specified before successful access of the information sources can be performed. They are known as Information Specification Forms (ISFs) (Dahlbäck and Jönsson 1999). Just like OPMs, the ISFs are application-

dependent and they are also used to record information in the dialogue tree, i.e. to inform the Dialogue Manager which parameters have to be provided by the user and which should be requested next.

By introducing this new structure in parallel with the old OPMs, the system can follow shifts in focus and maintain the mixed-initiative character of interaction with the user. For example, if the user has posed an incomplete complex request and the system asks for the parameters necessary to specify the request, the user might want to ask a clarifying question before giving an answer. This shift in focus can be represented in the dialogue by the introduction of an OPM. When the clarification question has been answered, the complex task represented by the ISF can be picked up again, see the example in Section5.3.4.

5.3.3 Domain knowledge management

The complex task-related requests also require capabilities to deal with descriptions that have not been necessary in LINLIN. The system must be able to map descriptions to entities (A16) and to detect and deal with ambiguous and erroneous descriptions (A17 and A18). It should also be able to derive new information from that provided by the user (A20) and to provide domain-specific default information (A22). All these capabilities rely on the availability of domain knowledge and domain reasoning.

The Domain Knowledge Manager was designed to supply the capabilities related to domain knowledge. The Domain Knowledge Manager maps descriptions to entities and reasons about where and how the information should be retrieved and how information from different domain knowledge sources and information systems should be integrated. If the Domain Knowledge Manager encounters a problem it cannot solve by using domain knowledge, a specification of the problem and the needed clarifying information is returned to the Dialogue Manager.

U1: I want to go to the city centre.
S2: The city centre is a big area. Can you point on the map or give more specific information like a landmark or a street?
U3: Are there any bus stops near the Garden square?
S4: There are several bus stops near the Garden square.
< Shows the bus stops in the map >
U5: Then I want to go there from the University.
S6: When do you want to go?
U7: On the 31st of April before lunch.
S8: The 31st is not a valid date, there are only 30 days in April. Give a new date please.
U9: The 30th of April.
S10: The alternative trips are shown in the table.
< Shows a table of trips >

Figure 5.8: A hypothetical dialogue with the ÖTRAF dialogue application.

To facilitate addition, replacement and reuse of domain knowledge sources, an agent-based architecture has been chosen for the Domain Knowledge Manager. The Domain Knowledge Manager consists of several agents: the **Control Agent**, the **Recipe Agent**, the **Integration Agent**, and **Domain Agents**. The agents provide different *services*, for example to retrieve some information given certain parameters, and can also request services from each other.

The Control Agent is a generic domain-independent agent that controls the processing of a request. For this purpose it utilises knowledge structures called *recipes*. A recipe consists of a series of services from different agents, which are executed in order to construct an answer to the request. The Recipe Agent is responsible for the construction of recipes that match the requests. The Integration Agent is a domain-independent agent that can integrate several response alternatives into one answer, utilising *integration rules*, which contain both domain heuristic and more general principles. The Domain Agents are responsible for the appropriate access of domain knowledge sources and are able to perform sophisticated knowledge reasoning in order to retrieve the information.

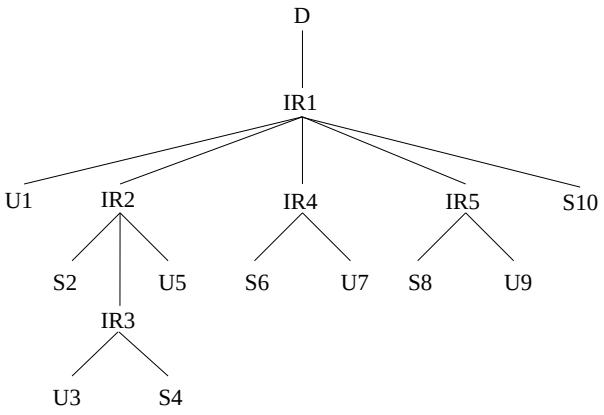

Figure 5.9: The dialogue tree resulting from the dialogue in Figure 5.8.

In the domain of local public transportation, four different domain agents are present. The **Temporal Reasoning Agent** contains a calendar and reasons about temporal expressions. The **Spatial Reasoning Agent** consists of a Geographical Information System and a reasoning mechanism used to deduce the relations between geographical objects. The **Time-table Agent** accesses an information source on the Internet, which contains the time-tables for local public transportation. There is also a **System and Help Information Agent** with system information, like references to human operators for questions outside the scope of time-table information, for example, on lost property.

5.3.4 Examples

To illustrate how the Dialogue Manager (DM) and the Domain Knowledge Manager (DKM) cooperate in processing requests clarifications, consider the hypothetical dialogue shown in Figure 5.8. The dialogue tree in Figure 5.9 shows the resulting structure of the dialogue.

The first utterance, U1, initiates a trip ISF. Information about the arrival location provided by the user is inserted into the ISF under Arr, which results in the structure presented in Figure 5.10. The ISF indicates that the departure place and time has to be further specified by the user with the marker *req* in Dep and TTime (TravelTime).

$$\begin{bmatrix} Type: & Trip \\ Arr: & [\ Area: \quad City\ center\ ] \\ Dep: & req. \\ TTime: & req. \\ TType: & opt. \end{bmatrix}$$

Figure 5.10: The ISF in IR1 after processing U1.

However, before continuing the dialogue and asking the user for the information that is missing in the ISF, the DM asks the DKM to validate the provided values. This validation is performed in order to detect vague or erroneous information that might have been given by the user.

The arrival location in a trip ISF will be used to find suitable bus stops that can be used to search the time-table database. The validation of the arrival location therefore means that the Spatial Reasoning Agent tries to map the location to a small set of bus stops. In this case it discovers that Area: *City Centre* is a too vague description since it corresponds to too many stops, in our case more than 5 stops. The DM is informed of this and is also given the information that more specific information like a point, a landmark or a street is required, Figure 5.11.

$$\begin{bmatrix} Status: & Error \\ Item: & [\ Area: \ City\ center\ ] \\ Type: & \begin{bmatrix} TooMany: & BusStops \\ & [\ Up: \ 5\ ] \end{bmatrix} \\ Solution: & \begin{bmatrix} SpecInfo: & \{Point, \\ & Landmark, \\ & Street\} \end{bmatrix} \end{bmatrix}$$

Figure 5.11: The response from the DKM to the domain validation of the arrival location.

Thus, the user will not be asked to provide the value of another parameter since it would be an implicit confirmation that the arrival place is correct, instead a new IR-unit, IR2 in the dialogue tree, is created and a clarification, S2, is initiated based on the information from the DKM that indicates the problematic item, the type of problem and a possible solution to the problem.

Instead of answering the system's question, the user takes the initiative by requesting new information, U3. This request results in a new IR-unit, IR3, to be inserted into the dialogue tree as a clarification of the system's clarification in IR2, as shown in Figure 5.9. The utterance is a simple request and the DM utilises an OPM to model this, Figure 5.12.

$$\begin{bmatrix} Obj: & \begin{bmatrix} \#1 & [\ Stop: \ ?\ ] \\ \#2 & \begin{bmatrix} Landmark: & Garden \\ & square \end{bmatrix} \end{bmatrix} \\ Prop: & \begin{bmatrix} Near: & \begin{bmatrix} Place1: & \#1 \\ Place2: & \#2 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 5.12: The OPM in IR3 after processing U3.

To answer this request means reasoning about spatial relations between geographical objects. The request is therefore sent to the DKM, which asks the Spatial Reasoning Agent for information. The request is successfully processed and some nearby bus stops are found

and sent back to the DM utilising the structure in Figure 5.13. The DM can then ask the generator to present them to the user, S4.

$$\begin{bmatrix} Status: & Success & \\ Stops: & \begin{bmatrix} \begin{bmatrix} Name: & Centrum\ Snickareg.\ 30 \\ Id: & 1268 \end{bmatrix} \\ \begin{bmatrix} Name: & Linnegatan \\ Id: & 1220 \end{bmatrix} \\ \begin{bmatrix} Name: & Stora\ torget \\ Id: & 450 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figure 5.13: The response from the DKM to the OPM in IR3.

The user responds to this answer by confirming his departure location, U5, and thereby responds to the request S2 of IR2. He also provides an arrival location. This new information is represented in the OPM of IR2, Figure 5.14.

$$\begin{bmatrix} Obj: & \begin{bmatrix} \#1 \begin{bmatrix} Landmark: & Garden\ square \end{bmatrix} \\ \#2 \begin{bmatrix} Landmark: & University \end{bmatrix} \end{bmatrix} \\ Prop: & \begin{bmatrix} Arr: & \#1 \\ Dep: & \#2 \end{bmatrix} \end{bmatrix}$$

Figure 5.14: The OPM in IR2 after processing U5.

The DM resumes processing the ISF in IR1 and updates it with the arrival and departure location based on the information in the OPM of IR2. Information about the arrival location is added to the previously provided information in the field Arr. The new information about the departure location is inserted into the field Dep, yielding the structure in Figure 5.15.

$$\left[\begin{array}{ll} Type: & Trip \\ Arr: & \left[\begin{array}{ll} Area: & City\ center \\ Landmark: & Garden\ square \end{array}\right] \\ Dep: & \left[\ Landmark: \quad University\ \right] \\ TTime: & req. \\ TType: & opt. \end{array}\right]$$

Figure 5.15: The ISF in IR1 after updates with information from the subtree in IR2.

Again, the DM asks the DKM for domain validation of the partially specified ISF. Since both locations can be mapped to a limited number of bus stops the ISF is approved by the DKM. The DM now needs to have a time to complete the ISF, and consequently a new IR-unit, IR4 in the dialogue tree, is created and the user is, in utterance S6, asked for this. The answer U7 is a valid response to S6 and produces a new OPM, see Figure 5.16.

$$\left[\begin{array}{ll} Obj: & \#1\left[\begin{array}{ll} Date: & \left[\begin{array}{ll} Day: & 31 \\ Month: & April \end{array}\right] \\ Time: & \left[\begin{array}{ll} POD: & lunch \\ Mod: & before \end{array}\right] \end{array}\right] \\ Prop: & \left[\ TTime: \quad \#1\ \right] \end{array}\right]$$

Figure 5.16: The OPM in IR4 after processing U7.

The new information from IR4 is then inserted as TTime in the ISF of IR1. This results in a fully specified Trip ISF, Figure 5.17.

$$\left[\begin{array}{ll} Type: & Trip \\ Arr: & \left[\begin{array}{ll} Area: & City\ center \\ Landmark: & Garden\ square \end{array}\right] \\ Dep: & \left[\ Landmark: \ University\ \right] \\ TTime: & \left[\begin{array}{l} Date: \left[\begin{array}{ll} Day: & 31 \\ Month: & April \end{array}\right] \\ Time: \left[\begin{array}{ll} POD: & lunch \\ Mod: & before \end{array}\right] \end{array}\right] \\ TType: & opt. \end{array}\right]$$

Figure 5.17: The ISF of IR1 after updates with information from IR4.

The ISF is again sent to the DKM for validation. When the Temporal Reasoning Agent tries to map the temporal description in TTime to a format suitable for time-table database search it discovers the erroneous date. The DKM then returns a response, Figure 5.18, to the DM informing it of the error. The DM initiates a new clarification IR-unit, IR5, and a clarification is formulated, S8.

$$\left[\begin{array}{ll} Status: & Error \\ Item: & \left[\ Date: \left[\begin{array}{ll} Day: & 31 \\ Month: & April \end{array}\right]\ \right] \\ Type: & \left[\ NotValid: \left[\begin{array}{ll} Month: & April \\ Up: & 30 \end{array}\right]\ \right] \\ Solution: & \left[\ SpecInfo: \ \{Date\}\ \right] \end{array}\right]$$

Figure 5.18: The response from the DKM to the domain validation of the time description.

The user responds to the system's clarification request and provides a new date, U9. The response is modelled in an OPM in IR5, Figure 5.19.

$$\begin{bmatrix} Obj: & \#1 \begin{bmatrix} Date: & \begin{bmatrix} Day: & 30 \\ Month: & April \end{bmatrix} \end{bmatrix} \\ Prop: & [\ TTime: \ \#1\ ] \end{bmatrix}$$

Figure 5.19: The OPM of IR5 after U9.

The information in the clarification request IR-unit, IR5, is propagated to the ISF of IR1 which is updated. This time, the new information replaces the old in TTime since it was erroneous. The resulting ISF is presented in Figure 5.20.

$$\begin{bmatrix} Type: & Trip \\ Arr: & \begin{bmatrix} Area: & Citycenter \\ Landmark: & Gardensquare \end{bmatrix} \\ Dep: & [\ Landmark: \ University\ ] \\ TTime: & \begin{bmatrix} Date: & \begin{bmatrix} Day: & 30 \\ Month: & April \end{bmatrix} \end{bmatrix} \\ Time: & \begin{bmatrix} POD: & lunch \\ Mod: & before \end{bmatrix} \\ TType: & opt. \end{bmatrix}$$

Figure 5.20: The ISF of IR1 after integration with the information in IR5.

Once more a validation of the ISF is performed by the DKM. This time no problems are detected and a search for suitable trips can finally be done. The DKM does this by first asking the Spatial Reasoning Agent to map the departure and arrival locations to two sets of bus stops, then asking the Temporal Reasoning Agent to map the vague temporal description to a precise time interval. Given this information, the DKM then searches the time-table database to find one or more trips that fulfill the requirements. The resulting trips are sent back to the DM and displayed to the user, S10.

5.4 The MALIN framework

ÖTRAF involves many modifications of the LINLIN framework. Some of these were introduced in a new framework, MALIN. The most noticeable is the change in architecture with the introduction of a separate module, the Domain Knowledge Manager, that deals with domain knowledge representation and reasoning.

Of the capabilities revealed by the corpus study (Figure 5.2), but not supported by LINLIN, capabilities A5-A6, A16-A18, A20 and A22 are related to domain knowledge and information source access.

Access of information sources can be problematic in two ways: no answer to the request can be produced (A5), or too many answers are found (A6). The first situation can occur if a request is inconsistent or if no object meets all the restrictions of the request. Two different approaches to dealing with this are for the system to try and fix it itself, or for the system to help the user handle the situation. The first approach includes relaxing some of the constraints or resolving the inconsistency, both of which require reasoning about the domain. If the system fails or does not try to solve the problem itself, it can give the user as much help as possible when he or she has to deal with the problem, for example by stating the cause of the problem and suggesting how the request should be modified. MALIN supports both approaches. The second problematic situation, where a request has resulted in too many answers from the information sources, can arise from requests that are not specific enough. The solution chosen in MALIN is to use the domain knowledge and decide which constraints should be asked for in order to specify the request, thus helping the user to formulate a more specific request.

Mapping descriptions to entities (A16) is similar to retrieving answers to requests. There are also two problems that have to be dealt with in this case; the descriptions can be ambiguous (A17) and correspond to several entities, or they can be unsatisfiable (A18) and not correspond to any entity. Ambiguous descriptions can be dealt with by either asking the user to choose one of the matching entities or asking the

user to provide a distinguishing feature. In MALIN, both approaches are used depending on how many alternatives there are. Unsatisfiable descriptions can be dealt with in three different ways: find and inform the user of faulty presuppositions that cause the description to be unsatisfiable, find and present near misses by relaxing some of the features in the description, or inform the user of the problem giving as helpful information as possible. MALIN uses the first and third of these; if a faulty presupposition is present it is presented to the user. Other problems are also brought to the user's attention together with helpful information on how they can be corrected.

Other capabilities that rely on domain knowledge include reasoning about and deriving new information from the information provided by the user (A20) and using domain-related default information about requests (A22). If a user has specified a request that is not complete, the system can fill in the empty spaces by inserting default information or by deriving it from the information that has been provided.

The use of a specialised Domain Knowledge Manager has a number of advantages. The first is that dialogue management becomes more focused as it only has to consider dialogue phenomena, while domain-specific reasoning is handled by the Domain Knowledge Manager. The second major advantage is that once an interface between the Dialogue Manager and the Domain Knowledge Manager has been specified, they can be developed and experimented with independently of each other. This, in turn, facilitates porting a system to new domains since domain-related issues are included in the domain knowledge sources. Another advantage is that the domain knowledge sources can easily be modified, exchanged and reused. Finally, with a separate module for domain knowledge management, the domain-dependent features and information sources are gathered in one location.

5.5 Implications for ontology use in dialogue systems

The implementation of the Domain Knowledge Manager in ÖTRAF and MALIN has two drawbacks. The representation of domain knowledge is varies greatly since the only requirement on the domain agents is the syntax of the services they provide. Thus, the type of knowledge and reasoning they provide can differ greatly. A more declarative and uniform approach to domain knowledge representation is desirable to further facilitate reuse and portability. The monopoly of the Domain Knowledge Manager on the domain reasoning also causes some tasks to be delayed. For example, ambiguities during interpretation can be resolved with the use of domain knowledge and focus management can also be improved if the Dialogue Manager has access to domain knowledge reasoning mechanisms. Thus, ontologies as representation format and a general Ontology Manager that can communicate with the Interpreter, the Dialogue Manager and the Domain Knowledge Manager is a possible solution.

