Typology for Information Contents

Responsible for the Report
Annelise Bech, Andrei Mikheev, Marc Moens, Costanza Navarretta

July 1993

ET-12 Project
Methodologies for Constructing Knowledge Bases
for Natural Language Processing Systems
Report 2

Contents

Preface **3**

1 Introduction **4**

Part 1: Conceptual Modeling: Towards a Typology for Knowledge Organisation **8**

2 Conceptual modeling **9**

2.1 The model 10

2.2 The knowledge base 12

2.3 Conceptual language 14

3 Types of knowledge **16**

4 The taxonomy **20**

4.1 Type hierarchy 20

4.2 Aggregation 22

5 Canonical structures **23**

6 Definitions **27**

7 Functional dependencies **30**

8 Schemata, prototypes and constraints **31**

9 Conceptual dependencies and episodes **35**

Part 2: Information Contents for the Knowledge Elicitation Process **40**

10 A working method for knowledge elicitation **41**

11 Types of information presupposed by texts **42**

12 Examples of the analysis of three text corpora **44**

12.1 Characteristics of the three text corpora 45

13 The analysis of some texts from the three corpora **47**

13.1 Analysis of one PDS 47

13.2 Analysis of a text from the motor car mechanical domain . . 50

13.3 Analysis ofa text from the terrorist corpus 52

13.4 Conclusion 55

14 References **56**

Preface

In the first phase of the project *Methodologies for Constructing Knowledge Bases for Natural Language Processing Systems* we made an analysis of the state–of–the–art in knowledge engineering for natural language understanding systems and adopted the 3-step methodology proposed in Hobbs (1984) and its 9-step extension as our starting point for further work. This proposal was argued for in Project Report Nr 1.

The objectives of the current phase of the project (Work Package 2) were twofold. One was to determine which *kinds* of knowledge should be encoded in natural language understanding systems and how the nature of this information has repercussions on the choice of knowledge engineering strategies. The current report (Project Report Nr 2) addresses these issues for what we argue are two related yet clearly different aspects of knowledge engineering, viz. knowledge elicitation and knowledge organisation. Making this distinction has allowed us to clarify and revise the methodology outlined in Project Report Nr 1. Project Report Nr 2 thus has two major parts: one dealing with organisation, one with elicitation. The paper on knowledge organisation was the responsibility of the Edinburgh team; it explores the nature and goals of conceptual modeling. The paper on elicitation was produced by the Copenhagen team and describes an information typology obtained by applying a particular knowledge elicitation technique.

The other objective of this phase of the project was an examination of the characteristics possible support material should have in such a knowledge engineering enterprise. This is reported on in Project Report Nr 3.

1 Introduction

The methodology we adopted as a starting point, viz. the methodology proposed by Hobbs (1984, 1986), can be summarized in three steps:

1. Select the facts relevant to the domain.
2. Organise the facts into domain clusters.
3. Formalise the facts.

Underlying this seemingly simple methodology were a number of well thought out principles governing the process of knowledge engineering for natural language understanding systems:

- The standard sublanguage approach to domain modeling (stating domain dependent semantic constraints) must be supplemented with the creation of a knowledge base containing extra-linguistic knowledge.
- The aim of the methodology is to extract knowledge that is presupposed but not explicitly mentioned in the analysed texts.
- The main principle in the process of knowledge base creation is selectivity: one should attempt only to encode facts directly relevant to the natural language processing task at hand.
- Word meanings should be tuned to domain usage. Instead of formalizing the general meaning of a word one has to find the aspects of the meaning which are relevant to the domain.

In Project Report Nr 1 we argued why we wanted to take this 3-step methodology and its underlying philosophy as the starting point for our work. We also reported how the methodology was extended by Hobbs and Navarretta into a more detailed 9-step method.

However, subsequent investigation has revealed that the 9-step extension misses some of the important characteristics underlying the original proposal. For example, the original methodology makes a clear distinction between knowledge elicitation and knowledge organisation. In the 9–step method this distinction is somewhat blurred: it assumes that the the knowledge organisation phase can be driven by the highest layers of world knowledge (the "core theories"), knowledge that is presupposed to exist in the system already. Little or no attention is paid to how these “core theories” can be extracted from the knowledge engineer and how this commonsense knowledge should be organised before encoding of it takes place.

A number of other weak points in both methodologies emerged during the current phase of the project. In particular, neither methodology appears to provide a formal framework for the process of domain organisation; i.e. there are no guidelines about how the extracted knowledge should be analysed, and it emerged from our practical application of the methodology that frequent appeals need to be made to the intuition of the knowledge engineer. Hobbs (1984) does list some useful strategies for axiomatizing but we would like to see these incorporated into a more rigorous general approach for organising the elicited facts into a general and consistent theory of the domain.

We also have a worry about the purely abductive nature of Hobbs’ axiomatization. It could be argued that domain organisation must be independent of the chosen inference engine and that tailoring a domain to the chosen inference engine should be a separate step after the domain model has been designed.

These comments and suggested corrections apply mainly to the phase of domain organisation. For the elicitation phase we remain with the original Hobbs framework, although it is a very general methodology which is potentially also resource-intensive. We will argue that there are other useful strategies that can be used for the elicitation of specific knowledge types. This will be described as a process of *specialisation*, a process which we will argue allows for more wide-ranging automatization and will therefore be less resource-intensive.

So the main lesson learnt so far in the project is that elicitation and organisation are of a very different nature.

In the elicitation phase the problem is how to extract information relevant to a particular domain. Here we presuppose that we have textual sources of information (a text corpus, written protocols, interviews with experts, etc) and a knowledge engineer as a source of commonsense knowledge.

The problems for which we want to develop a methodology comprise:

1. How do we determine which facts are relevant to the task?
2. What methods can be used for extracting them from the text data?

3. How do we extract presupposed information not mentioned in the texts?
4. What computer tools can be used for analysing a large text corpus?
5. What are the requirements for the corpus itself?
6. Which support materials are useful and/or necessary for the elicitation phase?

In the organisation phase or the conceptual analysis phase the question is how to analyse the extracted information in terms of the knowledge base to be built. The extracted facts need to be represented in a coherent way, bringing out mutual dependencies and constraints as well as general laws of the domain. The problems for which we want to develop tools and techniques in this phase comprise the following:

1. What are the general principles that underlie the knowledge base for a natural language understanding system? These principles determine the overall structure of the conceptual analysis and provide the operational framework for conceptual modelling.
2. What is the process of the text comprehension? In other words, how do we define the task the natural language understanding system is expected to perform? For example, what does "text comprehension" mean in a particular application?
3. What types of knowledge does the knowledge base have to contain for the natural language processing system to be able to perform the task?
4. What are the basic domain independent principles for organisation of each of the knowledge types? In other words, what are well-formed knowledge types.
5. How can these principles be applied for organising the elicited facts into conceptually coherent structures?
6. How can the conceptual structures be formally represented?

We will argue in this report that it is important to separate out the elicitation work from the conceptual modeling work, while at the same time showing how one can guide the other.

In the first paper of the report we provide the theoretical basis for the compositional design of domain models. Our approach consists of conceptualisation rather than axiomatisation, but this doesn't mean that the resulting structures cannot be represented in predicate logic. We also define a generic task of text comprehension. We then define levels of complexity for this task, and define knowledge types needed for the execution of this task at its different levels of complexity. Then we introduce general principles for the process of knowledge base organisation and define what a domain model is and how it can be formalised. The rest of the first paper is devoted to a more detailed yet informal exploration of the knowledge types.

The second paper in this report is concerned with the elicitation phase. Given the abstract specification of a domain model, a natural language understanding task, and knowledge types appropriate to them, the paper attempts to arrive at an understanding of how information presupposed in the texts can be extracted. As a working method for this, the paper suggests a combination of those items of the 3–step and 9–step methodologies which are relevant to the elicitation strategy. The paper reports on an application of this revised methodology to text corpora in three different domains.

Part 1:
Conceptual Modeling: Towards a Typology for Knowledge Organisation

Responsible: Andrei Mikheev and Marc Moens
HCRC Language Technology Group
University of Edinburgh

2 Conceptual modeling

Knowledge organisation or conceptual modeling is arguably one of the most important stages of knowledge engineering. Its aim is to interpret the vast amount of individual facts and rules gathered at the stage of knowledge elicitation and to organise these into a coherent model. The creation of such a model is by no means an easy task. Many of the steps and decision procedures that make up this process cannot be formalised and are guided instead by the knowledge engineer's intuitive expertise. But in spite of this informality, the overall framework for conceptual analysis *can* be given a formal characterization. This framework can then be used by knowledge engineers to guide them through the conceptual modeling phase. It even allows them to create different conceptual models for the same domain without creating inconsistencies.

The real world, or the part of it that the knowledge engineer is trying to model, typically does not consist of a finite number of clearly discernable facts. Nevertheless, the model the knowledge engineer needs to develop has to be finite and discrete. To eliminate this apparent contradiction a special utility is used, viz. the *concept*. A concept is an abstraction that selects features of importance for the task at hand and ignores task–irrelevant details and complexities. It represents an aspect of reality, a workable approximation. Concepts can be organised into complex conceptual structures. These structures are the basic material for conceptual modeling.

One can contrast two principles of ontological categorization (cf. Rosch 1977a, b):

- a concept aims at capturing as many features (properties) as possible shared by its instantiations; in other words it aims to be as specific as possible.
- a concept aims at capturing as many instances as possible in order to reduce the number of needed categories; in other words it aims to be as general as possible.

In this section we explore some techniques for representing the outside world in a discrete model. We determine some basic principles of knowledge organisation and suggest a notation for conceptual modeling. This investigation by and large remains at the conceptual and epistemological levels (Brachman

1979). Where natural language understanding tasks are concerned, there is also a particular relation between words and concepts, and throughout our investigation we will take seriously the problem of linguistic anchoredness.

2.1 The model

A conceptual model is a simplified approximation of a set of phenomena in the real world taking into account the particular task for which this model is needed. This set of phenomena is normally considered to form a coherent cluster or domain. The main problem with conceptual modeling is that it is extremely difficult to create a fully consistent model for a large domain. Most successful implementations of conceptual models are concerned with so-called *micro-world* models—i.e. models which encode knowledge about fairly tiny and very simplified subsets of the real world.

