# A Study on the Profits of Semantics in Object-oriented Information Modelling based on Natural Language.

*An Artificial Intelligence approach.*

Erik Sj. C. van de Ven


Master's Thesis No. 371, June 4, 1996

Department of Informatics
Faculty of Mathematics and Informatics

Katholieke *Universiteit* Nijmegen

Information Systems

# A Study on the Profits of Semantics in Object-oriented Information Modelling based on Natural Language.

*An Artificial Intelligence approach.*

**Afstudeerscriptie 371**,

ter verkrijging van de graad doctorandus
aan de Katholieke Universiteit Nijmegen,
Faculteit der Wiskunde en Informatica,
Afdeling Informatiesystemen

door:

Erik Sj. C. van de Ven

geboren te Epe, June 8th 1972

begeleider(s):

Paul Frederiks

June 4, 1996

# Contents

**Abstract** **3**

**1 Introduction** **9**

1.1 Problem Area . . . . . 9

1.2 Problem Statement . . . . . 11

1.3 Related Work . . . . . 11

1.4 Organization of this Paper . . . . . 11

**2 Expert Systems and Information modelling** **13**

2.1 Introduction . . . . . 13

2.2 Information Systems . . . . . 13

2.3 Information Systems and Decision Making . . . . . 14

2.4 Expert systems . . . . . 16

2.4.1 System overview . . . . . 16

2.4.2 System architecture . . . . . 17

2.4.2.1 The knowledge base . . . . . 18

2.4.2.2 The inference engine . . . . . 21

2.4.2.3 The user interface . . . . . 22

**3 Natural Language** **23**

3.1 Introduction . . . . . 23

3.2 Properties of Natural language . . . . . 23

3.2.1 Language is structured . . . . . 24

3.2.2 Language is meaningful . . . . . 24

3.2.3 Language is referential . . . . . 24

3.3 Syntax of Natural language . . . . . 25

3.4 Meaning of Natural language . . . . . 28

3.4.1 The meaning of words . . . . . 29

3.4.1.1 The definitional theory of meaning . . . . . 29

3.4.1.2 The prototype theory of meaning . . . . . 29

3.4.2 The meaning of sentences . . . . . 30

**4 Semantics of NL for Information Modelling purposes** **31**

4.1 Introduction . . . 31

4.2 A Semantical Linguistic theory . . . 32

4.2.1 The functional grammar paradigm . . . 32

4.2.2 A semantical typology of predications . . . 36

4.3 Application of the Semantical approach . . . 38

4.3.1 Different aspects of a UoD . . . 38

4.3.2 Data modelling . . . 38

4.3.3 Process modelling . . . 43

4.3.4 CPL and complex constraint modelling . . . 44

4.3.4.1 Static vs. dynamic constraints . . . 44

4.3.4.2 Time order . . . 44

4.3.4.3 Modality . . . 45

4.3.5 Formal syntax of CPL . . . 45

4.3.6 Formal semantics of CPL . . . 45

4.4 Transforming 'Object Action Involvement' and 'Object Life'-models into CPL . . . 46

4.4.1 'Object Action Involvement Model' transformation . . . 46

4.4.2 'Object Life Model' transformation . . . 50

**5 The role of the lexicon in the semantical approach** **53**

5.1 Introduction . . . 53

5.2 The lexicon . . . 53

5.3 Semantical relations and their Applications . . . 56

5.3.1 Instance relations . . . 58

5.3.2 Hypernym relations . . . 59

5.3.3 Lexical semantical consequence relations . . . 61

5.3.4 Meronym relations . . . 62

5.3.5 (Partial) synonym relations . . . 65

5.3.6 Antonym relations . . . 66

5.3.7 Semantic consequence relations operating on verbs . . . 70

5.3.8 Function relations . . . 74

5.3.9 Gerund relations . . . 76

5.3.10 Transmission relations . . . 77

5.3.11 Overview of Semantical relations . . . 78

5.4 Evaluation . . . 79

**6 Supporting Information Analyst and Domain Experts** **81**
6.1 Introduction . . . 81
6.2 PSM$^{2+}$ Meta model revisited . . . 81
6.3 Formulation by Navigation . . . 82
6.4 Interaction with the Inference Engine . . . 85
6.5 Verbalization . . . 87
6.5.1 Introduction of Syntax . . . 87
6.5.2 A Verbalization Algorithm . . . 89
6.5.2.1 Potential subject status . . . 89
6.5.2.2 Potential object status . . . 90
6.5.2.3 Verbalization of terms . . . 90
6.5.2.4 Verbalization of predicates . . . 92
6.5.2.5 Verbalization of simple predications . . . 94
6.5.2.6 Verbalization of complex predications . . . 97
6.6 Evaluation . . . 100

**7 Conclusions and Further Research** **103**
7.1 Summary . . . 103
7.2 Evaluation . . . 104
7.3 Further Research . . . 104

**A Implementation Forward Chaining** **107**

**B Implementation Backward Chaining** **109**

**C RDB schema of Information Modelling tool** **113**

**D Implementation of Verbalisation algorithm** **115**

**E Predicate verbalisations for modality MUSTand NEC** **119**

**F PSM$^{2+}$-expert prototype** **121**

**Bibliography** **123**

**Author Index** **125**

**Index** **127**

And though I have the gift of prophecy, and understand all mysteries,
and all knowledge;
and though I have all faith,
so that I could remove mountains,
and have not charity,
I am nothing.

(1 CORINTHIANS 13:2, The King James Version (1611))

# Abstract

The past decades have shown that the development of large software applications is a difficult problem, which is still far from fully understood. In fact, it appears that this problem only increases as application tend to become more and more complex. The development of *information systems*, one of the more frequently occuring application, is a typical example of this problem. The development of an information system, is based on a simplified description of the *Universe of Discourse* (UoD), called an *information model*. The correctness of this model is warrented by *verification* and *validation*. Verification checks whether the model has certain properties, using the rules of a technique. While on the other hand, validation checks whether a well-formed model, meets its intended requirements in the UoD. Validation is therefore to be performed by the *domain-expert*. However, the typical domain-expert, although considered as an expert on the UoD, has - in general - no (or too little) knowledge about traditionally used information modelling techniques to fully comprehend the model. To narrow the gap between the informal specifications of the requirements stated by the domain-expert and the formal modelling techniques, information modelling techniques have been introduced which are based on comminication means shared by both domain-expert and information analyst: *natural language* (NL). Some of these natural language based information modelling techniques primarily focus on structure (or *syntax*) of NL, while others primarily focus on meaning (or *semantics*) of NL.

In this study the semantic approach to information modelling is exploited, resulting into an unambiguous modelling technique, exceeding the *expressive power* of syntax-oriented approaches. The approach described in this document is based on the semantics-oriented linguistic theory of *Functional Grammar*. Additional profits of the semantics-oriented approach are reflected in easy automatic verbalisation of the informaton model, simulation of semantic inferences based on the human *mental lexicon* and establishment of a natural integration of data with *common sense* knowledge to realize intelligent communication between man and machine. The profits are demonstrated by a prototype tool, called $PSM^{2+}$-expert, with an easy and direct implementation in a *logic programming* language, called PROLOG.

# List of Figures


2.1 A communication-oriented architecture of an information system . . . . . 14
2.2 The different types of information systems . . . . . 16
2.3 Typical expert system architecture . . . . . 18
2.4 An ordinary family . . . . . 19
2.5 Example information architecture . . . . . 20

3.1 The meaning is separated from the reference . . . . . 25
3.2 The hierarchy of linguistic structures . . . . . 26
3.3 The parse tree of a sample sentence . . . . . 28
3.4 The model of the transformational - generative grammar. . . . . 28

4.1 The functional grammar paradigm . . . . . 34
4.2 Typology of SoAs . . . . . 38
4.3 Modelling techniques with their typical use . . . . . 39
4.4 A binary relation in PSM . . . . . 39
4.5 PSM conventions (a) specialization, (b) generalization, (c) aggregation, and (d) domain specification . . . . . 40
4.6 Cardinalities restricted by uniqueness constraint, (a) one-to-many, (b) one-to-one, (c) many-to-many . . . . . 41
4.7 Distributivity of cardinalities in CPL: (a) collective, (b) distributive . . . . . 42
4.8 Handling populations in PSM . . . . . 42
4.9 A BORROW action in the CPL paradigm . . . . . 43
4.10 Action 'give' conform $PSM^2$ . . . . . 46
4.11 $PSM^2$-schema to be transformed . . . . . 48
4.12 KISS model for object type musician . . . . . 51

5.1 An information system containing lugubrious facts . . . . . 54
5.2 The system lexicon and the mental lexicon, used during communication . . . . . 55
5.3 The lexicon as a top-layer on the data model . . . . . 55
5.4 Classification of semantical relations . . . . . 56
5.5 PSM meta schema of $PSM^{2+}$ . . . . . 57
5.6 A general $PSM^{2+}$-relation between one or more Objectypes. . . . . 58

5.7 A 'join'-relation in PSM$^{2+}$ notation. . . . . 58
5.8 Objectification of relations in PSM$^{2+}$. . . . . 60
5.9 Hypernym and Instance relations according to WordNet. . . . . 60
5.10 Contrasting terms based on a single direct antonym relation. . . . . 70
5.11 Semantical consequence relations between verbs: (a) simultaneity, (b) proper inclusion and (c) presupposition . . . . . 71
5.12 Illustration of presupposition . . . . . 73
5.13 Visualization of a 'gift'. . . . . 74
5.14 Gerund as denotation for objectified fact-types. . . . . 76
5.15 Derivation of the transmission relation. . . . . 78

6.1 Various ways of supporting . . . . . 82
6.2 The adapted version of the PSM$^{2+}$meta schema . . . . . 83
6.3 Construction of a PSM$^{2+}$-predication. . . . . 83
6.4 Predication formulation dialog of prototype Information Modeler. . . . . 84
6.5 Defining a complex predication (2) based on (1). . . . . 85
6.6 Definiteness expressed by a definite article . . . . . 87
6.7 PSM$^{2+}$-model to be verbalized . . . . . 97
6.8 Verbalization of a nested (complex) predication . . . . . 100

C.1 Relational database schema of PSM$^{2+}$-tool . . . . . 113

# List of Tables


3.1 The lexicon contains the context sensitive information . . . . . 30
3.2 An example of a selection restriction . . . . . 30

4.1 Semantic functions recognized by FG . . . . . 33
4.2 Satellites recognized by FG . . . . . 36
4.3 Semantic parameters recognized by FG . . . . . 37
4.4 Formal syntax of CPL . . . . . 45
4.5 Table representation of permitted order of actions . . . . . 50

5.1 Word-association test on test-word 'chair' . . . . . 54
5.2 Mathematical properties of class (A) meronym relations . . . . . 63
5.3 Overview of direct antonym-relations . . . . . 70
5.4 Overview of semantic relations. . . . . 79

6.1 Different verbalizations caused by distribution of Subject and Object . . . . . 89
6.2 Potential subject assignment in various languages of the world . . . . . 90
6.3 Potential object assignment in various languages of the world . . . . . 90
6.4 Term-verbalization inherent to distribution of subject and object assignment. . . . 91
6.5 Definiteness and Number properties affect term verbalization. . . . . 91
6.6 Plural form of terms . . . . . 92
6.7 Possible predicate verbalizations for modality FACTUAL . . . . . 95
6.8 Possible predicate verbalizations for modality PERMIT . . . . . 95
6.9 Morphological modifications of predicates . . . . . 96

E.1 Possible predicate verbalisations for modality MUST, NEC . . . . . 120

# Chapter 1

# Introduction

## 1.1 Problem Area

The past decades have shown that the development of large software applications is a difficult problem, which is still far from fully understood. In fact, it appears that this problem only increases as applications tend to become more and more complex. The development of information systems, one of the more frequently occuring applications, is a typical example of this problem.

An important and very difficult goal of the early phases of system development is the acquisition and representation of requirements. This part of system modelling is commonly referred to as *requirements engineering*. Information modelling is defined as that part of requirements engineering which involves a high-level, problem-oriented and implementation independent description of the problem domain. The description resulting from information modelling, is called an *information model*. An information model models part of a real or postulated world. This area of concern is referred to as the *Universe of Discourse* (UoD), see [Gri82]. An important role of the information model is to provide a common understanding of the UoD involved. The information model plays the central role between the informal world of the UoD and the formal information system.

As stated in [HW92] an information model contains at least two aspects: the static aspects and the dynamic aspects of the UoD. An information model therefore can be viewed from different perspectives, a *data* perspective (providing a description of the static aspects) and a *process* perspective (providing a description of the dynamic aspects). Modelling static aspects results into a *data model*, and modelling dynamic aspects results into a *process model*. Most information systems development methods tend to be dominated by one perspective. A clear separation between data and process can be useful, because data models are likely to be relevant for a longer period than process models which can be unstable ([AW91]).

The quality of an information model is of great importance because it forms the basis of the information system to be developed. The quality of the information system greatly depends on the quality of this model. The quality of the model can be assured by *verifying* and *validating* the model. Verification checks whether a model has certain properties, using the rules of a technique. If the model has a formal foundation, it is possible to perform verification automatically. Checking whether a model is well-formed according to a certain technique, is an example of verification. Examples of data modelling techniques with a formal foundation are PSM([HW93]), and IFO([AH87]). Validation, on the other side, checks whether a well-formed model meets its intended requirements in the UoD. The *validation* of a model can not only be done by introducing formal foundations. In order to validate a model we need the cooperation of a domain-expert. The domain-expert, although considered as an expert on the UoD, has - in general - no knowledge of the used traditional modeling techniques. However, he is the obvious person to validate the model. To resolve this paradox, the gap between information analyst and domain expert must be narrowed somehow. The domain-expert has to validate the model formulated by the information analyst.

As stated before an information model is to provide a common understanding of the UoD involved. This implies that the model should have a unequivocal meaning. In [HW92] five requirements on information modelling techniques are recognized:

- *formal*: Firstly, an information model should have a well-formed formal semantics.
- *expressive power*: In addition to that, the information modelling technique should have sufficient expressive power to the describe the UoD.
- *comprehensible*: As information models play a crucial role in the communication between information analyst and the domain-expert, they should be comprehensible.
- *executable*: Insight in the meaning of an information model can be improved by its execution. Therefore, information models should be executable.
- *conceptual*: And finally the last requirement states that an information modelling technique should be on a conceptual level. This prevents implementation decisions from having to be made in a too early phase, which could lead to suboptimal solutions during actual implementation. Models which are not on a conceptual level have generally a negative influence on the comprehensibility.

Four different approaches to information modelling can be distinguished (see [HW92] for more information):

- *Structured approaches* as JSD([Jac83]) and Yourdon([You89]), have some serious disadvantages. The first disadvantage is that they do not have a formal foundation. This implies that the specifications are not executable. The second disadvantage is that the process and data modelling techniques involved do not have sufficient *expressive power*.
- *Petri net based approaches* are suitable for modelling communication between parallel processes. A major disadvantage of these approaches, is the complexity of specifications and the moderate comprehensibility.
- *Object oriented approaches* Although with these approaches modelling data with complex structures and complex constraints, cannot be done adequately, they have an important advantage: the data and process perspectives are integrated in a natural way.
- *Formal specification languages* With regard to the above mentioned requirements one of its major drawbacks is the moderate comprehensibility.

All the above mentioned approaches have their advantages and disadvantages. The suitability of an approach is inherent to the type of UoD: real-time, data intensive, etc. Although we do not believe in one *silver bullet*, the combination of the object-oriented approach with the formal specification language approach is worth to be studied in more detail.

To narrow the gap between the informal specifications of the domain expert and the formal modelling techniques of the information analyst, modelling techniques have been introduced that are based on communication means that are shared by the domain expert and information analyst: *natural language* (NL). Some of these techniques focus their attention mostly on the syntax (read structure) of NL, while others try to capture semantics (read meaning) as well. These approaches to using natural language can be called the *syntactical* and *semantical* approach respectively.

In [BR95b] the formal specification language ([DR91]) approach based on natural language and the object oriented approach are combined successfully. In [Kri94] an informal approach to object orientation based on natural language is given. An example of the syntactical approach is given in [FKW95].

## 1.2 Problem Statement

This study attempts to clarify the essential differences between the semantical approach of [BR95b] and the syntactical approach of [FKW95]. Typical questions to be answered are:

- What advantages bring the involvement of semantics?
- How to exploit the semantical approach?

## 1.3 Related Work

In [BR95b] the formal specification language approach based on natural language ([DR91]) and the object oriented approach are combined successfully. This combination has resulted into a NL based knowledge representation languageknowledge!representation language, called *Conceptual Prototyping Language* (CPL). CPL has a formal foundation based on logic. Although this language is developed to be as close as possible to NL, it is still difficult (too formal) to read or write, reducing the comprehensibility of the specified model.

In [Kri94] another approach to object orientation based on natural language is given. This approach however lacks a formal foundation which leads to fuzzy concepts. Another problem is the low level of expressive power and comprehensibility. A very useful model, part of this method, is called the KISS-model. This model describes the possible life of an object of a certain type. In this model, the well-known *Jackson Structure Diagram* (JSD) concepts like sequence, selection and iteration are used. Also this model lacks a formal foundation but the suggestion of modelling this aspect of the UoD is valuable.

The so-called syntactical approach of [FKW95] combines different modelling techniques to capture the knowledge about the UoD. A formal conceptual modelling technique PSM is used to capture the static aspects. An object life model is introduced to define the dynamic aspects of object types. Affix grammars ([Kos91]) are used extensively to describe the different models, and to facilitate the verbalization of these models. The syntactical approach can therefore be regarded as a grammar based information modellinginformation modelling!grammar based - ([HPW94]) strategy.

## 1.4 Organization of this Paper

The following chapter - **chapter 2** - studies a special type of information system, called *expert system*. The composing components of expert systems are discussed with special attention to the crucial component, the *knowledge base*.

**Chapter 3** focuses on natural language which can be used to narrow the gap between information analyst and domain experts. The major properties of natural language, with special attention to *meaning*, will be discussed.

In **chapter 4**, the discussion on the meaning of natural language is detailed and it is studied how meaning of natural language can be used to enhance comprehensibility and expressive power of traditional information modelling techniques.

The crucial role and contents of the *lexicon* in the semantical approach is described in **chapter 5**. In this chapter it will be illustrated how consultation of the mental lexicon can be simulated with formal inference rules.

In **chapter 6** several ways of support of the semantical approach are discussed including information modelling, verbalization and interaction with the inference engine.

Finally, this document is closed with some conclusions and further research issues, mentioned in **chapter 7**.

# Chapter 2

# Expert Systems and Information modelling

## 2.1 Introduction

This chapter introduces the terms *information system*, *decision making* and *expert system* and describes how these concepts are related to each others. The typical architecture of expert systems is discussed and different reasoning strategies are described and illustrated with the help of examples. The characteristic properties and system components of expert systems are discussed with main focus on the knowledge base, which is the central component of an expert system.

## 2.2 Information Systems

An *information system* is defined here as a system that collects, processes, stores, and distributes information so that it can be used by people. The information may be about people, places, things, or events inside an organization or in the environment that surrounds it. People use the information to make decisions, to keep track of resources, and to plan for the future.

The last three decades have shown that the architecture of information systems has evolved from file-oriented, via data-oriented and communication-oriented towards an object-oriented view. Hand in hand with this architectural evolution the way a user communicates with the information system is changed ([FW96]). The communication-oriented architecture of an information system is depicted in figure 2.1. The system keeps track of the definition of data, and also administers the application programs (the *procedures* base) and the actual forms (*forms base*) which are used as a communication mechanism. The user sends update or retrieval requests, conform a certain *information grammar*, to the *information processor*. If a user is is authorized (*authorization base*) to perform a request, the information processor performs the request on the *information base*. The information (i.e. knowledge that can be communicated (b.d.)) base consists of the knowledge about the UoD.

In this architecture the system is seen as an important means of communication. The communication-oriented architecture offers the user much more freedom for manipulation and interpretation of data. Due to the extra facilities, the system now is capable of providing the user with interpreted data for example in the form of pie chart. These extra facilities support the user of the system with decision making, resource management and planning. In the following section focus is on supporting decision making.

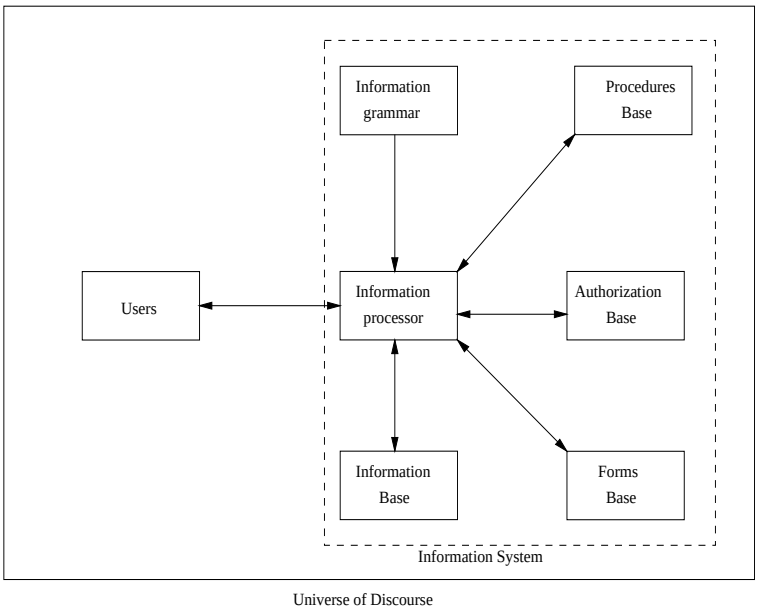


Figure 2.1: A communication-oriented architecture of an information system

## 2.3 Information Systems and Decision Making

As stated in the previous section, one of the tasks of an information system is to support *decision making* performed by the users of the system. In this section the term 'decision making' and its relationship with information systems is made clear.