Chapter 6

Domain Knowledge and Ontologies in BIRDQUEST

In this chapter a dialogue system, BIRDQUEST, *based on the* MALIN *framework, is presented with a focus on ontology usage.*

The next step towards a portable dialogue systems framework capable of sophisticated domain reasoning is described in this chapter. In the BIRDQUEST application, two new features are introduced, the use of domain ontologies and information extraction techniques to access unstructured textual information.

ÖTRAF and MALIN use structural information to provide answers to user requests. However, most of the information available in electronic formats comes as text, making up huge sets of unstructured information in natural language. One way to make this information available in dialogue systems is to use information extraction techniques that pull out relevant information from textual documents and compile it to a usable database.

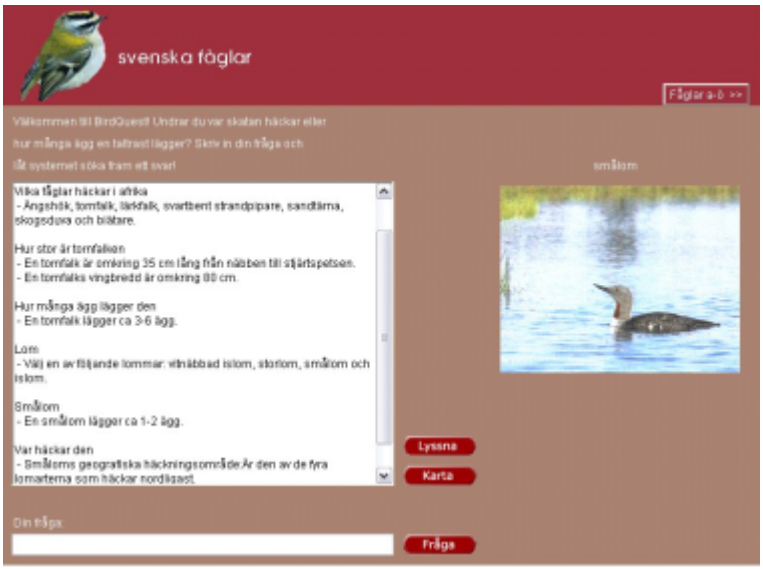

Figure 6.1: Graphical user interface of BIRDQUEST.

This approach has been used in the development of BIRDQUEST, a dialogue system that provides information collected from a bird encyclopedia. In BIRDQUEST, users can ask questions in natural language and also use menus to enter bird names. The interface is shown in Figure 6.1. A translation of the full dialogue in the interface is given in Figure 6.2, and another example of a dialogue with BIRDQUESTcan be found in Figure 1.1.

6.1 System architecture

The system architecture combines an interaction component with an information processing component and a set of knowledge sources that define the scope of the language and the domain of the system.

U1: Which birds nest in Africa?
S2: Montagut's harrier, common kestrel,[...], Stock dove and European Bee-eater.
U3: How large is the common kestrel?
S4: A common kestrel is about 35 cm long from the beak tip to tail. A common kestrel's wingspan is about 80 cm.
U5: How many eggs does it lay?
S6: A common kestrel lays approx. 3-6 eggs.
U7: Loon
S8: Select one of the following loons: Yellow-billed loon, black-throated loon, red-throated loon and great northern loon
U9: red-throated loon.
S10: A red-throated loon lays approx. 1-2 eggs.
U11: Where does it live?
S12: Red-throated loons geographic nesting area: Is the one of the four loons that nest northernmost
U13: What does a waterfowl look like?
S14: Choose one of the following Waterfowl: Barnacle Goose, Tufted Duck, Bean Goose, Whooper Swan, Gadwall,[...]
U15: gadwall
S16: You can receive information about the size and plumage of a Gadwall. Please specify what you are interested in.
U17: size
S18: A Gadwall has a length of 56 cm from beak tip to tail. Information about wingspan is missing for Gadwall
U19: Where do mute swans live?
S20: Information about a mute swan's range:
The mute swan can be found nesting mainly in western Europe, but also in southeast Europe and Turkey.
U21: How long does a mute swan live?
S22: Information about maximum age of a mute swan:
Maximum age in the wild: 24 years and 3 months
U23: Which birds can I see in Sweden in December?

Figure 6.2: A dialogue illustrating the BirdQuest system.

The Information Processing Component takes collections of unstructured or semi-structured documents and transforms them into structured information that can be used by the dialogue system during

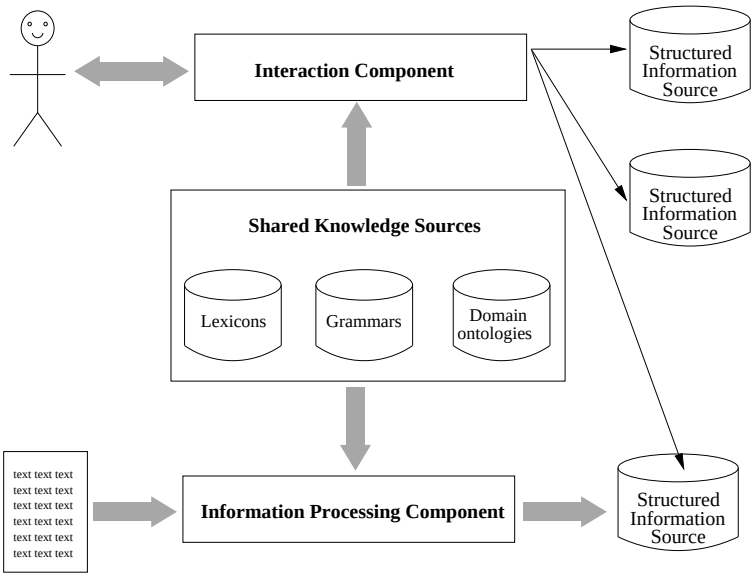

Figure 6.3: The processing modules and the shared knowledge sources of BirdQuest that are used for dialogue interaction and information extraction.

interaction with the user. The transformation is based on information extraction techniques, and the documents are analysed in several stages going through lexical, morphological, syntactical and semantic analysis, in each step adding more structure to the documents. A wide variety of pattern extraction rules are then applied to the documents. The objective is to fill the database with relevant information and ignore text segments that do not meet the information needs of the users. The ontology serves as an important knowledge source for the information processing component to identify the relevant information and provide semantics for the text processing (Flycht-Eriksson et al. 2003).

The Interaction Component has been implemented based on MALIN with a parser that combines full and partial parsing techniques, the Dialogue Manager and a Domain Knowledge Manager have been extended to utilise ontological knowledge. The generator is template-based and adapted to present values and text snippets extracted from the text source. A separate question analysis module that uses the ontology has also been developed but not yet integrated into the system. The architecture of the combined system is shown in Figure 6.3

6.2 The BIRDQUEST ontology

The BIRDQUEST ontology was developed manually following the usual stages of development and construction, cf. Section2.3:

1. Specification of scope and purpose
2. Ontology capture
3. Ontology coding
4. Evaluation

6.2.1 Scope and purpose

The primary purpose of the ontology in BIRDQUEST is to support the dialogue system's tasks of cooperatively formulating information requests together with the user and of accessing and retrieving the requested information. The ontology should also model the type of information to be extracted from the source document and stored in the database. In the case of a combined system for dialogue interaction and information extraction, it is crucial that the ontology captures both the system-oriented view to reflect the information sources and the user-oriented, often more naive, view of the domain.

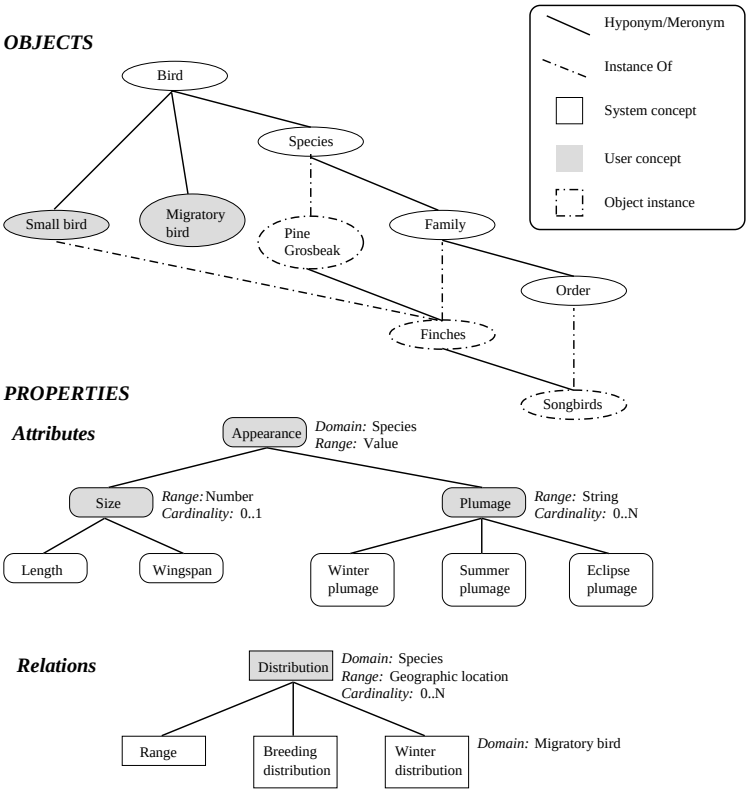

Figure 6.4: Part of the integrated ontology representing the conceptualisations of both bird encyclopedia and users.

The bird encyclopedia almost exclusively contains information of a factual character with a focus on objects. The ontology was therefore restricted to modelling this type of information in terms of objects, attributes and relations, leaving out events/processes.

6.2.2 Ontology capture

To identify relevant concepts to include in the ontology two corpora were analysed, the bird encyclopedia (Staav and Fransson 1991) and a question corpus.

The bird encyclopedia

The organisation and structure of the book were taken as a starting point for the identification of ontology concepts. It uses the K H Voous system for dividing birds into orders, families and species. For each of the categories, information about certain properties was presented, most of which was species specific. The book was manually analysed, to identify the objects, properties and relations relevant for the purpose of information extraction. The analysis gave a total of 8 categories (various groupings of birds, and geographical locations), 30 attributes and 3 relations, some of which are presented in Figure 6.4.

The question corpus

The corpus used consists of 264 questions about birds. It was collected by The Swedish public service television company on a web site for one of their nature programs, where the public could send in questions. The analysis of the corpus focused on questions that were deemed as within the boundaries of the application leaving out, for example, questions concerning veterinarian treatment of birds or explanations of behaviour. The analysis of the remaining questions in the corpus revealed that the users' view of the domain in most cases corresponds to the one found in the encyclopedia, but a small number of new categories and several new attributes were identified. These new concepts were of three types:

- Users sometimes utilised another way of categorising birds than the zoological oriented taxonomy in the bird encyclopedia, talking about "Spring birds", "Small birds", "Migratory birds", and "Birds of prey", etc.
- In many cases the properties of the birds were more general than the terms used in the book, for example questions about size which include both wingspan and length.
- A number of properties were not present in the bird encyclopedia but closely related to them, such as weight and speed of flight.

From the analysis of the encyclopedia, a conceptualisation underlying the structure and presentation of information to be extracted by the Information Processing Component was constructed. The result was a system-oriented domain ontology representing experts' (the book authors) view of the domain. The non-expert view of the domain, useful for dialogue interaction as provided by the question corpus, was then integrated in the following manner:

- By allowing multiple inheritance, new links between existing categories and new categories were added. Note, for example, how the category SmallBird is introduced and that Finches is multiple linked to both this category and Family in Figure 6.4.
- In a similar manner vague properties were introduced and linked to the existing properties. Figure 6.4 illustrates how two new levels were introduced, Wingspan and Length are sub-properties of Size, which in turn is a sub-property of Appearance.

Properties from the question corpus that could not be linked through hyponym relations to existing properties in the system-oriented conceptualisation where left out. This was because there was no existing information regarding these in the bird encyclopedia and neither was there any existing information in the database to be used by the dialogue system.

With 4 new objects and 6 new properties introduced by the analysis of the question corpus, the resulting ontology contains 12 objects, 36 properties and 3 relations.

6.2.3 Ontology coding

The ontological knowledge was represented in two different knowledge sources, one containing the concepts and their taxonomical relations, and one holding the individuals and facts, i.e. object instances and their taxonomical relations. The factual part was generated semi-automatically by extracting instances from the bird encyclopedia.

Each concept had a unique name, a definition in natural language, and a tag stating if it was system or user derived, i.e. if it came from the bird encyclopedia conceptualisation or the question corpus. Properties and Relations also had domain and range restrictions, which stated what type of objects they were applicable to. This is illustrated in Figure 6.4 where the relation winter distribution has domain Migratory bird and range Geographical location. There were also cardinality restrictions for the properties and relations.

6.2.4 Evaluation

To verify, among other things, that the ontology met its intended purpose it was evaluated in the context of its usefulness for dialogue interaction (Flycht-Eriksson and Jönsson 2003). The results are presented in Section6.5

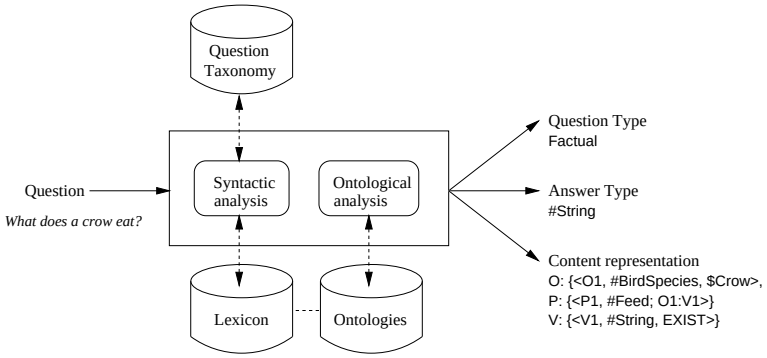

Figure 6.5: A question analysis module that includes two components for syntactic and ontological processing of questions and utterances. It utilises three types of knowledge sources, a question taxonomy, a semantic lexicon and an ontology.

6.3 Question Analysis

The module for question analysis in BIRDQUEST is new compared to the MALIN framework, which utilised a chart parser as a basis for syntactic analysis. The new module uses a commercial functional dependency grammar (FDG) developed by Connexor[1] (Tapanainen and Järvinen 1997), for syntactic analysis, and couples this with an ontology-based analysis.

The module can be easily adapted for various applications by exchanging the application-specific knowledge sources. Figure 6.5 shows the architecture of the module and the knowledge sources it uses. The question analysis module takes a question or information request as input and tries to determine the question type, answer type and content. The question type information is a category from the question type taxonomy, e.g. factual, explanation or definition. The answer type can be a boolean, string, number or an ontological entity. The

[1]The product is now called Connexor Machinese Syntax.

content representation models the objects, events, attribute and relations according to the domain ontology.

6.3.1 Representation format

In Figure 6.5 a representation of the question "What does a crow eat?" is presented. The way the content is modelled reflects to a high degree how knowledge of a domain is represented in the ontology. The tuples represent objects, attributes and relations. The first member of a tuple is always an identifier (O, A, R, or V for objects, attributes, relations and values respectively). For objects and events, the second member is an ontological type, and the third member can be either a quantifier or an instance in the knowledge base. For properties, i.e. attributes and relations, the second element contains also the argument structure expressed as identifiers of objects and values.

To produce the ontological content representation a semantic lexicon is used. As described in Section7.1.2 lexical items, i.e. word and phrases, are linked to ontological concepts through reference relations. Since words can be ambiguous and there can be synonyms, there exist a many to many relationship between lexical items and concepts.

In Figure 6.6 a few examples of words and their ontological correspondence are given. The lexicon is in Swedish but a column with an English translation has been added for illustration.

6.3.2 Syntactic and ontological analysis

The goal of the syntactic and semantic analysis is to construct a content representation of a question that capture the semantics, utilising as much linguistically motivated information as possible. The syntactic analysis of questions is divided into a number of different steps.

Lexical item		*Semantic*
Swedish	English	
familjer	families	$< O, Family, ALL >$
längsta	longest	$< A, Length; ? : V >$ $< V, Number, MAX >$
utseende	appearance	$< A, Appearance; ? :? >$
ser-ut	look-like	$< A, Appearance; ? :? >$
lever	live	$< A, OldestAge; ? :? >$
lever	live	$< R, Distribution; ? :? >$

Figure 6.6: Examples of items in a semantic lexicon. Two parts are shown, the word or phrase and the semantics in terms of ontological concepts.

The first step is to do a syntactic analysis of the question using the Functional Dependency Grammar (FDG) parser. It is a full syntactic parser that produces both morphological information for word-form tokens and functional dependencies representing relational information in sentences. The question is annotated with information about part-of-speech, dependencies, and syntactical functions.

The next step is to process the output from the FDG parser in order to identify noun and verb phrases, which hold the objects, properties, and relations of the question. This information can be extracted using information about dependencies and syntactic functions, i.e. it is implicitly available in the output from FDG.

At this point the analysis of the question is quite detailed on a syntactic level. Next, a semantic analysis is performed. The constituents of the question are looked up in a domain-specific semantic lexicon. The final step is to transfer the dependencies between the words to the semantic concepts that have been retrieved from the lexicon. However, they are no longer seen as directed, and are after this step simply referred to as *links*.