In conceptual modeling one can reuse the same concepts (or concept names) and relations (or relational names) in different domains. However, the organisation of concepts and relations is likely to be quite different for every single model thus constructed: different models can use the same words and concepts, but differences in the rules that apply to those concepts can make the micro-models inconsistent with one another. Nevertheless we will argue that it is possible to organise a complex model as a collection of several micro-models, some of them mutually inconsistent yet each of the micro-models being internally consistent. These micro-models all have complex patterns of relations and interaction with each other that determine the behaviour of the complex model itself.

When a natural language understanding system is used as the natural language interface for accessing a database, the domain model can be interpreted as a closed world. However, for many other tasks, natural language systems need to be equipped with open world models, where the model contains information that is true or false and the status of all other information that the system does not contain is considered as unknown. One could describe this kind of model as a "shallow" model: it only contains the information that is necessary for achieving the desired level of complexity—typically in natural language processing, the desired depth of understanding. The model may have semantic gaps that are irrelevant to the task.

It is possible in theory to extend an open-world model to a closed one, but in practice this requires a lot of effort and does not necessarily result in a dramatic increase in problem solving capacity.

Since the number of facts about almost any topic can be virtually limitless, it is necessary for successful domain modeling to first determine the following characteristics of the model:

Scope. The scope is correlated to the width of the domain of interest. Determining the scope of the model involves assessing how much of a domain needs to be encoded. Depending on this, a decision can be made on the size and the nature of subclusters in the model.

Granularity. The level of granularity of the model is correlated to the depth of knowledge required for representing the domain's concepts. Determining the required level of granularity involves looking for the most simple approximation or the highest level of abstraction appropriate to the task and to avoid unneeded detail.

Problem solving requirements. The required problem solving capabilities come from an analysis of the inferential operations the system will have to perform in performing the task at hand. In natural language understanding systems this comprises traditional reasoning capabilities but more importantly the capacity to use the model for linguistic processes such as resolution.

In the case of conceptual modeling for natural language understanding systems, the scope and the level of granularity of the domain are implicit in the source texts we are working from, as well as in the protocols of dialogues with experts. This means that conceptual modeling must start with a careful restructuring of the overall domain (and of its source texts) into subdomains, to then determine the level of abstraction for each of the subdomains. The result of these steps should be a structured organisation of the subdomains, both internally (a structuring of each subdomain) as well as externally (structuring the subdomains with respect to each other).

The third step, determining what problem solving capabilities are required, can lead to many different answers when doing knowledge engineering for natural language processing purposes. Apart from standard deductive technique, Hobbs (Hobbs et al. 1990) has argued for the pervasive use of abductive reasoning methods. Others have argued for other forms of defeasible reasoning (e.g. Mercer 1989). This suggests that the model should not be oriented towards a specific type of reasoning but should be constructed in such a way that it can be adapted to different inference engines.

2.2 The knowledge base

Amongst the properties of conceptualization are the following:

Internal interpretation. Any knowledge structure consists of two parts: *descriptive knowledge*, a description of the way the domain is organised, and *interpretation knowledge*, which interprets how the descriptive knowledge can be used for solving a particular task. One and the same instance of descriptive knowledge can have several interpretations and can serve towards solving different tasks. Descriptive knowledge is usually considered as a knowledge base whereas interpretation knowledge is usually considered as a machinery for knowledge processing.

Internal organisation. Descriptive knowledge constitutes the structure where concepts are linked to one another with a variety of relations. The semantics of a concept is determined by its attributes (properties); the semantics of a link is determined by its operational properties.

Compositionality. Building a structured knowledge organisation, and recursively embedding knowledge structures, can be done by means of the following techniques:

- **Abstraction.** This allows the knowledge engineer to organise individual concepts in types by having them share common properties.
- **Generalisation.** This allows the knowledge engineer to generalise from individual properties of conceptual types to common properties, and to construct complex type–supertype subsumption relations with multiple inheritance of properties.
- **Aggregation.** This allows the knowledge engineer to decompose complex concepts into their components. Each of these components can be decomposed further depending on the level of granularity.

Internal Dependencies. Concepts depend on each other in a number of different ways, e.g. functionally or causally. Knowledge about dependencies constitutes so-called dynamic knowledge, where change of one conceptual entity causes changes to other ones. There are special conceptual types for representing this type of of dependency, viz. *actors* and *rules*. Actors are active elements that change other elements of the knowledge base or create new ones. Rules describe the semantics of a transformation.

To summarize a Knowledge Base is a collection of concepts, conceptual relations and transformations that are organized in structural clusters by means of abstraction, generalisation and aggregation. Such a collection is used by different interpretation procedures for solving different tasks.

One of the basic problems in conceptual analysis is how to represent any given item. In many knowledge engineering efforts this problem has not seen tackled in a systematic way. The result of this is that the same knowledge unit is sometimes represented as a concept in one system and as a relation in another. In the field of expert systems where each implementation has its own unique and self-contained task to perform, this does not matter too much. But for natural language systems there must be a more systematic mapping from linguistic items into conceptual ones (concepts and relations) since the tasks performed by natural language systems are more general across implementations.

In order to tackle this problem, one could suggest the following principle: *a concept corresponds to any salient phenomenon of the world that is being modeled.* This means that not only objects are concepts, but also events, attributes, propositions, states and whatever else exists in the world. So apart from nouns, which are usually mapped into concepts anyway, verbs, adjectives and adverbs, which usually correspond to relations, would also be mapped into concepts. For example, for the adjective "red" we could have the conceptual type `red(x)`, where x stands for any instance of redness, instead of the relation `red(y)`, where y stands for an instance of an object. To represent the fact that the object y is red, we can use a relation like "characteristic" (or `char`): `red(x) & object(y) & char(x,y)`. For eventualities, something similar can be done, adopting a Davidsonian approach. One of the notations derived from this approach is Hobbs' ontological promiscuity notation (Hobbs 1985b). A further extension of the approach is also possible to include a subatomic event semantics (cf. Parsons 1990). Finally, it is important to note that each knowledge cluster, proposition, fact etc., itself has the status of a concept and thus can be used as a single unit in other knowledge structures.

To distinguish concepts from relations we can use Quine's adage, "To be is to be the value of a variable". This means that we can quantify over a concept and refer to its instances by means of a variable. In this vein, Sowa suggests the following tests for determining whether t is of type "concept":

- Can you say "There exists an instance of t" ?
- Can you count instances of t or measure some amount of t?
- Can you refer back to an instance of t?

Relations, on the other hand, correspond to the organisational properties of concepts. Some of these properties can be represented in natural language by means of prepositions, case endings, word order variations, etc. They are often referred to as *thematic relations* or roles. Another kind of relation corresponds to the knowledge organisation principles of abstraction, generalisation and aggregation. And there are still other kinds of relations, for example logical, causal, temporal etc. The initial set of relations to be used in the model must be chosen very carefully, and for every relation a semantics must be provided.

Taking this approach we can achieve a more direct mapping between natural language and logic. However such a representation is likely to be more detailed and complex than more traditional representations where relations are chosen in an *ad hoc* manner. If we want to maintain the more direct mappings but at the same time avoid these complexities, it is possible to define macro-relations which can be used as shorthand of entire knowledge structures. These macro-relations are defined in terms of the initial set of relations and can be expanded into appropriate knowledge clusters.

2.3 Conceptual language

It has often been argued (e.g. Lenat & Brown 1984) that the choice of representation formalism is of critical importance for getting useful results in the knowledge discovery process. It is quite common in knowledge engineering to choose a conceptual representation scheme which is very close to the target knowledge representation language.

This has the advantage that one can tailor the conceptual scheme taking into account the knowledge representation's own strengths and weaknesses. However, having closely tailored conceptual and knowledge representation schemes can also lead to an undesirable bias.

To avoid this bias, it is important to maintain the conceptual modeling phase as application independent as possible. However, this creates a need for a conceptual notation which is independent of the knowledge representation language. And extra effort will be required to adjust the conceptual notation to the real system.

One such application independent notation is the predicate calculus and its various extensions—such as many sorted extensions, ontologically promiscuous extensions, etc. However, it has been argued that this notation scheme does not map straightforwardly onto natural language, and that complex formulae in this notation are not very readable. For experienced computational linguists or knowledge engineers, this may not pose a real problem. But we are aiming for a situation where the computational linguist or knowledge engineer tries to construct a knowledge base for natural language processing purposes, with the help of an expert. It is desirable to be able to do this domain modeling in a notation which is easily understood by the knowledge engineer *as well as* by the domain expert. Normal predicate calculus notation could be argued not to have this property.

Graph notation for knowledge engineering has been advocated by many authors (e.g. Buzan 1974, Nowak & Gowin 1983) as particularly helpful in this regard. Unlike traditional notation schemes, graphs allow a change of focus at any time: any node in a graph can be considered as a hook for more graph material. The ability of graphs to grow from almost any point also closely corresponds to the free association techniques often used in the knowledge elicitation process, thus adding ease of encoding in that respect also.

When adopting a graph notation as a conceptual formalism for knowledge engineering for natural language processing purposes it is necessary to ensure

- that the expressive power of the graph notation is similar to that of some predicate calculus (and that either can be translated into the other);
- that it has a formally developed syntax and semantics;

- that it allows for a systematic mapping to and from natural language;
- that it is in a readable form with a close correspondence to natural language constructions.

Such a conceptual language can then be viewed as an intermediate representation between natural language and formal logic. Put differently, the conceptual language can be considered as a high level language with the predicate calculus as an assembler.

The Conceptual Graph (CG) formalism Sowa developed seems to fulfill these characteristics. It allows for a systematic translation from and to natural language; it is more readable or at least has a less steep learning curve than formal logic; and it has a formal translation to predicate calculus. The CG formalism embodies a logical system, based on the rules of inference developed by Peirce for existential graphs. CG can also be extended to other logics such as modal logic, temporal logic, etc. This means that CG can serve both as an intermediate conceptual language for translation to the other formalisms and as final knowledge representation language. The former property can be used for quick system prototyping.

In the remainder of this paper we will be using CG notation, but this does not mean that we want to argue that the knowledge engineering process can only be carried out by means of this formalism. We do want to claim that CG provides a useful notation formalism for the conceptual modeling aspect of knowledge engineering. But it is just that, a notation.

3 Types of knowledge

In this section we investigate different knowledge types that can be used by a system during a natural language understanding task. The approach we take here is to first determine a generic task specification for a natural language understanding system, to break this down into different levels of complexity, and to find out what extra-linguistic knowledge can contribute to these tasks.