The term 'decision making' refers to an area of psychology ([Gle91]) that tries to model the process of making decisions performed by humans. Typical questions to be answered in this area are:

1. Which information is used by the human decision maker?
2. How is this information selected and retrieved from memory?
3. Which reasoning strategy is used to draw a certain conclusion?
4. Which cognitive shortcuts (rules of thumb) are used ?
5. How does uncertainty of information affect the decision making process?
6. How is a decision evaluated?
7. How is feedback processed?

These questions are strongly related to a field of study that is called *artificial intelligence* (AI). From a research perspective, AI can be defined as follows ([RK91, page 3]):

> Artificial intelligence is the study of how to make computers do things which, at the moment, people do better.

The theory of artificial intelligence states that both computers and human beings are information processing systems. This basic assumption brings reasoning and decision making in relation with computers. The decision making process is a complex process that can be separated into different phases. Three different phases can be recognized ([Bem94] and [Sim60]).

1. *exploration* phase (intelligence) concerning information acquisition.

2. *problem-formulation* phase (design) concerning the formulation of the problem. This phase typically has an iterative nature.

3. *ixproblem-solving* phase (choice) concerning the actual implementation of the chosen solution.

Traditionally, making decisions belongs to the responsibility of the *user* of the information system. However, the last decades have shown that (in specific domains) the decision making can (partly) be performed by the system itself. Information systems can be classified according to which degree they are capable of making decisions ([Bem94]):

- *Traditional database systems* contain truths about *specific* facts of the UoD.
  - *Transaction Processing System* (TPS) are *not* capable of making decisions. A TPS is responsible for recording all relevant information and state-changes in the UoD. Sometimes these systems are called *operational systems*.
- *Knowledge base systems* not only contain truths about *specific* facts but also about *general* facts of the UoD ([Rei86]).
  - *Structured Decision Systems* (SDS) are capable of making decisions on their owns. A SDS is an information system that processes information according to fully structured and formalized procedures. The system is capable of making all the decisions without human intervention or support. It will be clear, that these systems can only be used in highly structured application domains. A system computing the optimal route between to cities on a road map, is an example of an SDS.
  - *Decision Support Systems* (DSS), *Expert System* (ES) or *Knowledge Base Systems* (KBS) are capable of *supporting* the decision making performed by humans. While a SDS is used in highly structured application domains, a DSS (ES or KBS), is used in domains that can not be formalized completely: not all the information is available, some information is uncertain, and some problems can only be solved by using heuristics in stead of algorithms. The system does not make the decisions itself, but supports the user in his decision making by providing insight in the typically complex application domain. The terms ES and KBS are sometimes used synonymously, however essential differences between expert systems and knowledge base systems exist. An expert system is a knowledge base system with the following additional properties ([Bra91]):
    * an expert system is capable of *explaining* its behavior and its decision to the user of the system. This additional features can be necessary in uncertain domains, in order to enhance the user's confidence in the results of the system. The system can show how a certain conclusion has been drawn, i.e. by showing the derivation.
    * an expert system has the ability of dealing with uncertainty and incompleteness. With this extension it is possible to express the uncertainty (expressed by a so-called *certainty factor*) of the available information. For example, we *may* not be quite sure that some symptom is present in the patient; some drug *may* cause some problem with a certain low probability, but usually does not; most 'typical' birds fly but a penguin (also a bird) does not.

Figure 2.2 shows how the different types of information systems can be related to each other. Transaction Processing Systems form the *infra-structural* basis for the other information system, i.e. they perform storage and retrieval of relevant information in the UoD. A DSS, ES or SDS uses a TPS to store and retrieve the knowledge that is used during decision making. This is visualized by the edges in figure 2.2.

A promising new trend in the field of artificial intelligence is the development of *expert systems* ([Gle91, p. 310]). This relatively new development will be discussed in more detail in the following section.

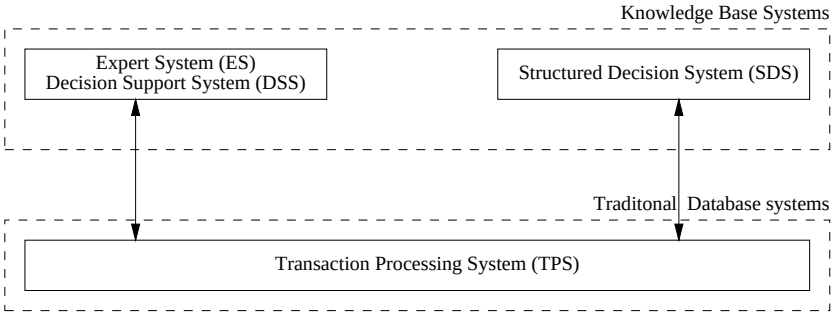


Figure 2.2: The different types of information systems

## 2.4 Expert systems

In this section focus is on a modern type of information systems that are able to perform as well as human experts in some specific application domain. These system are called *expert systems*. In the following sections expert systems are introduced by providing a system overview and a more detailed discussion on its composing components. In section 2.4.2.1 the knowledge base, which is the central component of an expert system, is discussed.

### 2.4.1 System overview

An *expert system* is a system that behaves like an expert for some - usually narrow - problem domain. Typical applications include tasks such as medical diagnoses, locating failures in certain kind of equipment, or interpreting measurement data. Expert systems are capable of solving problems that require expert knowledge in a particular domain. They should posses that knowledge in some form. When the knowledge is currently not available in the system the user may be asked to supply it (if possible). The authors of [NH89, page 9 and 10] even mention that expert systems have passed restricted *Turing tests* specific to particular universes of discourse.

One of the characteristic properties of an expert is the ability of drawing conclusions. These conclusions are based on the knowledge that is available to the expert. This knowledge includes facts, general rules and heuristics. A conclusion is drawn by using a reasoning strategy. Traditionally two different reasoning strategies can be distinguished ([Gle91, page 316]): *deduction* and *induction*. Deductive reasoning is about certainties: If certain premises (axioms) are true, then certain conclusions will follow. There are no exceptions. A general rule ("All men are mortal") is postulated and used to claim something about a particular cases ("John Smit is mortal"). In inductive reasoning however, this process is reversed: a number of different instances are considered and it is tried to determine - i.e. induce - what general rule covers them all. So, from particular cases a general rule is induced. Note that in contrast with deduction, induction can never be certain, only probable. In fact, the proposition *All men are mortal* is an induction!

Computer models of reasoning are based on the human reasoning models provided by psychologists. Therefore the different human models have resulted into different computers models of reasoning:

**Rule-based reasoning** is currently the most common form of reasoning performed by expert systems. The knowledge is described by the use of *if-thenif-then rules* rules (sometimes called *production* rules. In general, such rules are conditional statements, but they can have various interpretations, such as:

- **if** precondition P **then** conclusion C
- **if** situation S **then** action A
- **if** conditions C1 and C2 hold **then** condition C does not hold

**Frame-based reasoning** is directed to representing, in a structured way, large sets of facts. The set of facts is structured and possibly compressed: facts can be abstracted away when they can be reconstructed through inference.

**Case-based reasoning** is based on the premise that human beings use analogical or experiential reasoning to learn and solve complex problems. Problems are solved by performing the following steps: when a problem is encountered, the set of cases is consulted to retrieve cases that are most appropriate or similar to the current problem. Then a solution is created for the current problem by either modifying the solution or creating a new solution using the same process as was used in the similar past case.

**Pattern-based reasoning** is - as may be expected by its name - based on *pattern matching*. The underlying theory is that of artificial neural networks and connectionist models. Systems of this type are able to recognize patterns even when the data is noisy, ambiguous, distorted, or has a lot of variation.

In this document focus is on rule-based reasoning because this turns out to be a natural form of expressing knowledge. In [Bra91, page 334] the following additional desirable features of *If-then* rules are mentioned:

- *Modularity*: each rule defines a small, relatively independent piece of knowledge.
- *Incrementability*: new rules can be added to the knowledge base relatively independently of the other rules.
- *Modifiability* (as a consequence of modularity): old rules can be changed relatively independently of other rules.
- Support system's *transparency*: *If-then* rules facilitate the explanation of how conclusions are reached and why certain information is needed. This will be discussed in more detail in a following section called *The inference engine*.

The tasks of an expert system can be summarized as follows:

- storage and retrieval of information (facts and rules or derivation steps)
- support inference in order to solve the problem specified by the user
- checking whether hypotheses stated by the user are true or not. Or formulated alternatively: Does a derivation exist on base of the available information to prove the hypothesis?
- explaining to the user *how* new information was derived or problem was solved
- during the derivation-process, explaining to the user *why* certain information is needed

In the following section we describe the typical system architecture of an expert system in order to realize the above specified criteria.

### 2.4.2 System architecture

In this section we describe the architecture of the suggested expert system, and discuss its composing components in more detail. The suggested system architecture is depicted in figure 2.3. The following components can be recognized ([Bra91]):

**The knowledge base** is a set of statements that describe the knowledge about the truths of the actual world plus a set of constraints that describe statements that must be true in all possible worlds and statements that ought to be true in all possible worlds ([DR91]).

**The inference engine** knows how to use the available information in order to solve the problems in the domain.

**The user interface** caters smooth communication between the user and the system, also providing the user insight into the problem-solving process performed by the inference engine.

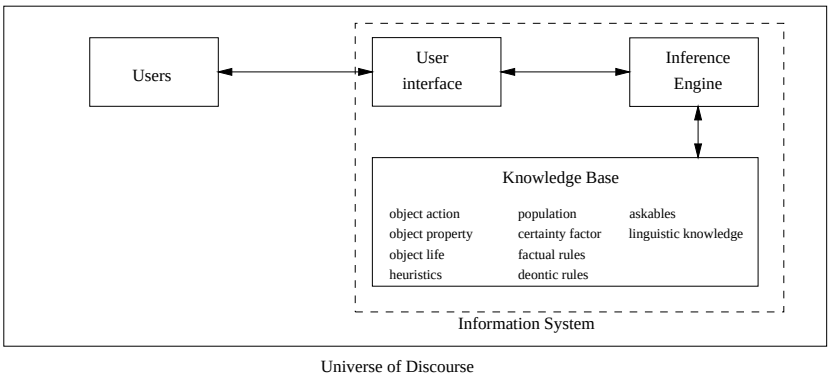


Figure 2.3: Typical expert system architecture

#### 2.4.2.1 The knowledge base

The knowledge base plays the crucial role in the system: it captures the domain specific knowledge. In principle any consistent formalism in which we can express knowledge about some problem domain can be used in an expert system([Bra91, page 334]). Within the OO-paradigm, the knowledge in the knowledge base comprises the following types of information (inspired by [Bra91], [FKW95] and [DR91].

**Object action involvement knowledge** This type of knowledge describes the types of objects and actions that can be found in the UoD. It states which object types are involved in what action types. This type of knowledge is normally captured by a data modelling technique like NIAM, ER or PSM in which fact types represent actions performed by autonomous objects.

**Object properties knowledge** This type of knowledge describes the static aspects of the UoD by modelling *properties* of objects. Strictly spoken, the introduction of properties belongs to the scope of implementation. However properties are introduced at this level to provide a more simple description mechanism. This choice is also reflected in so called 'conceptual' data modelling techniques that are capable of modelling properties by introduction of concepts like *label type* and *lexical object types* that facilitate property-modelling.

**Object life knowledge** This model considers the UoD from a historical perspective, and describes for each action type the *course of life* of its instances. Alternatively formulated, this information describes for each object type what actions *can* be performed in which order. This type of knowledge concerns dynamic behavior of objects.

**Heuristics** Heuristics are various tricks and rules of thumb that have often worked in the past and may do so again ([Gle91, p. 309]). Heuristics provide shortcuts to solutions when the problem space is too large or complex to be searched extensively (by brute force).

**Population knowledge** Describes the instances of the various object types.

**Certainty factors** Certainty factors can be used to express that some fact or rule is less than certain. Uncertainty can be modeled by assigning some qualification, other than just true or false, to assertions[1].

**Factual knowledge** (or *domain specific rules*) This type of knowledge is true at a certain moment in time, but may be changed by updates on the knowledge base. This knowledge can be used to derive new information from the knowledge base. Factual rules, which happen to be true, can be used to reduce the number of stored facts, as will be shown in the example below.

**Deontic knowledge** (or *constraints*) This knowledge consists of what is traditionally called *constraints* of the UoD. Deontic rules are used to check whether objects in the UoD behave in

[1]However, human experts seem to have trouble thinking in terms of actual probabilities; their likelihood estimates do not quite correspond to probabilities as defined mathematically

a permissible way. It is important to note that this type of knowledge - in contrast with factual rules - can **not** be used for inference purposes. Alternatively formulated, rules of this type of knowledge can not be used to derive new information because they describe a *desired* situation in stead of a *actual* situation. If the knowledge in the knowledge base is in contradiction with a deontic rule, this rule is said to be violated. If one or more rules are violated the knowledge base is in an *undesired* (but **not** inconsistent) state.

**Askables** This type of knowledge describes what kind of information may be asked from the user, and how the questions should be presented to the user. Interaction between man and machine can be realized by askables as discussed in section 6.4.

**Linguistic knowledge** is included to facilitate natural language processing and production. This part of the knowledge base is commonly known as the *lexicon*. Including linguistic knowledge serves the following goals ([BR95b]):

- to avoid semantic anomalies like *The dog buys a car.*
- to give more expressive power to modelling techniques.
- to cater smooth (natural language) communication between user and system
- to simplify the generation (and processing) of natural language sentences from the information models to support validation (see section 6.5).
- to simplify the information modelling process

Knowledge in the lexicon can be called *analytical knowledge* or *domain independent knowledge*). Analytical knowledge describes knowledge which is commonly referred to as *common sense*. Knowledge of this type is necessarily true. Since these rules are necessarily true, they cannot be changed by updates and moreover they have to be still true after any update. Of course, these rules can be used for inference purposes. If one or more rules of this type of knowledge are violated the knowledge base is said to be in in an inconsistent state.

**Example 2.1** *Consider a certain world in which persons can be male or female. A male person is called a man, and a female person is called a woman. Man and woman can be married and its possible for a man and a woman that they are having a baby. In this particular world we want men and woman to be married first, before they are having a baby. A person is married at a certain date, has a certain hair-color and dies at a certain date. In this domain all men have brown hair. (The assumed population is depicted in figure 2.4.)*

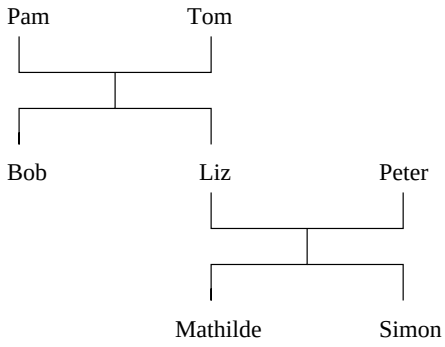


Figure 2.4: An ordinary family

This situation can be defined by using the formal conceptual data modelling technique PSM ([HW93]). The following lines provide a formal definition of the PSM schema depicted in 2.5:

(2) $\mathcal{E}$ = {person, man, woman}
$\mathcal{L}$ = {haircolor, sex, date}
$\mathcal{F}$ = {f, g, h, k, l, m, n}
where f ≡ {is_haircolor_of, has_haircolor}, g ≡ {is_parent_of, has_as_parent}, ...
$\mathcal{O}$ = {person, man, woman, haircolor, sex, date, f, g, h, k, l, m, n}
man Spec person, woman Spec person

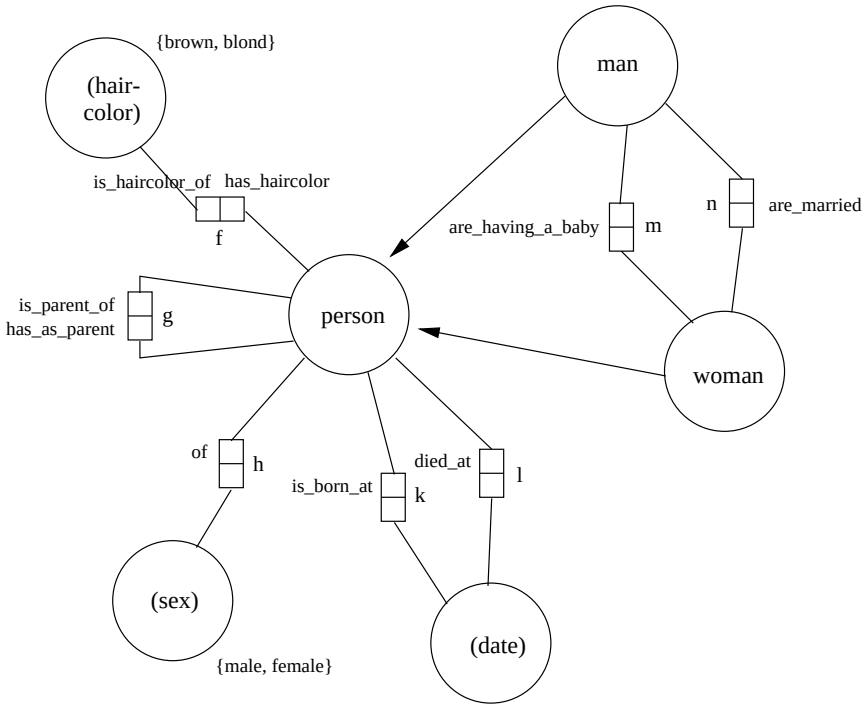


Figure 2.5: Example information architecture

A subtype defining rule is used to express how subtypes *man* and *woman* are derived from their superclass *person*. For this purpose the language LISA-D[2] ([HPW93]) can be used.

(3) man is person of sex 'male'.
woman is person of sex 'female'.

The different types of knowledge can now be illustrated with the help of this example. The following lines provide examples of factual, deontic and analytical rules. As stated before a factual rule is a domain-specific rule that can be used to derive new information. If for example it is known that all men have a brown haircolor, then we can remove the following facts from the population base,

(4) Pop(f) = {<Tom, brown>, <Bob, brown>, <Peter, brown>, <Simon, brown>}

because they can be derived from the single rule:

(5) if man(X) then has_haircolor(X, brown).

An example of a deontic rule is the following: In this particular world man and woman must be married if they are having a baby. This can be expressed by a single deontic rule:

(6) if having_a_baby(X, Y) then married(X, Y).

It is not allowed to derive new information from (6) because it expresses something that is *wanted* to be true, but not necessarily is. The following rules are examples of analytical rules, which are necessarily true, and therefore can be used to derive new information.

(7a) if parent(X, Y) then ancestor(X, Y).
(7b) if (parent(X, Y) and ancestor(Y, Z)) then ancestor(X, Z).
(7c) if (parent(X, Y) and of_sex(X, male)) then father(X, Y).
(7d) if (parent(X, Y) and of_sex(X, female)) then mother(X, Y).
(7e) if (father(X, Y) and (father(Y, Z) or mother(Y, Z)))
then grandfather(X, Z).

□ **End Example**

[2]LISA-D is an abbreviation of *Language for Information Structure and Access Descriptions*

#### 2.4.2.2 The inference engine

A stated before, the factual rules and the analytical rules can be used for inference purposes. The subsystem responsible for performing such inferences is called the *inference engine*. The inference engine can use the inference rules in two different ways, by performing a *backward chaining* or a *forward chaining* strategy.

The forward chaining strategy uses the *if-then* rules from left to right. The elements on the left hand side of the rule are input information, while those on the right hand side are derived information. Contrarily, the backward strategy starts with a hypothesis and works backwards, according to the rules in the knowledge base, toward easily confirmed findings. This strategy uses *if-then* rules from right to left. The following example which is partly based on the previous example shows how these different reasoning strategies can be applied.

**Example 2.2** The family structure of figure 2.4 can be defined appropriately with the help of a *logic programming language* like PROLOG(*Programming in logic*[3]) ([Bra91]) as follows:

```
parent(pam, bob).          of_sex(simon, male).
parent(pam, liz).          of_sex(peter, male).
parent(tom, bob).          of_sex(tom, male).
parent(tom, liz).          of_sex(bob, male).
parent(liz, mathilde).     of_sex(liz, female).
parent(liz, simon).        of_sex(mathilde, female).
parent(peter, mathilde).   of_sex(pam, female).
parent(peter, simon).
```

An expert system can use *if-then* rules to perform a backward or forward reasoning strategy. First, the forward chaining strategy is shown. As stated before, the forward chaining strategy uses the *if-then*-rules from left to right. The expert system search the knowledge base to derive all possible information by applying the analytical rules of (7) on the above specified facts about the family hierarchy. Some of the result of this reasoning strategy are given in (8). Appendix A provides a PROLOG implementation of a forward chaining algorithm.

(8)
```
father(tom, bob).
father(tom, liz).
father(peter, mathilde).
father(peter, simon).
mother(liz, mathilde).
mother(liz, simon).
grandfather(tom, mathilde).
grandfather(tom, simon).
...
```

□ **End Example**

The other strategy, backward chaining, uses the rules from right to left. This can be very useful, when we want the system to process a query, or to proof/refute a hypothesis. Take for example the following hypothesis: grandfather(tom, mathilde). The system tries to satisfy this predicate by using the available facts and rules. See the dialog below. Appendix B provides a PROLOG implementation of a backward chaining algorithm that generates a trace for explanation purposes.