The partial semantic interpretation that has been produced is then further processed and specified based on the ontology. Knowledge of domain and range restriction for properties and relations are utilised to filter out erroneous links between objects, values, properties and relations, and to add missing objects or values.

6.3.3 Examples

In the following section, we present four examples drawn from the dialogue in Figure 6.2, and in detail explain how the question analysis operates, and how the ontological analysis contributes to the creation of a knowledge-rich and unambiguous representation of question contents with explicit semantics.

The figures in this section have two parts. First the output from the parser used for syntactical analysis is presented. Then the output from the question analysis module presented. The number indicates the source in the syntactic analysis of an ontological concept. If a number is missing the information has been added by the ontological analysis. Strike-through text denotes information retracted by the ontological analysis.

The first example, presented in Figure 6.7, shows how the question "What does a waterfowl look like?" is interpreted. The output from the FDG parser basically consists of six different parts, some separated by tabs, and other by white spaces. The leftmost column contains numerical references to the constituents of the question interpreted. The second column contains the actual word and the third holds the word's stem. The fourth column represents the syntactic function of the constituent, as well as a reference to the constituent that it is dependent on. The last column has two different parts. The leftmost part contains the surface syntactic tag, while the rest of the column contains morphological information about the word.

The information provided by the FDG parser is then used to make a semantic analysis of the question. The first column refers to the con-

Output from the FDG parser

1	Hur	hur	man:>2	%AH ADV
2	ser	se	main:>0	%MV V PRES
3	en	en	det:>4	%>N DET SG NOM
4	and	and	subj:>2	%NH N SG NOM
5	ut	ut	advl:>2	%AH ADV
6	?	?		%INTERP INTERP QuestionMark

Output from the question analysis

4: O: Family:Waterfowl
2+5: P: Appearance
V: String

Links: {{5, 2}, {4, 2}}
Question type: Fact
Answer type: String: EXIST

Figure 6.7: Analysis of, "What does a waterfowl look like?" (swe. "Hur ser en and ut?").

stituents that give rise to the semantic entities; the second contains either an O, A, or R for objects, attributes and relations, respectively; the third contains the semantics proper.

First of all, the syntactic representation receives a content representation, by looking up the words in a semantic lexicon. This lexicon is basically a reflection of the ontology, i.e., they use the same basic categories to describe objects, attributes and relations in the world. Here, the fourth constituent in the question is mapped to the object Waterfowl. The second and fifth constituent are mapped to the attribute Appearance. In Swedish, "se ut" (eng. look like) is a particle verb, and in the current question, the particle 'ut' is located three constituents to the right of the verb 'se'. However, since 'ut' is dependent on 'se' and no constituents are dependent on 'ut', we can conclude that this is essentially a single unit. The dependencies are

Output from the FDG parser

1	Var	var	loc:>2	%AH ADV
2	lever	leva	main:>0	%MV V PRES
3	knölsvanar	knölsvan	subj:>2	%NH N PL NOM
4	?	?		%INTERP INTERP QuestionMark

Output from the question analysis

1: O: SpatialObject: EXIST
2: R: Distribution
3: O: BirdSpecies:MuteSwan
2: P: ~~OldestAge~~

Links: {{1, 2}, {3, 2}}}
Question type: Fact
Answer type: SpatialObject: EXIST

Figure 6.8: Analysis of, "Where do mute swans live?" (swe. "Var lever knölsvanar?".

also relabelled to *Links* and their direction is removed. They are kept since they are later used for ontological disambiguation. In this case, the interpretation is unambiguous since it contains one attribute and one object where the object is of the attribute's domain. The ontological analysis only adds the expected answer type *String* based on the range restriction of the attribute.

Figure 6.8 illustrates how the question "Where do mute swans live?" is analysed. Here we have an ambiguity because of the word 'lever' (eng. live), which can refer to both the distribution of a bird, as well as its life span. This is reflected as the second constituent receives two different readings. Disambiguation is performed using the ontology to compare the given objects with the attribute and the relation. Since the relation Distribution has the domain BirdSpecies and the range SpatialObject this is the preferred interpretation and the attribute OldestAge is ignored.

Output from the FDG parser

1	Hur	hur	ad:>2	%>A ADV
2	länge	länge	advl:>3	%AH ADV
3	lever	leva	main:>0	%MV V PRES
4	en	en	det:>5	%>N DET SG NOM
5	knölsvan	knölsvan	subj:>3	%NH N SG NOM
6	?	?		%INTERP INTERP QuestionMark

Output from the question analysis

1, 2, 3: P: OldestAge
5: O: BirdSpecies:MuteSwan

Links: {{1, 2}, {2, 3}, {5, 3}}
Question type: Fact
Answer type: Number: EXIST

Figure 6.9: Analysis of, "How long does a mute swan like?" (swe. "Hur länge lever en knölsvan?").

In the analysis of "How long does a mute swan live?" in Figure 6.9 we once again find the ambiguous word 'lever' (eng. live). In this case, however, no ambiguity arises since, "Hur länge lever" (eng. "How long ... live?") is treated as a single question phrase. There are a number of different question phrases, or generally multi-word units, that can be stored as single constituents in the semantic lexicon for more efficient processing.

The last example, presented in Figure 6.10, shows a more complex question "Which birds can I see in Sweden in December?". Here, we conclude that the user is interested in a number of birds matching certain criteria, hence the answer type BirdSpecies: ALL. The ontological analysis uses the ontology to reason about the proper interpretation of the spatial and temporal information. Temporal and spatial aspects are treated using two special types of relations, TemporalRestrict and SpatialRestrict that can have objects, attributes

Output from the FDG parser

1	Vilka	vilken	det:>2	%>N DET PL NOM
2	fåglar	fågel	obj:>5	%NH N PL NOM
3	kan	kunna	v-ch:>5	%AUX V PRES
4	jag	jag	subj:>3	%NH PRON SG NOM
5	se	se	main:>0	%MV V INF
6	i	i	advl:>5	%AH PREP
7	Sverige	sverige	pcomp:>6	%NH N SG NOM
8	i	i	advl:>5	%AH PREP
9	december	december	pcomp:>8	%NH N SG NOM
10	?	?		%INTERP INTERP QuestionMark

Output from the question analysis

1, 2: O: BirdSpecies: ALL
5: R: Distribution
7: O: Country:Sweden
9: O: Month:December
R: TemporalRestrict(5:9)
R: SpatialRestrict(5:7)

Links: {{1, 2}, {2, 5}, {7, 5}, {9, 5}}
Question type: Fact
Answer type: BirdSpecies: ALL

Figure 6.10: Analysis of, "Which birds can I see in Sweden in December?" (swe. "Vilka fåglar kan jag se i Sverige i December?").

or other relations as domain and TemporalObject or SpatialObject as range, respectively. In this case, they are attached to the relation Distribution since this has both temporal and spatial aspects, and a full interpretation of the question is derived.

6.4 Dialogue and domain knowledge management

The Dialogue Manager in BirdQuest utilises the OPM structures described in Section5.3.2 to represent requests. Since most requests regard an object and property, and these are simple concepts, there is no need for ISFs to describe complex objects.

Since only one information source is used, a relational database with the information extracted from the bird encyclopedia, the Domain Knowledge Manager in BirdQuest does not utilise the agent architecture of MALIN. Instead it communicates with an Ontology Manager that provides domain reasoning capabilities and utilises a tool, QUAC, to transform user requests sent to it by the Dialogue Manager to SQL questions (nlpFarm 2004).

The first step when a request is received is to check whether the object, property and/or value provided match. This means asking the Ontology Manager for information about the domain and range restrictions of the property. If they do not match, the ontology is once more used to see if there is a hypernym or hyponym of the object that does match. If there is a small number they can be used directly to gather information from the database, but if there are too many, a clarification can be sent back to the Dialogue Manager that asks the user to chose one of the instances collected from the ontology.

If object and property do match, the next step is to check if the property can be used for database access or if it has to be transformed into sub-properties. If the property or its direct children are system-oriented they are used, otherwise a clarification with the alternatives is sent back to the DM.

Next the DKM creates one or more SQL questions based on the request and accesses the database. The retrieved answers are returned to the DM. An example of the cooperation between DM and DKM is given in the next section.

6.4.1 Examples

To illustrate the work of the DM and DKM consider the sequence U13-S18 in the dialogue in Figure 6.2. This is an example of a request for a property value, which results in the interpretation presented in Figure 6.11. Since it contains both an object and a property it is considered fully specified by the Dialogue Manager, and is therefore sent to the Domain Knowledge Manager for retrieval of information.

```
<O1, Family, Waterfowl>
<A1, Apperance; O1:V1>
<V1, String, EXIST>
```

Figure 6.11: The request for "What do waterfowl look like?" sent by the DM to the DKM.

The DKM checks if the property Appearance is applicable to the object of type Family. This is not the case, as can be seen in the domain ontology in Figure 6.4. The ontology is therefore traversed to see if there are any hyponyms/hypernyms that can be used instead. The type BirdSpecies is, and the instances of bird species are collected from the knowledge base. As there are many bird species that belong to the family Waterfowl, the DKM decides to send a clarification message to the DM instead of creating new requests. The alternatives are returned as a list, see Figure 6.12.

```
{<O1, BirdSpecies, Barnacle>,
 <O2, BirdSpecies, Goose>,
 <O3, BirdSpecies, TuftedDuck>,
 ...,
 <O6, BirdSpecies, Gadwall>}
```

Figure 6.12: A list of bird species belonging to the family Waterfowl that can be used for a clarification, sent by the DKM to the DM.

Thus, the DM initiates a clarification sub-dialogue, see S14, asking for a more specific object. When the user has made her choice, U15,

the request is once again sent to the DKM. The same attribute and value is used but a new object has been inserted in the request, see Figure 6.13.

```
<O1, BirdSpecies, Gadwall>
<A1, Apperance; O1:V1>
<V1, String, EXIST>
```

Figure 6.13: The request for "What do a gadwall look like?" sent by the DM to the DKM.

This time, the object Gadwall and the property Appearance match. However, when checking the property in the ontology, the DKM finds that it is too vague to be used for database access since it has no immediate sub-properties that can be used for database access. Therefore a clarification message is sent to the DM, and a new clarification sub-dialogue takes place (S16-U17). This time the DKM returns two alternatives, that are the sub-properties of Appearance, see Figure 6.14.

```
<A1, Size; ?:?>
<A2, Plumage; ?:?>
```

Figure 6.14: A list of sub-properties for Appearance that can be used for a clarification, sent by the DKM to the DM.

When the user has specified the property, the request is sent back to the DKM. This time the object and value are the same with the property replaced, Figure 6.15.

```
<O1, BirdSpecies, Gadwall>
<A1, Size; O1:V1>
<V1, String, EXIST>
```

Figure 6.15: The request for "What is the size of a gadwall?" sent by the DM to the DKM.

This time the DKM can map the property to several sub-properties suitable for access to the structured information source, utilising the ontology. The DKM translates the request into two database questions and retrieves the information. The results for Length and Wingspan are then sent to the DM, Figure 6.16.

```
<O1, BirdSpecies, Gadwall>
<A1, Length; O1:V1>
<V1, String, "56 cm">
<A2, Wingspan; O1:V2>
<V2, String,  NULL>
```

Figure 6.16: An answer with values for length and wingspan for Gadwall, sent by the DKM to the DM.

6.5 Evaluation of BirdQuest

An evaluation of the system was performed with the goal of detecting problems concerning interpretation, dialogue management, and representation and use of domain knowledge.

6.5.1 Assessment of the question analysis approach

As the question analysis module was not integrated in the BirdQuest system, a preliminary assessment of the approach was done manually. The processing steps were applied to a sample of representative questions drawn from the question corpora. During the analysis of the sample of representative questions, three different types of problematic questions that were dealt with in a varying degree of success were identified:

Ambiguous general words

One problem encountered was the interpretation of general words that, in some questions, have very domain-specific meanings. For example, in the question, "Is the Oystercatcher really a type of Crow", 'is' denotes the relation IsA, see Figure 6.17.

```
<O1, BirdSpecies, Oystercathcer>
<R1, IsA; O1:O2>
<O2, Family, Crow>
```

Figure 6.17: The interpretation of "Is the Oystercatcher really a type of Crow".

However, in the question, "Which bird is widest between the tip of the wings" this interpretation of 'is' adds a second interpretation of the question that is not the intended. The correct interpretation gets a higher rank since it incorporates more of the information from the syntactic analysis, as shown in Figure 6.18. The parenthesis indicated that the faulty relation R1, has not been included in the interpretation.

```
<O1, BirdSpecies, EXIST>
<A1, Wingspan; O1:V1>
<V1, Number, MAX>
(<R1, IsA>)
```

Figure 6.18: The interpretation of "Which bird is widest between the tip of the wings".

Ambiguous domain words

A related problem is words that can have several interpretations, for example 'live' that can mean a location where birds live, or the time span of a bird's life, as exemplified in the questions presented in Figures 6.8 and 6.9. Another example is 'breed' that can be used together with 'where' to denote a location and together with 'how many' to refer to the number of couples in a population of birds. For

example, "Where do swans breed?" and "How many breeding swans are there in Sweden?" With the use of knowledge from the ontology regarding the domain and range restrictions of these, disambiguation is possible and the correct interpretation is produced in all cases.

Temporal and spatial restrictions

The use of the relations TemporalRestrict and SpatialRestrict to attach temporal objects and spatial objects to the proper object, property or relation works well. However, for some questions, further processing might be needed to derive a more precise meaning. For example, the question "What birds can I see in Sweden in December?" in Figure 6.10.

```
<O1, BirdSpecies: ALL>
<R1, WinterDistribution; O1:O2>
<O2, Country: Sweden>
```

Figure 6.19: The interpretation of "Which birds can I see in Sweden in December?".

In this case, the temporal restriction means that the relation Distribution with a TemporalRestrict should be transformed to the relation WinterDistribution, resulting in the interpretation shown in Figure 6.19

6.5.2 Assessment of dialogue and domain knowledge management

BirdQuest is intended to be used by casual users without previous experience of dialogue systems or extensive knowledge of birds. It was therefore evaluated in a walk-up and use situation similar to a real use situation during an Open House day at the University. In this respect, the situation resembles that of (Gustafson and Bell 2000), although slightly more controlled.

Table 6.1: User utterances

	No of utterances	Percentage of user utterances
Interpretable Requests	189	37%
Cooperative CR Responses	55	11%
Uncooperative CR responses	11	2%
Out of scope	121	23%
Mis-interpreted	141	27%

We had six work stations running BirdQuest during 2 hours and 30 minutes and collected dialogues from 27 users. They received minimal instructions in advance, they were only told that the system can answer questions on Nordic birds, that it understands Swedish, and that the dialogue would be recorded.

The resulting corpus consisting of 27 dialogues has a total number of 518 user utterances, with a mean of 19 for each user. However, with individual differences, for instance, three users posing more than 40 utterances to the system and three users posing less than 5.

Personal data about age, gender, interest in birds and knowledge of birds were collected together with each dialogue. The users were of varying age, 5 female and 22 male. Most of them had no interest in birds, nor any knowledge of birds. Thus, despite having no interest in birds, they were fairly representative of the intended users. Besides the logged dialogue, the users were also asked to fill out a brief questionnaire on how they liked to use the system. Most users thought the system was fun to use, on a 10-graded scale we had a mean of 7.1. The users also thought that it was fairly simple to use BirdQuest, mean 6.1. On the question how they liked the system we had a score of 4.7, i.e. the users neither disliked nor liked BirdQuest.

Table 6.2: System utterances

	No of utterances	Percentage of system utterances
Successful resp.	180	35%
Clarification req.	70	13%
Incorrect resp.	15	3%
Incorrect focus	16	3%
Error message	240	46%

As we had no predefined tasks, we did not have a situation that allowed for a controlled evaluation, as e.g. PARADISE (Walker et al. 1998) or PROMISE (Beringer et al. 2002). Instead, we used a combination of quantitative and qualitative approaches to analyse the collected dialogue corpus. The dialogues were tagged in order to provide statistics over successful and problematic information exchanges.

Table 6.1 shows that approximately half of the users' utterances (48%) were involved in successful information exchanges where the user initiated an information request or answered a clarification request from the system. We also see that 25% of the users' utterances were erroneous in some way and that BIRDQUEST failed to interpret 27% of the utterances, as will be further discussed in Section6.6.

From Table 6.2 we can see that BIRDQUEST presented 180 successful responses. A successful response is a response in which BIRDQUEST presents information found in the database. A response where the bird encyclopaedia does not include the information and BIRDQUEST responds e.g. "Information on wing span is missing for magpie." is also considered successful. The reason being that BIRDQUEST successfully accessed the database and presented whatever information was there, including cases where there was no information in the database. Among the 180, there are 55 such responses, so they are not rare, and show one of the many interesting problems we encountered in the development of a dialogue system based on information extraction from a text book.