The main goal of natural language understanding systems is to extract conceptual content of natural language statements in order to derive episodic or discourse structure that represents structural, causal, temporal, spatial, logical and other relations between domain concepts and propositions. The resulting interpretation can then be accessed and searched during the question answering process.

To perform this mapping of natural language expressions to and from conceptual knowledge, two kinds of knowledge are needed in the knowledge base of the natural language understanding system: *linguistic knowledge* and *conceptual knowledge*. Both of these have an interpretational component for mapping to one another.

But one can also distinguish different levels of complexity in the text comprehension task. Reaching each level involves performing more reasoning, and sometimes reasoning of a different kind. In what follows, we represent all these levels in a modular way, but in a real life application a model may be preferred which integrates all the relevant knowledge (including linguistic) and allows simultaneous access of the various knowledge sources.

1. **Linguistic-Semantic level**
   The first task of an NLU system is the transformation of linguistic content of a sentence to its conceptual representation. Morphology and grammar are used to parse a sentence and to create its formal syntactic structure. The mapping of the syntactic structure into the conceptual one assumes the existence of a *type hierarchy* based on abstraction and generalisation.

   There are different kinds of ambiguity at the level of linguistic processing which cannot be resolved without accessing knowledge about semantic compatibility of the concepts involved. For every concept a *canonical structure* must be defined which determines how different concept types fit together. This canonical structure imposes not only semantic constraints but also determines conceptual relations for type coexistence in one structure.

   At this level of complexity the system is capable of resolving various linguistic phenomena by means of contributions from the morphology component, preparsing strategies, the grammar, the conceptual taxonomy and canonical structures. However, at this level only explicitly mentioned information is being extracted.

2. **Static Pragmatic level**
At the second level of complexity the system enriches the explicitly defined semantic contents that was created at the first level by adding background knowledge. This background knowledge is based on the aggregation of different conceptual types into coherent clusters which are used to resolve semantic gaps. It can also contribute to the resolution of definite expressions, certain anaphoric references, prepositional attachment ambiguities, metonymy, ellipsis, etc. For example, in language a concept can be referred to by one of its components or the component is referred by means of its aggregate. Background knowledge enables the system to detect and perform so-called conceptual shifts to resolve such expressions. Also, the activation of background knowledge may allow the system to create a list of expected words, related to the current one. This list can be considered as a more advanced version of semantic clusters at the preparsing level and its predictive power may in some instances simplify the task of the parser.

   One can distinguish four types of static background knowledge:

   Definition. This knowledge type is akin to Aristotelian concepts: define a concept by means of its necessary and sufficient conditions. Thus a concept can be defined by means of its *genus* (its supertype) and its *differentia* (its properties that distinguish it from its genus). This approach allows one to decompose concepts into primitives and to represent an internal view of the concept. However, in many cases it is impossible to provide such a definition for a concept.

   Schemata. These represent an external view of the concept—i.e. how the concept is used and coexists with other concepts, and which knowledge structures it takes part in. This is a Wittgensteinian view of concepts: since for many concepts one cannot provide a formal definition by stating its necessary and sufficient conditions, it is sometimes possible to determine the knowledge cluster this concept is used in and to define the concept by describing its usage. One concept can have more than one schema, allowing the encoding of multiple views of the concept.

   Prototype. A prototype is a representation of a typical individual of a class. A prototype definition can be arrived at by specialising the schemata and assigning default values for certain characteristics of it. A prototype allows one to infer presupposed individual characteristics that were not mentioned explicitly.

Constraints. These are laws which, for example, determine what the possible quantitative or qualitative characteristics are of a given concept.

Apart from a more precise treatment of the linguistic data by means of conceptual shifts and word expectations, it is also possible at this level to infer implicit information by using static background knowledge encoded in definitions, schemata, prototypes and constraints.

3. **Dynamic Pragmatic level**
At the third level of complexity the system is equipped with *dynamic knowledge*. This is knowledge which allows the system to generate new information from explicitly information by using laws of dependencies within the domain. By applying these laws to the conceptual structure the system can infer new information that will fill the semantic gaps in its representation. Usually dynamic knowledge exists in two forms: production rules that determine different sorts of inference, and actors that activate corresponding production rules for knowledge base transformations. One can distinguish at least three types of dependency:

   - *conceptual dependencies* relate to laws which allow one to infer a new peace of information from the existing one. This inference can be organised in different ways. The most common way is to implement a logical inference machinery that allows the chaining of implications to deduce the new knowledge. In the work of Schank and Hobbs a different strategy is used, which allows conclusions that are not necessary logically valid. Such knowledge is usually represented by means of rules which determine which implications follow from certain explicitly stored data.
   - *functional dependencies* allow to calculate values of some characteristics that depend on explicitly mentioned information if they are linked by functional dependencies. To perform this sort of calculation actors are usually defined as a procedural call.
   - *dynamic correction* is used for complex simulation models that change their states cyclically in time. This kind of knowledge checks for the current state and performs an update whenever it

is necessary. However, for the purpose of language comprehension this depth of knowledge is usually not needed.

At this level of complexity it is possible to organise the comprehension of a complex discourse with implicit connections and presuppositions between sentences and as well as answering direct and indirect questions such a system would also be capable of providing explanations and of handling expectations that don't materialize.

Every knowledge type that was mentioned above must have special interpretational strategies that use this knowledge for problem solving. These strategies usually are part of the knowledge representation machinery and are provided to a knowledge engineer. However, to avoid implementation bias, the knowledge engineer should annotate each type of knowledge with a specification of how that piece of knowledge should be used for the task at hand.

4 The taxonomy

A knowledge taxonomy is the result of structuring knowledge in accordance with the rule of knowledge compositionality (i.e. abstraction, generalisation and aggregation). The first main kind of taxonomy is the *type hierarchy*, the result of knowledge abstraction and generalisation. Another type of taxonomy is one that encodes *mereological relations*, the result of knowledge aggregation.

4.1 Type hierarchy

A type hierarchy is defined by means of the IS-A link. It constitutes a partial order graph whose properties are reflexive, antisymmetric and transitive. The type hierarchy is constructed from a set of properties describing things in the world and value sets for these properties. Each type is an unique set of properties with assigned values.

However this IS-A relation in fact is overloaded by two relations: abstraction (INDIVIDUAL < TYPE) and generalisation (TYPE-1 < TYPE-2). Both of them have slightly different inferential features. Also, the relation (TYPE < INDIVIDUAL) is always considered ill-typed.

As was mentioned before, successful domain modeling is not possible for the world as it is. Instead, a world model could be considered as a collection of microworld models that co-exist by sharing the basic principles of their organisation. This assumption implies that every domain needs its own type hierarchy which may or may not be compatible with another domain. However two hierarchies must be compatible if they correspond to hierarchically related domains.

The main criteria for the type hierarchy are how useful the properties are for modeling the domain. It will not necessarily form a discriminational tree structure but rather a complex multiple inheritance structure with several main types at the top.

Generalizing the ontological study for conceptual types (Guarino 1991) we will distinguish three main types of concepts: entities, attributives and roles. Entities and attributives constitute the natural type. **Natural types** exist as such. They are semantically rigid, i.e. they don't depend on any particular situative framework. For instance, the concept types HUMAN, CAR, MOVE, RED, NUMBER OR TIME-POINT are natural. **Role types** are semantically non-rigid and exist only in the framework of some domain. For instance, the type DOCTOR exists only in the framework of professional relations. The notion of *foundedness* allows one to distinguish between two types of natural concepts. **Entities** are essentially independent and can be instantiated. **Attributives** are founded on entities and can be recognized only in presence of some entity. For example, the type COLOR is an attributive since one cannot point to a color without pointing to an object. Note that both COLOR and OBJECT are semantically rigid because they remain themselves in any possible situation. Roles are always founded on natural types. For example, the role type DOCTOR is founded on the natural type HUMAN because one cannot point to a doctor without pointing to a human. If a role is founded on an entity it inherits all the properties of the entity and will be called entity-role. Roles for attributives will be called attributive roles.

Natural types constitute an ontology that tends to be domain independent. Role types are domain dependent and usually constitute a lattice or arbitrary partial ordered graph.

Since the notion of type is very close to the notion of set, set theoretic operations like conjunction, disjunction and difference can be used for the dynamic creation of new types from existing ones. However, there is also sufficient difference between a type and a set. Two sets are considered equal if they consist of the same elements, but for types this is not true: extensional equivalence does not imply an intensional one.

There is one important thing to note about the conceptual taxonomy. It must have a mapping to the linguistic categories but it is purely semantic. This means that the knowledge engineer must avoid linguistic bias in the taxonomy and the properties for a type characterisation must be derived from the conceptual analysis rather then from the linguistic one. Word sense is represented by mapping lexical items to ontological concepts. Each type should be considered from the semantic point of view and in some cases it can have straight linguistic concordance but in many cases it does not.

4.2 Aggregation

Mereological relations represent the aggregation of different entities or their roles into a composite type. Together with IS-A links, these aggregative relations are amongst the most important sources of inference in the process of text comprehension. These relations allow us to decompose complex concepts to their components and define composite entities with other entities as components. A composite concept is an aggregative of the components that are linked to each other with the conceptual relations and constitute a knowledge cluster. Plural denotation also is a kind of aggregation of individuals into a set.

There are several mereological relations that are usually considered as PART-WHOLE relations. In fact logical and inferential properties of this relation depend on the types of the concepts it links. For example, property inheritance from the part to the whole and from the whole to the part can be very type dependent.

Examples of different types of the mereological relations include:

- Component—Integral Object
- Portion—Mass
- Member—Collection
- Stuff—Object
- Place—Area
- Activity—Component
- ...

In any single case careful mereological analysis must be done in order to determine the inferential properties and the exact type of mereology. Aggregation is a basic technique for constructing canonical structures, definitions, schemata, prototypes and other types of knowledge. However not all the links between concepts should be considered as aggregative. Aggregation is a connection between conceptual types that can be instantiated, i.e. entities and their roles. In this sense attributive types are not aggregated but rather attributed to a type.

5 Canonical structures

A canonical structure determines semantically valid type coexistence for a given type. It shows selectional constrains for permissible type combinations expected for the type and relations for the permissible combination. Usually these structures are based on deep semantic case relations between types. This allows to use the structures for checking semantic consistency of the parser output and cut inconsistent branches of the parsing in dynamic.