**Example 2.3**

*Question, please:*

**grandfather(tom, mathilde).**

*(grandfather(tom, mathilde)) is true.*

*Would you like to see how?* **yes.**

*grandfather(tom, mathilde).*
*was derived by rule (7e) from*

[3] PROLOG is by some fanatics even called the PROgramming Language Of God.

*father(tom, liz).*
*was derived by rule (7c) from*
*parent(tom, liz).*
*Is known as a fact!*
*and*
*of_sex(tom, male).*
*Is known as a fact!*
*and*
*mother(liz, mathilde).*
*was derived by rule (7d) from*
*parent(liz, mathilde).*
*Is known as a fact!*
*and*
*of_male(liz, female).*
*Is known as a fact!*

□ **End Example**

The combination of forward chaining and backward chaining can be very useful to support information modelling. Forward chaining can be used to derive all facts that can be derived by using the current knowledge, i.e. verbalization of the information model shows all consequences of the currently chosen model. When verbalization shows a (potential) design flaw of the current model, backward chaining can be used to inform the user about *how* a certain verbalization has been established.

#### 2.4.2.3 The user interface

The user interfaces should preferably contain a natural language interface. The natural language interface ensures that a natural form of communication between user and system is established. The user interface caters smooth communication by producing and parsing natural language. Production of NL on base of the modeled information is called verbalization, and is discussed in more detail in section 6.5. Parsing on the other hand, is involved in query processing and is not further discussed in this document . A graphical interface can be used to provide an abstract overview of some part of the knowledge present in the knowledge base.

# Chapter 3

# Natural Language

## 3.1 Introduction

In this chapter a short introduction to *natural language* (NL) is provided. In [BH89] it is argued that a sound definition of the term *language* covering all it's necessary and sufficient aspects can not be given. Although linguistics has an empirical nature, it can only be defined as that which is studied by linguistics. Although this definition of language might be unsatisfactory, alternative definitions are often incorrect or incomplete, take e.g. the following attempt:

**language** the sounds produced by the vocal tract which have a meaning and are used to communicate.

Note, however that language is not always produced by the vocal tract (writing) and language is not always used to communicate (muttering).

When studying natural language a clear separation between *competence* and *performance* of NL has to be made. Competence is the knowledge a human being possesses about his mother tongue. This can be seen as a description of language as a system of signs and meanings. The performance, on the other hand, is about *using* that competence by talking or writing natural language. Although the knowledge about natural language is reflected in the use of it, language is sometimes used in such a way that it conflicts with our knowledge of how it *should* be used. During conversation While talking, e.g. we restart, correct or even do not complete sentences.

The systematic part of language has been studied intensively by Noam Chomsky. Chomsky focussed his attention on the systematic part, i.e. competence, of natural language which he used as a the basis of his transformational-generative grammar (TGG) ([Fah91]). The distinction between competence and performance is also mentioned by Ferdinand De Saussure - a Swedish linguist, who stressed that the 'langue'(read competence) should be separated from the 'parole' (read performance). The 'langue' should be the subject of scientific analysis and should be studied by linguists. The 'parole', however, should be studied by psychologists and sociologists.

In this document our attention is focussed on competence. In the discussion of language, English will be used as our main example. But it is important to keep in mind that what is said about English generally goes for the other human languages as well. In the following sections the major properties like structure and meaning of NL are discussed.

## 3.2 Properties of Natural language

In [Gle91] five major properties are mentioned: language is *creative*, *interpersonal*, *structured*, *meaningful* and *referential*. The property of creativity states that humans are able to utter and understand sentences they have never heard before. Therefore using language is not a kind of

memorization that performs speech acts whenever the appropriate circumstances arise: language is a creative process. With the next property, inter-personality, it is stated that using language is mostly a social activity in which the thoughts of one mind are conveyed to another. The last three properties mentioned, will be discussed in more detail in the following sections.

### 3.2.1 Language is structured

Although using language is a creative process, it is also restricted: there are unlimited numbers of strings of English words that we would not utter. For example we do not say:

*The throws John ball*

Our utterances are conform the abstract principles of the language we use. The *structural principles* define how we can combine words into meaningful and comprehensible sentences. In general we are not aware of using these principles. Even so, these principles determine our use of language and allow us to compose and understand boundless new sentences. This structural part of language seems to be very systematic and therefore seems to be a good candidate for formalization. In a following section more details and examples of the structure of natural language are provided.

### 3.2.2 Language is meaningful

Traditionally natural language can be divided into:

- vocabulary, consisting of all the words of a natural language.
- grammar rules, or structural principles, describing the rules that are consulted to construct sentences by combining the words from the vocabulary.

Each word in the vocabulary represents a meaningful *idea* (also called *concept*) about something (e.g. *ball*), action (e.g. *throw*), abstraction (e.g. *justice*), quality (e.g. *heavy*), etc. In general the relation between word and concept is defined arbitrarily. The purpose of language is to express all the meanings of our utterances (references to concepts) to others, so we have no choice but to learn and memorize these relations.

But people talk in sentences rather than just one word at a time, and the structuring of words, i.e. composing sentences, form a major contribution to the meaning of our utterances. For example the words *John*, *Peter* and *killed*, can be combined in two different sentences with very different meanings.

*John killed Peter.*
*Peter killed John.*

Keep in mind that the difference in meaning between the above two sentences is a result of different ways of structuring the same words. Handling meaning of sentences, will be discussed in more detail in a following section.

### 3.2.3 Language is referential

Besides the fact that we know how to put words into meaningful sentences, we also know which words refer to which things, scenes and events in the real world around us. Take as an example the sentence:

*That's a rabbit.*

The words are put together into a sentence in a correct and meaningful way. However, when this sentence was uttered by a little boy while pointing at a dog, we would not think he has learned English very effectively. This problem is called the problem of *reference*: how to use language to describe the world of real things and events. This problem is quite complex and belongs to the field of psychologists.

The discussion on meaning and reference of words (and sentences) has also been studied by classical philosophers. But the modern history of the philosophical discussion on meaning has been started by John Locke.

John Locke (1632-1704), an English philosopher, described in his *Essay Concerning Human Understanding*, published in 1690, that words are used to represent the ideas of the speaker: the meaning of a word is the idea that a speaker has in mind when he uses the word and the idea, the listener creates mentally when he hears the word. These ideas arise from perceptions. So to his opinion meaning and reference are each others equals. This theory can be successfully applied on words that refer to concrete objects in our real world. However, it does not hold for words that refer to *abstract* entities like *justice* and *conclusion*.

In the twentieth century the meaning of natural language was greatly influenced by mathematical logic. In the article *Uber Sinn und Bedeutung*, published by Gottlob Frege (1848-1925) in 1892, the distinction between reference and meaning was made clear. Frege, a German mathematician and linguistic philosopher, separated the *reference* of a word (also called *denotation* or *extension*) from the *meaning*[1] of a word (also called *intension*).

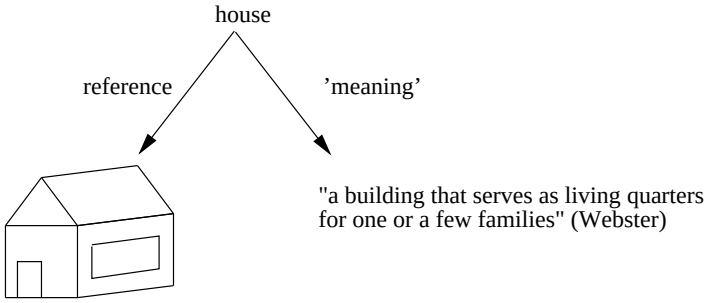


Figure 3.1: The meaning is separated from the reference

Note that in this theory two expressions with different meanings can refer to the same reference-object. Take e.g. the following two expressions:

*George Washington*
*the first president of the united states*

Note that the problem of words referring to abstract entities is unlinked from the meaning but still remains unsolved in the view of some philosophers. The problem of reference is an important issue in the field of psychology and philosophy but is beyond the scope of this document. Our attention will be focussed on the structure and the meaning (semantics) of natural language.

## 3.3 Syntax of Natural language

In this section the structure of NL will be investigated and different abstraction levels are distinguished.

Sentences produced by *natural language users* (NLUs) are built by grouping phrases. These phrases on their turn are composed of words, and these words can also be split in smaller units called *morphemes*. A morpheme is a sequence of *phonemes*. This hierarchy is depicted in figure 3.2.

Phonemes are described by symbols from the phonetic alphabet. With about 40 phonemes, the 80,000 or so morphemes, and the hundreds of thousands of words can be constructed. The study of phonemes is called *phonology* and the study of morphemes *morphology*. The term *syntax* (i.e. 'arranging together' (Greek)) is the name of the system that arranges (or groups) words together into meaningful sentences. This topic has been investigated intensively by the American linguist, Noam Chomsky.

As stated before, syntax of natural language consists of two parts:

[1] The meaning is given by means of a definition. In a following section other approaches to the problem of representation of meaning are presented. The problem of meaning is still a research issue.

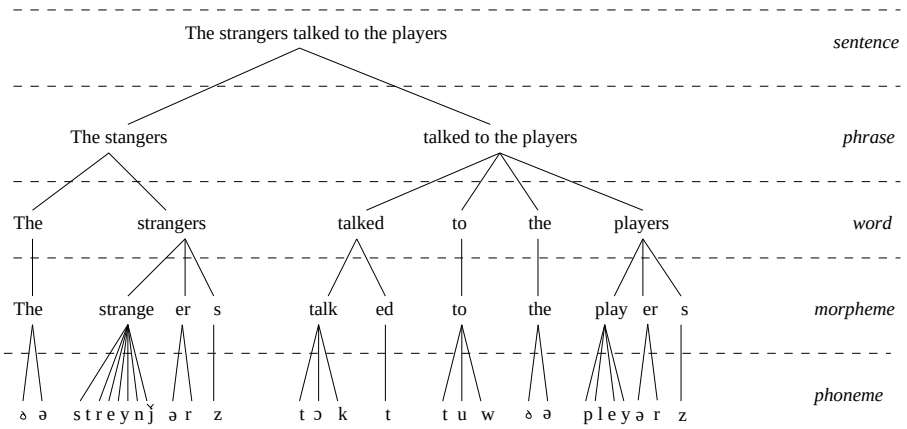


Figure 3.2: The hierarchy of linguistic structures

- *syntactical categories* i.e. the word families. All elements from a certain family can be exchanged with each other without changing the structure of the sentence.
- *syntactical rules* describing the possible arrangements of the syntactical categories.

Traditionally ten syntactical categories can be distinguished ([Mil93]): *noun*, *adjective*, *verb*, *adverb*, *pronoun*, *determiner*, *preposition*, *conjunction*, *numeral* and *interjection*. A category can be divided into sub-categories; these sub-categories can be divided into sub-subcategories again. Take for example the category of nouns. The noun can be divided into *proper nouns* (e.g. 'John', 'Mary', 'The Beatles') and *common nouns*. The common nouns consist of *countable nouns* (book, table) and *non-countable nouns* (milk, grain, confidence).

Another example is the category of verbs. The verb-category can be split into three subcategories: *lexical verb* (*to hear*, *to see*, *to register*), *auxiliary* (*to have*, *to be*) and verbs expressing *modality* (*can, must, may, shall, will*). The lexical verbs can be divided again into *intransitive*, *transitive* , *pseudo-transitive* and *ditransitive* verbs. An intransitive verb (e.g. *to sit*, *to sleep*, *to smile* and *to talk*) can not be combined with a direct object. For example we do not say[2]:

⋆(1) John sleeps the bed.
⋆(2) Mary smiles the dog.

Transitive verb (e.g. *to build*, *to adore*, and *to devour*) however need a direct object. See the examples below:

(3) John adores his mother-in-law.
⋆(4) Mary built.

Sometimes verbs belong to both categories mentioned above. These verbs *can* have a direct object, but this is not necessary. This category of verbs is called the pseudo-transitive category and contains verbs like *to eat*, to drink and to *read*. This situation is shown by the examples given below:

(5) Peter drinks his milks.
(6) Peter drinks.
(7) Anne reads a book.
(8) Anne reads.

The last category of verbs is called ditransitive and contains verbs like *to sell* and *to give*. The verbs in this category can have both a direct and indirect object.

(9) John sells the car.
(10) John sells the car to Mary.
(11) Anne gives the book to Peter.

[2] A ⋆ indicates the 'invalidity' of the sentence.

Now, different categories for words have been defined, it needs to be studied how words and categories are related to each other. Obviously a category contains one or more words. On the other hand, a word can belong to more than one category. A word that belongs to more than one category is a hot topic of discussion. In English, it occurs very often that a word belongs two more than one category (e.g. *house*, *work*, *play*, *back*, *paper*, *surface* etc.). Some believe that such a word should be seen as a single word, while others say that we have to do with different words. The word *house* as a noun has a phonetic representation that slightly differs from the word *house* as a verb. In written language however this difference can not be observed and may lead to confusions. Take for example the sentence below. This sentence has two possible interpretations because both words (*bottle* and *smell*) belong to two different categories, namely noun and verb.

(12) The French bottle smells.

When *bottle* comes out as a verb, the sentence is telling us something about what the French put into bottles - namely smells (i.e. perfumes). In another interpretation, the sentence is telling us something about the French bottles.

A sentence that can be interpreted in more than one way, is called *ambiguous*. In the following lines a description of the syntactical rules is provided. With these rules ambiguity can be defined formally.

As mentioned before, syntactical rules describe how syntactical categories can be combined into sentences. Noam Chomsky has researched this intensively, and a lot of theories are based on his results ([BH89]). In his book *Aspects of The Theory of Syntax*, published in 1965, Chomsky started the syntactical research. This has resulted in a set of rules that describe how to combine the different syntactical categories. These rules can be expressed by so-called production rules of a context-free grammar ([AU77]).

A sentence S can be seen as a sequence of a *noun phrase* (NP) followed by a *verb phrase* (VP).

(SR1) $S \rightarrow NP\ VP$

A noun phrase consists of a noun, that can be preceded by a *determiner* (Det) and followed by a prepositional phrase (PP). A noun can be preceded by an unlimited number of adjectives (A). A prepositional phrase is a preposition (P) followed by a noun phrase.

(SR2) $NP \rightarrow (Det)\ \bar{N}\ (PP)$
(SR3) $\bar{N} \rightarrow A\ \bar{N}$
(SR4) $\bar{N} \rightarrow N$
(SR5) $PP \rightarrow P\ NP$

Because of the different syntactical subcategories of a verb, a verb phrase can be constructed in different ways. The intransitive verb has no (in)direct object, the ditransitive can have both a direct and indirect object. In the following lines the necessary syntactical rules for the intransitive, transitive and ditransitive verbs are provided. A verb is represented by the symbol V.

(SR6a) $VP \rightarrow V$
(SR6b) $VP \rightarrow V\ NP$
(SR6c) $VP \rightarrow V\ NP\ PP$

With these rules sample sentences can be parsed and generated automatically ([DKNZ92]). The above mentioned rules are also called rewrite rules. A sentence can be represented by a parse tree in which each internal node represents the application of one of the rewrite rules. In figure 3.3 the parse tree of the sentence *The neighbor of my brother sells the car to Mary* is depicted.

Ambiguity can now be defined as follows: a sentence is ambiguous if it has more than one parse tree. The sentence *The French bottle smells* is indeed ambiguous conform this definition.

Although these rewrite rules are a powerful mechanism to analyze and describe sentences from our language, they are not powerful enough to describe sentences like (13b) and (14b).

(13a) Peter read the book.
(13b) Did Peter read the book?
(14a) The barbarians destroyed Rome.
(14b) Rome was destroyed by the barbarians.

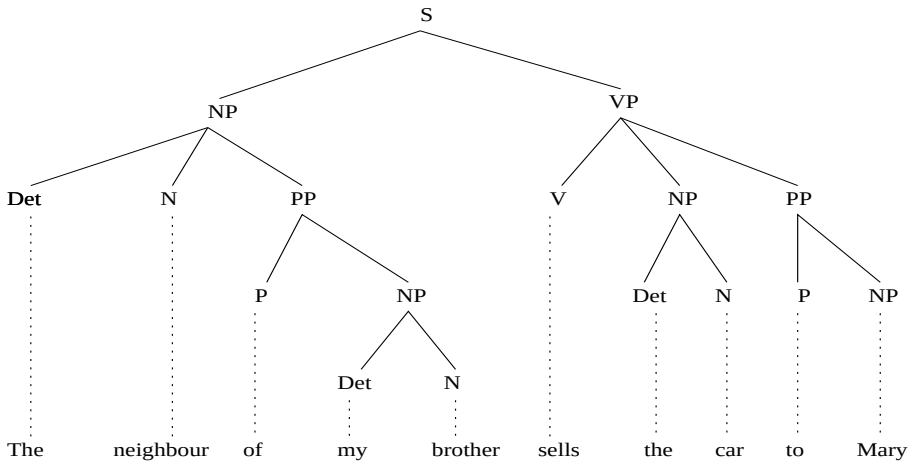


Figure 3.3: The parse tree of a sample sentence

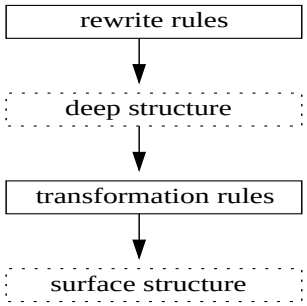


Figure 3.4: The model of the transformational - generative grammar.

Chomsky choose not to extend the rewrite rules with rules to produce interrogative (13b) and passive (14b) sentences, but introduced another type of rule: the so-called . It is clear that in contrast with (13b) and (14b), sentence (13a) and (14a) can be produced by the above mentioned set of rewrite rules. A transformation rule can now be applied to (13a) to produce (13b). This rule transforms a declarative sentence into an interrogative sentence. Another transformation rule is used to transform the active sentence (14a) to the passive sentence (14b). So, to recapitulate, the sentences (13b) and (14b) are produced indirectly, first by the rewrite rules - producing a so-called *deep structure* - secondly by the transformation rules. producing the so-called *surface structure* ([BH89]). This situation is depicted in figure 3.4.

Transformation rules can be defined arbitrarily. A transformation rule can transform everything into everything. Restrictions are needed to distinguish meaningful transformation from senseless transformation rules. The restrictions on these rules can only be obtained by induction on an empirical basis. The last 25 years of research on syntactical aspects of language has been focussed on finding empirical restrictions and on formulating these restrictions adequately. These issues still play an important role in current research.

Rewrite and transformation rules facilitate the description of language and to check sentences on well-formedness on a syntactical level. In the following section it is described how semantical anomalies, i.e. sentences without an explicable meaning (e.g. *The table writes a letter*), can be excluded.

## 3.4 Meaning of Natural language

In this section is is described how the meaning of our utterances can be defined and used. A discussion on the meaning of single words is provided, followed by a discussion on meaning of complete sentences.

### 3.4.1 The meaning of words

It is generally accepted by linguists that the meaning of a word is a *concept*. So a word is just a representative of a meaningful idea (or concept) in the mind of the person using that word. During communication we try to express these concepts to others. The question, "How to express these concepts ?", is one of the most difficult questions in the study of language. The subfield that attempts to answer this question is called the study of *semantics*.

As stated before, one of the oldest approaches to the topic of meaning equates concepts of words and phrases with reference. We have already seen this theory of meaning results into several difficulties. Therefore two other approaches are defined: the *definitional theory of meaning* and the *prototype theory* of meaning.

#### 3.4.1.1 The definitional theory of meaning

In figure 3.1 the word *meaning* is placed between quotation marks, because - as just stated - the meaning of a word is a concept instead of a definition. A concept is an idea in someone's mind, a definition however is written by the author of a lexicon. It is the definitional theory of meaning that assumes that these two fulfill the same role.

This approach states that meanings can be analyzed into a set of subcomponents, organized in our minds much as they are in standard dictionaries. Various meaning (or semantical) relationships exist between words and phrases. Some words are similar in meaning (*synonym* relationship) and other are opposites (*antonym* relationship). According to this approach these relations can be explained by assuming that words are sets of *semantic features*. The concept of *bird* for example contains the semantic features *feathers*, *flies*, *animal* and *wings*.

Taken together, the semantic features constitute a definition of a word. According, to this theory, we carry such definitions in our heads as the meaning of words in a so-called *mental lexicon* ([Mil93, pages 121-143]).

Although this approach is intuitively satisfying, some problems are brought into being: language itself is used to express the meaning of language-elements. To break this vicious circle, philosophers - from Plato to Leibniz - have been trying to make a list of so called axiomatic 'base-words', that can be used to describe the meaning of all other words ([Mil93, page 155]). Up to now all these attempts have failed.

#### 3.4.1.2 The prototype theory of meaning

Another approach that attempts to define concepts is the *prototype theory* of meaning. This theory explains the fact that some members of a meaning category appear to exemplify that category better than others do. The definitional theory, lists the necessary and sufficient semantic features that *define* a concept. However an *armchair* seems to be a better example of the concept of *furniture* than a *reading lamp*. An armchair is a typical piece of furniture, a reading lamp is not. This difference cannot be explained by the definitional theory of meaning. It claims to have said it all by the following definition of *furniture* (conform Webster's dictionary).

**furniture** movable articles used in making a room ready for occupancy or use

The prototypical theory states that a concept is defined by a whole set of features, no one of which is individually either necessary or sufficient.