Clarifications

It is notable that a fair amount of the dialogue moves involve clarifications. The system initiates 70 clarification sub-dialogues in order to transform a vague information request to a specific question, as exemplified by the excerpt U13-S18 in Figure 6.2.

The first type of clarification where the object needs to be specialised was initiated in 31 of 180 user requests for information. The second type regarding vague user properties occurred in 28 instances. Finally, mapping vague properties to ones suitable for database access was done in 64 cases. There were also 5 cases in which the user requested information about birds of the categories 'Small birds', 'Migratory birds' and 'Sedentary birds' which had not been implemented and could therefore not be handled. The high number of clarifications and property mapping for database access shows the usefulness of the domain ontology.

S22: You can receive information about size and plumage of a Blue Tit. Please specify what you are interested in.
U23: blue tit

Figure 6.20: An example of an uncooperative answer to a clarification request.

The users responded cooperatively to 55 clarification requests from the system and incorrectly 11 times. A typical example of the latter is seen in Figure 6.20.

Dialogue management, such as clarification sub-dialogues, thus plays an important role for the performance of BirdQuest. Contextual interpretation and dialogue history management are other important dialogue phenomena from MALIN that are frequently utilised in the dialogues. Managing dialogue history is, however, not trivial. There are 16 cases in the corpus, termed Incorrect focus in Table 6.2, when BirdQuest presents doubtful responses because of how dialogue history is handled, as will be further discussed in Section6.6.1.

Utterances out of scope for BirdQuest

Approximately half of the non-successful user utterances (23% of all user utterances) were questions that BirdQuest would never be able to answer. Beringer et al. (2002) use the term uncooperative user for users who, "fall out of the role or purposely misuse the system.", and propose to exclude them from evaluations. We include such users in our corpus, but group them together in a wider category called, Out of scope.

Out of Scope utterances include user requests for information that is outside the scope of the application, such as "How do you kill crows?", or socialisation utterances (Gustafson and Bell 2000), such as, "How are you?". Utterances can also be out of the database' scope, e.g. "How high does a magpie fly?" is such an utterance since there is no information on how high birds fly in the Bird encyclopaedia. These type of requests are further discussed in Section6.6.5

The reason for grouping such utterances together is that BirdQuest can never present information on them. Instead, we need to add a number of well-designed responses that inform the user of the system's abilities. Utterances that are out of the system's scope require different types of responses from the system, and the corpus gave us valuable insights into the importance of system help messages that describe what BirdQuest can and cannot do.

Utterances where BirdQuest fails

Finally, there are those utterances where the system failed, i.e. those where an answer can be found in the encyclopaedia, but in which BirdQuest fails to present a successful response for various reasons. Such utterances comprise 27% of the users' input.

We have further analysed these and categorised them as being 1) spelling mistakes, 2) lexical gaps or 3) grammatically out of scope,

Table 6.3: User utterances not interpreted by BirdQuest

	No of utterances	Percentage of system utterances
Misspelled	28	5%
Not in lexicon	64	12%
Not in grammar	50	10%

as seen in Table 6.3. Table 6.3 includes only utterances that can be successfully responded to, not, for instance, misspellings in utterances that are out of the system's scope.

Table 6.3 only gives a very brief indication of the nature of non-interpretable utterances in the corpus. For instance, each utterance is tagged as being of one type only, with misspellings having the highest priority and missing grammar rules the lowest. Furthermore, there could be several misspellings in one utterance.

It is also the case that the categories overlap, i.e. utterances can belong to more than one category, e.g. a misspelled word can also be missed in the lexicon. There are three such utterances tagged misspelled that also contain words not in the lexicon. 11 of the utterances tagged misspelled cannot be handled because of missing grammar rules. Thus, with a spelling checker, BirdQuest can handle another 14 utterances, i.e. misspelling on its own only accounts for 2.5% of the utterances that BirdQuest cannot handle.

6.6 Implications for development

Our main goal was to collect data for further development. Utterances in the categories involving unsuccessful information exchanges, such as incorrect responses and mis-interpretations, were therefore analysed more closely.

6.6.1 Problematic focus management

Focus management and methods for determining topic shifts play an important role in many dialogue systems. Our corpus includes a variety of problematic cases that need further consideration, both in how focus is handled in the Dialogue Manager and how the grammar rules are constructed.

When a user poses a question that gives rise to a partially interpreted request the Dialogue Manager seeks the dialogue history to see if it can be contextually specified. This works well in most cases when utterances are elliptic but in some cases, especially when the interpreter has failed to produce a complete interpretation, it can cause faulty responses, see the example in Figure 6.21.

U35: How old does a common gull get?
S36: Information about the greatest age of a Common Gull:
Greatest age: 29 years and 7 months
U37: What migratory birds exist?
S38: Information about range is missing for Common Gull.

Figure 6.21: An example of problematic focus management due to partial interpretations.

The reason for not being able to correctly interpret U37 is that migratory birds is not a well defined concept in the bird encyclopaedia [2]. Thus, only a partial interpretation with the property Range, based on interpreting 'exist' as connected to a location, is produced. The partial interpretation is then combined with the object Common gull from U35 and an incorrect response is produced in S38.

A more difficult problem to handle is when there has been a long segment of uninterpreted user utterances. These are often caused by sequences of misspellings or requests outside the scope of the system. Normally, such sequences of utterances are contextually interpreted

[2] Many bird species do not migrate every year, and often some exemplars stay when most birds of that specie migrate etc. (cf. Section6.6.4).

utilising the dialogue history and a successful response can be presented to the user. There are, however, also "faulty" cases, as seen in the dialogue fragment in Figure 6.22[3]. Here, the user tries to shift focus and thus the property Feed should not have been inherited from the previous question (U11). A more appropriate response in S18 would have been "You can receive information about a song thrush's flight, feed, breeding location, appearance,... Please specify what you are interested in.".

U11: What does the honey buzzard eat
S12: Information about feed for a Honey Buzzard:
Main feed consists of Honeycomb,[...]
U13: What birds can imitate speech
S14: Failed to interpret the question. Please try again
U15: Talking birds
S16: Failed to interpret the question. Please try again
U17: Song thrush
S18: Information about feed for a Song Thrush:
Worms, larvas and mollusc, etc[...]

Figure 6.22: An example of problematic focus management due to undetected focus shift.

The problem of dialogue history and focus shifts does not have a simple solution. Leceuche et al. (2000) use focus rules based on relations between the various things in the domain to track focus shifts. A similar approach is to encode the domain in a type hierarchy and use under-specified feature structures to determine clarification requests (Denecke 1997). Jokinen et al. (1998) use a topic model based on a tree organisation of domain knowledge to detect topic shifts. Such approaches would correctly handle the dialogue above and start a new focus space after U13, as there is no relation between the items in U11-S12 and U13. However, as we assume that it is a good idea to keep items in focus even after user utterances with no previous relation to the current focus (Jönsson 1995), such mechanisms are not applicable here. For instance, if the user typed instead "U13b:

[3]The Swedish name of Song thrush, (swe. taltrast), can be literally translated as "Talking thrush"

magpi", we would not have a relation between the items in the utterances, and then will not be able to correctly respond if the user corrected that to "U15b: magpie".

Another approach is proposed by (Chu-Carroll 2000) who distinguishes between task initiatives and discourse initiatives that are identified based on cues. These are then utilised to adopt different interaction strategies. In BIRDQUEST, one solution would be to maintain focus for discourse initiatives and drop focus for new task initiatives. In the dialogue in Figure 6.22, U13 is to be considered a new task initiative, whereas U13b is not a new task initiative and focus is maintained.

6.6.2 Unnecessary clarifications

Since information extraction is used to populate the database with information there can be holes in it, i.e. for some properties values are missing due to shortcomings in the information extraction component or lack of information in the original text source. This causes unnecessary clarifications. In the example in Figure 6.23 a more appropriate answer to U19 would have been to give information about size instead of making a clarification since there is no information about plumage in the database.

U19: What does a mute swan look like?
S20: You can receive information about the size and plumage of a Mute Swan. Please specify what you are interested in.
U21: What does the plumage of a mute swan look like?
S22: Information about eclipse plumage is missing for Mute Swan.
Information about winter plumage is missing for Mute Swan.
Information about summer plumage is missing for Mute Swan.
Information about breeding plumage is missing for Mute Swan.

Figure 6.23: An example of unnecessary clarification.

The database has many such missing values as there often was no information in the information source. It is, thus, not possible to adopt general dialogue strategies, for instance, for clarification requests. Instead, the domain knowledge base must be consulted to decide on a proper dialogue strategy for each property.

In BirdQuest, the unnecessary clarifications can be dealt with through extended cooperation between the Dialogue Manager and the Domain Knowledge Manager. When a vague property is encountered, the Dialogue Manager can send the request to the Domain Knowledge Manager for information on suitable clarifications. By traversal of the ontology, the DKM can produce a set of sub-properties that can be used to access the database. Should there be only a few pieces of information these can be returned directly as an answer to the request. Otherwise, the proper clarification is decided based on the relation between the given property and the sub-properties that produced the values.

Note, however, that such a strategy can violate the learning aspect discussed below.

6.6.3 Partial and empty answers

A problem related to unnecessary clarifications is how partial and empty answers should be presented to the user when a vague property has been mapped to several sub-properties, for example, Plumage in S22 in the example above, or Size in the example in Figure 6.24.

S2: You can receive information about the size and plumage of a Magpie. Please specify what you are interested in.
U3: size
S4: Information about wingspan is missing for Magpie.
A Magpie has a length of 46 cm from beak tip to tail.

Figure 6.24: An example of partial answers.

In the case of empty answers, S22 above, an alternative could be to just state that "Information about plumage is missing for mute swan", and for partial answers the sub-properties with missing information could be omitted. However, including all the sub-properties has a point in so far as it helps the user learn what type of information the system can provide. A solution is to compromise and make the system adaptive. This means that the first time a vague property is mapped to a set of sub-properties, all of them are presented, thereafter empty answers can be omitted.

6.6.4 Ontological interpretation failures

Some seemingly simple interpretation failures that have been tagged as missing information in the lexicon are actually more complicated and relate to the representation of domain knowledge in the ontology. For example, users tend to ask what colour a specific bird is, see U1 in Figure 6.25[4]. However, in the bird encyclopaedia colour is linked to the body parts of a bird. The complex relation between the concept Bird and the property Colour via the body parts thus has to be captured in the ontology so that this knowledge can be used for interpretation and dialogue management.

U1: What colour is a blue tit?
S2: Failed to interpret the question. Please try again

Figure 6.25: An example of ontological interpretation failure.

With more information about how entity types and properties are related, for example, through a chain of hypernym and meronym relations, requests like U1 can be dealt with. Since a bird consists of body parts and body parts have colours the missing link in the request is a body part and based on this a clarification can be produced "A

[4]Many of the requests for appearance can be handled by presenting a picture of the bird. However, the pictures in our bird encyclopaedia are copyrighted and can therefore not be presented.

bird has several colours for different body parts. What part of the blue tit are you interested in?"

6.6.5 Questions outside database coverage

Users at the Open House often posed questions that were outside the database coverage. A successful interpretation of such a request and an appropriate response is seen in the dialogue in Figure 6.26.

U1: What is the wingspan of an Albatross?
S2: There is no information on non-Nordic birds. The available birds are listed in the menu "Birds a-ö". Please try again

Figure 6.26: An example of questions outside database coverage.

The same strategy can be used to handle similar questions concerning non-Nordic birds like canaries and ostriches. However, this would require the addition of information of non-Nordic birds to the ontology collected from other sources than the bird encyclopedia.

Nevertheless, in most cases the requests concerned properties that are not covered by the database but which users often ask about, for example Weight and Flight-speed as in "How much does a Sea Gull weigh?" or "How fast can an Eagle fly?". The common response to these types of questions were "Failed to interpret the question. Please try again." or, in some cases, a partial interpretation was made which led to inappropriate responses. A more desirable response would be to give more informative error messages and explain to the user that it cannot answer questions about these topics.

The properties similar to those in the database, such as Weight, Flight-speed, would be added to the ontology as user-oriented properties. Since the DKM always has to map these types of properties to the system-oriented sub-properties before database access, the DKM could conclude that, if a user-oriented property does not have any user-oriented sub-properties, it is outside database coverage and an

appropriate answer can be given. If these properties are related to others, for example, Weight is a sub-property of Appearance, the system can then even suggest some of the sibling properties, in this case, Size and Plumage.

Another strategy is to have BIRDQUEST respond with help phrases explaining how to pose valid requests, as is done in Targeted Help (Gorrell et al. 2002). Targeted help is used for improving user behaviour in speech interfaces. It utilises the SLM-based recognition and categorised help message templates to present targeted help when the grammar-based recogniser fails. Thus, a system must learn the most common types of mistakes, which in turn must be classified to provide a targeted help. Unfortunately, we do not yet have a large enough BIRDQUEST corpus for such classification.

6.7 Implications for ontology use in dialogue systems

The development of BIRDQUEST and a question analysis module show the potential of domain ontologies as domain knowledge sources that support question analysis, dialogue management and domain knowledge management. The evaluation illustrates how useful ontological knowledge can be for disambiguation of questions, clarification initiatives, and database access. It also points out other potential usages for focus management, more intelligent clarifications and helpful error messages for questions outside the scope of the system.

These features will be developed and generalised to a framework for ontology usage in dialogue systems, presented in Chapter 7.

Chapter 7

A Framework for Use of Ontologies in Dialogue Systems

In this chapter, a framework for the use of ontologies in information-providing dialogue systems is presented.

A framework for use of ontologies in dialogue systems, consisting of an ontology specification and algorithms for various functionalities that can be provided by ontologies in information-providing dialogue systems is presented in this chapter. The framework includes models for question analysis, models for dialogue management tasks and models for domain knowledge management tasks.

The framework has been developed based on experiences with the development and evaluation of BirdQuest, see Chapter 6, and an ontology designed for usage in dialogue systems, AnOntology, that is the realisation of the design presented in Chapter 4.

7.1 The AnOntology specification

The AnOntology specification is intended to be used for the creation of mixed ontologies that are natural language-oriented and provides semantic information for lexicons, as well as domain representation and reasoning in information-providing dialogue systems. It includes all features necessary to support a variety of functionalities in dialogue systems.

7.1.1 Design decisions

AnOntology includes the following design decisions, which are based on the analysis presented in Chapter 4:

Concepts Atomic concepts are chosen over concepts with internal structure since they provide sufficient functionality and are less complex to handle.

Attributes Local, instance, inherited and polymorphic attributes are the basis since these are necessary in dialogue systems. Own and class attributes that are useful in some domains, can be specified by inference rules if desired.

Facets of attributes Since type constraints are necessary, and default values and cardinality restrictions can be useful for many tasks, these are supported. Operational definitions of values are not included, but the framework should be possible to extend with such definitions later.

Relations Binary relations are sufficient for the needs of information-providing dialogue systems and are therefore chosen over n-ary relations. Since atomic concepts are used, domain and range restrictions that link the relations to the entity types are used.

Time and space Time and space are recommended to be treated separately in their own ontologies and with inference rules that can be used together with other ontologies.

Taxonomy The taxonomy is expressed by axioms that link concepts together with taxonomic relations. It is possible to have single concepts, several distributed taxonomies, as well as one common taxonomy, for instance a tree. There is no specific ontological back-bone with top-level distinctions but the modular architecture allows several different ontologies to be combined, and thereby a backbone can be designed and reused (an example of a meta-ontology is provided).

Inheritance Inheritance is specified in definitions, which allow for specification of non-monotonic and multiple inheritance. The meta-ontology described in Section7.1.3 provides definitions for simple monotonic inheritance.

Axioms No axioms are used since sufficient knowledge is provided through the taxonomy.

Instances The type of instances allowed are individuals and facts, which are represented in the KB.

Many of the guidelines presented in Section2.2.2 are aimed at the knowledge engineering work during ontology capture and coding, but some of them are related to the design of the framework. Guideline DG1 says that definitions should be based on identity criterions. This is not directly implemented in AnOntology since definitions are strings of text. This means that natural language descriptions can be used, but more formal definitions are of course also allowed. Guidelines DG4 and DG6 talk about different kinds of entities and roles/attributes. This is implemented in AnOntology through a separation of entity types, properties and lexical items. For guideline DG10 most issues concerning naming concepts are left to the knowledge engineer but polysemy is supported, while delimiters are not allowed, only Alfa-numeric characters can be used. The guidelines concerning the taxonomic organisation, DG11 and DG12, are also considered, which means that entity types and properties are organised in trees with disjoint categories, and multiple inheritance is allowed. Guidelines for extendibility and maintenance, DG14 and DG15, are also considered, and extendibility and modularisation are supported by AnOntology.

7.1.2 Formal specification

A formal specification of AnOntology has been done and is used for the algorithmic description of ontology usage in dialogue systems presented in the following sections. The specification includes three parts; the ontology specification that is general for a domain, the knowledge base that is application-specific, and the lexicon that is language- and application-specific.

A taxonomy is a structure $\langle C, \{R\}_i \rangle$, where:

- C is a set of concepts, the universe of discourse, that names the entities of a domain
- $\{R\}_i$ is a finite family of relations where each $R_i \subseteq C \times C$ is a relation expressing a partial ordering of the ontological entities

Let $\langle C, \{R\}_i \rangle$ be a taxonomy.