Ancestors of the canonical structures are Tesniere's dependency grammar, Fillmore's case grammar, Katz and Fodor's semantic constraints and Allerton's valency theory. Canonical structures add further information about the semantic roles that each argument plays. For example, thematic roles of a subject and an object can play different semantic roles in the verb "like" where subject is an experiencer and in the verb "use" where the subject is an agent.

Canonical structures are more general then binary features of simple case frames. They can handle:

- complex interconnections between events by allowing them to be interconnected in one canonical structure;
- disambiguate noun phrases or other stable patterns of word collocations;
- resolve long-distance dependencies;
- contain nested contexts for embedded propositions, time and modality.

Canonical structures are based on the type hierarchy and are inheritable from a type to its subtype with possible specialization of the structure.

Not all the conceptual types have rich canonical structures. The richest ones can be provided for events (verbs), attributives (adjectives) and role types. Canonical structures for natural objects usually consist from a single conceptual node. The following example shows the canonical structure for the event ADMIT in the hospital domain:

[ADMIT : every]-

$-(agnt)- > [DOCTOR]$

$-(subj)- > [PATIENT : x]$

$-(to)- > [HOSPITAL]$

$-(from)- > [HOSPITAL]$

$-(reas)- > [HEALTH - PROBLEM]$

$-(purp)- > [SITUATION : [PATIENT : *x] < -(ptnt) - [UNDERGO] - (subj)- > [HOSPITAL - TREATMENT]]$

$-(when)- > [DATE]$

All types that are used in the canonical structure must be represented in the type hierarchy. For example, both DOCTOR and PATIENT are role types of the type HUMAN. Nominalized verbs inherit the canonical structures from the verbs they are derived from. So the noun "admittance" will have the same structure as above. Note also complex attachement of the purpose for the structure. The variable *x represents the cross-reference between two nodes, i.e. the same instance of the type PATIENT.

However not only events can have canonical structures:

$[EASY : every]$

$< -(attr) - [OBJECT] < -(ptnt) - [ACT : x]$

$< -(manr) - [ACT : *x]$

This says that every concept of the type EASY can be linked either to some object by the attribute link or to some act by the manner link and the object is a patient of the act. The variable *x represents the cross-reference between the two nodes. This structure allows us to recognize ill-formed usages of the words. For example: *John is easy to do the homework* - is wrong because John stands in the agent relation instead of the valid patient one.

The canonical structure for "eager" could be given as

$[EAGER : every]$

$< -(attr) - [ANIMATE] < -(agnt) - [ACT : x]$

$< -(manr) - [ACT : *x]$

Using this canonical structure it is possible to recognize ill-formed usage of "eager" in : *The homework is eager for John to do.* In this example John doesn't stand in the valid agent relation.

Prepositions can also be provided with canonical structures. For instance, the preposition "with" can be mapped onto at least two different conceptual relations:

1) $[ACT] - (instrument) - > [notANIMATE]$

2) $[OBJECT] - (accompaniment) - > [OBJECT]$

A pronoun like "he" can be represented by means of a structure like the following:

[MALE : #] (# - means that a reference has to be resolved to the previously introduced information).

For objects we can provide following structure:

$[OBJECT : every]$

$< -(arg) - [EVENT]$

$-(char)- > [ATTRIBUTE]$

$-(stat)- > [STATE]$

This canonical structure allows one to recognize possible combinations of different types with the OBJECT type. (The relation "argument" is a supertype for relations like agent, patient, etc.)

Canonical structures for role types usually include the framework the role is defined in:

$[DOCTOR : every]$

$< -(agnt) - [CURE]$

$-(rcpt)- > [ANIMATE]$

$-(ptnt)- > [ILLNESS]$

This structure allows to bind the type DOCTOR with types or subtypes of CURE, ILLNESS and animate objects when they are used in the same lexical scope.

In resolving a compound noun phrase the canonical structures for every word in it are matched for a maximal join. There can be four cases:

- A canonical structure is found only for the head noun. In this case using type constraints the modifier is joined.
- A canonical structure was found only for the modifier. In this case the modifier usually specifies the role for the head if the modifier is of the object type. If the modifier is of the event or attributive type it has a reach canonical structure that can absorb the head noun.
- A canonical structure was found for both. In this case using type constraints both of the structures are tried for the maximal join.
- No canonical structures were found for either of them. This is the most ambiguous case where no decision can be made about how to join the concepts.

The canonical structures sometimes intersect with type definitions but generally they are not as detailed and represent the implicit patterns of relationships necessary for semantically well-formed sentences. However people often use metaphorical references and metonymy that violate semantic constraints represented in a canonical structure. There are several approaches for resolving this kind of ambiguity. For instance, a “conceptual shift” allows one to subsume an integral concept by one of its components. This knowledge can be provided by definitions and schemata.

6 Definitions

Type definition is an Aristotelian approach to represent a conceptual type by the set of properties that determine the necessary and sufficient conditions for a concept to belong to the type—the concept's *essence.* In this case, the concept inherits all the properties from its supertypes and will typically also have some unique properties that distinguish it from all other conceptual types with the same supertypes. A definition must always be true because if it were false for some object the object wouldn't belong to that type. Providing the definition of a type makes it possible to decompose the type into primitive types and their properties.

The possible meanings of a word are obviously domain dependent. It may in some cases be possible, however, to distinguish the most abstract meaning and to organize the domain-dependent meanings as specialisations of the main one. In this sense, the main meaning could be considered as a canonical structure and the adjusted version can be considered as a subtype of the main abstract type. During sentence processing the main meaning can be used first and when further information is obtained, this meaning can be replaced by its specialized subtype.

Here we will suggest a way of providing definitions in lambda calculus style, where a type definition is an assertion that some type label is equivalent to a particular lambda abstraction. It is important to note that conceptual relations can be defined in the same way.

There are at least four ways for providing definitions for new types:

1. by assigning particular values to the features of the supertype. For example:

type [MAN : every x] is $[HUMAN : *x] - (char)- > [SEX] - (val)- > [MALE]$

This says that for x to be of the type MAN is to be of the type HUMAN with fixed value of the characteristic sex and this value must be of type MALE. Note that MAN was quantified universally and HUMAN existentially; this allows one to say that every man is a human.

2. by providing aggregative differentia to the type. For example, the definition for the type HOSPITAL shows differentia through the aggregation of the conceptual types MEDICAL TREATMENT and PATIENTS with it:

   type [HOSPITAL: every x] is

   $[BUSINESS - ESTABL : *x] < -(loc) - [MED - TREATMENT] - (ptnt)- > [PATIENTS : \{*\}]$

   This says that hospital is a business establishment where patients are treated. Plural denotation for patients is represented by {*}.

3. by describing how it can be constructed from primitive types. Types defined in this way may not have a supertype. For example:

   [INTERVAL: every x]-

   $-(char)- > [POINT : y]$

   $-(char)- > [POINT : z]$

   $-(meas)- > [MEAS - UNIT] - (val)- > [NUMBER : n]$

   $[POINT : *z] - (arg - 1)- >< diff > -(rslt)- > [NUMBER : *n]$

   $[POINT : *y] - (arg - 2)- >$

   This definition determines the aggregative construction with complex relations inside it. This construction is called an interval. For the interval definition we used two primitive types—POINT and NUMBER. We can slightly adjust the definition by defining role types UP-POINT and LOW-POINT of type POINT and use them respectfully. Here also we used an *actor* “diff” for measuring difference in measure units between the points. Note that in this case the type INTERVAL does not have any supertype.

4. by describing laws the type is to obey. For example, the type definition for triangle is “three points that are not on the same line”. To represent this we designed the constraint rule that checks values of intervals between the points.

   [TRIANGLE: every x]-
   $-(char)- > [POINT : y]$
   $-(char)- > [POINT : d]$
   $-(char)- > [POINT : z]$

[INTERVAL: i1]-
$-(char)- > [POINT : *y]$
$-(char)- > [POINT : *d]$
$-(meas)- > [MEAS - UNIT] - (val)- > [NUMBER : i1v]$

[INTERVAL: i2]-
$-(char)- > [POINT : *y]$
$-(char)- > [POINT : *z]$
$-(meas)- > [MEAS - UNIT] - (val)- > [NUMBER : i2v]$

[INTERVAL: i3]-
$-(char)- > [POINT : *z]$
$-(char)- > [POINT : *d]$
$-(meas)- > [MEAS - UNIT] - (val)- > [NUMBER : i3v]$

not $[[NUMBER : *i1v] - (arg - 1)- >< diff > -(rslt)- > [NUMBER] < -(=)- > [NUMBER : *i3v]]$

$[NUMBER : *i2v] - (arg - 2) - -$

Role types are usually defined by specializing attributes to some framework. For example, role type AGE is founded on the type TIME-INTERVAL which is a specialization of the type INTERVAL.

type [TIME-INTERVAL: every x] is

$[INTERVAL : *x]-$

$-(char)- > [LOW - POINT] - (kind)- > [DATE : y]$

$-(char)- > [UP - PNT] - (kind)- > [DATE : z]$

$-(meas)- > [TIME - MEAS - UNIT] - (val)- > [TIME - VALUE : n]$

$[DATE : *z] - (arg - 1)- >< minus > -(rslt)- > [DATE : *n]$

$[DATE : *y] - (arg - 2) - -$

This specialization is represented by using time-dependent types for the points and the measure units. The value is calculated by the actor "minus" instead of more general actor "diff".

Since type AGE is a role it is defined in respect with the appropriate situative framework:

type [AGE : every x] is

$[TIME - INTERVAL : *x]-$

$-(char)- > [LOW - POINT] - (kind)- > [DATE] < -(when) - [BIRTH] - (ptnt)- > [ANIMATE]$

$-(char)- > [UP - PNT] - (kind)- > [DATE - OF - SPCH]$

$-(meas)- > [MEAS - UNIT] - (val)- > [TIME - VALUE]$

This says that the type AGE is a time interval where the low-point is the date of birth of some animate creature and the up-point is a date of the utterance. Procedure of calculating the exact value of the age is inherited from its supertype.

We can also define conceptual relation age(x,y) in terms of the conceptual type AGE:

(age: every y)(z, v) is

$[AGE : x]-(char)- > ...- > [ANIMATE : *z]-(link : *y)- > [TIME-VALUE : *v] < -(val)-...-[AGE : *x]$

This definition says that relation age is a link between elements of the same instance of the concept AGE (reference x - *x) and these elements are the animate object and the value of the AGE concept.

Both definitions for the concept AGE and the relation "age" allow one to expand initial representation in terms of primitives used in the definitions or contract the representation by hiding low-level primitives and using types themselves.