Consider e.g. the concept of 'bird'. Prototype theorists claim that 'birdiness' is determined by the total number of features a given creature exhibits. Animals that have few (penguins and ostriches) will be judged to be pore members of the bird family, while those that have many (such as robins) will seem to be a strong member (or prototype).

It appears that both the definitional and the prototypical approaches to word meaning have something to offer. To prototype theory explains why a robin is a 'better' bird than an 'ostrich', and the definitional theory says that an ostrich should nevertheless be classified as a bird (conform Webster's dictionary).

| lexical item | smile | eat | sell |
|---|---|---|---|
| category | V | V | V |
| sub-categorization | <> | $< (NP) >$ | $< NP(PP) >$ |

Table 3.1: The lexicon contains the context sensitive information

| lexical item | talk |
|---|---|
| category | V |
| sub-categorization | $< (PP_{[animate]}) >$ |

Table 3.2: An example of a selection restriction

**bird** any of a class of warm-blooded vertebrates distinguished by having the body more or less completely covered with feathers and the forelimbs modified as wings

### 3.4.2 The meaning of sentences

So far, it has been discussed how to handle the meaning of words. When studying the meaning of sentences we come to a problem at a higher level of complexity. It has been shown that the meaning of a single word is a concept which can be 'described' by means of a definition or a prototype. The meaning of a complete sentence on the other hand, has to do with relations *between* these concepts. Evidently, the context sensitive nature of NL - ignored by the rewrite-rules of the transformational-generative grammar - supply a major contribution to the meaning (i.e. semantics) of NL.

Basic sentences introduce some concept that they are about (called the *subject* of the sentence). Simon C. Dik mentions ([Dik89, page 23]) that the term 'subject' only makes sense when a certain context (i.e. the *predicate* of the sentence) is provided. In a sentence like (15) *The boy* is called the subject, and what is proposed or predicated of this concept is that he *hit the ball.*

(15) The boy hit the ball.

Generalizing from this example, the meaning of a sentence (sometimes called *proposition*) can be regarded as a sort of miniature drama in which the verb is the action and the nouns are the performers, each playing a different role. In the example above of the *boy-hitting-ball* miniature drama, the *boy* is the "doer", *the ball* the "done-to" and *hit* is the action itself ([Gle91]). This *action-oriented* viewpoint treats verbs as basic frames to be filled with concepts. This approach enables us to describe the meaning of a sentence precisely.

However, the linguistic theory of the transformational-generative grammar, is a formal attempt at structuring syntactical categories in terms of rules of formal syntax to be applied *independently* of the meanings. In this theory syntax is thus given priority over semantics. Of course, semantics is recognized by this theory, but is not its basic assumption. How this linguistic theory handles semantics will be described shortly.

The transformational-generative theory states that context sensitive information should be stored, separately from the rewrite-rules, in a so-called *lexicon*. This implies that the context sensitive information describing to which sub-category a verb belongs, should also be stored in the lexicon (see table 3.1).

Note that this context sensitive information is needed for the benefit of a correct deep-structure generation. Therefor the lexicon is consulted *during* deep-structure generation by choosing the appropriate lexical items to fill the structure. With this information semantic anomalous sentences like *John walks the house.* can be excluded.

Another form of context sensitive information are the so-called *selection restrictions*. The selection restriction expressed in table 3.2, states that the PP of *to talk* should be animate. With this restriction rule, semantic anomalous sentences like *John talks to the table* can be excluded.

Another linguistic theory, called functional grammar, focuses on semantics, and therefore seems to be a better candidate to deal with semantics of NL. This approach will be discussed in the following chapter.

# Chapter 4

# Semantics of NL for Information Modelling purposes

## 4.1 Introduction

In the previous chapter the major properties of natural language were discussed. An introduction to semantics of natural language was also provided. In this chapter a more detailed discussion on semantics is given. The linguistic theory introduced in this chapter attempts to analyze and describe natural language from a different point of view. In contrast with the syntax oriented approach of the transformational-generative grammar, this approach focuses on semantics and can therefore be called the *semantical approach*. The underlying linguistic theory of this semantical approach is the theory of *Functional Grammar* (FG) ([Dik89]). The following sections describe the suitability of (the semantical representation of) NL used for information modelling. A formal semantical representation of NL is introduced to show the expressive power of semantic-oriented modelling techniques.

Philosophers like Wittgenstein and Quine state that all knowledge is expressible in (natural) language and even that only natural language can be used to express all knowledge. This makes natural language a good candidate for describing our knowledge about the UoD. As discussed in chapter 2, the knowledge base contains all available knowledge about the UoD. The language that is used to express this knowledge, is called a *Knowledge Representation Language* (KRL). Therefore it seems that NL is a good candidate for KRL. The following advantages of using NL as KRL can be mentioned (see the requirements stated in section 1.1):

- *expressive power*, as suggested by Wittgenstein, all knowledge is expressible in (natural) language.
- *comprehensibility*, humans at a certain age are assumed to be able to produce and understand natural language. This makes NL a universal communication medium between all individuals belonging to the group of information analysts, domain experts and end-users.

Although the above mentioned advantages are tremendous, natural language also has some major drawbacks that have a negative influence on the suitability of NL as a KRL:

- *formal*: A well-formed formal semantics is needed to avoid ambiguities and inconsistencies. A single formalism should be able to capture all knowledge. When the idea of a single formalism is related to natural language, we encounter the following problems:
  - As illustrated in chapter 3 natural language is notoriously *ambiguous*. Providing a complete formalism as suggested by the transformation-generative grammar of Chomsky seems to be an enormously complex job.

  - Another problem is the existence of so many natural languages[1]. Choosing a specific language makes the specification heavily dependent on the syntax of that specific language. Because language is just a means to express meanings and thoughts in someone's mind, it is more probable that agreement can be reached on a semantic than a syntactic level.

- *executable*: The lack of a formal foundation makes it impossible to execute the specification.

To solve the above specified problems the *semantic representation* of natural language in stead of natural language itself can be used. The semantic representation of natural language defines the exact meaning of a sentence and thus resolves the problem of ambiguity. On the other hand the semantic representation of natural language is less dependent on the natural language, because this should be the same for all languages, providing a universal semantical framework. In [Mos79] this hypothesis is put into words as follows:

> "It is one of the central hypothesis of case grammar and of other semantically-based models of language that, while languages may differ in their syntactic rules and surface structures, they share a universal semantic base."

## 4.2 A Semantical Linguistic theory

This section introduces a linguistic theory that focuses on semantics instead of syntax. This linguistic theory provides a theoretical foundation for the semantic representation of NL.

### 4.2.1 The functional grammar paradigm

According to Functional Grammar a language is in the first place seen as an instrument of social interaction among human beings. This in contrast with the *transformational-generative grammar* (TGG) theory that in the first place treats language as an abstract formal object. In TGG a grammar is conceptualized primarily as an attempt to characterize this formal object by applying rules of formal syntax independently of their meanings.

In his book *The Theory of Functional Grammar*, Simon C. Dik, criticizes the transformational-generative grammar [Dik89, page 16 and 17] as follows:

> "In the early seventies, many studies appeared with arguments to the effect that "X is really Y", trying to demonstrate that what at first sight looked like X (where X could be a category or construction) was in fact only the outward manifestation of some "deeper" category or construction Y. Thus, there were arguments that "pronouns were really noun phrases", "articles were really pronouns", "quantifiers are really predicates", etc. ...
>
> ...I believe that a simple lesson can be drawn from this situation: Take languages seriously. Whenever there is some overt difference between two constructions X and Y, start out on the assumption that this difference has some kind of functionality in the linguistic system. Rather then pressing X into the preconceived mould of Y, try to find out why X and Y are different, on the working assumption that such a difference would not be in the language unless it had some kind of task to perform."

The functional theory suggests to specify the *relations* of constituents to the construction in which they appear, instead of specifying the intrinsic properties of the constituents. So the functional approach takes *relations* between the concepts as its basic assumption. This *relation-oriented* approach can be illustrated with the following statements clarifying the differences between functional (read relational) and categorical notions.

[1] There are about 4,000 languages now in use on earth ([Com87])

(1) *The old man* is a noun phrase (NP)
(2) *The old man* is subject

Statement (1) makes sense as it stands. A meta-linguistic predicate such as "being a NP" tells us something about the intrinsic properties of constituents, and can thus be predicated of such constituent independently of its context. Statement (2) however, does not make sense without a context. The predicate of "being a subject" needs a context to be meaningful. As a result the notion of subject defines a relation (or function) *between* constituents. The theory of FG takes these functional statements as its basic concepts and states that linguistic expressions are networks, characterized by functional relations operative at different levels. FG recognizes the following functional relations at three different levels ([Dik89, page 24]):

**Semantic functions** specify the roles played by the referents of the terms involved within the *State of Affairs*[2] (SoA) described by the predication. Semantic functions recognized by FG are described in table 4.1. The first column shows the position in a verb-frame at which the semantic function can be applied. Intransitive verbs (e.g. *to walk*) have only one argument position, while a ditransitive verb, e.g. *to give*, has three argument positions (*to_give(Arg1, Arg2, Arg3)*). Semantic functions play a crucial role in the remaining part of this document .

**Syntactic functions** specify the "perspective" from which a SoA is presented in a linguistic expression (examples are *subject* and *object*). The role of syntactic functions is explained in the discussion on verbalization in section 6.5.

**Pragmatic functions** specify the "informational" status of a constituent within the wider communicative setting in which it occurs (examples are *theme*, *topic* and *focus*). A detailed discussion on the complex study of pragmatics is outside the scope of this document .

| Position | Function | Description / Example |
|---|---|---|
| First | Agent | the entity controlling an action |
| | | *(John)$_{Ag}$ kissed Mary.* |
| | Positioner | the entity controlling a position |
| | | *(John)$_{Po}$ kept his money in an old sock.* |
| | Force | the non-controlling entity instigating a process |
| | | *(The earthquake)$_{Fo}$ moved the rock.* |
| | Processed | the entity that undergoes a process |
| | | *(The apple)$_{Proc}$ fell from the tree.* |
| | Zero | the entity primarily involved in a state |
| | | *(The cup)$_{Zero}$ was on the table.* |
| Second | Goal | the entity affected or effected by the operation of some[3] controller (Agent/Positioner) or Force |
| | | *John kissed (Mary)$_{Go}$.* |
| Second or third | Recipient | the entity into whose possession something is transferred |
| | | *John gave the book to (Mary)$_{Rec}$.* |
| | Location | the place where something is located |
| | | *John landed on (Mars)$_{Loc}$.* |
| | Direction | the entity towards which something moves/is moved |
| | | *John drove to (London)$_{Dir}$.* |
| | Source | the entity from which something moves/is moved |
| | | *The apple fell from (the tree)$_{So}$.* |
| | Reference | the second or third term of a relation with reference to which the relation is said to hold |
| | | *The boy resembles (his father)$_{Ref}$.* |

Table 4.1: Semantic functions recognized by FG

[2] A State of Affairs is the conception of something that can be the case in some real or postulated world.
[3] Goal is used in the sense of *Patient* (as in [Blo33])

In (4) we see that the linguistic expression of (3) can be described by using functional statements on constituents. In (4) the constituents *the boy* and *the ball* are related to the constituent *to hit* by the semantic functions *agent* and *goal* respectively.

(3) The boy hits the ball.
(4) hit(The boy$_{Agent}$, The ball$_{Goal}$).

Figure 4.1 shows how the linguistic expression *I believe that John gives the book to the librarian* can be formed. Linguistic expressions are the result of a mapping on the underlying clause structure. This mapping of clause structure to linguistic expression is controlled by so-called *expression rules*[4]. Expression rules play an important role in verbalization of an underlying clause structure into a linguistic expression. This is discussed in more detail in section 6.5.

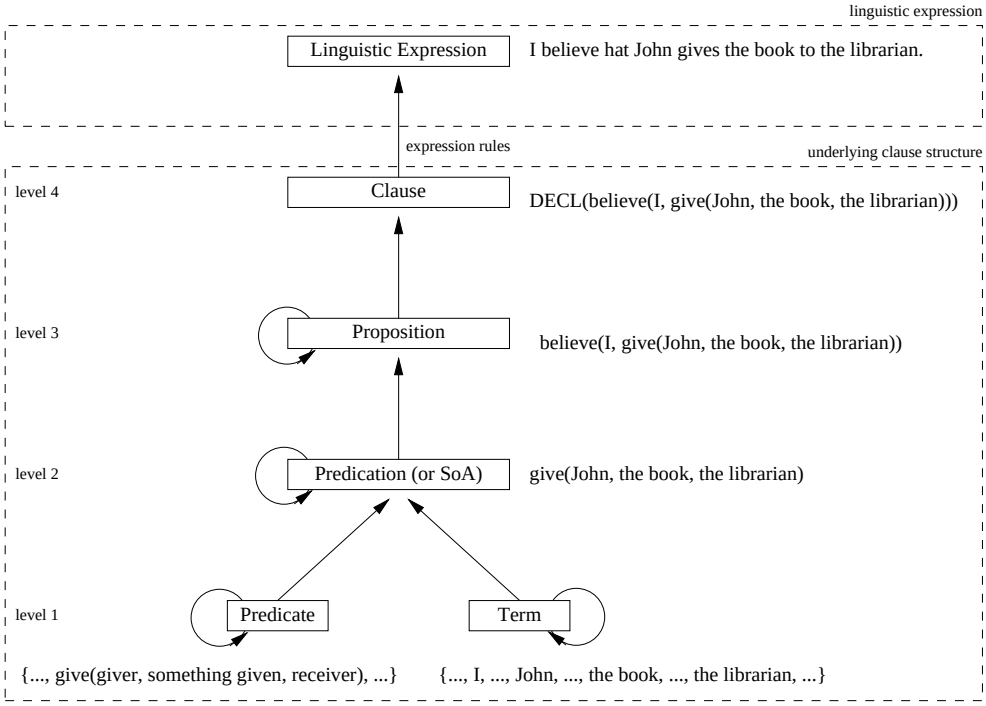


Figure 4.1: The functional grammar paradigm

Predicates and terms can be simple or complex. Simple predicates and terms are contained in the lexicon. Complex predicates and terms are not stored in the lexicon but are produced by rules. These formation rules are stored in the lexicon. A predicate can be seen as a kind of frame. Terms fill the argument positions of a predicate frame. A predicate specifies various types of information. These types of information will be discussed with the help of the following sample predicate frame of the English verb *to give*.

(5) give$_V$ ($x_1$:<animate>($x_1$) $_{Ag}$, $x_2$ $_{Go}$, $x_3$ : <animate> ($x_3$) $_{Rec}$ ).

**form** : by convention, the lexicon provides the written form of the infinitive. However in some cases, more information like the *stem* and irregular forms should also be provided by the lexicon.

**syntactic category** of the predicate, in this case V(erb). Examples of other categories are adjectival (A) and nominal(N).

**quantitative valency** specifies the number of arguments the predicate can take. In this case three, denoted by $x_1$, $x_2$ and $x_3$.

[4]Psychologists believe ([Gle91, page 350 and 351]) that *Natural Language Users* (NLUs) actually, mentally represent sentences in two different ways. The first is the surface structure, that contains the linguistic expression as uttered by the NLU. The second is the underlying structure of the sentence. This two level hierarchy corresponds to the two level hierarchy of the functional paradigm.

**qualitative valency** specifies the types of arguments the predicate can take. The type of an argument is specified by means of a semantic function. As stated before, semantic functions define the roles played by the arguments of the predicate. In this case the first argument plays the role of *agent*, while the second and third argument play the role of *goal* and *recipient* respectively.

**selection restriction** such as <animate> define the semantic type of the terms which can be inserted into the argument position ( non-metaphorical use of language is assumed).

The predication (or SoA) forms the kernel of the underlying clause structure. A predication is the result of a predicate applied to an appropriate set of terms. The predicate of (5) applied to the terms *John*, *the book* and *the librarian* results in the predication *give(John, the book, the librarian)* as shown in figure 4.1. While simple predications are constructed by combining a predicate with the appropriate terms, a complex predication can be constructed by combining a predicate with one or more predications and a set of terms. The construction of complex predication is visualized in figure 4.1 by the loop at level 2. An example of a complex predication is given in (6).

(6a) I see that John gives the book to the librarian.
(6b) $\text{see}_V$(I, $e_1$)
where $e_1 \equiv \text{give}_V(\text{John}_{Ag}, \text{the book}_{Go}, \text{the librarian}_{Rec})$

Now consider a linguistic expression like (7).

(7) I believe that John gives the book to the librarian.

Although the subordinate clause of (7) has the same form as in (6), its semantic status is different. The things that people can say to believe or not are not SoAs, rather they are *propositions*. Propositions can be said to be known or thought about; they can be reason for doubt; the can be mentioned, denied, rejected and remembered; and they can be said to be true or false. Therefore the predicate believe takes propositions rather than predications for its second argument. Thus a SoA (of level 2) can be built into a higher-order structure: the *proposition* (of level 3). The predication of (6b) however does not fully describe the linguistic expression of (6a): the *speech act* is not yet included. The sentence of (6a) is declarative in stead of interrogative or imperative. This is stated by the keyword DECL that is used to built a proposition (of level 3) into a *clause* (of level 4). The clause of (8) provides a full analysis of the sentence *I believe that John gives the book to the librarian.*

(8) DECL( $\text{believe}_V$(I, $\text{give}_V$(John, the book, the librarian)))

Now consider the linguistic expressions (9a), (10a) and (11a). These sentences represent the same SoA, but differ only in time and space. In FG these differences can be made with the help of *satellites* and *operators*. Terms like *in the library* in (10a) are used to provide more information about the SoA and are called *satellites*. While semantic functions provide the *necessary* information of a SoA, satellites provide *additional* information about the SoA. Satellites recognized by FG are listed in table 4.2. The *predication operator* PAST can be used to locate a SoA in the time interval preceding the moment of speaking $t_0$. Another essential dissimilarity between operators and satellites exist. Operators are used to capture those modifications and modulations which can be brought about at the relevant level by *grammatical means*; satellites those that can be brought by *lexical means*. FG distinguishes operators and satellites at all levels from one to four. Satellites and operators recognized by FG are discussed in (more) detail in [Dik89, pages 184-185, 192-197, 202-208, 251-253, 255-260]. Table 4.2 provides some examples of level 1 and 2 satellites.

(9a) John gives the book to the librarian.
(9b) $\text{give}_V(\text{John}_{Ag}$, the $\text{book}_{Go}$, the $\text{librarian}_{Rec})$
(10a) John gives the book to the librarian in the library.
(10b) $\text{give}_V(\text{John}_{Ag}$, the $\text{book}_{Go}$, the $\text{librarian}_{Rec})$ (in the library)
(11a) John gave the book to the librarian.
(11b) PAST[$\text{give}_V(\text{John}_{Ag}$, the $\text{book}_{Go}$, the $\text{librarian}_{Rec})$]

An example of a typical level 2 operator is the *tense operator*. Tense operators are used to grammatically code temporality. Temporality distinctions serve to locate the SoA at some time interval along the time axis. Examples of the tense operator are given in (12), (13) and (14).

| Level | Satellite | Description / Example |
|---|---|---|
| 1 | Beneficery | is the person or institution for whose benefit (sometimes against whose interest) the SoA is effected. |
| | | *John bought some flowers for (Mary)$_{Ben}$.* |
| | Company | specifies an entity together with whom the SoA is effected. |
| | | *John went to Paris with (Mary)$_{Com}$.* |
| | Instrument | specifies the tool with which some action is carried out or some position is maintained. |
| | | *The thief broke the window with (a brick)$_{Instr}$.* |
| | Manner | specifies the way in which an action is carried out, a position is maintained or a process occurs. |
| | | *Wendy danced (gracefully)$_{Man}$.* |
| 2 | Location | the place at which a certain SoA takes /took place |
| | | *John met Peter on (the platform)$_{Loc}$.* |
| | Time | time at which/from which/until which a SoA takes place |
| | | *John met Peter at (five o'clock)$_{Time}$.* |
| | Cause | specifies a motivation which is not ascribed to any of the participants in the SoA, but which is advanced by the speaker as an explanation of the SoA. |
| | | *The car slipped because (the street was wet)$_{Cause}$.* |

Table 4.2: Satellites recognized by FG

(12a) give$_V$(John, the book, the librarian).
(12b) John *gives* the book to the librarian.
(13a) PAST[give$_V$(John, the book, the librarian)]
(13b) John *gave* the book to the librarian.
(14a) FUTURE[give$_V$(John, the book, the librarian)]
(14b) John *will give* the book to the librarian.

Sometimes a linguistic expression has more than one underlying clause structure. In such a case, the linguistic expression is called *ambiguous*. A sentence with an ambiguous meaning like *The French bottle smells* has the following underlying clause structures (15) and (16). This example shows that the semantic representation of natural language does **not** contain any ambiguities, because it represents the exact meaning of the sentences.

(15) bottle$_V$(The French$_{Ag}$, smells$_{Go}$).
(16) smell$_V$(The French bottle$_{zero}$).

### 4.2.2 A semantical typology of predications

As stated before, a predication (or SoA) forms the kernel of the underlying clause structure. A predication is formed by selecting a predicate from the lexicon and filling this predicate with appropriate terms. Therefore the typology of predications as discussed in this section, strongly depends on predicates and terms. As an example consider the following sentences:

(17) John is painting a portrait.
(18) John is painting portraits.