For every $c_1 \in C$:

- $GetSuperConcepts_{R_i}(c_1) := \{\ c_2 \mid \exists\ \langle c_1, c_2 \rangle \in R_i\ \}$
- $GetSubConcepts_{R_i}(c_1) := \{\ c_2 \mid \exists\ \langle c_2, c_1 \rangle \in R_i\ \}$

Definition 1: *An ontology* is a structure $O := \langle C, T_e, T_p, S, A, L, V \rangle$, where:

- $C = E \cup P$ is the universe of discourse, which is the union of the two disjoint sets, entity types and properties
- T_e is a taxonomy of entity types $\langle E, \{R\}_i \rangle$
- T_p is a taxonomy of properties $\langle P, \{R\}_i \rangle$, where $P = P_r \cup P_a$, with two disjoint subsets, relations and attributes

- Let Values be a set of primitive datatypes such as integers, strings, and booleans, then
  - $S = S_r \cup S_a$ where
  - $S_r : P_r \rightarrow E \times E$ is a signature function that expresses domain and range restrictions for relations
  - $S_a : P_a \rightarrow E \times Values$ is a function that expresses domain and range restrictions for attributes
- $A : P \rightarrow Pos \times Pos$ is a function that expresses the signature arity (cardinality) of a property's range, where Pos is the set of positive integers
- L is a set of labels that denote the various origins of elements in C, for example, from the users or the system
- $V : C \rightarrow L$ is a function that maps a concept to a label that expresses its origin

Operators on ontologies

Let $\langle C, T_e, T_p, S, A, L, V \rangle$ be an ontology and π_i a projection function that maps a tuple on its ith element.

Let R_i be a relation in $T_e = \langle E, \{R\}_i \rangle$. For every $e \in E$:

- $GetSuperEntitytypes_{R_i}(e) := GetSuperConcepts_{R_i}(e)$
- $GetSubEntitytypes_{R_i}(e) := GetSuperConcepts_{R_i}(e)$

Let R_i be a relation in $T_p = \langle P, \{R\}_i \rangle$. For every $p \in P$:

- $GetSuperProperties_{R_i}(p) := GetSuperConcepts_{R_i}(p)$
- $GetSubProperties_{R_i}(p) := GetSubConcepts_{R_i}(p)$
- $GetPropertyDomain(p) := \pi_1(S(p))$

- $GetPropertyRange(p) := \pi_2(S(p))$
- $GetMinCardinality(p) := \pi_1(A(p))$
- $GetMaxCardinality(p) := \pi_2(A(p))$

For every $c \in C$:

- $GetVocabulary(c) := V(c)$
- $GetPropertiesForDomain(c) :=$
  $\{\ p \in P \mid GetPropertyDomain(p) = c\ \}$
- $GetRelationsForRange(c) :=$
  $\{\ p \in P \mid GetPropertyRange(p) = c\ \}$

For every $c_1,\ c_2 \in C$:

- $GetRelations(c_1, c_2) := \{\ p \mid \exists\ \langle c_1, c_2\rangle \in S(p)\ \}$

Definition 2 *A Knowledge Base* w.r.t. O is a structure $\langle I, \tilde{E}, F\rangle$, where:

- I is a set called individuals (instances of entity types)
- $\tilde{E} \subseteq I \times E$ is a relation expressing mapping of individuals to entity types
- F is a set of facts (instances of properties) expressed as tuples
  - $\langle p, e_1, e_2\rangle$ where $\langle e_1, e_2\rangle \in S(p)$
  - $\langle p, e, v\rangle$ where $\langle e, v\rangle \in S(p)$
  - $\langle p, i_1, i_2\rangle$ where $\exists\langle e_1, e_2\rangle \in S(p)$:
    $\langle i_1, e_1\rangle \in \tilde{E} \wedge \langle i_2, e_2\rangle \in \tilde{E}$
  - $\langle p, i, v\rangle$ where $\exists\langle e, v\rangle \in S(p) : \langle i, e\rangle \in \tilde{E}$

Operators on knowledge bases

Let $\langle C, T_e, T_p, S, A, L, V \rangle$ be an ontology where $T_e = \langle E, \{R\}_i \rangle$ and $T_p = \langle P, \{R\}_i \rangle$, and $\langle I, \tilde{E}, F \rangle$ a knowledge base.

For every $i \in I$:

- $GetConceptsOf(i) := \{\ e \mid \exists\ \langle i, e \rangle \in \tilde{E}\ \}$

For every $e \in E$:

- $GetInstances(e) := \{\ i \mid \exists\ \langle i, e \rangle \in \tilde{E}\ \}$

For every $e \in E$ and $p \in P_a$:

- $GetDefaultValue(e, p) := \{\ v \mid \exists\ \langle p, e, v \rangle \in F\ \}$

Definition 3 *A Lexicon* w.r.t. O is a structure $\langle I, \tilde{C} \rangle$, where:

- I is a set whose elements are called lexical items
- $\tilde{C} \subseteq S \times C$ is a relation called lexical reference assignments

Operators on lexicons

Let $\langle C, T_e, T_p, S, A, L, V \rangle$ be an ontology and $\langle I, \tilde{C} \rangle$ a lexicon.

For every $i \in I$:

- $GetConcepts(i) := \{\ c \mid \exists\ \langle i, c \rangle \in \tilde{C}\ \}$

For every $c \in C$:

- $GetLexicalItems(c) := \{\ i \mid \exists\ \langle i, c\rangle \in \tilde{C}\ \}$

7.1.3 Meta-ontology

A complement to the AnOntology framework is a meta-ontology that extends AnOntology with the most basic distinctions needed in information-providing dialogue systems, such as various taxonomic relations.

Taxonomy

The AnOntology framework differentiates between entity types and properties, where properties are further divided into attributes and relations. The meta-ontology extends these with distinctions between objects and events for entity types and defines a number of taxonomic relations. These relations are based on traditional hyponymic relations and some meronymic relations. The concepts are named using only alfanumeric characters and in singular form. The taxonomy is one common tree with no multiple inheritance, illustrated in Figure 7.1.

Inheritance

Inheritance is specified by recursive definitions similar to the operators in Section 7.1.2. To separate information held in the ontology from that derived, new operators are introduced, for example $IsDomain(p_1, e_1)$ corresponds to $GetPropertyDomain(p_1) = e_1$.

Since properties should be placed at the most general level (DG7), it most be possible to infer that they are applicable to sub-concepts, i.e.

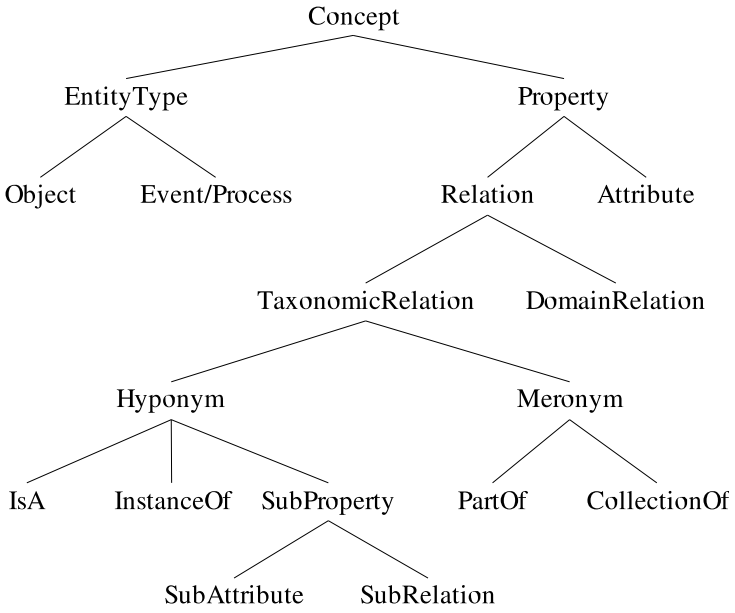

Figure 7.1: The taxonomy of the meta-ontology for AnOntology.

that properties are inherited downwards in the entity type taxonomy. Properties should also be propagated to some type of parts from a whole, for example, to a member from a collection.

Let $\langle C, T_e, T_p, S, A, L, V \rangle$ be an ontology where $T_e = \langle E, \{R\}_i \rangle$ and $T_p = \langle P, \{R\}_i \rangle$

For every $p \in P$ and $e_1, e_2 \in E$:

$IsDomain(p, e_1)$ *is true if and only if*
$(GetPropertyDomain(p) = e_1)$ *or*
$\exists \langle e_1, e_2 \rangle \in IsA : GetPropertyDomain(p) = e_2$

$IsRange(p, e_1)$ *is true if and only if*
$(GetPropertyRange(p) = e_1)$ *or*
$\exists \langle e_1, e_2 \rangle \in IsA : GetPropertyRange(p) = e_2$

Values for attributes can also be inherited through similar specifications. These are especially useful if default values are used.

Let $\langle C, T_e, T_p, S, A, L, V \rangle$ be an ontology where $T_e = \langle E, \{R\}_i \rangle$ and $T_p = \langle P, \{R\}_i \rangle$ and $\langle I, \tilde{E}, F \rangle$ a knowledge base.

For every $p \in P$, $e_1, e_2 \in E$ and $v \in V$:

$IsDefaultValue(e_1, p, v)$ *is true if and only if*
$(GetDefaultValue(e_1, p) = v)$ *or*
$\exists \langle e_1, e_2 \rangle \in IsA : GetDefaultValue(e_2, p) = v$

7.2 Ontology use in question analysis

Based on the AnOntology framework, robust question analysis can be achieved. Robust techniques for syntactic partial parsing can be combined with ontological analysis to create semantically consistent interpretations of utterances in dialogue systems. The main functionalities supported by the use of an ontology during interpretation are the ability to disambiguate senses of words and phrases, i.e. to find the intended ontological concept, and the ability to relate properties to objects and events.

The syntactic analysis of user utterances results in a number of lexical items, e.g. words and phrases, and links between these. A semantic lexicon, described above, is accessed to map the lexical items to ontological entities. Since lexical items can be ambiguous, i.e. they are mapped to a set of concepts, these need to be disambiguated, i.e. one concept has to be chosen over the others. This is done in two steps, where properties and entity types are treated separately.

7.2.1 Disambiguation of properties

The semantic interpretation of an utterance results in four sets of concepts, a set of unambiguous properties, P, a set of ambiguous properties that can be further divided into subsets with the alternatives for each ambiguous lexical item, AP = {AP1, AP2,...,APn}, a set of unambiguous entity types and values, E, and a set of ambiguous entity types that can be further divided into subsets with the alternatives for each ambiguous lexical item, AE = {AE1, AE2,...,AEn}. For disambiguation of properties E and AE are joined to form one set of entity types.

Ambiguous interpretations of properties are resolved by the iterative application of an algorithm to the subsets APi in AP, described in Figure 7.2.

Domain and range restrictions are used together with the links produced by the syntactic analysis, to test which of the alternatives of an ambiguous property that can be related to the entity types provided. Thus, the alternative that is most semantically coherent with the rest of the utterance will be chosen.

The algorithm consists of several levels where restrictions are relaxed for each level. The results are ranked corresponding to their motivation in terms of present information, and the first receives the highest rank. Thus, the alternative with the highest rank is the most semantically consistent with the rest of the utterance and should be chosen over the others. The algorithm returns a list with all the results, if however, several alternatives of the same rank are produced (`ExistSeverlOfHighestRank`), a clarification is returned instead.

```
DisambiguateProperty(APi: Set of ambiguous properties,
                     E: Set of entity types, L: Set of links)

for all p in APi do
  for all e1 in E do
    for all e2 in E do
    if (AreRelated(p,e1,e2) and (e1 != e2) and
        ((p,e1) in L) and ((p,e2) in L))
    then add p with rank 1 to Result

    else if (AreRelated(p,e1,e2) and (e1 != e2) and
             ((p,e1) in L))
         then add p with rank 2 to Result

    else if (AreRelated(p,e1,e2))
         then add p with rank 3 to Result

    else if (IsDomain(p,e1))
         then add p with rank 4 to Result

if (ExistSeveralOfHighestRank(Result))
then return clarification(APi, AmbiguousProperty, Result)
else return Result


areRelated(p: Property, e1: Entity Type, e2: Entity Type)

if ((IsDomain(p,e1) and IsRange(p,e2)) or
    (IsRange(p,e1) and IsDomain(p,e2)))
then return TRUE
else return FALSE
```

Figure 7.2: The algorithm for ontological disambiguation of properties.

For example, as mentioned before, 'live' is an ambiguous lexical item that can be mapped to both the attribute OldestAge and the relation Distribution. For the utterance "Where do mute swans live?" the syntactic analysis produces the following sets of concepts and links:

```
E:  {<O1, SpatialObject, EXIST>,
     <O2, BirdSpecies, MuteSwans>}
AP: {{<P1, OldestAge; ?:?>, <P1, Distribution; ?:?>}}
P:  {}

Links: {{P1, O1}, {P2, O1}}
```

The properties derived from 'live' is disambiguated by the algorithm. Since `AreRelated(Distribution, BirdSpecies, SpatialObject)` and the property and the entity types are linked, Distribution is added to the result with rank 1. Since `IsDomain(OldestAge, BirdSpecies)` and the property is linked to the entity type, OldestAge is also added to the result with rank 4. Thus, Distribution will be chosen over OldestAge since it has a higher rank.

7.2.2 Disambiguation of entity types

Entity types are disambiguated in a way very similar to properties. This time the algorithm is iteratively applied to the subsets AEi in AE, described in Figure 7.3. P is extended with the results from the disambiguation of AP. Domain and range restrictions are used in a similar way to find out what entity type that corresponds to the given properties. The algorithm is almost identical but includes one extra level. If there is no provided property that links any of the entity types, but there exists such a property in the ontology, a semantic coupling between the ambiguous entity type and another entity type has been found and the ambiguous entity type is added to the result.

```
DisambiguateEntitytype(P: Set of properties,
                       AEi: Set of ambiguous entity types,
                       E: Complement set of AE, L: Set of links)

for all e1 in AE do
  for all e2 in E do
    for all p in P do
    if (AreRelated(p,e2) and ((p,e1) in L) and ((p,e2) in L))
    then add e1 with rank 1 to Result

    else if (AreRelated(p,e1,e2) and ((p,e1) in L))
         then add e1 with rank 2 to Result

    else if (AreRelated(p,e1,e2))
         then add e1 with rank 3 to Result

    else if (IsDomain(p,e1))
         then add e1 with rank 4 to Result

    else if (GetRelations(e1,e2) != NULL)
         then add e1 with rank 5 to Result

if (ExistSeveralOfHighestRank(Result))
then return clarification(AEi, AmbiguousEntitytype, Result)
else return Result
```

Figure 7.3: The algorithm for ontological disambiguation of entity types.

Names of entities can be ambiguous, for example, in the SVT Text multi-domain 'Linköping' can refer to both a city and a hockey team. The question "How did Linköping do in the hockey game?" will result in the following semantic interpretation:

```
P:  {<A1, Result; ?:?>}
AE: {<O1, City, Linköping>,
     <O1, HockeyTeam, LHC>}
E:  {<O3, HockeyGame, EXIST>}

Links: {{O1,A1}{O3,A1}}
```

Since `GetRelations(HockeyGame, HockeyTeam)` results in the non-empty set {Participate}, LHC is added to Result with rank 5. This is the only result and thus it will be chosen, and the ambiguity is resolved.

7.2.3 Relating entity types and properties

When the ambiguous entity types and properties have been resolved and two unambiguous sets of properties and entity types have been produced, it remains to relate these. An algorithm similar to those used for disambiguation is utilised for this purpose, see Figure 7.4.

If two entity types or one entity type and one value in the set E can be related to a property in P, i.e. domain and range restrictions are fulfilled, this relationship is added to the result. If only one entity type of the right domain for a property is available the missing entity or value type is collected from the ontology and the relationship added to the result. This is often the case when the user has asked a question, in this case the added information corresponds to the expected answer type.

The next step is to add properties that are indicated by the presence of two entity or value types. This happens when vague expression are used that cannot be mapped to a property during interpretation.

```
RelateEntitytypesAndProperties(P: Set of properties,
                               E: Set of entity types,
                               L: Set of links)
for all p in P do
  for all e1 in E do
    for all e2 in E do
    if (AreRelated(p,e1,e2) and (e1 != e2) and
        ((p,e1) in L) and ((p,e2) in L))
    then add (p;e1:e2) with rank 1 to Result

    else if (AreRelated(p,e1,e2) and (e1 != e2) and
             ((p,e1) in L))
         then add (p;e1:e2) with rank 2 to Result

    else if (AreRelated(p,e1,e2))
         then add (p;e1:e2) with rank 3 to Result

    else if (IsDomain(p,e1))
      then for all e3 in getRange(p,e1) do
           add (p;e1:e3) and (e3, exist) with rank 4  to Result

    else if (getProperties(e1,e2) != NULL)
         then for all p2 in getProperties(e1, e2) do
              add (p2;e1:e2) with rank 5  to Result

return Result
```

Figure 7.4: The algorithm for ontological relating of entity types and properties.