Unfortunately in many cases it is not possible to provide a definition for a type by stating the necessary and sufficient conditions for it. In this case the type can be described by its schema.

7 Functional dependencies

Functional dependencies represent a connection among values of the elements of a type. These dependencies can be represented by a special type of relation, viz. the **actor**. An actor represents some law that can be stated either procedurally by attachment to an external procedure or declaratively by table look-up. A specific feature of actors is that they are active and are automatically activated when any of their functionally dependent values is modified. They recalculate the other dependent values. An example of an actor can be found in definition of the type AGE above.

The actor *minus* allows one to calculate a person's age from his/her date of birth and the date of utterance. It is also possible to provide actors that calculate the second or the third of these values from the two others. This usage of actors is called "if-added": if some of the functionally dependent value was modified, the actor activates itself.

Actors can also be used for finding the exact values for a concept characteristics. These are so-called *skolem functions* that can be defined whenever some variable depends on a universally quantified concept. For example:

$[DOCTOR: every] < -(benf) - [EARN] - (ptnt)- > [SALARY] - (val)- > [AMOUNT]$

This fact can be represented by skolem function:

$[DOCTOR: every] - (arg)- >< DOC - SALARY > -(rslt)- > [AMOUNT]$

Function DOC-SALARY retrieves the exact value of the salary for every particular doctor.

Skolem functions usually are defined if there are many individual facts or when the knowledge base is connected with a database. In this case a skolem function is just a query function for the database. This usage of actors is known as "if-needed", i.e. they are activated when we want to retrieve a value that is functionally defined.

8 Schemata, prototypes and constraints

If a concept definition can be provided it represents a narrow view of the concept by imposing obligatory conditions needed for making type distinctions. However, a concept can also be defined by describing all the things that are associated with. This is a **schema** for the concept. It represents a semantic cluster associated with the concept and forms a broad view of it by linking it with its *accidental* properties. While there is only one definition for a type, the number of schemata for the type can be arbitrarily large, even in the same domain, representing different points of view. The amount of detail and the scope of the schema is highly dependent on the task.

For example, a possible schema for the type HOSPITAL could look as follows:

[HOSPITAL: every]-

$-(part)- > [WARD: \{*\}]$

$-(part)- > [DEPARTMENT: \{*\}]$

$< -(loc) - [WORK] - (agnt)- > [STAFF]$

$-(char)- > [ADDRESS]$

$-(char)- > [SPECIALISATION]$

$-(char)- > [CHARGE - RATE]$

This schema represent the following facts:

Every hospital has wards. It consists of different departments. It is a place where hospital staff works. It has an address, a specialisation and a charge-rate.

Apart from the expansion of the concept of the type HOSPITAL into a complex structure, this schema also allows to resolve metonymy when the type HOSPITAL is used instead of its elements. For example:

This hospital is more expensive than another one.
(hospital is used instead of the concept CHARGE RATE)
The whole hospital voted against the new legislation.
(hospital is used instead of the concept STAFF).

The schema shows the task-dependent aspectual association of the type HOSPITAL. Having another task we would represented the type HOSPITAL by another set of associates or enlarged the current association. Usually if both the definition and the schema are provided for a type, the definition is included into the schema.

In case when there is no definition for a concept, the schema is supplied with the supertype (if any) but the differentia is not provided at all. Instead of the differentia there is a semantic cluster that can be applied to the family of the concepts that constitute the type in accordance with the Wittgensteinian principle of family resemblance.

For example, we cannot provide the differentia between the type WORK and its supertype ACTIVITY because we cannot find any property that is obligatory for an activity to be of the type WORK. But we can define loose association of properties of different activities we call work and associate them into a schema.

WORK < ACTIVITY

$[WORK]-$

$-(agnt)- > [ANIMATE]$

$-(loc) - [PLACE]$

$-(duration)- > [TIME - INTERVAL]$

$-(purp)- > []$

$-(rules)- > []$

For every specific kind of work we will define the purpose, the rules, etc.

Schemata allow us to describe a concept not in the terms of primitives but rather in the terms of the entire framework where different concepts are defined in the terms of each other. The following example defines the role type PATIENT in the framework of medicine:

```
type [PATIENT : every x] is
[MED − TREATMENT] -
−(ptnt)− > [HUMAN : ∗x] − (char)− > [HEALTH − STATE] − (meas)− > [HL − UNITS] − (val)− >
[VAL : z]
−(agnt)− > [DOCTOR : y]
−(purp)− > [SITUATION :
    [HUMAN : ∗x] − (char)− > [HEALTH − STATE] − (meas)− > [HL − UNITS] − (val)− > [VAL :
∗v] < −(>)− > [VAL : ∗z]
            ]
−(char)− > [AGE]
−(char) − [SEX]
```

This says that every patient is a human who underwent some medical treatment with the purpose of increasing his or her health-state. The type doctor can be defined in the same way in terms of the types MED-TREATMENT and PATIENT.

Events have more complex schemata that are called episodes. Episodes will be explored in the following section.

Prototypes represent typical instances of conceptual types. Prototypes allow us to enrich the semantic representation with the background knowledge about typical individuals of a given type by assigning *default values* to the elements of the schema. These values may be or may not be true in every particular case and can be modified in accordance with the particular case by changing default values for explicitly mentioned ones.

For example, in the schema for the type WORK we can assign the default values for the duration of the work, its purpose and its place :

```
[WORK]−
−(agnt)− > [ANIMATE : x]
−(loc)− > [PLACE] − default : − > (kind)− > [BUSINESS − ESTABL]


−(duration)− > [TIME − INTERVAL] − default : − > (meas)− > [HOUR] − (val)− > [8]
−(purp)− > default : − > [SITUATION : [ANIMATE : ∗x] < −(ptnt) − [EARN] − (obj)− > [SALARY]]
```

To create a prototype for the type PATIENT we can assign default values to the health state, the age, the sex etc. For example most patients in certain wards of a hospital may be over a particular age.

Constraints are general principles that must be true for the instances of the type including its internal structure and external relationships. Like definitions, constraints must always be true. But the reason is different. If constraints were false this would mean some law was violated or the thing could not be used for its normal purpose. For example, constraints can determine mutually disjoint concepts that cannot co-exist in one schema, or constraints can be applied for a number of the same links for a type. Constraints represent the state of affairs in the particular world and can be different for another possible world.

Constraints can be defined either in the schema itself or by IF-THEN rules the schema is supplemented with. We can add constraint for the schema of the type PATIENT that human age must be in range from 0 up to 110 years. If this constraint is a general constraint for a human's age then it is better to encode it in the HUMAN-AGE schema directly:

IF $[HUMAN - AGE] - (meas)- > [YEARS] - (val)- > [NUMBER : x]$

THEN $[NUMBER : *x] - (>=)- >$ [0] and $[NUMBER : *x] - (<)- >$ [150]

Every relation in a schema includes an implicit constraint for being the only possible or allowing several of them. The general case is that when we want to represent that there can be more than one relation of certain type we use plural denotation. For example if we want to represent the fact that a car cannot have less than three wheels we will write in a schema for the type CAR:

[CAR: every]-

$-(part)- > [WHEEL : \{*\}] - (quant)- > [NUMBER] - (>)- >$ [2]

$< -default : -(=)- >$ [4]

Another example of a constraint defines disjoint sets of men and women:

IF $[HUMAN : *x] - (char)- > [SEX] - (val)- > [MALE]$

THEN $not[[HUMAN : *x] - (char)- > [SEX] - (val)- > [FEMALE]]$

Frame languages usually are combined with logical languages for the constraints representation. We will use logical operators for augmenting conceptual graph notation.

9 Conceptual dependencies and episodes

Conceptual dependencies allow us to reconstruct implications and presuppositions from explicitly mentioned information. They consist of two parts: *conceptual relations* and *production rules* for the inference generation based on these relations. When a new piece of information turns out to have a conceptual dependent structure within the knowledge base, the production rules are to be activated to perform the inference and to generate the implied information. Since conceptual dependencies constitute the dynamic part of a knowledge base they usually correspond to events.

An important difference between logical inferences and conceptual inferences is that conceptual inferences are generated not on the demand of a user but from the newly acquired information. Another difference is that conceptual dependencies are not logical relations and conceptual inferences do not necessarily constitute logically valid deductions.

There are different types of conceptual inferences. An initial representation of an event can be enriched with a knowledge structure that is presupposed to be a *purpose* of this event. A *result inference* can be made whenever an event is present and no information contradicts the inferred result. Inferences can be made for simulating *changes of states* of the participants of the event. A chain of the causally related events can be reconstructed having the resulting situation. Different production rules are activated to reconstruct the knowledge structures related by the same set of conceptual relations.

Production rules also include some control information about the circumstances under which the rule can be fired. The *scope* of the production determines the kind of inferences the production can be used for. The *conditions* impose constraints for the production to be used in every particular case. The *body* of the production is the rule itself. It consists of the IF-THEN part and can be supplied with the ELSE part. *Actions on the production realisation* determine what should be done after the production rule was fired. If the action is another production we have an algorithmic model. In that sense conceptual dependencies can be represented as actors that correspond to production rules. They are of the "if-added" type and are activated whenever certain information is explicitly added to the knowledge base.

An **episode** is a kind of schema that is built around a number of events for representing a temporal-causal chain of world state changes in time. Apart from including schemata for its events an episode also has an intention associated with it, called the **goal**, and the chain of events that will lead to the realisation of the goal, called the **plan**. There can be different plans for the realisation of the same goal, and the same chain of events can be used for the realisation of different goals. Episodes also include conceptual dependencies between different parts of schemata for the events and provide a basis for inferences that can be drawn from any point of the episode for reconstructing the preceding part of the episode or the following part. An important property of episodes is that they contain not all possible conceptual dependencies that can be drawn but rather some restricted set that is relevant to the episode's intention.

A schema for an event includes *actants* that participate in the event, a *script* or chain of subevents that are necessary or usually occur in the realisation of the event, information about the event's *purpose*, its *resulting changes* and any other conceptually relevant knowledge. Here we will use the terms episode and event schema interchangeably.