The semantic difference between (17) and (18) is more than just the number of portraits that are painted by John. Both sentences are based on the same predicate, namely 'paint(<painter>$_{Ag}$, <something to be painted>$_{Go}$)'. The event described in (17) however, is expected to reach a certain natural terminal point in which the SoA is fully accomplished. While sentence (18), is not expected to have such a natural terminal point but can go on indefinitely. These subtle semantic differences show that introducing a semantic typology on the level of predication is more appropriate than a typology on the level of predicates[5]. Predications (or SoAs) can be categorized according to

[5]In contrast with this, the authors of [Dig89], [BR95b] and [Mos79] argue to define semantic parameters like *dynamic*, *telic*, etc. on the level of predicates.

semantical criteria. Applying these criteria to predications results into categories of predication of e.g. the following forms: *experience*, *action*, *process* and *state*.

(19) John saw a bird. (*Experience*)
He shot at it. (*Action*)
The bird fell down. (*Process*)
It was dead. (*State*)

Criteria used to classify the SoAs are called *semantic parameters*. A semantic parameter is a semantic property Prop of a SoA that can be positive (+Prop) or negative (-Prop). FG recognizes the following semantic parameters:

- dynamism
- telic
- momentaneous
- control
- experience

Table 4.3 describes these semantic parameters and shows some examples. The semantic parameters of FG are combined to constitute the typology of SoAs as depicted in figure 4.2. As can be seen, a [+dyn] and [+con] SoA is called an Action, while for example a [-dyn][-con] SoA is called a State.

| semantic par. | + or - | Description / Example |
|---|---|---|
| dynamism | +dyn | The SoA does involve some kind of change. |
| | | *John opened the door.* |
| | -dyn | The SoA does not involve any change. |
| | | *The substance was red.* |
| telic | +tel | If the SoA is fully achieved it reaches a natural terminal point. |
| | | *John was painting a portrait.* |
| | -tel | The SoA never reaches a natural terminal point but can go on indefinitely. |
| | | *John was painting.* |
| momentaneous | +mom | The SoA does not have a duration, its beginning coincides with its terminal point. |
| | | *John reached the summit.* |
| | -mom | The SoA occupies a certain stretch of time. |
| | | *John was painting a portrait.* |
| control | +con | The first argument of the SoA is able to determine whether or not the SoA will obtain. |
| | | *John opened the door.* |
| | -con | The first argument of the SoA is **not** able to determine whether or not the SoA will obtain. |
| | | *The apple fell from the tree.* |
| experience | +exp | The SoA describes that some animate being perceives feels, wants, conceives, believes, thinks or otherwise experiences something. |
| | | *John believes the story.* |
| | -exp | The SoA does **not** describe an experience of a animate being (as defined by +exp). |
| | | *John opened the door.* |

Table 4.3: Semantic parameters recognized by FG

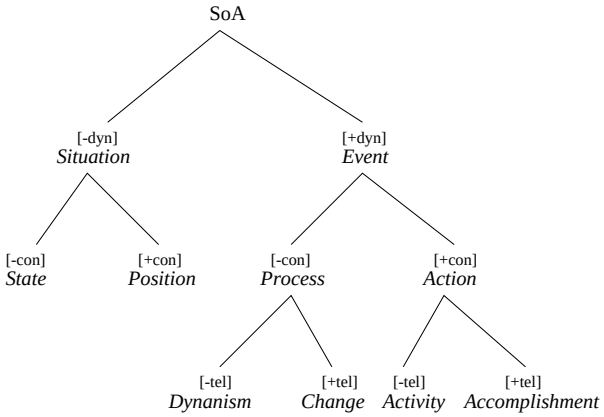


Figure 4.2: Typology of SoAs

Note that the semantic parameter *experience* can be applied to all semantic categories depicted in figure 4.2. FG does **not** recognize 'experience' as a semantic function but treats it as an property of semantic functions like *goal* (20) and *recipient* (21).

(20) John scared Mary$_{Go[+exp]}$.
(21) John apologized to Peter$_{Rec[+exp]}$.

## 4.3 Application of the Semantical approach

The knowledge in the knowledge base is represented in a so called Knowledge Representation Language (KRL). As stated in section 2.4.2.1, any consistent formalism in which we can express knowledge about some problem domain can be used as a KRL. The popularity of rule-based approaches in general and that of the KRL of *if-then* rules in particular has also been mentioned. In this section another rule-based KRL, called *Conceptual Prototyping Language* (CPL) developed by F.P.M. Dignum ([DR91], [Dig89]) is introduced. CPL is based on the linguistic theory of FG as introduced in the previous section. With CPL application of the semantical (Functional Grammar) approach to NL is demonstrated in the field of information modelling.

### 4.3.1 Different aspects of a UoD

Traditionally modelling techniques capture a single type of knowledge of the UoD. This implies that modelling techniques focus on a certain aspect of, or have a certain view on, the UoD. For example the following types of modelling techniques can be distinguished: data modelling, process modelling and (complex) constraint modelling techniques. Often these different techniques are combined to capture all knowledge about the application domain. Figure 4.3 shows some modelling techniques with their typical use[6].

As visualized in figure 4.3, CPL is a member of the intersection of data , process and constraint modelling techniques. This position represents the expressive power claimed by CPL. How CPL attempts to model all these different aspects of the UoD is described in the following subsections. CPL is introduced by means of examples of the three different modelling aspects of the UoD that show the expressive power of CPL in the field of information modelling, followed by a formal definition of its syntax and semantics.

### 4.3.2 Data modelling

This section provides a discussion on the correspondence between conceptual data modelling techniques and CPL. In this study PSM ([HW93]) is used to exemplify the set of conceptual data

[6] Some data- and process modelling techniques are capable of modelling some simple constraints, complex constraints however are modeled by an external constraint modelling technique with more expressive power

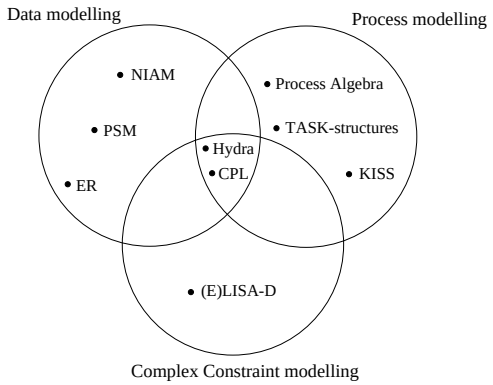


Figure 4.3: Modelling techniques with their typical use

modelling techniques as depicted in figure 4.3. The correspondence between PSM and CPL is investigated by introducing PSM concepts with their equivalent CPL specification. The theoretical value of this section, is the suggested expressive power of CPL.

Conceptual data modelling techniques are able to express the knowledge of the UoD by means of describing the existence of different object types and relations between object types in the UoD. In section 4.2 it has been shown that the semantical representation of FG can be used to capture the user-defined relations between object types (see for example (3) and (4)). The examples given in this section show that the translation from a FG representation to a CPL specification is very straightforward. The semantical representation of *A boy can hit a ball* is formalized in CPL as follows (its PSM equivalence is shown in figure 4.4):

(5) hit(ag = the boy)(go = the ball).

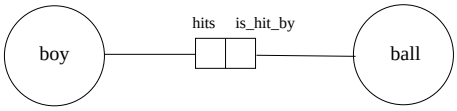


Figure 4.4: A binary relation in PSM

It can be observed that the CPL formalization in (5) is very close to the FG expression of (4). The verb *to hit* defines a relation between two object types. The object types *boy* and *ball* participate in this relation in the role of *agent* and *goal* respectively. Satellites[7] can be added to the predication in CPL as in (6b).

(6a) The boy hit the ball with the baseball bat.
(6b) hit(ag=the boy)(go=the ball)(instr=the baseball bat).

In addition to user-defined relations, like *to hit*, CPL recognizes special relations that capture conceptual data modelling notions like *label type*, *specialization*, *generalization*, *instantiation* and *aggregation*. These notions are captured by the introduction of special relations isa , hasa , exists and value_of. Examples of these notions and their corresponding representation conform CPL convention are provided in (7)-(12). Their equivalents in PSM notation are provided in figure 4.5.

[7] Strictly spoken, satellites are not part of the SoA itself, but provide additional information *about* the SoA.

(7a) An adult is a special type of person. (*specialization*)
(7b) isa (zero = adult)(zero = person).
(8a) A product is a car or a house.(*generalization*)
(8b) isa (zero = car)(zero = product).
isa (zero = house)(zero = product).
(9a) A car consists of a break, engine and accelerator. (*aggregation*)
(9b) hasa (zero = car)(go = break).
hasa (zero = car)(go = engine).
hasa (zero = car)(go = accelerator).
(10a) Their exist n instances of person[8]. (*instantiation*)
(10b) exists (<n> person).
(11a) An age is represented by an integer value. (*domain specification*)
(11b) value_of(zero = age)(zero = integer)

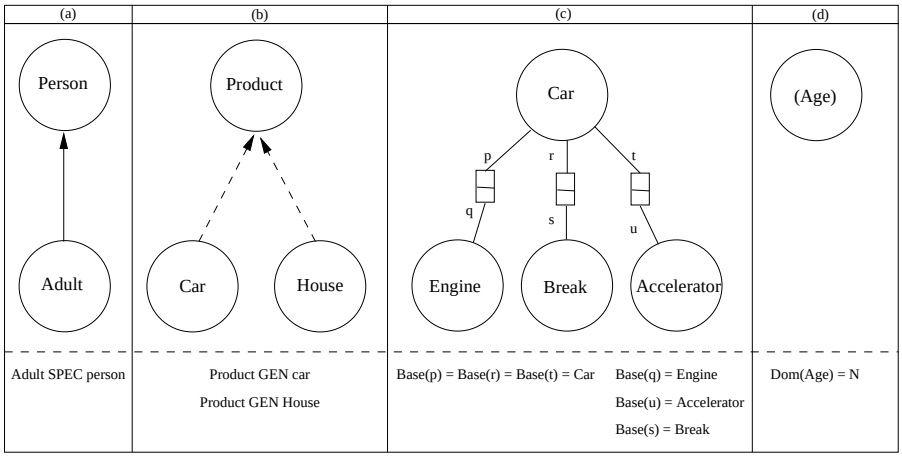


Figure 4.5: PSM conventions (a) specialization, (b) generalization, (c) aggregation, and (d) domain specification

The following lines show how CPL deals with meronym relations expressed by the verb *to have*[9].The verb *to have* is used in CPL in two different ways: *have* and *has*. Firstly, the ordinary *(to) have* relation is used to express possession between two object types. For example the knowledge *A student has a girlfriend* is expressed in CPL as follows: have(zero = student)(go = girlfriend). Secondly, CPL uses the term *has* for three different purposes

- to express possession of properties of an object type. Take for example *A student has a name*: has(zero = student, go = name).
- to express adjectives. For example the sentence *The green book* actually means *The book has a green color*, expressed as follows: has(zero = book) (go = A in color)(id: A = green). The variable 'A' represents an instance of type color, and the (id: A = green) is specified to express that that color is green. Later on, the use of these constructs will be discussed in more detail.
- to express membership of an object type[10]. For example *Erik is a student* is defined as follows: has(zero = student)(go = A in name)(id: A = Erik)

Some simple constraints - known from data modelling techniques - like *total*, *occurrence frequency* or *uniqueness* constraints are handled by CPL as follows. The syntax of CPL is extended with the

[8]The notation n+ and n- is used to express 'n or more' and 'n or less' instances of an object type.
[9]Meronym relation expressed by the verb *to have* is discussed in more detail in section 5.3.4.
[10]In [Dig89, page 30] the author states that a special relation like *member_of* is **not** invited to force the user to specify the identifying property (or properties) of an object. However, as mentioned by [FKW95, page 21], the object-oriented paradigm states that each object has a unique identification (the availability of an implicit label type is assumed). This implicit label type is used to assign each instance of an object type a unique object identifier.

symbol $*$ to express the total constraint on Person in the following sentence: *All students have a student-number*[11].

(12) MUST: PERF: has(< $*$ > zero=student)(go = student-number).

Occurrence frequency constraints are used to restrict the minimum or maximum number of instances of object types that participate in a certain fact-type like: *A student has at least 1 bike*. In contrast with CPL, PSM has no facilities to constraint the number of objects of a certain object type as defined in (14)-(16).

(13) MUST: PERF: have(zero=student)(< 1+ > go=bike).
(14a) There are 3 students.
(14b) exists (< 3 > zero=student).
(15a) There are at least 3 students.
(15b) exists (< 3+ > zero=student).
(16a) There are at most 3 students.
(16b) exists (< 3− > zero=student).

Finally the uniqueness constraint can be used to specify the cardinality of a relation resulting into one-to-one, one-to-many, or many-to-many relations. Their equivalent PSM specifications are depicted in figure 4.6.

(17a) A student can borrow one or more books (*one-to-many*).
(17b) MUST: PERF: borrow(< 1 > ag=student)(< $*$ > go=book).
(18a) Marriage between man and woman (*one-to-one*).
(18b) MUST: PERF: marry(< $*$ > ag=man)(< 1− > go=woman).
and
MUST: PERF: marry(< $*$ > ag=woman)(< 1− > go=man).
(19a) A student can enroll one or more courses, a course can be enrolled by one or more students (*many-to-many*).
(19b) MUST: PERF: enroll(< $*$ >ag=student)(< 0+ >go=course).
and
MUST: PERF: enroll(< 0+ >ag=student)(< $*$ >go=course).

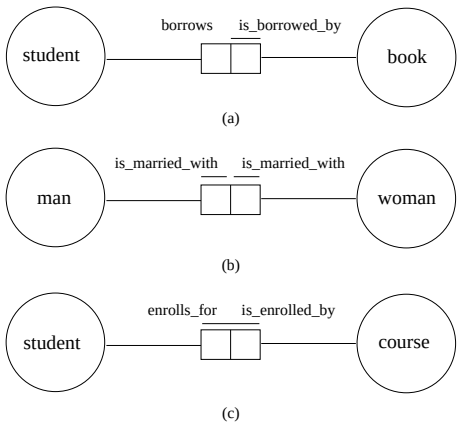


Figure 4.6: Cardinalities restricted by uniqueness constraint, (a) one-to-many, (b) one-to-one, (c) many-to-many

Besides cardinality CPL offers possibilities to define distributivity of the cardinalities in sentences like *Two boys kiss three girls*. A sentence like this has two semantic interpretations as shown in figure 4.7: 'Two boys kiss *together* three girls' (*collective*) or 'Two boys kiss *each* three girls' (*distributive*). These different semantic interpretations can be defined as follows.

(20) kiss(< $2, c >$ ag = boy)(< $3, c >$ go = girl).
(21) kiss(< $2, d >$ ag = boy)(< $3, c >$ go = girl).

[11] The CPL symbols MUSTand PERF are discussed in more detail in section 4.4

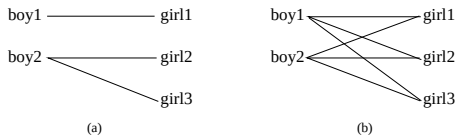


Figure 4.7: Distributivity of cardinalities in CPL: (a) collective, (b) distributive

In addition to specifying the structure of information, some data modeling technique provide formal techniques to define a population. Definition of populations in PSM and CPL is illustrated in the example below.

**Example 4.1** *Assume a world in which persons can join a band. Both 'person' and 'band' have a unique name referred to as 'p-name' and 'b-name' respectively. The data model of this (small) UoD is visualized in figure 4.8. The definition of populations is given below*:

$$
\begin{aligned}
\textsf{Pop}\ (\text{musician}) &= \{\text{m1, m2}\} \\
\textsf{Pop}\ (\text{band}) &= \{\text{b1, b2}\} \\
\textsf{Pop}\ (\text{m-name}) &= \{\text{'Keith', 'John'}\} \\
\textsf{Pop}\ (\text{b-name}) &= \{\text{'The Beatles', 'The Rolling Stones'}\} \\
\textsf{Pop}\ (\text{has}_{m-name}) &= \{\text{<m1, 'Keith'>, <m2, 'John'>}\} \\
\textsf{Pop}\ (\text{has}_{b-name}) &= \{\text{<b1, 'The Rolling Stones'>, <b2, 'The Beatles'>}\} \\
\textsf{Pop}\ (\text{join}) &= \{\text{<m1, b1>}\}
\end{aligned}
$$

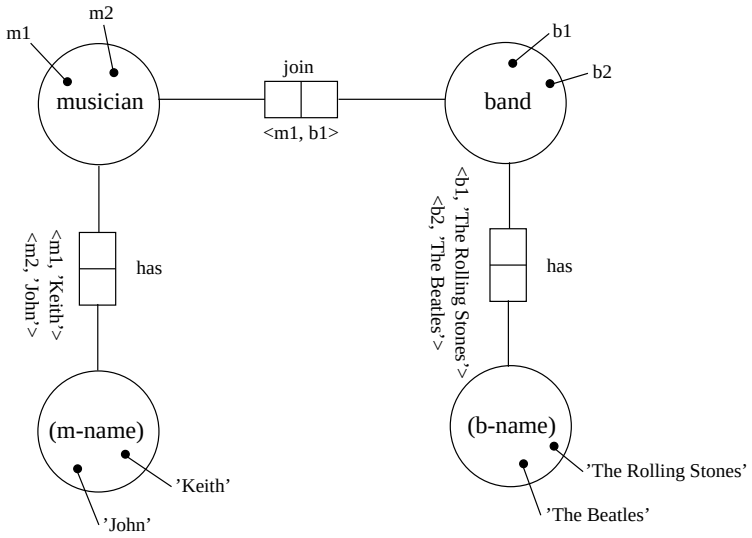


Figure 4.8: Handling populations in PSM

The corresponding CPL-specifications are given below. The first specification defines 'Keith joins The Rolling Stones', while the second and third define 'John is a musician' and 'The Beatles is a band' respectively.

FACTUAL: join(ag = M1 in musician)(go = B1 in band)
(id: has(<1> zero = M1 in musician)(<1> go = Name in m-name)
(id: Name = 'Keith'))
(id: has(<1> zero = B1 in band)(<1> go = Name in b-name)
(id: Name = 'The Rolling Stones'))

FACTUAL: has(<1> zero = M in musician)(<1> go = Name in m-name)
(id: Name = 'John')

FACTUAL: has(<1> zero = B in band)(<1> go = Name in =b-name)
(id: Name = 'The Beatles')

□ **End Example**

### 4.3.3 Process modelling

Process modelling concerns modelling dynamic behavior of objects. As stated in section 2.4.2.1, the dynamic behavior of objects is described by the so-called *Object Life Model* (OLM). In this section the correspondence between object life models of [FKW95] and [Kri94] at one side and CPL ([BR95a]) at the other, is discussed.

A process model describes all possible sequences of actions corresponding to the course of life of an object. In the paradigm of CPL, a process is seen as a combination of "primitive" or complex events (called actions). An action is miniature-drama expressed by a predication. E.g. sentence (5) shows an example of the action *The boy hits the ball.* The performance of an action results into the birth of a new relation. The death of a relation on the other hand is the result of the birth of another action. Take for example the action *borrow* which is equivalent to the insertion of a borrow relation. The return action on the other hand, is equivalent to the death of the borrow action (see figure 4.9).

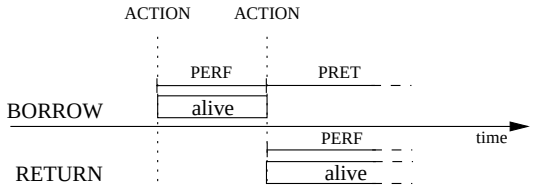


Figure 4.9: A BORROW action in the CPL paradigm

The current state of the world (UoD) is a set of actions that are performed in that UoD. In general, every action can be characterized by the effects it has on that state of the world. The birth of an action is denoted by ACTION (the representation of the actual action or present tense), the fact that a relation is alive is denoted by PERF (perfect tense) and the fact that it is dead is denoted by PAST (past tense).

(22) The student borrows the book.
ACTION: borrow(ag = student)(go = book).
(23) The student has borrowed the book.
PERF: borrow(ag = student)(go = book).
(24) The student borrowed the book.
PAST: borrow(ag = student)(go = book).

A possible sequence of actions can be described by defining for each action A the actions that necessarily precede A. This necessary precedence is expressed in CPL by the tokens PERMITand dest. These symbols are used as depicted in the following frame:

(25) PERMIT: ACTION: A
(dest: B )

Necessary preconditions are stated in B while the action is in A. The semantics of this statement is as follows: if A is performed, then B necessarily should be true. We can use this CPL-construction to model the course of life mentioned in section 2.4.2.1. See for example the following restriction: *A person can only leave a club, after he joined that club*

(26) PERMIT: ACTION: leave(ag = P in person)(go = B in band)
(dest: PERF: join(ag = P in person)(go = B in band))

In (26) two different variables P and B are used to reflect dependencies between objects of more than one object type, namely *person* and *band.* Since an object life model defines sequences of

actions that can be performed by a single object type, this restriction cannot be defined by an object life model. In [FKW95] this limitation is recognized and handled as a constraint (in the sense of a deontic rule). This 'solution' however conflicts with the strict separation between constraints that can be violated (deontic rules) and constraint that never can be be violated (analytical rules) as introduced in section 2.4.2.1.

### 4.3.4 CPL and complex constraint modelling

A data model defines the *possible* structure of the information in the UoD and a process model defines the *possible* sequence of actions in the UoD. These models therefore describe all *possible* situations. A constraint model, i.e. a set of constraints, on the other hand defines a *desired* situation. When the situation of the UoD is in contrast with a constraint in the constraint model, this constraint is said to be *violated*. In the following subsections constraints are classified according to three dimensions:

- static versus dynamic constraints
- time order
- modality

A deontic constraint is defined in CPL conform the general template of (27). A deontic constraint states that a certain predication 'Pred' must be true under certain circumstances defined by the situation 'sit'. Definition of these circumstances may be omitted resulting into a constraint that must be true under all circumstances. In (12)-(12) examples have been provided of such constraints.