For example, the question "Are there any nature-programs or science-programs tonight", produces the structure:

```
<O1, Nature-program, EXIST>
<O2, Science-program, EXIST>
<O3, TimeIntervall, TONIGHT>
```

Two instances of the relation TemporalRestrict are inferred, since it has the domain TV-program and the range TemporalObject:

```
<R1, TemporalRestrict, O1:O2>
<R2, TemporalRestrict, O1:O3>
```

The number of levels in the algorithm that are used can be adapted to the context in which the analysis is done. For example, in dialogue systems the last step, where properties can be added, might be unnecessary since fragmentary input should be allowed. Another choice is whether to further process questions that cannot be fully interpreted, i.e. all properties and entity types cannot be related. In a dialogue system, other modules may be able to compensate for this, for example, a Dialogue Manager can ask a clarification question or a Domain Knowledge Manager can derive missing information.

7.3 Ontology use in dialogue management

For dialogue management, the ontology can be used for resolving anaphora, resolving ellipsis, contextual interpretations and clarifications.

The dialogue management models described here rely on a representation of the dialogue history, DH, that utilises the same type of content representation as presented in Section 6.3.1, i.e. entity types are described in tuples `<Id, Name, Instance/Quantifier>` and properties in tuples `<Id, Name; Arg1:Arg2>`. When used in the algorithms a property or an entity type usually refers to the name, otherwise the tuple is indicated explicitly. The dialogue history also includes information on syntactic and pragmatic features of user and system utterances. In the BIRDQUEST and the ÖTRAF systems, the dialogue histories were implemented as dialogue trees but other representations, such as stacks or lists, can also be used.

7.3.1 Resolving anaphora

The resolution of anaphora deals with expressions like he, she, it, they and them. Typically, a question contains one of these expressions and a property, for example, "What do they eat?" or "When does it start?". The resolution is done using a combination of syntactic and ontological information on domain and range restrictions to find a suitable referent from the dialogue history, described by the algorithm in Figure 7.5. The syntactic information is used to check for agreement (`CheckSyntacticFeatures`), for example, in number and gender.

The entity type referent typically only indicates the syntactic features, for example, that the number for 'it' is singular. The list of entity types retrieved from the dialogue history is ordered with the most recent first in the list, and it contains information on syntactic features as well as the ontological concept. If there are several entity types present in a turn they are grouped together as a set. Thus, the list of possible referents, E, from the dialogue history consists of sets of entity types.

The following dialogue illustrates the need for ontological knowledge to resolve anaphora:

U1: Where do mute swans live?
S2: ...
U3: How many eggs do they lay?
S4: ...
U5a: What do they eat?
U5b: What do they look like?

```
ResolveAnaphora(r: entity type reference, p: Property,
                E: List of sets of entity types from DH)

for all E1 in E do
  for all e in E1 do
    if ((CheckSyntacticFeatures(r, e)) and
        (IsDomain(p, e) or IsRange(p, e)))
    then add e to Result

  if (NumOfElements(Result) > 1)
  then return clarification(r, AmbiguousReferent, Result)

  else if (Result != NULL)
       then return Result

return error(r, UnresolvedAnaphora)
```

Figure 7.5: The algorithm for ontological resolution of anaphora.

As described above, the first utterance, U1, is disambiguated and results in the interpretation:

```
<O1, SpatialObject, EXIST>
<O2, BirdSpecies, Eagles>
<R1, Distribution; O1:O3>
```

When U3 is analysed, a partial interpretation is found:

```
<O1, Egg, EXIST>
<R1, LayEgg; ?:O1>
<A1, NumberOf; O1:V1>
<V1, Number, EXIST>
```

To resolve the referent 'they' the dialogue history is traversed. Since `IsDomain(LayEgg, BirdSpecies)` is true and the syntactic feature of 'they' and 'eagles' corresponds, Eagle is added as a referent.

If U5a is asked next the following partial interpretation is produced:

```
<A1, Appearance; O1:V1>
<V1, String, EXIST>
```

The entity types from the dialogue history are inspected to find a suitable referent to 'they'. Since both 'eagles' and 'eggs' have compatible syntactic features and fulfil the domain restrictions of Appearance both will be added as possible referents, and a clarification is posed, `clarification('they', AmbiguousReferent, {Eagle, Egg})`.

If U5b was posed instead, a unique referent could have been found. Since `IsDomain(Feed, BirdSpecies)` is true the complete interpretation after anaphora resolution would have been:

```
<A1, Feed; O1:V1>
<O1, BirdSpecies, Eagles>
<V1, String, EXIST>
```

This fully specified request can then be sent to the Domain Knowledge Manager for retrieval of an answer.

7.3.2 Resolving ellipsis

Elliptical utterances are common in dialogue systems. In such the user only provides partial information, for example, an entity type, a property or a value. For example the user can pose a question "Which swans live in Sweden? and then ask "crows" and mean "Which crows live in Sweden?". Such utterances can often be resolved from the previous discourse. Thus, for a property the entity type should be inherited from the discourse and vice versa. Depending on what has been provided by the user different algorithms are used.

```
ResolveEllipsis(e: Entity Type,
                P: List of sets of properties from DH)

for all P1 in P do
  for all (p;arg1:arg2) in P1 do
    if (IsDomain(p, e) and not IsRange(p, e))
    then add (p;e:arg2) to Result

    else if (IsRange(p, e) and not IsDomain(p, e))
    then add (p;arg1:e) to Result

    else if (IsRange(p, e) and IsDomain(p, e))
    then return clarification(p, AmbiguousDomainOrRange, e)

  if (NumOfElements(Result) > 1)
  then return clarification(e, AmbiguousProperty, Result)

  else if (Result != NULL)
       then return Result

return error(e, UnresolvedEllipsis)
```

Figure 7.6: The algorithm for ontological resolution of ellipsis where only an entity type has been provided.

For entity types, a property that has the entity type as either domain or range is sought. Similarly to resolution of anaphora, the list of properties from the dialogue history is an ordered list of sets. If several properties match the entity type a clarification is produced. This is also the case if a property is found for which the entity type fulfils both the domain or range restriction, as shown in Figures 7.6.

The algorithm can be illustrated by the following dialogue excerpt:

U1: What owls live in Sweden?
S2: ...
U3a: waterfowl
U3b: Norway

The first utterance, U1, results in:

```
<O1, Family, Owls>
<O2, Country, Sweden>
<R1, Distribution; O1:O2>
```

If U1 is followed by U3a the algorithm resolves it in the following way. Since `IsDomain(Distribution, BirdSpecies)` is true, Owls is replaced by Waterfowl and the property and second entity type, Sweden, is inherited, resulting in:

```
<O1, Family, Waterfowl>
<O2, Country, Sweden>
<R1, Distribution; O1:O2>
```

If U1 is followed by U3b, `IsRange(Distribution, Country)` would be fulfilled and the result would be:

```
<O2, Country, Norway>
<O1, Family, Owls>
<R1, Distribution; O1:O2>
```

Elliptical properties are more straightforward to deal with. The dialogue history is traversed to find an entity type that matches the domain of the property, see Figure 7.7.

```
ResolveEllipsis(p: Property,
                E: List of sets of entity types from DH)

for all E1 in E do
  for all e in E1 do
    if (IsDomain(p, e))
    then add e to Result

  if (NumOfElements(Result) > 1)
  then return clarification(p, AmbiguousEntitytype, Result)

  else if (Result != NULL)
       then return Result

return error(p, UnresolvedEllipsis)
```

Figure 7.7: The algorithm for ontological resolution of ellipsis where only a property has been provided.

The algorithm in Figure 7.7 can be illustrated by the following dialogue excerpt:

U1: What do crows eat?
S2: ...
U3: and look like?

Utterance U3 is interpreted as the attribute Appearance. Crows can be retrieved from the dialogue history and since `IsDomain(Appearance, BirdSpecies)`, `<O1, BirdSpecies, Crows>` is added to form a complete request that can be sent to the Domain Knowledge Manager for further processing.

When a value is provided, cardinality restrictions are used to decide if it should replace or be added to the old. If the maximum cardinality is 1 or if the maximum number of values are reached, then the value should be replaced, otherwise added, see Figure 7.8.

```
ResolveEllipsis(v1: Value,
                A: List of sets of attributes from DH)

for all A1 in A do
  for all (a;e:v) in A1 do
    if (IsRange(a, v1))
    then if (GetMaxCardinality(a) > NumOfElements(v))
         then add v1 to (a;e:v) and add to Result

         else if ((GetMaxCardinality(a) = 1) or
                  (GetMaxCardinality(a) = NumOfElements(v))
              then add (a;e:v1) to Result

  if (NumOfElements(Result) > 1)
  then return clarification(v1, AmbiguousAttribute, Result)

  else if (Result != NULL)
       then return Result

return error(v1, UnresolvedEllipsis)
```

Figure 7.8: The algorithm for ontological resolution of ellipsis where only a value has been provided.

The use of the algorithm is exemplified by:

U1: What birds have a wingspan of 50 cm?
S2: ...
U3: 1m?

The first request is interpreted as:

```
<O1, BirdSpecies, ALL>
<A1, Wingspan:O1:V1>
<V1, Measure, 50cm>
```

The interpretation of U3 produces a new value, `<V1, Measure, 100cm>`. This value replace the old since `IsRange(Wingspan, Measure)` is true and `GetMaxCardinality(Wingspan) = 1`:

```
<O1, BirdSpecies, ALL>
<A1, Wingspan:O1:V1>
<V1, Measure, 100cm>
```

7.3.3 Contextual interpretation

Contextual interpretation is similar to ellipsis resolution, it involves the integration of the new information provided by the user with old information from the discourse. This step relies on a task model that describes the entity types and properties that should be specified to form a complete request, for example, an ISF that specifies what properties are required, as described in Chapter 5. This can be represented as a list or set of properties that specifies the required parameters in terms of missing objects. Below is an example of a task model for a bus trip specification:

```
{<O1, Trip, EXIST>,
 <R1, ArrivaLoc; O1:?>,
 <R2, DepartureLoc; O1:?>,
 <R3, ArrivalTime; O1:?>,
 <R4, DepartureTime; O1:?>,
 <R5, TrafficTypeRestrict; O1:?>,
 <R6, BusLineRestrict; O1:?>}
```

Contextual interpretation involves several steps. First, answers to clarification questions need to be integrated with the information from the questions. For example, if the system asks for the value of a property, and the user replies with only an entity type, then they are coupled if the range restriction of the property is fulfilled by the entity type. The algorithm is shown in Figure 7.9.

```
IntegrateAnswerAndQuestion(p: property, e: entity type)

if (IsRange(p,e))
then return (p;?:e)
else return error(p, NoMatchingEntity)
```

Figure 7.9: The algorithm for integration of answers to clarification questions.

```
ContextualInterpretation(p: Property, arg: Argument to p,
                         T: Task model)

for all (p1;arg1:arg2) in T do
  if (p = p1)
  then replace (p1;arg1:arg) in T and add arg to T
       return T


ContextualInterpretation(e: entity type, T: Task model)

P := GetPropertiesWithRange(T, e)

if (NumOfElements(P) = 1)
then replace (p1;arg1:arg) in T and add arg to T
     return T

else if (NumOfElements(P) > 1)
     P2 := GetPropertiesWithMissingEntity(P)

     if (NumOfElements(P2) = 1)
     then replace (p1;arg1:arg) in T and add arg to T
          return T
     else return clarification(e, AmbiguousRelationToProperty, P)

else return error(e, NoMatchingProperties)
```

Figure 7.10: The algorithms for contextual interpretation.

After that the contextual interpretation proceeds with integration of the provided information into the task model. If both a property and an entity type are available, the integration is straightforward, if the property is a part of the task model the information in the task model is replaced by the information provided by the user. If only an entity type is available, all properties with matching range are retrieved from the task model (GetPropertiesWithRange). If there is only one with the right range restriction, the entity type is inserted in the task model. If there are several possible properties to attach the entity type to, a clarification is posed. If there are no properties an error message is returned. The error message has to be handled by the Dialogue Manager, and it can indicate several things, for example, that the user has shifted focus and a new task has been initiated, or that the user has been uncooperative. The algorithms are shown in Figure 7.10.

The last step is to add default values. If there is a property with a missing entity type for the range, a default value can be added from the ontology. The algorithm is shown in Figure 7.11.

```
AddDefaultValues(T: Task model)

for all (p1;arg1:arg2) in T do
  if ((arg2 = NULL) and IsDefaultValue(arg1, p1, v) and
      (v != NULL))
  then replace (p1;arg1:v1) in T

return T
```

Figure 7.11: The algorithm for addition of default values.

The algorithms for contextual interpretation are illustrated by the dialogue excerpt below:

U1: I want to go to Norrköping at noon
S2: From where do you want to leave?
U3: the University by express bus

The first utterance results in the following interpretation.

```
<O1,Trip,EXIST>,
<R1, ArrivaLoc; O1:O2>,
<R2, DepartureTime; O1:O3>,
<O2, Area, Norrköping>
<O3, TimePoint, Noon>
```

Since it concerns a Trip, a task model for trips is created. The information provided by the user can be integrated into the task model since the objects Area and TimePoint match the range restrictions SpatialObject and TemporalObject of the relations ArrivalLoc and ArrivalTime. Default values for TrafficTypeRestrict and BusLineRestrict are also collected from the ontology and integrated into the task model:

```
{<O1, Trip, EXIST>,
 <R1, ArrivaLoc; O1:?>,
 <R2, DepartureLoc; O1:O2>,
 <R3, ArrivalTime; O1:?>,
 <R4, DepartureTime; O1:O3>,
 <R5, TrafficTypeRestrict; O1:O4>,
 <R6, BusLineRestrict; O1:O5>
 <O2, Area, Norrköping>
 <O3, TimePoint, Noon>
 <O4, TrafficType, ALL>
 <O5, BusLine, ALL>}
```

The answer, U3, to the clarification question posed by the system, U2, is interpreted as:

```
<O1, Location, University>
<O2, TrafficType, ExpressBus>
```

The first step in the contextual interpretation of this is to match the object University in the answer to the property DepartureLoc

in the question. This can be done since `IsRange(DepartureLoc, Location)` is true. Next, the produced property and entity type pair is integrated into the task model. TrafficType has no property associated with it, but since there is only one property in the task model of the right range, it can also be integrated into the task model, resulting in:

```
{<O1, Trip,EXIST>,
 <R1, ArrivaLoc; O1:O6>,
 <R2, DepartureLoc; O1:O2>,
 <R3, ArrivalTime; O1:?>,
 <R4, DepartureTime; O1:O3>,
 <R5, TrafficTypeRestrict; O1:O7>,
 <R6, BusLineRestrict; O1:O5>
 <O2, Area, Norrköping>
 <O3, TimePoint, Noon>
 <O5, BusLine, ALL>
 <O6, Location, University>
 <O7, TrafficType, ExpressBus>}
```

Based on the task model the next step is to ask the user for an arrival time.

7.3.4 Clarifications

If the user has indicated a focus shift and only provided a partial request, or if the ellipsis resolution fails, a clarification request should be presented to the user. To be helpful, the clarification should indicate the type of appropriate answer. This means that if an entity type has been provided, the suitable properties should be indicated, and vice versa. This is done by `GetPropertiesForDomain(Entitytype)` or `GetPropertyDomain(Property)`, respectively.

7.4 Domain knowledge management

Domain knowledge management involves the verification and transformation of requests, and reasoning about answers.

7.4.1 Verification of requests

The first step is to verify that the request is valid for database access, i.e. to decide if there is any information for the entity types and properties related to the answer type in the data base. This is decided based on the vocabulary of the concept in the ontology. The labels used for the vocabulary are used to indicate the origin of a concept, and here system-oriented means that it is a database concept. If it is user-oriented and is a leaf node, then it cannot be used for data base access and a helpful error message should be sent to the user instead. A proposal implemented by the algorithms in Figure 7.12 is to look for related concepts for which there is information available and suggest these as alternatives. This is done by traversing the ontology moving one step up and then down again to find all sibling concepts. These are inspected and if suitable added as possible alternatives.

For example, the question "How much does a Sea Gull weigh?" produces the following interpretation:

```
<P1, Weight; O1:V1>
<O1, BirdSpecies, SeaGull>
<V1, Measure, EXIST>
```

Since `GetVocabulary(Weight) = USER` and there are no sub-concepts of Weight, the siblings Length and Wingspan are retrieved based on the existence of a common super-concept Size. Both Length and Wingspan have vocabulary `SYSTEM` and thus they are added to Alternatives, and a clarification is constructed, `clarification(Weight, OutsideDB, {Length, Wingspan})`

```
VerifyRequest(C: Set of concepts)

for all c1 in C do
  if ((GetVocabulary(c1) = USER) and (GetSubConcepts(c1) = NULL))
  then for all c2 in GetSuperConcepts(c1) do
       add GetSubConcepts(c2) to Siblings

       for all c3 in Siblings do
         if ((GetVocabulary(c3) = SYSTEM) or
             ((GetVocabulary(c3) = USER) and
              (GetSubConcepts(c3) != NULL)))
         then add c3 to Alternatives

  if (Alternatives != NULL)
  then return clarification(c1, OutsideDB, Alterantives)
  else return error(c1, OutsideDB)
```

Figure 7.12: The algorithm for verification of a request.