For example, an episode for an event like BUY needs to comprise a schema that can be invoked by passing actual instances to it. So this schema has a header and a body:

$[BUY : everyx](BUYER : y, SELLER : s, THING : z, PAYMENT : p, DATE : d, LOCATION : l)$

$[TRANSACTION : *x]$ -

$-(agnt)- > [BUYER : *y]$,

$-(ptnt)- > [THING : *z]$

$-(from)- > [SELLER : *s]$

$-(for)- > [PAYMENT : *p]$

$-(when)- > [DATE : *d]$

$-(where)- > [LOCATION : *l]$

$-(purp)- > [SITUATION : sit1 : [BUYER : *y] - (expr)- > [OWN] - (ptnt)- > [THING : *z]]$

$-(precondition)- > [SITUATION :$

$[BUYER : *y] - (expr)- > [OWN] - (ptnt)- > [PAYMENT : *p]$

$[SELLER : *s] - (expr)- > [OWN] - (ptnt)- > [THING : *z]$

$[BUYER : *y] - (expr)- > [WANT] - (ptnt)- > [SITUATION : *sit1]$

....

$]$

$- < nes - script > - > [SITUATION : sit2$

$[SELLER : *s] < -(ptnt) - [GIVE : g1] - (agnt)- > [BUYER : *y]$

$[PAYMENT : p] < -(obj)-$

$[SELLER : *s] < -(agnt) - [GIVE : g2] - (ptnt)- > [BUYER : *y]$

$[THING : *z] < -(obj)-$

$[GIVE : *g1] < -(elaborate)- > [GIVE : *g2]$

$]$

$- < plaus - script > -default :> [SITUATION :$

$[THING : *z] < -(ptnt)-[CHOOSE : ch]-(agnt)- > [BUYER : *y]$

$[SELLER : *s] < -(ptnt) - [ASK : a] - (agnt)- > [BUYER : *y]$

$[PAYMENT : *p] < -(about)-$

....

$[CHOOSE : *ch] - (then)- > [ASK : *a] - ......- > [SITUATION :$

$*sit2]$

$]$

$- < rslt > -default : - > [SITUATION :$

$[BUYER : *y] - (expr)- > [OWN] - (ptnt)- > [THING : *z]$

$[SELLER : *s] - (expr)- > [OWN] - (ptnt)- > [PAYMENT : *p]$

$not[[SELLER : *s] - (expr)- > [OWN] - (ptnt)- > [THING : *z]]$

$not[[BUYER : *y] - (expr)- > [OWN] - (ptnt)- > [PAYMENT : *p]$

$]$

$- < create > - > [SELL](BUYER : *y, SELLER : *s, THING : *z, PAYMENT : *p, DATE : *d, LOCATION :$

$*l)$

The event SELL is a different event. Buyer and seller have different purposes and they may also have different attitudes towards their respective actions. However the schema for SELL also has a lot in common with the schema for BUY. To represent the fact that every BUYing event is necessary a SELLing event we use the actor $<$ *create* $>$ that creates an instance of the selling event with the same participants that was passed to it as arguments.

The actors "result", "necessary script", "plausible script", "create", etc., perform a dynamic creation of corresponding knowledge structures. However, each of the actors has its own features for using. For example, the actors "create" and "necessary script" are fired without checking any conditions. The actor "result" on the other hand generates different resulting situations in accordance with explicitly mentioned information. A default resulting state is generated if there is no explicit information about other possible result states. The production rule that corresponds to the "result" actor can look in the following way:

```
Name: BUY-SELL result
Scope : BUY event; SELL event
Condition: apply if there is no explicitly mentioned result
Body:
        IF check: precondition-1 = TRUE OR precondition-1 = UNKNOWN
         ......................................
        THEN apply: default result
        ELSE ..............
Post actions: none
```

This production is to be applied whenever instances of the buying or the selling events are added to the discourse. Upon firing the rule the production checks if there is no explicitly mentioned result. The rule itself checks the preconditions and other available information and activates the corresponding branch.

To the BUY and the SELL schemata we can also add actors and productions for reconstructing the preconditions for these events. This is presupposed information which is usually not mentioned in the text.

The essential advantage of episodes over the static representation of events is that episodes allow us to produce context sensitive conceptual inferences. Moreover, there is a particular class of verbs—causative-inchoative verbs—that refer exactly to the resulting state without directly mentioning the caused event. Such verbs like "damage", "cure", etc., represent the final stage of some event or chain of events. In this case the resulting stage is predefined but the script of the exact chain of events that caused the state is unknown and can be only retrieved from the text. Careful analysis is needed for every verb in accordance with which parts of the episode are necessary presupposed which are plausible and which inferences can be generated.

The episode HOSPITALISATION is the central point in the domain of patient discharge summaries (provided by an Irish hospital group for application of our results). This episode defines the framework for arranging the information from the medical reports into a coherent, logically and conceptually related cluster. This episode includes the stable chain of events (script) that necessarily occurs: ADMITTANCE, CURE, DISCHARGE arranged in temporal order. The goal of the episode is to improve patient's health state. Each of the events has conceptual consequences that are necessary for the realisation of the next event. Production rules check these necessary conditions and perform the conceptual movement in the chain. They also determine what is to happen if a particular action fails. Each of the events can be decomposed for sub-goals and further sub-plans.

In certain cases an even deeper understanding can be provided by incorporating the knowledge about the agents' *motivations* and *emotions* into an episode. This knowledge allows one to reconstruct an account of why the goal has arisen and how it relates to the emotional state of the actor and what the motivation was of the actor to seek the fulfillment of the goal. There are different theories for simulating human emotions. Some of them, like Plot Units and the theory of affect (Dyer 1989 and Lehnert 1989), were implemented in natural language understanding systems and showed (or simulated) a rather deep level of comprehension. It also makes the system more robust in the face of incomplete or even ungrammatical input.

An episode is a structured way of representing corresponding activities and can be freely combined with other episodes into a plan for a more comprehensive episode. Some episodes don't have any chains of events for their realisation at the defined level of abstraction. Such episodes we will call elementary. However, any elementary episode can be provided with its script at another level of abstraction. This feature highlights the difference between the episodes on the one hand and the script and MOP theories (Schank and Abelson 1977, Schank 1982) on the other. Episodes don't require a set of global primitives for the concept to be decomposed but rather are adjusted to a particular level of the granularity required by the task.

Part 2
Elicitation: Typology for Information Contents

Responsible: Annelise Bech, Costanza Navarretta
CST
Copenhagen

10 A working method for knowledge elicitation

The process of knowledge elicitation concerns the extraction of relevant information from the texts. In this part of the document we will determine which information presupposed in the texts is relevant in an NLU system.

The nine-step method which we adopted at the end of WP 1 (Bech et.al. 1993) presupposes that the TACITUS commonsense knowledge base described in (Hobbs 1986, 1988) is available. Thus it does only give guidelines about how to organize and insert new information in this existing base (steps 5–9 of the method). Because we want to define a general method and because we cannot get the TACITUS commonsense knowledge base, we have redefined the steps of organizing a knowledge base (Part 1 of this report). We have also combined the first four steps of the nine-step method with the original Jerry Hobbs' three–step strategy (Hobbs 1984) that gives good guidelines for a first analysis of the texts and a first division into subdomains (clusters) of the background knowledge presupposed by the texts.

The resulting working method for knowledge elicitation from a text corpus which we have adopted is the following:

- Make an extensive list of the content words in the text corpus to be processed and an extensive list of general relevant facts about the text corpus and about the content words in it.
- Collect morphologically related words.
- Divide the resulting groups of morphologically related words into subdomains (”clusters”).
- Give a first organization of the knowledge in each subdomain.

Next for each content word (or for each group of morphologically related words) do:

- Look for all occurences of the word in the text corpus to see the contexts in which the word is used. When necessary, look at previous or following sentences to resolve anaphora.
- Reduce the citations to their predicate argument relations. Examine the contexts and ask what facts about the word are required to justify each of the occurrences of the word.

- Make a preliminary division of these predicate-argument relations into heaps, according to a first analysis of which predicates should go together. This first analysis is based i.a. on the knowledge enterer's linguistic knowledge and on his knowledge about the text corpus. Some patterns must be split up because more facts are presupposed in the citation.

- Give an abstract characterization of the facts about the word that justify each of the heaps. This is a matter of making explicit the first analysis that underlay the classification into categories in the previous step, in fact this and the previous step will often be done in tandem. Recognizing a more abstract characterization may lead one to join two heaps, and failure to find a single abstract characterization may lead one to split a heap.

11 Types of information presupposed by texts

In this section we will determine which kinds of background knowledge are necessary to understand texts. The strategy adopted to identify the information to be extracted from texts has been to apply the working method for knowledge elicitation described in the previous section to small text corpora belonging to different domains and annotate the background knowledge that is required by the texts. Some examples of the results of this analysis can be found in section 12.

Background knowledge comprises linguistic and extra-linguistic (world) knowledge. The discussion whether linguistic and extra-linguistic knowledge should be handled in distinct ways is still not resolved but the field of lexical semantics is actually having a revival. Because linguistic and extra-linguistic knowledge are strictly interrelated, we will not try to find a clear-cut separation line for the two kinds of information but we will, when possible, define the linguistic phenomena that indicate the presence of presupposed background knowledge. Then it will be possible to investigate regularities and dependencies between the two kinds of information because most of the facts presupposed in the texts are exactly the facts that are necessary to resolve/disambiguate linguistic phenomena such as anaphora, compound nouns, metonymies etc.

Results from lexical semantics that systematize regularities in the lexical and compositional behaviour of words are of course of big interest and can be "(re)-used".

The working typology we present in this section is not exhaustive, but it gives a pretty good indication of the kinds of knowledge presupposed in texts. To make a more complete list more text corpora should have been analysed (possibly though the list would be still incomplete).

- General knowledge about the type of corpus to be analysed is relevant because it clarifies many linguistic and pragmatic features of the texts. This type of information is relevant when determining the important facts about the text corpus in the first phases of the elicitation process. The elements about the actual text corpus that should be taken into consideration are at least
  - the genre
  - the style
  - the medium

  Particular attention must be given to the type of language used: linguistic conventions of the genre, formal/informal texts, discourse strategy, informational density, length and complexity of the sentences, types of subordinate clauses, temporal and causal adverbials etc.
- Knowledge about the communicative situation and the communicative competence (the addressor and the addressee of the texts and their qualifications, the purpose of the communication, the extent of shared knowledge...) is relevant to determine the granularity of the domain (degree of technicality) and to establish many facts presupposed by the texts (e.g. the purpose of the texts).
- Knowledge about the communicative situation together with knowledge about the task the system should accomplish, is also important to decide the granularity of the system.
- Both overt and covert connections among sentences must be discovered.
- Relations (temporal and causal) among sentences must be explicated.