(27) MUST: <Pred>
(sit: ...).

#### 4.3.4.1 Static vs. dynamic constraints

Static constraints are constraints on [-dyn] SoAs. As discussed in section 4.2.2 this type of SoA represents a situation like *The book is green* or *John was sitting in the garden*. The situation or predication of a constraint is called Dynamic if the predicate in the respective part is [+dyn] and is referred to as action, i.e. an event is taking place. If both Situation and Predication are static, the constraint is called static ((29)-(30)). If the Situation or the Predication is dynamic then the constraint is said to be dynamic ((30)-(31)).

(28a) The color of a book must be 'green'.
(28b) MUST: has(zero = book)(go = A in color) (id: A = 'green').
(29a) A student has a maximum age of 27.
(29b) MUST: has(zero = student)(go = A in age) (id: A < 27).
(30a) A person that drinks beer must be at least 21 years old.
(30b) MUST: has(zero = P in person)(go = A in age)
(id: A > 21)
(sit: PERF: drink(ag = P in person)(go = B in drink)
(id: B = 'beer')).
(31a) A student must return the book he/she borrowed.
(31b) MUST: ACTION: return(ag = S in student)(go = B in book)
(sit: PERF: borrow(ag = S in student)(go = B in book))

#### 4.3.4.2 Time order

Another dimension to classify (dynamic) constraints is the dimension of time order which specifies a desired order of actions. Another 'semantic function' called 'tmp' and a set of time operators (*before*, *after*, *in*, etc.) are introduced to facilitate time order specification. An example of such a constraint is the following fact: *A student must return a borrowed book within three weeks* (32) or *A person must have paid subscription before he joins the band* (33).

(32) MUST: ACTION: return(ag = A in student)(go = B in book) (tmp = T2 in time)
(id: T2 < T1 + 3∗week).
(sit: PERF: borrow(ag = A in student)(go = B in book) (tmp = T1 in time)).
(33) MUST: ACTION: pay(ag = P in person)(go = B in band) (tmp = T1 in time)
(id: T1 before T2)
(sit: PERF: join(ag = P in person)(go = B in band)(tmp = T2 in time)).

#### 4.3.4.3 Modality

The last dimension to be mentioned is called *modality*. With respect to modality, CPL recognizes two types of constraints. These are prescriptive (MUST) and permissive (PERMIT). A prescriptive constraint states that the SoA described by the static part of the predication *must* be valid and the actions described by the dynamic part of the predication *must* be executed under the conditions stated in the Situation. A permissive constraint *permits* the SoA described by the static part of the predication to be valid and the actions described by the dynamic part of the predication to be executed under the condition stated in the situation. Examples of permissive constraints have been given in (25) and (26), while examples of prescriptive constraints are provided in (28)-(33).

### 4.3.5 Formal syntax of CPL

The formal definition of the syntax of CPL as provided in this section, is adapted from [DR91] and [Dig89] [12].

| | | |
|---|---|---|
| <CPLSpec> | → | <Spec>* |
| <Spec> | → | <Mode> ':' <Predication> |
| | | [<Situation>] |
| <Predication> | → | <Tense> ':' [ ∼ ] <Rel> <Arg>$^+$ <Satellite>* <Ident>* |
| <Situation> | | '(sit:' <Predication> ')' |
| | | <Ident>* |
| <Arg> | → | '(' [<Card> [, <Distr>]] <ROLE> '=' <Participant> ')' |
| <Satellite> | → | '(' <SAT> '=' LINGEXPR ')' |
| <Ident> | → | '(id:' VAR '=' EXPR ')' |
| | \| | '(id:' <Predication> ')' |
| <Rel> | → | <SpecRel> \| VERB |
| <SpecRel> | → | exists \| is_a \| value_of \| has |
| <Participant> | → | <Object> |
| | \| | <ObjectClass> |
| <Object> | → | VAR ' in' <ObjectClass> |
| <ObjectClass> | → | NOUN \| @ |
| <Tense> | → | ACTION \| PERF \| PAST \| PROSP \| DONE |
| <Mode> | → | FACTUAL \| PERMIT \| MUST \| NEC |
| <ROLE> | → | *<a semantic function recognized by FG>* |
| <SAT> | → | *<a satellite recognized by FG>* |
| <Distr> | → | 'c' \| 'd' |
| <Card> | → | '<' (INT \| '*') ['+' \| '-'] '>' |

Table 4.4: Formal syntax of CPL

### 4.3.6 Formal semantics of CPL

As stated in [HW92] an information modelling technique needs a formal foundation to avoid deficiencies like inconsistency, lack of structure, over-specification, incompleteness, ambiguity and redundancy. CPL has such a sound formal semantics such that each CPL construct can be expressed in a combination of various types of logic:

[12] The terms VAR(iable), EXPR(ession), VERB, NOUN, LINGEXPR (linguistic expression) and INT(eger) have their usual interpretation.

- *predicate logic* used to define relations between objects
- *deontic logic* used to define constraints that ought to be true
- *modal logic* used to express necessity
- *temporal logic* used to describe temporal aspects

This section only provides some examples (34)-(35) of CPL constructs with their corresponding logical interpretation[13]. A far more detailed discussion on this topic is provided in [Dig89].

(34a) John has borrowed a book.
(34b) FACTUAL: ACTION: borrow(<1> ag = A in person)(<1> go = B in book)
(id: has(<1> zero = A in person)(<1> = C in name)
(id: C = 'John'))
(34c) $\exists A(person(A) \wedge \exists C(name(A,C) \wedge C = \text{'John'})$
$\exists B(book(B) \wedge \mathsf{PERF} : borrow(A,B)))$
(35a) If the book is present, then the person can borrow the book.
(35b) PERMIT: ACTION: borrow(<1> ag = A in person)(<1> go = B in book)
(sit: has(<1> zero = B in book)(<1> go = C in availability)
(id: C = present))
(35c) $\forall A \forall B(person(A) \wedge book(B) \wedge$
$\exists C(availibility(B,C) \wedge C = \text{'present'}))$
$\rightarrow P(borrow(A,B))$

## 4.4 Transforming 'Object Action Involvement' and 'Object Life'-models into CPL

This section discusses how simple *object action involvement* and *Object Life models* ([FKW95] can be transformed into a corresponding CPL-specification.

### 4.4.1 'Object Action Involvement Model' transformation

The first problem encountered when transforming the object action involvement model into a corresponding CPL specification is the problem of *semantical load* . The semantical load is the amount of semantical information captured by a model. The example below shows that the semantical load of an ordinary object action involvement expressed in $PSM^2$ model is relatively low, resulting into ambiguities. The problem of semantical load is a result of the fact that $PSM^2$ is syntax-oriented while CPL is semantics-oriented, and thereby providing more expressive power. Ambiguities in $PSM^2$ specifications are solved in CPL specifications, as shown below.

**Example 4.2** *Consider a world (UoD) in which the structure of information is modeled by the object action involvement model of 4.10.*

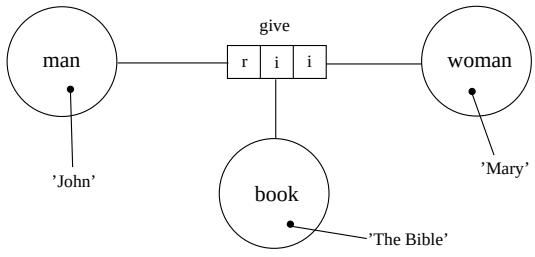


Figure 4.10: Action 'give' conform $PSM^2$

Figure 4.10 shows a give-relation between three object types *man*, *woman* and *book*. Strictly spoken, this schema contains ambiguities with respect to *tense*, *modality* and *semantic roles*, leading to the following different interpretations (read meanings).

[13] In (35c), the fact that $\alpha$ is permitted is denoted by $P(\alpha)$

- *tense*:
  - *<man> gives <book> to <woman>*, or in general the actual action is being performed now (ACTION).
  - *<man> has given <book> to <woman>*, or in general the action is dead, but the relation still exists (PERF).
  - *<man> gave <book> to <woman>*, or in general the action and relation are both dead (PAST).
  - *<man> will give <book> to <woman>*, or in general the action will be performed in future (PROSP).
- *modality*:
  - *<man> gives <book> to <woman>* (FACTUAL)
  - *<man> is able/permitted to give <book> to <woman>* (PERMIT)
  - *<man> has to give <book> to <woman>* (MUST)
  - *<man> is obliged to give <book> to <woman>* (NEC)
- *semantical role*: Semantical roles solve ambiguities presented by the different interpretations of:
  - *<man> gives <book> to <woman>*
  - *<man> gives <woman> to <book>*

  In the first - most probable - interpretation, the role of goal is played by the <book> and the role of recipient by the <woman>. Strictly spoken, on the base of the model of figure 4.10, the second (semantic anomalous) interpretation can also be chosen. In some situations however, the use of semantic role is more subtle. Consider for example the semantical difference between between *'John' moves the table* and *The earthquake moved the rock*. Does the earthquake actually have control over moving the rock? This question has to be answered negatively, because their exists a semantical difference between *person moves table* and *earthquake moves rock*. In the first sentence the person plays the semantical role of *agent* while in the second sentence the earthquake is involved in a MOVE-relation in the role of *force*. Semantical roles have been discussed in detail in section 4.2.

□ **End Example**

In the example above the lack of semantical information in PSM$^2$ specifications is illustrated. Ambiguities resulting from that lack are solved in CPL by modal and tense operators placed between brackets in the example above. When we make a transformation from a syntax-oriented approach to a semantical-oriented approach the additional desired semantical information should be provided by the information analyst. Recapitulating, the syntax-oriented schema of 4.10 must be extended with the following semantical information:

- for each action specify
  - tense $\in$ {ACTION, PERF, PROSP}
  - modality $\in$ {FACTUAL, PERMIT, MUST, NEC}
  - for each object type involved in action specify
    - semantic role $\in$ {agent, positioner, force, processed, zero, goal, recipient, location, direction, source, reference}

In the following lines a simplified algorithm is introduced that is able to transform simple PSM$^2$-models into a corresponding CPL-specification. First, the PSM$^2$-model of a sample UoD is provided with an equivalent CPL-specification. Then we generalize from this example to constitute the algorithm that performs the PSM$^2$ to CPL transformation.

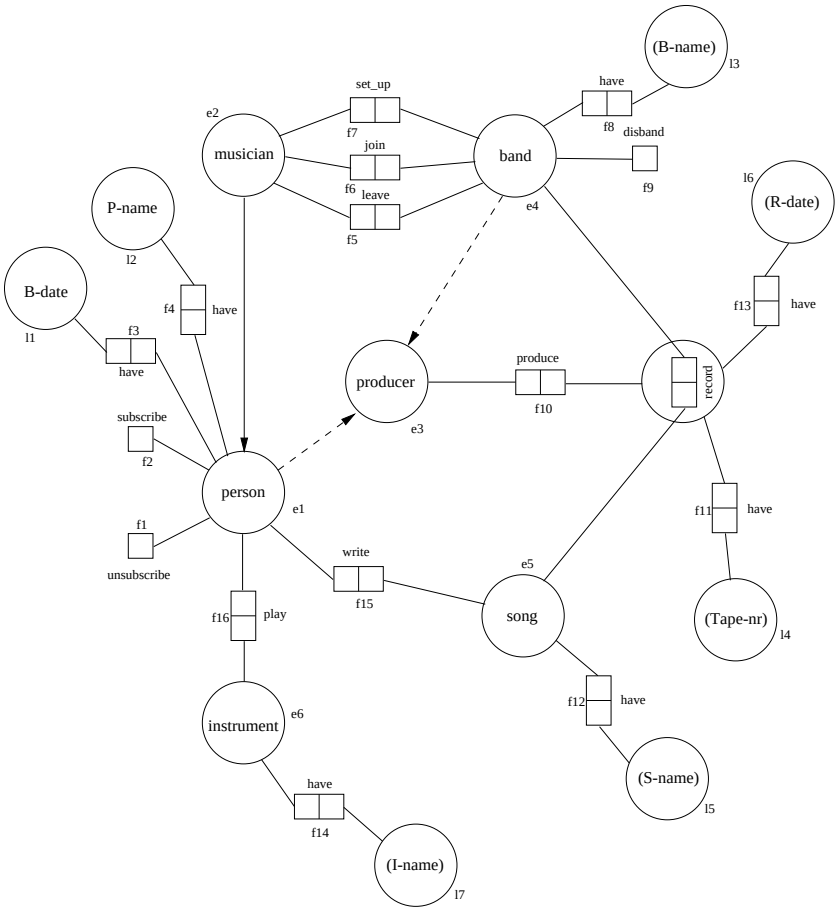


Figure 4.11: PSM$^2$-schema to be transformed

**Example 4.3** *A UoD is assumed of which the structure of the relevant information is defined by the PSM-schema depicted in figure 4.11.*

The corresponding CPL-specification is given below:

has(zero = person)(go = P-name).
has(zero = person)(go = B-date).
has(zero = band)(go = B-name).
has(zero = record(ag = band)(go = song))(go = Tape-nr).
has(zero = record(ag = band)(go = song))(go = R-date).
has(zero = song)(go = S-name).
has(zero = instrument)(go = I-name).

subscribe(zero = person).
unsubscribe(zero = person).
set_up(ag = musician)(go = band).
leave(ag = musician)(go = band).
join(ag = musician)(go = band).
record(ag = band)(go = song).
produce(ag = producer)(go = record(ag = band)(go = song)).
write(ag = person)(go = song).
play(ag = person)(go = instrument).
disband(zero = band).

isa(zero = musician)(zero = person).
isa(zero = person)(zero = producer).
isa(zero = band)(zero = producer).

□ **End Example**

In the following lines a simple algorithm is provided that performs the transformation under the following assumptions:

- the symbol '+' denotes concatenation of strings.
- $\mathcal{I}$ is an information structure such that $\mathcal{I}$ = <$\mathcal{P}$, $\mathcal{O}$, $\mathcal{L}$, $\mathcal{E}$, $\mathcal{F}$, Base, Spec, Gen>
  with $\mathcal{L}$ = {l1, l2, l3, l4, l5, l6, l7}, $\mathcal{E}$ = {e1, e2, e3, e4, e5, e6}, $\mathcal{F}$ = {f1, f2, ..., f16} and $\mathcal{O}$ = $\mathcal{L} \cup \mathcal{E} \cup \mathcal{F}$
  F is a set of unairy and binairy facttypes, E is a set of entitytypes and L is a set of labeltypes.
- an injective function Lexicon defines the name for each element from $\mathcal{O}$ (Lexicon = {<l1, B-date>, ..., <l7, I-name>, <e1, person> , ..., <e6, instrument>, <f1, unsubscribe>, ..., <f16, play>}
- the symbol <Role> is used to denote the appropriate semantical function recognized by FG
- Specialisation is captured by a binairy relation Spec = {<e2, e1>}
- Generalisation is captured by a binairy relation Gen = {<e3, e1>, <e3, e4>}
- procedure OUTPUT produces the output of CPL-strings to screen.

```
proc PSM2CPLTransformation(I : Information Structure)
        Let I = <P, O, L, E, F, Base, Spec, Gen>
        for each f ∈ F
                PSM2CPL(f)
        for each o ∈ Spec
                Let o = <x, y>
                OUTPUT('isa' + PSM2CPL(x) + PSM2CPL(y))
```

```
        for each o ∈ Gen
                Let o = <x, y>
                OUTPUT('isa' + PSM2CPL(y) + PSM2CPL(x))
endproc P2M2CPLTransformation


proc PSM2CPL(o ∈ 𝒪)
        Name := Lexicon(o)
        switch(o):
                case o ∈ ℱ:
                        if |o| = 1 then
                                Let o = {p}
                                OUTPUT(Name + PSM2CPL(Base(p)))
                        else
                                Let o = {p, q}
                                OUTPUT(Name + PSM2CPL(Base(p)) + PSM2CPL(Base(q)))
                case o ∈ ℒ ∪ ℰ:
                        OUTPUT('(' + <Role> + '=' + Name + ')').
endproc PSM2CPL
```

### 4.4.2 'Object Life Model' transformation

In this section a transformation algorithm is introduced that transforms *Object Life Models* (or KISS models) into a corresponding CPL-specification. To show the strategy followed by the algorithm, the discussion is started on base of an example. Consider the KISS-model depicted in figure 4.12. As stated in [Kri94, page 322], a KISS-model can be defined by a matrix which shows which actions are permitted to be performed under which circumstances. The corresponding matrix of figure 4.12 is given in table 4.5. After that, this table can be used to produce the correct CPL-specifications, by reading the necessary predications of the actions on each row. This is done for action 'write' in the example below.

**Example 4.4** *We assume a UoD in which all possible orders of actions that can be performed by objects of object type musician are defined by the KISS-model provided in figure 4.12.*

This KISS-model can be transformed to table 4.12 which shows for each action the a set of necessary precondition. One of these preconditions should have preceded that action directly.

| | Preconditions | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Action | subscribe | write | play | produce | unsubscribe | set_up | join | leave |
| write | True | True | True | True | | True | True | True |
| play | True | True | True | True | | True | True | True |
| produce | True | True | True | True | | True | True | True |
| unsubscribe | True | True | True | True | | True | True | True |
| set_up | | | True | | | True | True | True |
| join | | | True | | | True | True | True |
| leave | | | True | | | True | True | True |

Table 4.5: Table representation of permitted order of actions

Table 4.5 shows that action *write* can be performed under the following circumstances.

PERMIT: ACTION: write(ag = M in musician)(go = song)
(dest: DONE: subscribe(zero = M in musician).
or (dest: DONE: write(zero = M in musician)(go = song).
or (dest: DONE: play(zero = M in musician)(go = instrument).
or (dest: DONE: produce(zero = M in musician)(go = recording)).
or (dest: DONE: set_up(zero = M in musician)(go = band).
or (dest: DONE: join(zero = M in musician)(go = band).

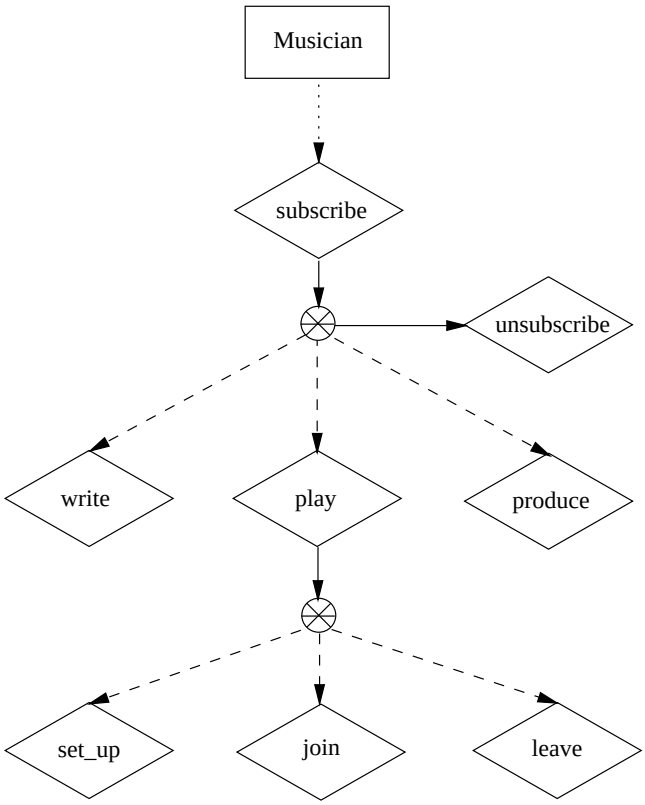


Figure 4.12: KISS model for object type musician

or (dest: DONE: leave(zero = M in musician)(go = band).

□ **End Example**

Analogously, each row of table 4.5 results into a CPL-specification. Below a simple algorithm is introduced that performs the transformation of a simple KISS-model k to a CPL-specification under the following assumptions:

- procedure OUTPUT produces the output of CPL-strings to screen.
- a KISS-model is a structure <ObjType, InstAction, Actions, Precon, IsTrue>.
  - variable ObjType contains the object type of the KISS-model (ObjType = e2)
  - variable InstAction contains the first action that should be performed by the object in the KISS-model. This action is the so-called *Initialization Action* (InstAction = f2)
  - a set Actions contains the actions mentioned in table 4.5 (Actions = {f1, f5, f6, f7, f10, f15, f16}).
  - a set Precon contains the preconditions mentioned in table 4.5 (Precon = Actions ∪ {InstAction}
  - a ternary relation IsTrue contains the truth values recorded in table 4.5 (IsTrue = {<f15, f2, True>, <f15, f15, True>, ..., <f15, f1, False>, ...}).

```
proc KISS2CPLTransformation(k : KISSModel)
        Let k = <ObjType, InstAction, Actions, Precon, IsTrue>
        for each action ∈ Actions
                KISS2CPL(action, k)
endproc KISS2CPLTransformation
```

```
proc KISS2CPL(action ∈ Actions, k : KISSModel)
        Let k = <ObjType, InstAction, Actions, Precon, IsTrue>
        Name := Lexicon(action)
        OUTPUT('(PERMIT: ACTION:' + Name)
        KISSAction(action, k)

        First := True
        for each prec ∈ Precon
                NamePrec := Lexicon(prec)
                if IsTrue(action, prec) then
                        if not First then
                                OUTPUT('or')
                        OUTPUT('(dest: DONE:' + NamePrec)
                        KISSAction(prec)
                        First := False
endproc KISS2CPL


proc KISSAction(action ∈ Actions ∪ Precon)
        if |action| = 1 then
                Let action = {p}
                OUTPUT('(' + <Role> + '=X in ' + Lexicon(Base(p)) + ')')
        else
                Let action = {p, q}
                OUTPUT"('(' + <Role> + '=X in ' + Lexicon(Base(p)) + ')')
                OUTPUT('(' + <Role> + '=' + Lexicon(Base(q)) + ')')
endproc KISSAction
```

The exercises performed in this section, show that a restricted translation from typical static and dynamic models to CPL can be made. The translations provide insight in the expressive power of CPL. Especially the choice of semantical roles, shows that CPL - as a semantical approach to information modelling - has more expressive power than syntax-oriented modelling. The following chapter shows that exactly these roles provide excellent grips for intelligent interaction between man and machine.