7.4.2 Transformation of properties

To transform the properties for database access, the ontology must be accessed to decide if a property is vague and has to be mapped to a more specific. In such cases the sub-properties are collected and if necessary, the request is partitioned into several object and property pairs, and each is used to collect the requested information from various data bases. The algorithm for this is described in Figure 7.13

For example, the request "What does a waterfowl look like?" is interpreted as:

```
<O1, Family, Waterfowl>
<A1, Apperance; O1:V1>
<V1, String, EXIST>
```

When Appearance is transformed the sub-properties Size and Plumage are retrieved from the ontology, and since both of these have vocab-

```
TransformProperty(p: Property)

if (GetVocabulary(p) = USER)
then for all p1 in GetSubProperties(p) do
       if (GetVocabulary(p1) = SYSTEM)
       then add p1 to NewProperties
       else add p1 to Alternatives

     if (NewProperties != NULL)
     then return NewProperties

     else if (Alternatives != NULL)
          then return clarification(p, VagueProperty, Alterantives)

     else return error(p, OutsideDB)

else return {p}
```

Figure 7.13: The algorithm for transformation of a request with a vague property.

ulary USER they are added to Alternatives, and a clarification is posed, `clarification(Appearance, VagueProperty, {Size, Plumage})`

7.4.3 Transformation of entity types

For entity types the database holds information for instances, so if the user has requested information about a certain property for an instance, the domain and range restrictions are tested to see if they match. If the entity types do not correspond, the taxonomy is traversed in order to find a sub-entity type of the right type, and then the knowledge base is accessed to find instances of these. The new instances are tested to see if they are taxonomically related to the original instance. The algorithm for this is described in Figure 7.14

```
TransformEntitytype(e: Entity Type, i: Instance of e, p: Property)

If (GetDomain(p) != e)
then for all e1 in GetSubEntitytypes(e) do
       if (GetDomain(p) = e1)
       then add e1 to NewEntitytypes

       for all ne in NewEntitytypes do
         for all i1 in GetInstances(ne) do
           if (i1 Hyponym i)
           then add i1 to NewInstances

         if (NewInstances != NULL)
         then add (ne, ni) to Alternatives

      if (NumOfElements(Alternatives) > N)
      then return clarification(i, VagueEntitytype, Alternatives)

      else if (Alternatives != NULL)
           then return Alternatives

      else return error(i, OutsideDB)

else return {e}
```

Figure 7.14: The algorithm for transformation of a request with a vague entity type.

For example, in the request "What does a waterfowl look like?", Waterfowl is of the entity type Family, and `GetDomain(Appearance)` is not. Thus, the sub-entity types of Family are collected from the ontology. This set only contains BirdSpecies, and thus all the instances of this entity type that are also hypernyms of Waterfowl are retrieved from the knowledge base. Since there are too many instances, a clarification is produced, `clarification(Waterfowl, VagueEntitytype, {Barnacle, Goose, ..., Gadwall})`

7.4.4 Reasoning about answers

In combination with transformation of requests, reasoning about answers is needed. Since a request can result in several new requests, but there are not necessarily answers to all of them, the database access can result in three situations; no answer is found, a suitable number of answers are retrieved, or too many answers are found. In the first case, a new transformation of the request can be attempted using algorithms 7.13 and 7.14. If there are too many answers, i.e. more than suitable for presentation in the media used in the dialogue system, a clarification should be initiated. The algorithms for reasoning on this is described in figures 7.15 and 7.16.

```
InspectAnswers(e: Original entity type, i: Instance of e,
               p: Original property,
               NP: Set of derived properties)

for all np in NP do
add AccessDB(e,i,np) to DBAnswers

if (NumOfElements(DBAnswers) = 0)
then for all np in NP do
  add TransformProperty(np) to NP2

if (NP2 != NULL)
InspectAnswers(e, i, p, NP2)

else if (NumOfElements(DBAnswers) > N)
     then return clarification(p, VageProperty, NP)

else return DBAnswers
```

Figure 7.15: The algorithm for reasoning about answers based on a vague property.

```
InspectAnswers(e: Original entity type, i: Instance of E,
               p: Original property, NE: Set of derived entity types)

for all (ne,ni) in NE do
add AccessDB(ne, ni, p) to DBAnswers

if (NumOfElements(DBAnswers) = 0)
then for all (ne,ni) in NE do
  add TransformEntitytype(ne, ni,p) to NE2

if (NE2 != NULL)
InspectAnswers(e, i, p, ne2)

else if (NumOfElements(DBAnswers) > N)
     then return clarification(e, VageEntitytype, NE)

return DBAnswers
```

Figure 7.16: The algorithm for reasoning about answers based on a vague entity type.

7.5 Summary

The framework for the use of ontologies in information-providing dialogue systems provides algorithms for the tasks necessary to achieve a natural and graceful interaction. User utterances can be robustly interpreted since the ontology supports word sense disambiguation and methods to move from partial to full interpretations. Dialogue management is improved since ontological-based resolution of anaphora and ellipsis give better results than syntax-based. Ontological contextual interpretation allows for flexible task management and clarifications can be more helpful. Domain knowledge management with verification and transformation of requests can map user concepts to system concepts, and gives better error message if it fails to retrieve the information.

The framework is intended as a support for the development of dialogue systems where the algorithms deemed necessary can be implemented using an ontology. The ontology design that supports this framework, AnOntology, has been implemented in Java with an API that includes the minimal set of operators needed. The module is thus straightforward to use, it is however possible to use other ontology tools in the framework as long as they support the same type of functionality.

Chapter 8

Summary and future work

This chapter summarises the results presented in the thesis and discusses some future directions in work with ontologies and dialogue systems.

The goal of the work presented in this thesis has been to investigate how ontologies can facilitate the development of information-providing dialogue systems with capabilities for domain knowledge reasoning and high portability. To achieve this goal an approach that combine analysis and synthesis of issues concerning design of ontologies used in dialogue systems with practical work on design and construction of dialogue applications that perform domain reasoning using ontologies, has been utilised.

The results are mainly of a theoretical character and include a requirements analysis of ontology design and a framework for usage of ontologies in information-providing dialogue systems. The results illustrate the potential of ontologies as domain knowledge sources that support portable and knowledge-rich approaches to interpretation, di-

alogue management, and domain knowledge management in dialogue systems. They are useful for the research community and the industry who develop dialogue systems that utilise ontological knowledge in some form. The character of the results also make them suitable as a stepping stone for further theoretical and practical research in this area.

In this chapter the research issues that has been addressed and the related contributions are summarised and discussed. Some interesting future research issues are also presented.

8.1 Issues and contribution

What type of functionality in dialogue systems can ontologies support?

An analysis of existing dialogue systems that utilise ontologies, presented in Chapter 3, revealed that ontologies can support interpretation of questions and requests, where ontological knowledge is used primarily for disambiguation, and dialogue management, specifically reference resolution, clarifications and contextual interpretations.

In the new framework for use of ontologies in information-providing dialogue systems, presented in Chapter 7, these functions were implemented along with support for transformation of partial to full interpretations, ellipsis resolution in dialogue management, and domain knowledge management functionality, i.e. verification and transformation of requests and reasoning on answers. A primary function for domain knowledge management is also to give helpful error messages or clarification requests if database access fails.

How can ontologies be incorporated in a dialogue system framework to support high portability?

A specification of the ontology functionality in terms of general operators and a modular architecture of the ontology allows for usage of ontological knowledge in several dialogue system tasks, as described by the algorithms in Chapter 7, with high portability. For a new domain, declarative knowledge in terms of an ontology, a knowledge base, and a database can be plugged in.

How should ontologies be designed to support the domain reasoning necessary in dialogue systems?

A compilation of design choices and design guidelines in the ontology research area made from the point of view of design of ontologies for dialogue systems, were presented in Chapter 2. These were then, in Chapter 4, analysed for various tasks in dialogue systems, interpretation, dialogue management, and domain knowledge management, based on empirical investigations of existing dialogue systems and corpora. The result was a design specification of ontologies that are to be used in dialogue systems, Chapter 7.

How can ontologies be developed to capture and integrate different views of a domain?

In Chapter 6, the development of an integrated ontology that captured both user and system conceptualisations was presented. In this case the merging of the different conceptualisations to create a shared domain ontology was rather straightforward. As user conceptualisations differ due to variations in interest and knowledge of the domain, more sophisticated methods for integration of ontologies might be required. It is possible that merging into one ontology is not possible and that it will rather be a question of mapping between several ontologies.

8.2 Future work

There are many areas and issues in which to continue the work, for instance refinement of guidelines, methods and tools for design and development, and various aspects of use of ontologies in dialogue systems and other related NLP areas. Here I will mention three in particular, corpora-based development of ontologies, integration of Information Extraction in information-providing dialogue systems, and the use of ontologies in various types of dialogue systems

8.2.1 Corpora-based development of ontologies

A survey of existing development methods and tools was presented in Chapter 2. Most of these relies on the work being carried out by an experienced knowledge engineer, or a community of users that can provide the domain knowledge. In NLP, an important source for information is corpora of various types. To make the development more efficient and flexible, corpora-based tools should be investigated. The use of corpora as a source for ontological knowledge also support portability, and simple maintenance and update in dynamic domains.

8.2.2 Use of ontologies for Information Extraction in dialogue systems

As stated in Chapter 6 much information today is available in unstructured documents rather than databases. A future direction in information-providing dialogue systems is therefore to integrate information extraction techniques that can mine the requested information on the fly, similar to how answers are retrieved in Q&A systems. The ontology in dialogue systems that integrate information extraction might need extended functionality to also support the information extraction tasks.

8.2.3 Use of ontologies in dialogue systems

The design requirements and the framework that have been presented in this thesis are aimed at dialogue systems that provide information, but many of the tasks, especially interpretation and dialogue management, are also applicable for other types of dialogue systems, such as problem solving. It is therefore interesting to take these as a starting point for future investigations of functionality of ontologies used for other tasks than retrieval and presentation of information.

Bibliography

Alexandersson, J. and T. Becker (2003). The formal foundations underlying overlay. In *Proceedings of the Fifth International Workshop on Computational Semantics, IWCS-5*, Tilburg, The Netherlands.

Alexandersson, J., E. Maier, and N. Reithinger (1994). A robust and efficient three-layered dialogue component for speech-to-speech translation system. Technical Report 50, DFKI GmbH, `www.dfki.uni-sb.de/cgi-bin/verbmobil/htbin/doc-access.cgi`.

Alexandersson, J. and N. Reithinger (1995). Designing the dialogue component in a speech translation system. In *Proceedings of the Ninth Twente Workshop on Language Technology, TWLT-9*, pp. 35–43.

Allen, J., D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent (2000). An architecture for a generic dialogue shell. *Journal of Natural Language Engineering, Special Issue on Best Practices in Spoken Language Dialogue Systems Engineering 3*(6), 1–16.

Allen, J., D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent (2001). Toward conversational human-computer interaction. *AI Magazine 22*(4), 27–37.

Allen, J., L. Schubert, G. Ferguson, P. Heeman, C. Hwang, T. Kato, M. Light, N. Martin, B. Miller, M. Poesio, and D. Traum (1995). The TRAINS project: a case study in building a conversational planning agent. *Journal of Experimental*

and Theoretical Artificial Intelligence 7, 7–48.

Appelt, D. and D. Israel (1999). Introduction to information extraction technology. A Tutorial fot IJCAI'99. `www.ai.sri.com/ appelt/ie-tutorial/IJCAI99.pdf`.

Arpiréz, J., O. Corcho, M. Fernández-López, and A. Gómez-Pérez (2003). WebODE in a nutshell. *AI Magazine 24* (3), 37–48.

Aust, H., M. Oerder, F. Seide, and V. Steinbiss (1995). The Philips automatic train timetable information system. *Speech Communication 17*(3-4), 249–262.

Bagga, A., J. Chai, and A. Biermann (1996). The role of WordNet in the creation of a trainable message understanding system. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, Menlo Park, California, USA, pp. 941–948.

Bateman, J. (1990). Upper modeling: organizing knowledge for natural language processing. In *Proceedings of 5th International Workshop on Natural Language Generation*, Pittsburgh, USA.

Bateman, J. (1991). The theoretical status of ontologies in natural language processing. In *Proceedings of workshops on Text Representation and Domain Modelling - Ideas from Lingustics and AI*, KIT Report 97. Technical University Berlin.

Bennacef, S., L. Devillers, S. Rosset, and L. Lamel (1996). Dialog in the RAILTEL telephone-based system. In *Proceedings of the International Conference on Spoken Language Processing, ICSLP'96*, Volume 1, Philadelphia, USA, pp. 550–553.

Beringer, N., U. Kartal, K. Louka, F. Schiel, and U. Türk (2002). Promise - a procedure for multimodal interactive system evaluation. In *Proceedings of the Workshop Multimodal Resources and Multimodal Systems Evaluation*, Las Palmas, Gran Canaria, Spain.

Beveridge, M. and D. Millward (2003). Combining task descriptions and ontological knowledge for adaptive dialogue. In *Proceedings of the 6th International Conference on Text, Speech and Dialogue, TSD'03*, Ceské Budejovice, Czech Republic.

Blaylock, N., J. Allen, and G. Ferguson (2002). Synchronization in an asynchronous agent-based architecture for dialogue systems. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialog*, Philadelphia, USA.

Blazquez, M., M. Fernández, J. Garcia-Pinar, and A. Gómez-Pérez (1998). Building ontologies at the knowledge level using the ontology design environment. In *Proceedings of the 11th Knowledge Acquisition Workshop, KAW'98*, Banff, Canada.

Carlson, R. and S. Hunnicut (1996). Generic and domain-specific aspects of the Waxholm NLP and dialog modules. In *Proceedings of International Conference on Spoken Language Processing, ICSLP'96*, Volume 2, Philadelphia, USA, pp. 677–680.

Chandrasekaran, B., J. R. Josephson, and V. R. Benjamins (1999). What are ontologies, and why do we need them? *IEEE Intelligent Systems 14*(1), 20–26.

Chu-Carroll, J. (2000). MIMIC: An adaptive mixed initiative spoken dialogue system for information queries. In *Proceedings of 6th Applied Natural Language Processing Conference*, pp. 97–104.

Corcho, O. and A. Gómez-Pérez (2000). Evaluating knowledge representation and reasoning capabilities of ontology specification languages. In *Proceedings of ECAI'00 Workshop on Applications of Ontologies and Problem-Solving Methods*, Berlin, Germany.

Cunningham, H. (2000). *Software Architecture for Language Engineering*. Ph. D. thesis, University of Sheffield. `gate.ac.uk/sale/thesis/`.

Dahlbäck, N. and A. Jönsson (1999). Knowledge sources in spoken dialogue systems. In *Proceedings of Eurospeech'99*, Budapest, Hungary.

Dahlbäck, N. (1991). *Representations of Discourse, Cognitive and Computational Aspects*. Ph. D. thesis, Linköping University.

Dahlbäck, N., A. Jönsson, and L. Ahrenberg (1993). Wizard of oz studies – why and how. *Knowledge-Based Systems 6*(4),

258–266. Also in: *Readings in Intelligent User Interfaces*, Mark Maybury & Wolfgang Wahlster (eds), Morgan Kaufmann, 1998.

Dahlgren, K. (1995). A linguistic ontology. *International Journal Human-Computer Studies 43*(5-6), 809–818.

Davis, R., H. Shrobe, and P. Szolovits (1993). What is a knowledge representation? *AI Magazine 14*, 17–33.

Degerstedt, L. and A. Jönsson (2001). Iterative implementation of dialogue system modules. In *Proceedings of Eurospeech 2001*, Aalborg, Denmark.

Denecke, M. (1997). An information-based approach for guiding multi-modal human-computer-interaction. In *Proceedings of IJCAI'97*, Nagoya, Japan, pp. 1036–1041.

Denny, M. (2002). Ontology building: A survey of editing tools. XML.com
`www.xml.com/pub/a/2002/11/06/ontologies.html`.

Dzikovska, M. (2004). *A practical semantic representation for natural language parsing.* Ph. D. thesis, University of Rochester.

Dzikovska, M., M. Swift, and J. Allen (2003). Integrating linguistic and domain knowledge for spoken dialogue systems in multiple domains. In *Proceedings of the IJCAI'03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico.

Embley, D., D. Campbell, S. Liddle, and R. Smith (1998). Ontology-based extraction and structuring of information from data-rich unstructured documents. In *Proceedings of CIKM'98*, Washington, D.C., USA.

Fernández, M. (1999, August). Overview of methodologies for building ontologies. In *Proceedings of IJCAI'99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends.*

Fikes, R., A. Farquhar, and J. Rice (1997). Tools for assembling modular ontologies in ontolingua. Technical report, Knowledge Systems Laboratory,
`www.ksl.stanford.edu/software/ontolingua/project-papers.html`.