Together with the above general points there are recurrent linguistic phenomena that need background knowledge to be disambiguated. The frequency of the kind of these phenomena vary from text type to text type. Some phenomena are more common than others, e.g. we hardly expect metaphors in an owner's manual or in a patient discharge letter, while we can expect noun compounds in nearly all texts. The linguistic phenomena to be taken in consideration are at least the following:

- compound nouns[1]
- pronominal anaphora
- definite reference
- attachment ambiguity (prepositional phrases)
- metonymy
- ellipses
- metaphors
- belief reports

Common to all domains (but with different granularity) are knowledge about scale, physical objects, space, change, causality, time, functionality, etc. that are exactly the "clusters of commonsense knowledge" identified by most researchers in the field of knowledge engineering for both NLP systems and expert systems (i.a. Hayes 1979, Herzog & Rollinger eds. 1991, Hobbs & Moore eds. 1985, Lenat & Feigenbaum 1987).

12 Examples of the analysis of three text corpora

Among the text corpora available at our sites we have selected three to be used as working material in WP 2 according to the following main criteria:

[1]Sometimes compound nouns must be considered as being a single item and then cannot be disambiguated. In many cases though it is necessary to use some background knowledge to disambiguate the relation among the constituents. Different interesting approaches have been suggested to analyse compound nouns (e.g. Downing 1977, Levi 1978, Anick and Pustejovsky 1990).

the text corpora should belong to different domains, they should have different degrees of technicality, so that it would be possible to compare them in the light of the present issue, and for practical purposes they should be small.

The first text corpus selected belongs to the motor car mechanical domain. It contains descriptions of different topics (viz. towing, wheel changing, engine oil, coolant, fan belt) from owner's manuals of seven cars from different firms. It has been collected at CST for a project on knowledge based machine translation (English, Danish).

The second text corpus belongs to the medical domain and consists of sixteen PDSs, i.e. Patient Discharge Summaries. It is a subset of a corpus used by HCRC - LTG in a project whose aim is to automatically extract information from PDSs written in different European countries.

The third text corpus is about terrorist attacks. It has been selected for our project from the set of test texts for MUC3. Our text corpus concerns only bomb attacks and contains texts from written sources, i.e. the texts transcripted from radio and TV broadcasts have not been included.

12.1 Characteristics of the three text corpora

The three corpora are quite different in style, technicality and complexity (by complexity in this context we mean the amount and the kind of information which is presupposed by the texts).

The motor car mechanical and the PDS text corpus are less complex than the terrorist corpus because they presuppose more specific background knowledge. The motor car mechanical and the PDS text corpus also belong to more delimited subject areas which makes it 'easier' to delimit the domain specific knowledge the texts presuppose. In the terrorist corpus commonsense (general world) knowledge and domain specific knowledge have no clear-cut boundaries.

The type of the motor car mechanical text corpus is owner's manuals. The readers are people with a driving licence and hence presumably have some layman's knowledge of the functionality of cars; the senders are experts in the domain. The information is therefore explicated as much as possible.

The discourse strategy for the text corpus is a 'step by step' one (Quirk *et al.* 1989), i.e. each fact follows the other in a linear way. The sentences are generally short and contain instructions expressed with imperatives (do this, don't do that).

Pictures supporting the information contained in the instructions were originally attached to the text. This extra information is lost in the actual "text-only" version.

The PDSs are written by doctors for doctors. The type is "discharge summary letters". The style is formal and the letters are composed in a nearly schematic way. Most sentences are short. Longer sentences contain lists (medicine, treatments etc.). The connection between the different sentences is not as linear as in the motor car mechanical corpus and the causal connections between the sentences are not always obvious.

In the PDS corpus both the sender and the addressee are experts. A lot of the implicit and/or background knowledge in the texts is therefore specific to the medical domain and, in the present case, mostly to the heart disease domain.

The terrorist corpus is much more complex. It contains articles from newspapers and news agencies' bulletins etc. The senders are journalists, the addressees are both journalists (news agencies' bullettins) and ordinary people who read newspapers. The style of the corpus is not homogenous and some of the texts are translations from other languages.

The texts contain information about bomb attacks that took place in Latin America over a period of more than a year. A lot of complex world knowledge (economics, international and local politics, social relations etc.) is presupposed.

To understand the texts in the corpus it is necessary to know the political situation in the area, to deal with beliefs and to have knowledge about the original source of the news (a local radio that sympathizes with a particular terrorist group; an international news agency; a newspaper owned by the government ...).
Take as an illustrative example

> THE MANUEL RODRIGUEZ PATRIOTIC FRONT (FPMR), WHICH THE PINOCHET REGIME CONSIDERS TO BE THE COMMUNIST PARTY'S ARMED BRANCH, ANNOUNCED FOLLOWING THE U.S. INVASION OF PANAMA THAT...

13 The analysis of some texts from the three corpora

In the following we will only describe the results of the analysis of one text from each corpus. In describing these results we distinguish between knowledge belonging to different domains (viz. domain independent (commonsense knowledge), domain specific, e.g. medical domain, motor car mechanical domain). How many domains can be distinguished in a corpus depends on the granularity one applies in the analysis.

In this paper we adopt a very coarse grain (the most conspicuous) because the relevant matter here is to define which types of knowledge are useful to understand a text and not to give a complete list of elements for each type.

We will also relate words to different clusters of knowledge or "core theories"[2].

13.1 Analysis of one PDS

The PDS we will discuss in the following is:

```
PATIENT NAME : JOHN AITCHROM

ADDRESS : 71 PARK DRIVE          SEX : M
          OLDCASTLE              DOB : 20.02.1947

ADMITTED :28.03.1991        DISCHARGED :30.03.1991
CONSULTANT: Prof L.T.G. Wilson
    MAIN DIAGNOSIS : G340.... Coronary atherosclerosis

GP : DR L.T.G. AWESOME
     31 ROADSIDE
     OLDCASTLE
```

[2]"Core theories" have been defined in (Hobbs et al. 1987) as the basic ontologies and structures of various commonsense domains that figure in virtually every domain of discourse. Examples of these domains are scalar notions, granularity, time, space, material, physical objects, causality, functionality, force, and shape.

> Dear Dr Simpson,
>
> Mr Aitchrom was admitted on 28th March 1991 for cardiac catheterisation. A report of this is enclosed.
>
> With double vessel disease and good left ventricular function, the prognosis with medical and surgical treatment is similar and a decision for surgery would be largely on the basis of symptoms.
>
> On full therapy, Mr Aitchrom is having much less angina.
>
> We plan to have a thallium study and I will review him when this is available.
>
> He should continue on his current drugs and he was discharged on 30 March 1991.

The content words (and the abbreviations) in the text have been divided into the following groups:

- **Domain independent knowledge**: available, basis, continue, current, dear, decide, decision, DOB (date of birth), dates (28.03.1991 etc.), double, enclose, full, function, good, largely, left, less, letter, main, mr, much, name, plan, report, review, secondary, sex, similar, study.
- **Domain specific knowledge** (medical domain): admit, angina, atherosclerosis, cardiac, catheterisation, consultant, coronary, diagnosis, discharge, disease, dr, drug, hospital, G.P., medical, patient, prognosis, surgery/surgical, symptom, thallium, therapy, treatment, ventricular, vessel.

To understand the text it is necessary, inter alia,

- to disambiguate **compound nouns** as *discharge letter, vessel disease, thallium study...*
- to resolve **pronominal anaphora**: *report of this is enclosed, review him when this is available...*
- to resolve **definite reference**: *...the prognosis with...*
- to resolve the **attachment ambiguity**: *With double vessel disease and good left ventricular function, the prognosis....*

The words which are domain independent can be divided into the following subdomains (clusters):

full —> set

28. . . —> numbers

basis —> causality

date
DOB
continue
current —> time and change

available
function —> functionality

double —> measurement

good
less
much
similar
largely
main
secundary —> scale

decision —> plan, causality, change

admit
discharge —> change, movement

left —> space, orientation.

We have also divided the domain specific words in following clusters:

angina
atheriosclerosis
disease —> disease (process)

catheterisation
surgery/surgical
medical
treatment —> treatment (process)

cardiac
coronary

ventricular
vessel —> patient's body-parts (related-to body-parts)

hospital —> space

drug —> material

consultant
doctor
G.P.
hospital
diagnosis
patient
prognosis —> health-organisation, goal-directed systems.

The following facts which are also presupposed by the text are also necessary to understand the texts:

- Hospitals are places where people with a disease (patients) are treated and/or cured.
- Doctors are persons that treat (surgically/medically) patients inside and/or outside hospitals.
- A disease is a process that make some parts of/the body malfunction. It can cause pain.
- On the basis of different signs doctors can decide a treatment and make a prognosis for a disease.
- Doctors at hospitals write discharge letters to inform doctors outside the hospital (or at other hospitals) about the situation of a patient when he/she is dismissed.

13.2 Analysis of a text from the motor car mechanical domain

Below is the chosen example from the motor car mechanical text corpus:

> Never tow an automatic transmission model with the rear wheels
> raised (with the front wheels on the ground) as this may cause

> serious and expensive damage to the transmission. If it is necessary to tow the vehicle with the rear wheels raised, always use a towing dolly under the front wheels.

The words in the text can be divided into the following groups:

- **Domain independent knowledge**: always, automatic, cause, damage, expensive, front, ground, may, necessary, never raise, rear, serious, under, use, with.
- **Domain specific knowledge**: dolly, model, tow, transmission, vehicle, wheel.

The words in the text can be divided into following subdomains:

under
front
rear —> orientation

ground —> space, orientation

tow
raise —> space, movement, causality

necessary
cause —> causality

always never —> time

with —> instrumentality, causality

expensive —> scale, economics

damage —> causality, goal-directed systems, functionality

serious —> scales

vehicle
transmission
wheels
dolly
model —> artifacts, goal-directed-systems

automatic —> functionality.

To understand the text it is necessary to:

- Disambiguate **compound nouns**: *transmission model...*
- Resolve **pronominal anaphora**: *this may cause...*
- Resolve **definite reference**: *Never tow an automatic transmission model with the rear wheels raised* (here a so called *bridging inference*), *the transmission...*
- Resolve **attachment ambiguities**: *tow an automatic transmission model with the rear wheels raised...*

The facts that are presupposed in the text are, inter alia:

- General knowledge about cars (e.g. they have four wheels, a transmission etc.).
- The texts contain instructions about how to use a vehicle in a correct manner, because incorrect procedures can damage it.
- There is a connection between the front wheels and the transmission in automatic car models.
- When towing vehicles, a dolly is often used so that the vehicle's wheels are off the ground (cf. the connection wheels-transmission).