# Chapter 5

# The role of the lexicon in the semantical approach

## 5.1 Introduction

In this chapter the advantages of the semantical approach to natural language, will come to light when studying communication between user(s) and information system. The semantical approach to information modelling, as suggested in chapter 4, provides fruitful clues for establishing a natural integration of data and general world knowledge. The profits of integration of data and general world knowledge are reflected in:

1. the enhancement of communication between user(s) and information system
2. the support of information modelling process

This chapter mainly focuses on how to enhance communication between user(s) and system while chapter 6 shows how to support information modelling. The following sections discuss why and how a lexicon can be used.

## 5.2 The lexicon

When humans communicate (with information systems) they use words. As discussed in chapter 3, these words are - mostly arbitrary chosen - denotations that represent an idea (or concept) in someone's mind. Such a concept is viewed as a bundle of semantical features. Sometimes a sentence triggers semantic features that conflict with each others. These sentences cannot be understood when non-metaphorical use of language is assumed. Sentences like these are called *semantically anomalous*. An example of such a sentence is *My brother is pregnant*. Further demonstrations with anomalous sentences show that a word like *bachelor* contains additional features such as *unmarried* and *adult* ([Gle91, page 345]). In 1910 two psychiatrists G.H. Kent and A.J. Rosanoff reported the results of a so-called *word-association test*. With this test a large number of testes was asked to say the word that came in their mind first when they heard a certain test-word. Table 5.1 shows the top ten results of this test applied to test-word 'chair' on a total number of 1000 man and woman ([Mil93, page 159]).

Results of tests like these made psychologists believe that meaning of words are stored as *relations* among words, called *semantic relations*. Semantic relations between words and features are stored in someone's memory in a so-called *mental lexicon*. Examples of semantic relations are *synonym* (seat), *hypernym*(furniture) and *meronym* (wood). Only a few semantic relations are recognized by conceptual data modelling techniques (*specialization*, *generalization* and *aggregation*) but the usefulness of semantic relations is often underestimated. During communication NLUs

| Frequency | Response word |
|---|---|
| 191 | table |
| 127 | seat |
| 107 | sit |
| 83 | furniture |
| 56 | sitting |
| 49 | wood |
| 45 | rest |
| 38 | stool |
| 21 | comfort |
| 17 | rocker |

Table 5.1: Word-association test on test-word 'chair'

(unconsciously) consult their mental lexicon to understand sequences of words uttered by their communication partner. This mental lexicon enables NLUs to make inferences like:

1. Someone who is killed, is dead.
2. Someone who manages something is called a manager.
3. Someone who is not male, must necessarily be female.
4. Someone who is not young, is not necessarily old (and vice versa).
5. Someone who is young, is not old.
6. Someone who is having a spouse, is married.
7. Someone who snores, also sleeps.
8. Someone who returns a book to a library, has been borrowing that book.
9. If a person has an arm, and that arm has a finger, then that person has a finger.

Inferences like these are so patently obvious for most NLUs that they are never really explicitly stated during conversation. On the contrary, these inferences are assumed to be known to realize effective communication. However, communication between humans and information systems has a more laborious nature. Usually humans are forced to express their meanings conform a strict syntax, and to use the same words that were initially put into the model by the information analyst in cooperation with the domain expert.

**Example 5.1** *Assume an information system that keeps track of lugubrious facts like 'Who killed whom and when?' as modeled by the PSM-schema depicted in figure 5.1.*

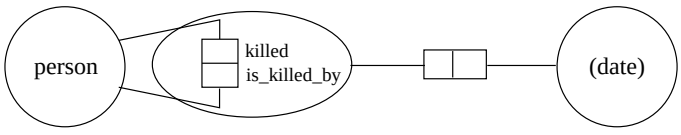


Figure 5.1: An information system containing lugubrious facts

Usually, information systems - based on the information model of figure 5.1 - are unable to answer questions like: *Who is dead?*, *Who is murdered?*, *Who is a murderer?* and *When died 'Johnson'?*. Although, the needed information to answer these questions is available, common sense knowledge is missing to relate these questions to the appropriate stored data.

□ **End Example**

The misunderstanding, sketched in the example above, that comes into being can be attributed to the fact that communication (between man and machine) is not about the words itself, but

about the *meaning* of the words. The information system does not have disposal of a kind of lexicon similar to the mental lexicon of humans. To solve this problem the information system can also be extended with a lexicon, called *system lexicon*. The system lexicon is a formal lexicon that approaches the mental lexicon as close as possible, in order to 'understand' the words and provide the appropriate answers to the queries formulated by the user. In concreto, this means that the system lexicon must provide a formal description of a (as large as possible) subset of the mental lexicon relevant to the UoD. This situation is depicted in figure 5.2. The contents of the system lexicon corresponds to the linguistic (analytical) knowledge referred to in section 2.4.2.1 and highlighted in figure 5.2.

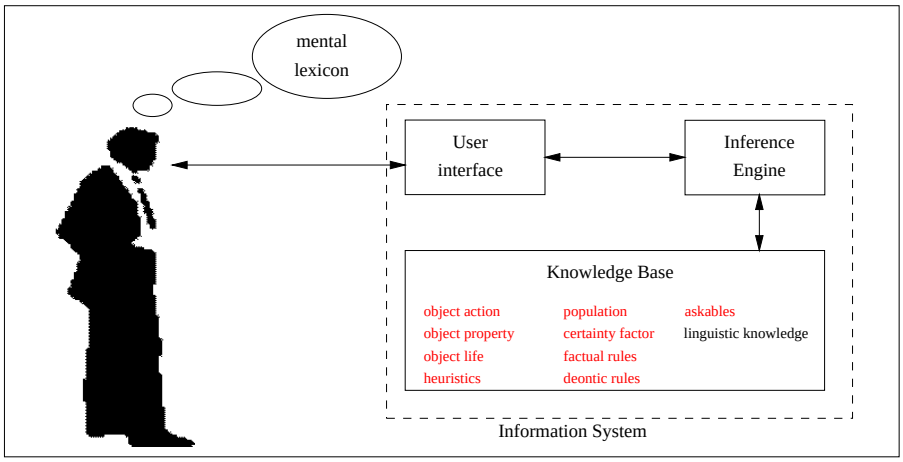


Figure 5.2: The system lexicon and the mental lexicon, used during communication

While the previous lines discussed *why* to use a lexicon, the following question to be answered is: *How* to use a lexicon? As stated before the system lexicon can be regarded as just another part of the knowledge base. The inference engine produces the answers by combining information modelling knowledge (i.e. the union of object action, object property and population knowledge) with general world knowledge (i.e. linguistic knowledge ). Now the advantages of semantical data modelling, as discussed in section 4.3, over traditional data modelling come to light. Traditional data modelling techniques (like PSM, NIAM and ER) only specify *which* entities are involved in a certain relation. In addition to that, semantical data modelling techniques (like CPL) also specify *how* entities are involved in relations, i.e. what is the *meaning* of their involvement or *which role* do they play in a relation. The system lexicon (from now on, simply called 'lexicon') can be seen as a top layer on the information model. This top layer brings more intelligence[1] in the system. It is the semantical approach that provides the handles to relate population knowledge to general world knowledge from the lexicon. The hierarchy of data, data structure and semantic information is depicted in figure 5.3.

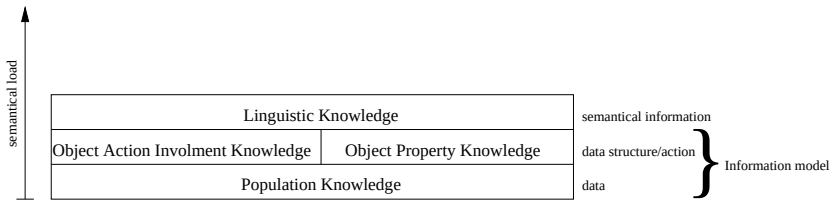


Figure 5.3: The lexicon as a top-layer on the data model

**Example 5.2** *This example shows how the semantical information model enables a natural integration of general world knowledge and modeled information. Some informal descriptions are*

[1] Intelligence is defined here as the capacity to apprehend facts and propositions and their relations and to reason about them [Webster's dictionary]

*provided that capture the inferences, mentioned in the previous section and the previous example.*

- *Someone who is killed is dead* means that the one who participates as *goal* in a *kill*-relation is said to be dead.
- *Who is a murderer?* can be answered by retrieving the one who participates as a *agent* in a *kill*-relation.
- *When died Johnson?* can be answered by retrieving the date of the *kill*-relation in which 'Johnson' participated as *goal*

□ **End Example**

As shown in the example above, the lexicon must be filled with formal definitions of inferences that are normally contained in the *mental* lexicon. In the following section the semantic relations in the mental lexicon are discussed and it is studied how these relations can be defined formally to fill the system lexicon in order to enable intelligent communication between user(s) and system. Linguistic knowledge consists of three distinct parts:

1. terms
2. predicates
3. semantical relations

While the role of terms and predicates is discussed in the following chapter, the following section starts a discussion on the importance of semantical relations. This study on semantical relations is assisted by examples that show their applications in the field of information systems.

## 5.3 Semantical relations and their Applications

Semantical relations are relations between words. In the OO-paradigm words are used to represent objects, methods and properties of objects. Objects can be represented by nouns while methods of objects can be represented by verbs. Verbs specify a relation (or action) between object types. Application of adjectives on objects presuppose the presence of object properties. This situation makes that in the OO-paradigm semantic relations can be viewed as relations operating on domains of objects, relations and adjectives. Binary semantic relations can be classified according to the domain on which they operate. Figure 5.4 shows different classes of semantical relations that can be distinguished. Class (A) relations, as an example, are relations between two objects, while class (B) relations constitute an association between an object and a relation.

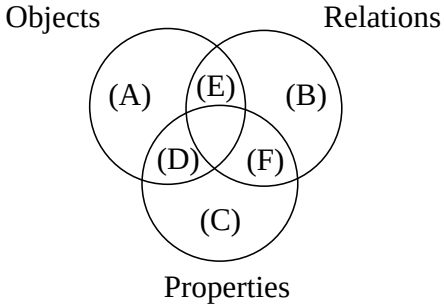


Figure 5.4: Classification of semantical relations

In this section different semantical relations are introduced and it is studied how these relations can enhance communication (between man and machine). Improving communication is achieved by adding semantical relations to the knowledge base. These semantical relations can be used during

backward and forward inference processes as discussed in section 2.4. The formalism of *if-then* rules is used for knowledge representation. This formalism is extended with semantical information like *tense*, *modality* and *roles* to avoid ambiguities (see section 4.4) and make reasoning about PSM$^{2+}$-models possible. A PSM$^{2+}$-model is a PSM$^2$-model ([FKW95]) with the following modifications:

- Verbs can be used to express an action, process or certain state, leading to the following classification:
  - *Action verbs* describing a state or process (*to run*, *to walk*, *to play*, *to fall*, etc.)
  - *State verbs* describing a state (*to know*, *to lie*, *to admire*, *to contain*, *to have*, *to possess*, etc.)

  In addition to the action verbs considered by PSM$^2$, the importance of state verbs is recognized by PSM$^{2+}$ in order to achieve more expressive power.
- A tense is added to each relation.
- A modality is added to each relation.
- Semantical roles, recognized by the theory of functional grammar, are added to each object type that participates in a certain relation.

Figure 5.5 shows a PSM meta schema of PSM$^{2+}$. The underlying formalism of PSM$^{2+}$ is a simplified version of CPL which can be implemented directly in PROLOG. The formalism is discussed in the following lines. Applications of inferences are illustrated with a prototype tool called PSM$^{2+}$-Expert, described in appendix F.

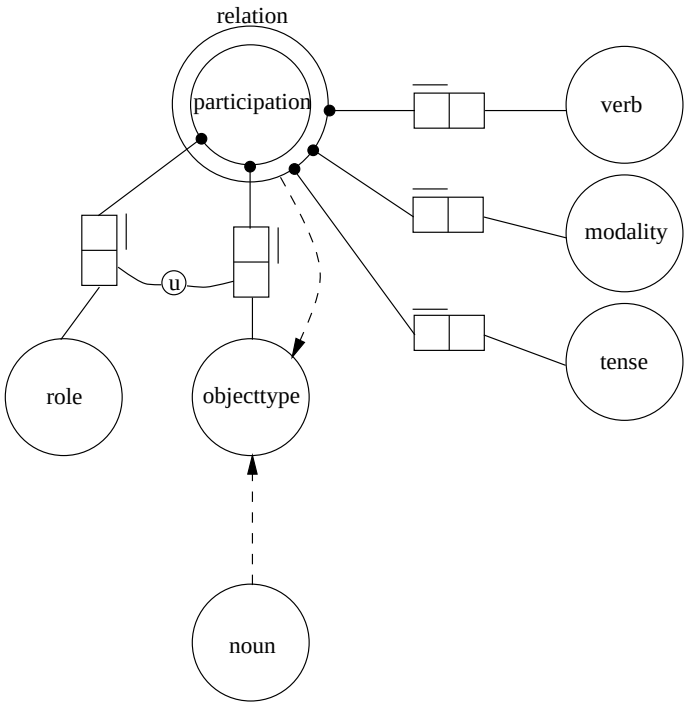


Figure 5.5: PSM meta schema of PSM$^{2+}$

A PSM$^{2+}$-schema consists of nouns and verbs. Nouns are used to represent object types while verbs are used to represent a relation between object types. A relation expressed by a verb can be seen as a complex object type. Object types participate in a certain relation, by playing a semantical role like *agent*, *goal*, etc. Therefore a relation can be viewed as a set participations of Object types. Each relation has a tense and modality providing more semantical information about the verb associated with that relation. Figure 5.6 shows a general template that can be combined to constitute a complete PSM$^{2+}$-model.

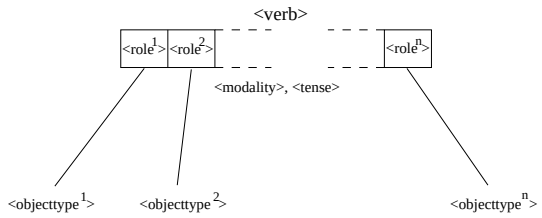


Figure 5.6: A general PSM$^{2+}$-relation between one or more Objectypes.

The PSM$^{2+}$-schema of figure 5.6 can be defined as follows:

- Relation(<modality>, <tense>, <verb>, [(<role$^1$>, <objecttype$^1$>), (<role$^2$>, <objecttype$^2$>), ..., (<role$^n$>, <objecttype$^n$>)]).
- <modality> $\in$ {FACTUAL, PERMIT, MUST, NEC}
- <tense> $\in$ {ACTION, PERF, PAST, PROSP}

**Example 5.3** *Assume a world (UoD) in which we register musicians, bands and musician that have joined a band and did not leave it afterwards. This situation is formally defined as*
Relation(FACTUAL, PERF, join, [(ag, musician), (go, band)]).
*and visualized in figure 5.7.*

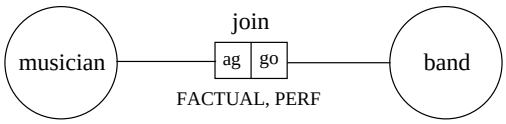


Figure 5.7: A 'join'-relation in PSM$^{2+}$ notation.

□ **End Example**

### 5.3.1 Instance relations

An instance is a single object of a certain object type. An instance relation is often reflected by the use of the words 'is a'. Take for example the following sentence:

(1) *John is a student.*

Sentence (1) states that 'John' is an instance of the set of objects referred to as student. For the sake of simplicity, 'John' is used as a denotation for a unique instance of object type *student*. And therefore it is not allowed to conclude that every *John* is a student. The semantical relation Instance is introduced to capture this part of semantical information. The corresponding semantical load of (1) can be defined as follows: Instance('John', student). The specification of instances of relations can also be handled with the same Instance relation. Remember that a relation is also an object type and can therefore be defined analogously. The fact that 'The Rolling Stones' has been joined by 'Keith' in the scope of example 5.3 can be viewed as an instance of relation *join* (2).

(2) Relation(FACTUAL, PERF, join, ([(ag, Instance('Keith', musician)), (go, Instance('The Rolling Stones', band))]).

**Definition 5.1** INSTANCE DEFINITION
*For the sake of simplicity, a shorthand notation for instance relations is provided:*
$\phi$(X) $\equiv$ Instance(X, $\phi$).

□ **End Definition**

**Example 5.4** *This example shows some applications of the previous definition.*

- Instance('Keith', musician). ≡ musician('Keith').
- Instance('The Rolling Stones', band). ≡ band('The Rolling Stones').

□ **End Example**

**Definition 5.2** OBJECT-PARTICIPATION
*The role 'Role' played by an instance 'Instance' of object type 'ObjectType' in a certain relation with modality 'Modality', tense 'Tense' and arguments 'Args' is captured by the following relation:*
ObjectPart (Modality, Tense, Relation, Args, Object Type, Instance, Role)

which is defined as follows[2]:
Relation(Modality, Tense, Relation, Args)
∧ Exec(member((Role, instance(Instance, ObjectType)), Args))
≡ ObjectPart (Modality, Tense, Relation, Args, ObjectType, Instance, Role).

□ **End Definition**

**Example 5.5** *This example shows applications of the definition of object-participation as given above:*
Relation(FACTUAL, PERF, join, [(ag, musician('Keith'), (go, band('The Rolling Stones'))]) ⊢

1. ObjectPart (FACTUAL, PERF, join, [(ag, musician('Keith'), (go, band('The Rolling Stones'))], musician, 'Keith', ag).
2. ObjectPart (FACTUAL, PERF, join, [(ag, musician('Keith'), (go, band('The Rolling Stones'))], band, 'The Rolling Stones', go).

□ **End Example**

**Example 5.6** *This example illustrates application of instances and object-participation on objectified relations. Assume a world in which persons have recorded songs. Some of these recordings have been produced by a producer. This situation is depicted in figure 5.8 and formally defined as:*

1. Relation(FACTUAL, PERF, record, [(ag, person(p1)), (go, song(s1))]).
2. Relation(FACTUAL, PERF, produce, [(ag, producer(p2)),
(go, Relation(FACTUAL, PERF, record, [(ag, person(p1)), (go, song(s1))])]).

□ **End Example**

### 5.3.2 Hypernym relations

The (well-known) data modelling concept *specialization / generalization* is used to define a *is a*-relation between object types. An *is a*-relation forms a relation between a so-called *subtype* and *supertype*. A linguistic designation for the superclass is *hypernym* while the term *hyponym* is used to refer to the subclass. An object type Y is a hypernym of object type X, iff every instance of X is an instance of Y. A hypernym relation can be found between student and person, because every student is a person. This is formally defined as follows: Hypernym(student, person). Hypernym-relations constitute a partial order and thus have a transitive nature which is illustrated in (3):

[2] As a consequence of the *Closed World Assumption* (CWA) the truth values of propositions is completely determined by their definitions as recorded in the knowledge base. The knowledge base can be seen as a formalized system of rules and facts. Propositions are true, only if they can be derived by performing inferences based on these facts and rules. All propositions that should be true, must in one way or another be derivable from the knowledge base. Some propositions however, are so elementary that including proper definitions of facts and rules to derive these propositions would only result into pollution of the knowledge base. Most such propositions are namely already defined outside the formal rule-system, and are considered to be standard propositions (or operations). Execution of such a definition is indicated by the symbol Exec.

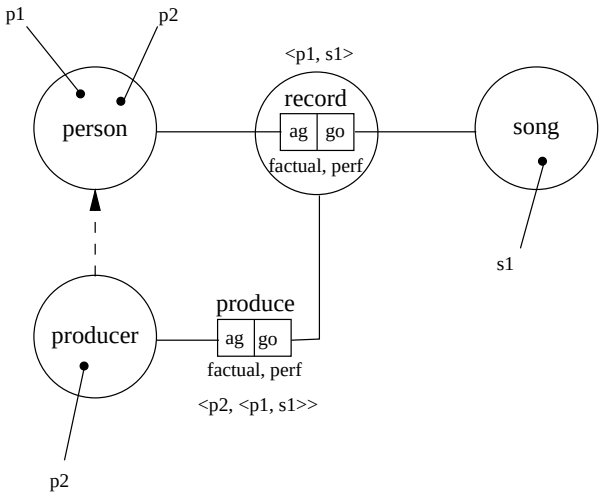


Figure 5.8: Objectification of relations in PSM$^{2+}$.