Flycht-Eriksson, A. (1999, August). A survey of knowledge sources in dialogue systems. In *Proceedings of IJCAI'99 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Stockholm, pp. 41–48.

Flycht-Eriksson, A. (2000). A domain knowledge manager for dialogue systems. In *Proceedings of the 14th European Conference on Artificial Intelligence, ECAI 2000*. IOS Press, Amsterdam.

Flycht-Eriksson, A. (2001). Domain knowledge management in information-providing dialogue systems. Licentiate Thesis 890, Linköping Studies in Science and Technology, Linköping University.

Flycht-Eriksson, A. (2003). Design of ontologies for dialogue interaction and information extraction. In *Proceedings of IJCAI'03 workshop on Knowledge and reasoning in practical dialogue systems*, Acapulco, Mexico.

Flycht-Eriksson, A. and A. Jönsson (2000). Dialogue and domain knowledge management in dialogue systems. In *Proceedings of the 1st SIGdial Workshop on Discourse and Dialogue*, Hong Kong.

Flycht-Eriksson, A. and A. Jönsson (2003). Some empirical findings on dialogue management and domain ontologies in dialogue systems - Implications from an evaluation of BirdQuest. In *Proceedings of 4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan.

Flycht-Eriksson, A., A. Jönsson, M. Merkel, and H. Sundblad (2003). Ontology-driven information-providing dialogue systems. In *Proceedings of 2003 Americas Conference on Information Systems*, Tampa, Florida, USA.

Fried, D., D. Wilkins, and E. Grois (2003). The Gerona knowledge ontology and its support for spoken dialogue tutoring of crisis decision making skills. In *Proceedings of the IJCAI'03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico.

FZI and AIFB (2004). The karlsruhe ontology and semantic web tool suite, kaon.
`kaon.semanticweb.org/`.

Gangemi, A., N. Guarino, C. Masolo, and A. Oltramari (2001). Understanding top-level ontological distinctions. In *Proceedings of IJCAI'01 Workshop on Ontologies and Information Sharing*, Seattle, Washington, USA.

Gangemi, A., N. Guarino, C. Masolo, and A. Oltramari (2003, Fall). Sweetening WordNet with DOLCE. *AI Magazine 24*(3), 13–24.

Gangemi, A., D. Pisanelli, and G. Steve (1999). An overview of the ONIONS project: Applying ontologies to the integration of medical terminologies. *Data Knowledge Engineering 31*(2), 183–220.

Gazauskas, R., K. Humphreys, S. Azzam, and Y. Wilks (1997). *Concepticons vs. Lexicons: An Architecture for Multilingual Information Extraction*, Volume 1299 of *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, Lecture Notes in Artificial Intelligence*, Chapter 3, pp. 28–43. Springer.

Genesereth, M. and R. Fikes (1992). Knowledge interchange format. Technical Report Logic-92-1, Computer Science Department, Stanford University.

Genesereth, M. R. and N. J. Nilsson (1987). *Logical foundations of Artificial Intelligence*. Los Alts, California, USA: Morgan Kaufman.

Gennari, J., M. Musen, R. Fergerson, W. Grosso, M. Crubézy, H. Eriksson, N. Noy, and S. Tu (2002). The evolution of Protégé: An environment for knowledge-based systems development. Technical Report SMI-2002-0943, Stanford University.

Gerstl, P. and S. Pribbenow (1995). Midwinters, end games, and body parts: a classification of part-whole relations. *International Journal Human-Computer Studies 43*(5-6), 865–889.

Goddeau, D., E. Brill, J. Glass, C. Pao, M. Philips, J. Polifroni, S. Seneff, and V. Zue (1994). GALAXY: A human-language interface to on-line travel information. In *Proceedings of International Conference on Spoken Language Processing, ICSLP'94*, Yokohama, Japan, pp. 707–710.

Gómez-Pérez, A. (1995). Some ideas and examples to evaluate ontologies. In *Proceedings of the Eleventh Conference on Artificial Intelligence Applications*. IEEE Computer Society Press.

Gómez-Pérez, A. (1999). Ontological engineering: A state of the art. *Expert Update. British Computer Society. 2*(3).

Gómez-Pérez, A., M. Fernández, and A. de Vicente. (1996). Towards a method to conceptualize domain ontologies. In *Proceedings of the ECAI'96 Workshop on Ontological Engineering*, Budapest, Hungary, pp. 41–52.

Gorrell, G., I. Lewin, and M. Rainer (2002). Adding intelligent help to mixed initiative spoken dialogue systems. In *Proceedings of the International Conference on Spoken Language Processing, ICSLP'02*.

Gove, P. and Merriam-Webster (2000). *Webster's Third New International Dictionary* (3rd ed.). Merriam-Webster, Inc.

Grenon, P. (2003, Augusr). Tucking RCC in Cyc's ontological bed. In *Proceedings of IJCAI'03*, Acapulco, Mexico.

Grosz, B. (1977). *The representation and use of focus in dialogue understanding*. Ph. D. thesis, University of California, Berkeley.

Gruber, T. R. (1993a). Towards principles for the design of ontologies used for knowledge sharing. Technical report, Stanford University, Palo Alto, California, USA.

Gruber, T. R. (1993b). A translation approach to portable ontology specification. *Knowledge Acquisition 5*, 199–220.

Grüninger, M. (1996). Designing and evaluating generic ontologies. In *Proceedings of the ECAI'96 Workshop on Ontological Engineering*, Budapest, Hungary.

Grüninger, M. and M. Fox (1995). Methodology for the design and evaluation of ontologies. In *Proceedings of the IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing*.

Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representaiton. *International Jouranl of Human-Computer Studies 43*, 625–640.

Guarino, N. (1998a). Formal ontology in information systems. In N. Guarino (Ed.), *Formal Ontology in Information Systems, Proceedings of FOIS'98*, Trento, Italy, pp. 3–15. IOS Press, Amsterdam.

Guarino, N. (1998b). Some ontological principles for designing upper level lexical resources. In *Proceedings of the First International Conference on Language Resources and Evaluation*, Granada, Spain, pp. 527–534.

Gurevych, I., R. Porzel, E. Slinko, N. Pfleger, J. Alexandersson, and S. Merten (2003). Less is more: Using a single knowledge representation in dialogue systems. In *Proceedings of the HLT-NAACL'03 Workshop on Text Meaning*, Edmonton, Canada, pp. 14 – 21.

Gustafson, J. and L. Bell (2000). Speech technology on trial: Experiences from the august system. *Natural Language Engineering 6*(3-4), 273–286.

Hagen, E. (1999). An approach to mixed initiative spoken information retrieval dialogue. *User modeling and User-Adapted Interaction 9*(1-2), 167–213.

Harabagiu, S., S. Maiorano, and M. Paşca (2003). Open-domain textual question answering techniques. *Natural language engineering 9*(3), 231–267.

Harabagiu, S., D. Moldovan, M. Paşca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Gîrju, V. Rus, and P. Morărescu (2000). FALCON: Boosting knowledge for answer engines. In *Proceedings of the Ninth Text REtrieval Conference, TREC-9*, Gaithersburg, Maryland, USA.

Heisterkamp, P., S. McGlashan, and N. Youd (1992). Dialogue semantics for a spoken dialogue system. In *Proceedings of the International Conference on Spoken Language Processing, ICSLP'92*, Banff, Canada.

Holsapple, C. and K. Joshi (2002, February). A collaborative approach to ontology design. *Communications of the ACM 45*(2), 42–47.

IBM (1999). Building object-oriented frameworks.
`www.ibm.com/java/education/oobuilding/index.html`.

Johnson, R. (1997). Frameworks home page.
`st-www.cs.uiuc.edu/users/johnson/frameworks.html`.

Jokinen, K., H. Tanaka, and A. Yokoo (1998). Context management with topics for spoken dialogue systems. In *Proceedings of the 36th Annual Meeting of the Association of Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL'98*, Montreal, Canada, pp. 631–637.

Jones, D., T. Bench-Capon, and P. Visser (1998). Methodologies for ontology development. In *Proceedings of IT & KNOWS Conference, XV IFIP World Computer Congress*, Budapest, Hungary.

Jönsson, A. (1993). *Dialogue Management for Natural Language Interfaces*. Ph. D. thesis, Linköping University.

Jönsson, A. (1995). Dialogue actions for natural language interfaces. In *Proceedings of IJCAI'95*, Montréal, Canada.

Jönsson, A. (1997). A model for habitable and efficient dialogue management for natural language interaction. *Natural Language Engineering 3*(2/3), 103–122.

Jönsson, A. and N. Dahlbäck (1988). Talking to a computer is not like talking to your best friend. In *Proceedings of the First Scandinavian Conference on Artificial Intelligence*, Tromsø, Norway.

Jönsson, A. and L. Strömbäck (1998). Robust interaction through partial interpretation and dialogue management. In *Proceedings of the 36th Annual Meeting of the Association of Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL'98*, Montreal, Canada.

Kifer, M., G. Lausen, and J. Wu (1990). Logical foundations of object-oriented and frame-based languages. Technical Report TR-90-003, Department for Mathematics and Computer Science, University of Mannheim.

Kishore, R., R. Ramesh, and R. Sharman (2003, August). Computational ontologies: Foundations, representations, and methods. Tutorial at AMCIS'03.

Lakemeyer, G. and B. Nebel (1992). Foundations of knowledge representation and reasoning. In *ECAI Workshop on Knowledge Representation and Reasoning*, pp. 1–12.

Larsson, S., L. Santamarta, and A. Jönsson (2000). Using the process of distilling dialogues to understand dialogue systems. In *Proceedings of 6th International Conference on Spoken Language Processing, ICSLP 2000*, Beijing, China.

Leceuche, R., D. Robertson, C. Barry, and C. Mellish (2000). Evaluating focus theories for dialogue management. *International Journal on Human-Computer Studies 52*, 23–76.

Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM 38*(11), 33–38.

Mahesh, K. (1996). Ontology development for machine translation: Ideology and methodology. Technical Report MCCS-96-292, Computing Research Laboratory, New Mexico State University.

Mahesh, K. and S. Nirenburg (1995). A situated ontology for practical NLP. In *Proceedings of IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, Canada.

Miller, G., R. Beckwith, C. Fellbaum, D. Gross, and K. Miller (1993). Introduction to wordnet: an on-line lexical database. `ftp.cogsci.princeton.edu/pub/wordnet/5papers.ps` .

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM 38*(11), 39–41.

Milward, D. and M. Beveridge (2003, August). Ontology-based dialogue systems. In *Proceedings of the IJCAI'03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico.

Milward, D. R. (2002). Exploiting the advantages of task and linguistically orientated dialogue management. Deliverable d4.4, Project SIRIDUS, `www.ling.gu.se/projekt/siridus/Publications/publications.html`.

nlpFarm (2004). nlpFarm open source project.
`nlpfarm.sourceforge.net/`.

Noy, N., R. Fergerson, and M. Musen (2000). The knowledge model of protégé-2000: Combining interoperability and flexibility. In *Proceedings of 2th International Conference on Knowledge Engineering and Knowledge Management, EKAW'2000*, Juan-les-Pins, France.

Noy, N. and C. Hafner (1997). The state of the art in ontology design, a survey and comparative review. *AI Magazine 18*(3), 53–74.

Noy, N. and D. McGuinness (2001). Ontology development 101: A guide to creating your first ontology. Technical report, Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, Stanford.

Oberle, D., R. Volz, B. Motik, and S. Staab (2004). An extensible ontology software environment. In S. Staab and R. Studer (Eds.), *Handbook on Ontologies*, pp. 311–333. Springer.

Ontolingua (2004).
`www.ksl.stanford.edu/software/ontolingua/`.

Ontological Engineering Group, T. U. o. M. (2003, November). WebODE ontology engineering platform.
`delicias.dia.fi.upm.es/webODE/`.

Ortiz, A. M., V. Raskin, and S. Nirenburg (2002, June). New developments in ontological semantics. In *Proceedings of LREC'02*, Spain.

Pfleger, N., J. Alexandersson, and T. Becker (2003). A robust and generic discourse model for multimodal dialogue. In *Proceedings of the IJCAI'03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico.

Porzel, R., N. Pfleger, S. Merten, M. Loeckelt, I. Gurevych, R. Engel, and J. Alexandersson (2003, August). More on less: Further applications of ontologies in multi-modal dialogue systems. In *Proceedings of the IJCAI'03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Acapulco, Mexico.

Protege-2000 (2004).
`protege.stanford.edu/`.

Quantz, J., M. Gehrke, U. Küssner, and B. Schmitz (1994). The VERBMOBIL domain model version 1.0. Technical Report 29, Technishe Universität Berlin.

Qvarfordt, P. (2003). User experience of spoken feedback in multimodal interaction. Licentiate Thesis 1003, Linköping Studies in Science and Technology, Linköping University.

Rangemalm, E. L. (1996). Student diagnosis in practice; bridging a gap. *User Modelling and User Adapted Interaction 2*(5), 93–116.

Raynor, W. (2000). *International Dictionary of Artificial Intelligence*. AMACOM Publishing.

R.Porzel, I. Gurevych, and C. E. Muller (2003). Ontology-based contextual coherence scoring. In *Proceedings of 4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan.

Russel, S. and P. Norvig (1995). *Artificial Intelligence: A modern approach*. Prentice Hall.

Santos, J. and S. Staab (2003, October). Fonte - factorizing ontology engineering complexity. In *Proceedings of International Conference on Knowledge Capture, ACM K-Cap 2003*, Sanibel Island, Florida, USA.

Seneff, S., D. Goddeau, C. Pao, and J. Polifroni (1996, October). Multimodal discourse modelling in a multi-user multi-domain environment. In *Proceedings of the International Conference on Spoken Language Processing, ICSLP'96*, Philadelphia, USA, pp. 192–195.

Seneff, S., E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue (1998). GALAXYII: A reference architecture for conversational system development. In *Proceedings of the International Conference on Spoken Language Processing, ICSLP'98*, Volume 3, Sydney, Australia, pp. 931–934.

Smith, B. (forthcoming). *Stanford Encyclopedia of Philosophy*, Chapter Ontology and Information Systems. Stanford university.

Smith, B. and C. Welty (2001). Ontology: Towards a new synthesis. In C. Welty and B. Smith (Eds.), *Formal Ontology in Information Systems, Proceedings of FOIS'01*, Ongunquit, Maine, pp. iii–x. ACM Press.

Smith, R. and D. Hipp (1994). *Spoken Natural Language Dialog Systems: A Practical Approach*. New York: Oxford University Press.

Sowa, J. (2000). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove, California, USA: Brooks Cole Publishing Co.

Staab, S. and A. Maedche (2000). Axioms are objects, too – ontology engineering beyond the modeling of concepts and relations. Technical Report 399, Institute AIFB, Univ. of Karlsruhe.

Staav, R. and T. Fransson (1991). *Nordens fåglar*. Stockhom: Norstedt.

Strömbäck, L. and A. Jönsson (1998). Robust interpretation for spoken dialogue systems. In *Proceedings of the International Conference on Spoken Language Processing, ICSLP'98*, Sydney, Australia.

Tapanainen, P. and T. Järvinen (1997). A non-projective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pp. 64–71.

Uschold, M., M. King, S. Moralee, and Y. Zorgios (1995). The enterprise ontology.

Ushold, M. and M. Gruninger (1996, June). ONTOLOGIES: Principle, methods and applications. *Knowledge Engineering Review 11*(2).

Valente, A. and J. Breuker (1996). Towards principled core ontologies. In B. Gaines and M. Mussen (Eds.), *Proceedings of KAW'96*, Banff, Canada.

van der Vet, P. and N. Mars (1998). Bottom-up construction of ontologies. *IEEE Transactions on Knowledge and Data Engineering 10*(4).

W3C (2004a). Extensible markup language (XML). `www.w3.org/XML/`.

W3C (2004b). Owl web ontology language overview. `www.w3.org/TR/2004/REC-owl-features-20040210/`.

W3C (2004c). Resource description framework (RDF). `www.w3.org/RDF/`.

Wahlster, W., N. Reithinger, and A. Blocher (2001). SmartKom: Towards multimodal dialogues with anthropomorphic interface agents. Technical Report 15, DFKI Saabrücken.

Walker, M., D. Litman, C. Kamm, and A. Abella (1998). Paradise: A framework for evaluating spoken dialogue agents. In M. Maybury and W. Wahlster (Eds.), *Readings in Intelligent User Interfaces*. Morgan Kaufmann.

Wee, L., L. Tong, and C. Tan (1999). Textual information extraction in the face of information deluge. In *Proceedings of the IJCAI'99 Workshop Text Mining, Foundations, Techniques and Applications*, Stockholm, Sweden.

Winston, M., R. Chaffin, and D. Hermann (1987). A taxonomy for part-whole relations. *Cognitive Science 11*(4), 417–444.

Zajac, R. (2001). Towards ontological question answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, ACL'01, Workshop on Open-Domain Question Answering*.

Zukerman, I., R. McConachy, and K. Korb (2000). Using argumentation strategies in automated argument generation. In *Proceedings of the First International Natural Language Generation Conference, INLG'2000*, Mitzpe Ramon, Israel, pp. 55–62.