13.3 Analysis ofa text from the terrorist corpus

We will describe the analysis of following text from the terrorist corpus:

> MEDELLIN, 27 AUG 89 (AFP) – [TEXT]
> TODAY, MEDELLIN, COLOMBIA' S SECOND LARGEST CITY, ONCE AGAIN EXPERIENCED A TERRORIST ESCALATION WHEN SEVEN BANK BRANCH OFFICES WERE SHAKEN BY EXPLOSIVES THAT CAUSED HEAVY DAMAGE BUT NO FATALITIES, ACCORDING TO RADIO REPORTS BROADCAST IN BOGOTA (500 KM TO THE SOUTH).

> THE TARGETS OF THE ATTACKS WERE THE BANCO CAFETERO BRANCHES AND ITS OFFICES IN MEDELLIN'S MIDDLE, WESTERN, AND SOUTHEASTERN AREAS.

The words in the text can roughly be classified in the three groups mentioned below:

Domain independent knowledge: according, again, area, banco/bank, Bogota, branch, broadcast, by, cause, city, Colombia, damage, dates (27 august 1989), escalation, experience, fatalities, heavy, in, its, km, largest, Medellin, middle, of, office, once, radio, report, second, seven, shook, southeastern, to, today, western, when.

Domain specific knowledge (terrorist domain): attack, explosive, target, terrorist.

We have divided the words in the target text into the following domains:

27 august 1989
when
again
once
today —> time, change

cause —> causality, change

500 —> set, scale, numbers

escalation
largest
heavy —> scale

by —> causality, instrumentality

in —> space

western
southeastern
middle
area —> space, orientation

km —> space, scale

city
bank (banco)
branch
office
radio —> space, organization, goal-directed systems

bank (banco) —> economy

branch, office —> subset

report
broadcast —> communication, plan&goal

shook —> change, movement, causality

second
seven —> set, numbers

to —> movement, change

target —> plan, goal-directed system, space

terrorist —> goal-directed system

attack —> change, movement, plan

explosive —> physical object, material, plan

according to —> belief

To understand the text it is necessary, inter alia:

- To disambiguate **compound nouns**: *bank branch offices, radio reports broadcast, Banco Cafetero branches...*
- To resolve **pronominal anaphora**: *the Banco Cafetero Branches and its offices...*
- To resolve **definite reference**: *the targets of the attacks* (where *attacks* refers to explosives and terrorist escalation).
- To deal with **belief reports**: *according to...*

For understanding the text it is inter alia necessary to have access to the following presupposed facts:

- terrorists use violence to achieve political results;

- explosives cause damage to things and can injure peoples;
- damage can have different degrees (damage to properties, damage to the environment, damage to human life);
- banks are economical institutions;
- *Banco* is the Spanish word for bank;
- news from a radio broadcast can be incomplete.

13.4 Conclusion

Applying the method for knowledge elicitation to the three corpora we have found out that the main types of background knowledge in a somehow homogenous text corpus can in many case be 'discovered' by applying the method to a subset of the source text corpus. After which criteria an "homogeneous" subcorpus should be composed will be inquired in the second report for this phase of the project *Criteria of possible "support" material.*

The elicitation method we have applied in this phase of the project is very general. The definition of strategies, techniques and heuristics aimed to facilitate the extraction of the different semantic types of knowledge described in the first part of this document is one of the issues to be investigated in the next phases of the project.

14 References

Anick and Pustejovsky [1990] P. Anick and J. Pustejovsky. "An Application of Lexical Semantics to Knowledge Acquisition from Corpora". In *COLING Proceedings*, Helsinki, Vol.2, pp. 7–12.

Bech et al. [1993] A. Bech, M. Moens, C. Navarretta. "Strategies in NLP knowledge engineering". ET-12 Project, Report 1.

Braasch [1991] A.D. Braasch. "Delprojekt 3: Oversaettelsesteori i Maskinoversaettelse. Valg af Tekstsort – Korpus – Undersögelsesaspekter". In A.L. Jakobsen (ed.), *Oversaettelse af fagsproglige tekster.* ARK 5, Köbenhavn, pp. 117–131.

Brachman [1979] R.G. Brachman. "On the epistemological status of semantic networks". In N.V. Findler (ed.): *Associative Networks: Representation and Use of Knowledge by Computers*, 3–50, New York, Academic Press.

Buzan [1974] T. Buzan. "Use Both Sides of Your Brain". E.P.Dutton, New York.

Clark and Gerrig [1983] H.H. Clark and R.J. Gerrig. "Understanding Old Words with New Meanings". In: *Journal of Verbal Learning and Verbal Behavior*, Vol. 22, pp. 591–608.

Downing [1977] P. Downing. "On the Creation and Use of English Compound Nouns". In: *Language*, Vol. 53, No. 4, pp. 810–842.

Dyer [1989] M.G. Dyer. "Knowledge interactions and integrated parsing for narrative comprehension" In: D.L. Waltz (ed.) *Semantic Structures: advances in natural language processing.* LEA.

Guarino [1991] N. Guarino. "Concepts, Attributes and Arbitrary Relations". In: *Data and Knowledge Engineering. Special issue on Linguistic Instruments and Knowledge Engineering.*

Hayes [1979] P.J. Hayes. "The Naive Physics Manifesto". In: D. Michie (ed.) *Expert Systems in the Micro-electronic Age* Edinburgh University Press, Edinburgh, pp. 242–270.

Herzog and Rollinger [1991] O. Herzog and C.-R. Rollinger (eds.) "Text Understanding in LILOG". Berlin: Springer-Verlag.

Hobbs [1984] J.R. Hobbs. "Sublanguage and Knowledge". Technical Note 329. SRI, California.

Hobbs & Moore [1985] J.R. Hobbs & R.C. Moore (eds.) "Formal Theories of the Commonsense World". Ablex.

Hobbs [1985a] J.R. Hobbs. "Granularity". In: *Proceedings of the Ninth International Joint Conference on Artificial Intelligence. Vol.1*, pp. 432–435.

Hobbs [1985b] J.R. Hobbs. "Ontological Promiscuity". In: *Proceedings, 23rd Annual Meeting of the Association for Computational Linguistics*, pp. 61–69.

Hobbs et al. [1986] J.R. Hobbs, W. Croft, T. Davies, D. Edwards, K. Laws. "Commonsense Metaphysics and Lexical Semantics". Technical Note 392. SRI, California.

Hobbs [1986] J.R. Hobbs. "Overview of the TACITUS Project". In: *Computational Linguistics*, Vol 12, No.3.

Hobbs [1987] J.R. Hobbs. "World Knowledge and Word Meaning". In Y. Wilks (ed.): *Proceedings of TINLAP-3*, Las Cruces, New Mexico, pp. 16–21.

Hobbs et al. [1988] J.R. Hobbs, W. Croft, T. Davies, D. Edwards, K. Laws. "The TACITUS Commonsense Knowledge Base" (Draft). SRI, California, May 17.

Hobbs et al. [1990] J.R. Hobbs, M. Stickel, D. Appelt and P. Martin. "Interpretation as Abduction". Technical Note 499. SRI, California.

Lenat & Brown [1984] D.B. Lenat & J.S. Brown. "Why AM and EURISCO Appear to Work". In Artificial Intelligence. Vol 23, no 3, pp. 269–294

Lenat & Feigenbaum [1987] D.B. Lenat and E.A. Feigenbaum. "On the Thresholds of Knowledge". MCC Technical Report Number AI-126-87, Austin Texas.

Lenat & Guha [1988] D.B. Lenat, R.V. Guha. "The World According to CYC". MCC Technical Report Number ACA-AI-300-88, Austin Texas.

Lenat et al. [1985] D.B. Lenat, M. Prakash, M. Shepherd. "Cyc: Using Common Sense Knowledge to Overcome Brittleness and Knowledge Acquisition Bottlenecks". MCC Technical Report Number AI-055-85, Austin Texas.

Lehnert and Loiselle [1989] W.G. Lehnert and C.L. Loiselle "An introduction to plot units" In D. L. Waltz (ed): *Semantic Structures: advances in natural language processing.* LEA.

Levi [1978] J. Levi. "The Syntax and Semantics of Complex Nominals". Academic Press, New York.

Marslen-Wilson and Tylor [1987] W. Marslen-Wilson and L.K. Tylor. "Against Modularity". In: J.L. Garfield ed., *Modularity in Knowledge Representation and Natural Language Processing*, MIT Press, Cambridge, Massachusetts, pp. 37–62.

Nowak and Gowin [1984] J.D. Nowak and D.B. Gowin. "Learning How to Learn". Cambridge, Cambrigde University Press.

Nunberg [1978] G. Nunberg. "The Non-uniqueness of Semantic Solutions: Polysemy". In: *Linguistics and Philosophy 3*, Reidel Publishing Company, pp. 143–184.

Parsons [1990] T. Parsons. "Events in Semantics of English: a study in subatomic semantics". MIT Press.

Quirk et al. [1989] R. Quirk, S. Greenbaum, G. Leech, J. Svartvik. "A Comprehensive Grammar of the English Language". Longman Group Lt. UK.

Rosch [1977a] E. Rosch. "Classification of real world objects: Origins and representation in cognition". In P.N. Johnson-Laird and P.C. Wason, editors, Thinking: Reading in Cognitive Science, pp. 212–222, Cambridge.

Rosch [1977b] E. Rosch. "Linguistic relativity". In P.N. Johnson-Laird and P.C. Wason, editors, Thinking: Reading in Cognitive Science, pp. 501–521, Cambridge.

Sag and Hankamer [1984] I.A. Sag and J. Hankamer. "Toward a Theory of Anaphoric Processing". In: *Linguistics and Philosophy 4*, Reidel Publishing Company, pp. 325–345.

Schank [1982] R.C. Schank. ”Reminding and memory organization: An introduction to MOPs”. In W.Lehnert & M.Ringle (Eds.), Strategies for natural language processing, pp. 455–493. Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank & Abelson [1977] R. Schank & R. Abelson. ”Scripts, plans, goals and understanding”. Hillsdale, NJ: Lawrence Erlbaum Associates.

Sowa [1984] J.F. Sowa. ”Conceptual Structures: Processing in mind and machine”. Addison-Wesley Publishing Company Inc. 1984.