(3) from:
*Every student is a person*
and *Every person is a life form*
it may be concluded:
*Every student is a life form*

A simple inference rule (4) can be added to the lexicon to capture this transitive nature of hypernym-relations. Simple rules like these extend the reasoning capabilities of the inference engine considerably. Consider figure 5.9 in which Hypernym and Instance relations are combined. When hypernym and instance-relations are combined, new conclusions can be drawn. The inferences made in (5) are a direct result of the defined hypernym and instance-relations visualized in figure 5.9. The conclusions from (5) can be drawn by adding inference rule (6).

(4) If (Hypernym(X, Y) and Hypernym(Y, Z))
then Hypernym(X, Z).

(5) *John is a student*
*John is an enrollee*
*John is a learner*
...

(6) if (Instance(X, Y) and Hypernym(Y, Z))
then Instance(X, Z).

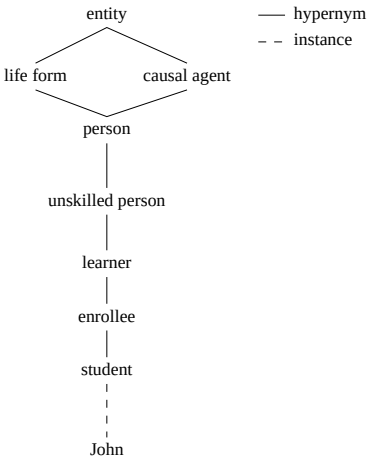


Figure 5.9: Hypernym and Instance relations according to WordNet.

**Axiom 5.1** TRANSITIVITY OF OBJECT-PARTICIPATION
*Object-participation is transitive with respect to hypernym links. This transitivity is captured by the following rule*:

if ObjectPart (Modality, Tense, Relation, Args, Sub, Instance, Role)
and Hypernym(Sub, Super)
then ObjectPart (Modality, Tense, Relation, Args, Super, Instance, Role).

□ **End Axiom**

**Example 5.7** *This example shows an application of the transitivity of object-participation as given above:*

ObjectPart (FACTUAL, PERF, play, [(ag, person('John')), (go, instrument(trumpet))], person, 'John', ag)
∧ Hypernym(person, life_form)
⊢ ObjectPart (FACTUAL, PERF, play, [(ag, person('John')), (go, instrument(trumpet))], life_form, 'John', ag)

□ **End Example**

### 5.3.3 Lexical semantical consequence relations

A lexical semantical consequence is a complex relation, in the sense that it is actually a *set* of different relation types. A lexical semantical consequence relation defines the analytical inferences that belong to what is commonly known as common sense. A lexical semantical consequence relation describes the 'obvious' inference that is contained in the meaning of the words. Consider for example the inference below:

*John is a bachelor* ⊢ *John is not married*

The inference can be explained by the introduction of the concept of *meaning postulate*. This concept was first introduced by Carnap ([Car56]). The inference is not warranted by the logical form of the antecedent, but rather by a semantical property of the predicate *bachelor*. In order to capture this property a semantical consequence relation between *bachelor* and *not being married* can be adopted. More examples are provided in (7)-(10).

(7a) A person who manages something, is a manager.
(8a) A life form which sleeps, is a sleeper.
(9a) A person who plays a game, is a player.
(10a) A person who plays an instrument, is a musician.

As illustrated in (7a) the role in which objects are involved in a relation is needed to enable appropriate definition of semantical consequence relations. In this case the semantic role of *agent* can be mentioned. In (8a) the transitive nature of object-participation is used to generalize and provide a single definition to predicate something about all sleeping life forms, e.g. persons and animals. Additional complexity is found in (9a) and (10b) in which in addition to role and relation, the involved *goal*-object type (*game* or *instrument*) is needed to provide the appropriate definition.

(7b) if ObjectPart (FACTUAL, PERF, manage, Args, person, Instance, ag)
then Instance(Instance, manager).
(8b) if ObjectPart (FACTUAL, PERF, sleep, Args, life_form, Instance, ag)
then Instance(Instance, sleeper).
(9b) if ObjectPart (FACTUAL, PERF, play, Args, person, Instance, ag)
and ObjectPart (FACTUAL, PERF, play, Args, game, _Game, go)
then Instance(Instance, player).
(10b) if ObjectPart (FACTUAL, PERF, play, Args, person, Instance, ag)
and ObjectPart (FACTUAL, PERF, play, Args, instrument, _Instrument, go)
then Instance(Instance, musician).

The definitions in (7b) and (8b) show that in principal every action performed by an agent forms a *subtype defining action*. This can be generalized by the introduction of a special binary relation

ActionAgent that captures trivial semantical relations between agents and actions performed by that agent ((11) and (12)). A general inference rule (13) is introduced to make the inferences.

(11) ActionAgent(manage, manager).
(12) ActionAgent(sleep, sleeper).
(13) if ObjectPart (FACTUAL, PERF, Relation, Args, person, Instance, ag)
and ActionAgent(Relation, Agent)
then Instance(Instance, Agent).

The enrichment of the retrieval language as illustrated above has also been studied in [HPW94]. In that report the extension of the 'information grammar' is established by macro-definitions. These macro-definitions however show an essential difference with the approach sketched above. Macro-definitions are domain specific in the sense that these definitions strongly depend on the arbitrary chosen names of predicators in the underlying information model. A predicator defines the participation of an object type in a certain relation. The function of a predicator name is to explain the meaning of a participation as well as possible. Knowledge about these chosen names is necessary in order to define appropriate macro-definitions. Because PSM$^{2+}$ is based on the semantical linguistic theory of functional grammar, the meaning of participations modeled by PSM$^{2+}$ is restricted to a small set of semantical roles. Focus is therefore directly on the meaning of the participation instead of on the name explaining the meaning as well as possible. Semantical roles unambiguously define the meaning of the participation. The use of these roles takes away the need of precise knowledge about the underlying information model. Semantical roles therefore enable universal (domain independent) definitions exceeding the scope of macro-definitions.

While in (7)-(9) trivial semantical relations are defined between morphologically related words like *manage-manager*, *sleep-sleeper* and *play-player* ,(10) shows an example of a semantical relation between words that are not morphologically related. The following lines provide more examples of non-morphologically related words. Consider the inferences in (14)-(15).

(14a) A life_form which is being killed, is dead.
(14b) if ObjectPart (FACTUAL, PERF, kill, Args, life_form, Instance, go)
then is_dead(Instance).
(15a) A person who kills somebody, is a murderer.
(15b) if ObjectPart (FACTUAL, PERF, kill, Args, person, Instance, ag)
then murderer(Instance).

### 5.3.4 Meronym relations

A meronym-relation is a complex relation in the sense that it is actually a set of different relation types. The types of meronym-relations in this set, have a different semantical load. Some of these relations are transitive, while others are not. It is important to recognize these different types of meronym-relation to avoid strange, undesired or illegal inferences. An example of an undesired inference is given in (16). In this example two different meronym relations are illegally combined in a transitive way to form an inference. The result of that inference does not correspond with our *mental vision* on how our world is arranged. The resulting statement seems to be semantically anomalous.

(16) from:
*The car park contains one car.*
and *That car has a spoiler.*
it may **not** be concluded:
* *The car park has a spoiler.*

Frequently, meronym (often confused with *part-of* or *part-whole* relations) and hypernym ('is a') relations are compared with each other because both are assumed to be asymmetric and transitive and can bring about a hierarchy (or taxonomy) of terms. This section however, shows that a *part-of* relation is a *specific type* of meronym relation. Although some meronym relations can be used to constitute such a hierarchy, example (16) has shown that other meronym relations do not. Therefore the complex meronym relation has to be studied in more detail. In the following lines different meronym-relations types are distinguished, and their properties are investigated. The authors of [Mil93] have distinguished seven meronym-relation types, listed below.

1. object-componentrelation!meronym!object-component (*tree-branch*)
2. set-element (*forest-tree*)
3. quantity-portion (*cake-portion*)
4. object-material (*airplane-aluminum*)
5. region-place (*Holland-Amsterdam*)
6. process-phase (*grow up-adolescence*)
7. activity-symptom (*shopping-pay*)

As can be observed, the first five meronym-relation types operate on objects (nouns), while the last two relation types *process-phase* and *activity-symptom* include verbs. The existence of a meronym relation is often reflected by verbs like *to have*, *to possess*, *to contain*, *to consist (of)*, etc. Note that the verb *to have* is also used for other purposes, that strictly spoken do not represent a meronym-relation. The first purpose is to express an attribute (or property)-relation which specifies the property of an object. An example of such a relation is *A person has a name*. Secondly, the verb *to have* can be used to express a kind of possession between autonomous object types, .e.g. *boy-girlfriend* ([Dig89, page 29]):

1. object-property (*person-name*)
2. object-object (*boy-girlfriend*)

To make reasoning about meronym relations possible, the logical behavior of these different meronym types have to be studied. The author of [Sch94] provides a formal mathematical discussion on a.o. meronym types. Studies like these found the extension of the inference engine to enable intelligent communication. The knowledge base can be extended with a simple inference rule to capture transitivity of meronym-relation types. Results of this study are used to provide an overview as represented in table 5.2. To start with, some trivial remarks can be made: none of these relations is reflexive nor symmetric and all meronym types are asymmetric ([Gri89]).

| | Mathematical property | | | | | |
|---|---|---|---|---|---|---|
| Meronym-type | reflexive | irreflexive | symmetric | asymmetric | transitive | a-transitive |
| Object-component | | * | | * | * | |
| Set-element | | * | | * | | |
| Quantity-portion | | | | * | * | |
| Object-material | | * | | * | * | |
| Region-place | | * | | * | * | |

Table 5.2: Mathematical properties of class (A) meronym relations

Special relations are introduced to capture the meronym relations of table 5.2: ObjectComp, SetElement, QuantPort, ObjectMat and Regplace. Transitivity of these relations can be modeled straightforward by a single inference rule. The inference rule that reflects the transitive nature of the ObjectComp-relation is given below:

if ObjectComp(A, B)
and ObjectComp(B, C)
then ObjectComp(A, C)

The semantical anomaly sketched in (16) can now be explained as follows. The sentence *The car park contains one car* is of type *set-element* while the sentence *The car has a spoiler* belongs to another type, namely *object-component*. Note that one of these relation is not even transitive.

The remaining relations *object-object* and *object-property* reflected by the use of the verb *to have* can now be discussed. The use of *to have* as a principal verb in natural language as expressed by

an *object-object* relation can be handled in our PSM$^{2+}$-formalism by the ordinary Relation symbol. See (17) which expresses the relation *A boy has a girlfriend.*

(17) Relation(FACTUAL, PERF, have, [(zero, boy(B), (go, girlfriend(G))]).

To avoid confusion with *object-property* relations, a special relation Property is introduced.

**Definition 5.3** OBJECT PROPERTY
*Iff an object 'Object' of object type 'ObjectType' has a property 'Property' with value 'Value', then the following statement holds:*
Property(ObjectType, Instance, Property, Value).

which is defined as follows[3]:
ObjectPart (FACTUAL, PERF, Have_Prop, Args, ObjectType, Instance, zero)
$\wedge$ ObjectPart (FACTUAL, PERF, Have_Prop, Args, Property, Value, go)
$\equiv$ Property(ObjectType, Instance, Property, Value).

□ **End Definition**

**Example 5.8** *This example shows some applications of the definition of object property*[4].

- Property(boy, _Instance, age, _Value). $\equiv$ A boy has an age.
- Property(boy, 'John', age, '12'). $\equiv$ John, who is a boy, is 12 years old.

□ **End Example**

Properties have a transitive nature with respect to hypernyms. Alternatively formulated, properties of a subtype are inherited of a supertype. If for example the statements *A person has an age* and *Every student is a person* hold, then it may be concluded *A student has an age.* This inference is enabled by introduction of the following inference rule.

**Axiom 5.2** INHERITANCE OF PROPERTIES
*If an object type 'Super' has a certain property 'Property' and their exists an hypernym relation between object type 'Sub' and 'Super' then 'Sub' has that same property:*
if Property(Super, Instance, Property, Value)
and Hypernym(Sub, Super)
then Property(Sub, Instance, Property, Value).

□ **End Axiom**

Conform the OO-paradigm, every object has a unique identifier. In PSM$^{2+}$-models the instance name (e.g. 'John') is used as such an identifier. This identifier distinguishes the instance from all other instances of all object types. The following lemma is based on this unique instance-identification.

**Lemma 5.1** Property(Sub, I, P, V), Hypernym(Sub, Super) $\vdash$Property(Super, I, P, V).

**Proof**

$$\dfrac{\dfrac{\dfrac{\text{Property(O, I, P, V)}}{\text{ObjectPart(FACTUAL, PERF, have\_prop, A, O, I, zero)}}\,\text{Def. 5.3} \quad \text{Hypernym(O, Super)}}{\text{ObjectPart(FACTUAL, PERF, have\_prop, A, Super, I, zero)}}\,\text{Ax. 5.1} \quad \dfrac{\text{Property(O, I, P, V)}}{\text{ObjectPart(FACTUAL, PERF, have\_prop, A, P, V, go)}}\,\text{Def. 5.3}}{\text{Property(Super, I, P, V)}}\,\text{Def. 5.3}$$

□ **End Lemma**

[3] A special relation Have_Prop is introduced to distinguish possession of properties from all other relations.
[4] An underscore '_' denotes an anonymous variable.

### 5.3.5 (Partial) synonym relations

One of the most well-known semantical relations is the synonym-relation . A synonym relation describes a relation between two terms with identical meaning. Examples of such combinations are *bike-bicycle* and *house-lodging*, etc. Although the logical behavior of synonym relations seems to be very straightforward (a synonym relation is reflexive, symmetric and transitive), inferences may produce undesired results as illustrated in example (18).

(18) from:
*A house is a building.*
and *A public lavatory is a building.*
it may **not** be concluded:
* *A house is a public lavatory.*

The problem arises from the fact that *house*, *public lavatory* and *building* are not pure synonyms, but can be regarded as synonyms when a certain *context* is assumed. When this context is ignored, the resulting inferences are illegal[5]. Synonyms which are only valid in a certain context are called partial (or quasi)-synonyms. Often, it seems to be hard to demarcate the necessary and sufficient context and to formulate this context adequately. Only simple and straightforward answers to these questions can be found when partial synonyms are defined on *adjectives*[6]. Useful results can be obtained when the context of adjectives is provided by means of the object type on which adjectives operate. Consider for example the (partial) synonym of *bad* in the context of object type *bread*. If *bread* is the opposite of *fresh* it is said to be *stale*, however if *meat* is the opposite of *fresh* it is said to be *tainted*. *Stale* and *tainted* can therefore both be regarded as partial synonyms of the opposite of *fresh*. However, the use of the terms *stale* and *tainted* is restricted to *bread* and *meat* respectively, because a statement like *The stale meat* or *The tainted bread* is undesired. Such opposites like *bad-fresh*, *stale*-fresh and *tainted-fresh* will be discussed in more detail in one of the following sections about *antonym*-relations. Two special relations Synonym and PartialSynonym are introduced to capture (partial) synonymous terms. (Partial) synonym relation can be defined on objects(nouns), relations(verbs) and adjectives (operating on properties).

**Definition 5.4** SYNONYMOUS TERMS
*A special Synonym-relation is introduced to capture that 'Term$_1$' is considered as a synonym of 'Term$_2$'. This Synonym relation can operate on verbs, nouns, and adjectives.*
Synonym(Term$_1$, Term$_2$)

□ **End Definition**

**Axiom 5.3** SYMMETRIC PROPERTY OF SYNONYMOUS TERMS
*If term 'Term$_1$' is a synonym of 'Term$_2$' then 'Term$_2$' is also a synonym of 'Term$_1$':*
if Synonym(Term$_1$, Term$_2$)
then Synonym(Term$_2$, Term$_1$)

□ **End Axiom**

**Axiom 5.4** TRANSITIVE PROPERTY OF SYNONYMOUS TERMS
*If term 'Term$_1$' is a synonym of 'Term$_2$' and 'Term$_2$' is a synonym of 'Term$_3$' then 'Term$_1$ is also a synonym of 'Term$_3$':*
if Synonym(Term$_1$, Term$_2$)
and Synonym(Term$_2$, Term$_3$)
then Synonym(Term$_1$, Term$_3$)

□ **End Axiom**

**Definition 5.5** PARTIAL SYNONYMOUS ADJECTIVES
*When partial synonyms are defined on adjectives the context is provided by specifying the object type 'ObjectType' on which the adjectives operate. A special PartialSynonym-relation is introduced for this purpose:*
PartialSynonym(ObjectType, Adjective$_1$, Adjective$_2$).

[5] In WordNet this context is provided by means of different senses of a term.
[6] A discussion on partial synonym relations operating on relations and objects falls outside the scope of this document.

□ **End Definition**

**Example 5.9** *This example shows some applications of partial synonym relations. The synonym relations defined below have a restricted application domain and are therefore classified as partial synonyms.*

- PartialSynonym(meat, bad, tainted).
- PartialSynonym(bread, bad, stale).

□ **End Example**

**Axiom 5.5** SYMMETRIC PROPERTY OF PARTIAL SYNONYMOUS TERMS
*If term 'Term$_1$' is a partial synonym of 'Term$_2$' in the context of object type 'ObjectType' then 'Term$_2$' is also a synonym of 'Term$_1$' in that same context:*
if PartialSynonym(ObjectType, Term$_1$, Term$_2$)
then PartialSynonym(ObjectType, Term$_2$, Term$_1$)

□ **End Axiom**

**Axiom 5.6** TRANSITIVE PROPERTY OF PARTIAL SYNONYMOUS TERMS
*If, in the context of object type 'ObjectType', term 'Term$_1$' is a synonym of 'Term$_2$' and 'Term$_2$' is a synonym of 'Term$_3$' then 'Term$_1$' is also a synonym of 'Term$_3$' in that same context:*
if PartialSynonym(ObjectType, Term$_1$, Term$_2$)
and PartialSynonym(ObjectType, Term$_2$, Term$_3$)
then PartialSynonym(ObjectType, Term$_1$, Term$_3$)

□ **End Axiom**

### 5.3.6 Antonym relations

In this section focus is on a special semantical relation between properties of objects (class (C)). The presence of properties is reflected by the use of adjectives in sentences like (19) and (20). Sentence (19) presupposes the presence of property *age* of object type *boy*, while sentence (20) describes the *color* of the book involved.

(19) The old man.
(20) The green book.

An antonym-relation is a relation between adjectives that represents each others opposite semantical load, e.g. *hot* versus *cold* and *young* versus *old*. A more detailed discussion on adjectives is provided before entering the topic of application of antonym-relations in inference processes. Although antonym relations can also be defined on verbs (class (B)), e.g. Antonym(*to send*, *to receive*)), in this section focus is on antonym relations between adjectives. Adjectives can be used in different positions within a sentence while representing the same semantical load, compare sentence (21) with (22).

(21) The nice girl.
(22) The girl is nice.

While in sentence (21) adjective *nice* is placed before noun *girl*, in (22) it is placed after noun *girl* and connected to *girl* via the verb *to be*. The use of adjectives in a construction like (21) is called *attributive* while a construction like (22) is called *predicative*. Some adjectives can be applied in both construction-types, as illustrated in (21) and (22). An example of an adjective that can only be used in a predicative construction is *ready*.

(23) The job is ready.
*(24) The ready job.

More often, it occurs that adjectives can only be used in *attributive* constructions. Consider adjectives like *former*, *lean*, *civil* and *average*. Adjectives which can not be applied in a predicative construction are called *non-predicative adjectives* , while adjectives that can not be applied in attributive constructions are called *non-attributive adjectives*. Linguists have found correlations between adjective construction-types and other linguistic phenomena ([Mil93]):

- predicative and non-predicative adjectives can **not** be combined with words like *and*, *or* and *but*: ∗*The complex and civil law.*
- non-predicative adjectives are not gradual: ∗*The very civil law.*
- nominalization of non-predicative adjectives is impossible: ∗*The 'leanity' of the milk.*

The closer observation in the remaining part of this section leaves the attributive-adjectives aside. The word-association test referred to in section 5.2 has also been applied on adjectives. Tests like these have shown valuable results. The response word on adjective *hot* is in almost all cases *cold*: its antonym! This made psycho-linguists recognize the importance of antonym-relations in our mental lexicon. A more formal and detailed discussion on adjectives and antonym-relations is needed to extend the knowledge base with antonym-relations and to incorporate this knowledge during inferences. The following types of antonym-relations are recognized in literature ([]). All these types of antonym-relations have one thing in common: the relation expresses an opposite or contrast on a given domain or parameter. Obviously, antonym-relations are symmetric.

1. contradictory terms (*complete-incomplete*, *dead-alive*)
2. contrary terms (*black-white*, *big-small*)
3. reversal terms (*constructive-destructive*, *loose-tight*)
4. contrasting terms (*watchful-carefree*, *dry-damp*)
5. relative terms (*brother-sister*)
6. complementary terms (*question-answer*)

As can be observed above, antonym relations exist between adjectives (1-4) and nouns/verbs(5-6). Because in this section focus is on class (C) relations, relative and complementary term-pairs fall outside the scope. Contradictory, contrary, reversal and contrasting terms on the other hand are discussed in the following lines.

At first sight, the typology of adjectives sketched above seems to be chosen arbitrary. However, this is actually not true: the typology groups adjectives in types with different semantical loads. The first criterion used to classify adjectives is *polarity*. Polarity specifies the number of dimensions of the parameter referred to by the adjective. A one-dimensional parameter has two poles, and is therefore called *bipolar*. An example of a bipolar parameter is age, height and temperature but also beauty, legibility and readability. Examples of multi-dimensional or n-polar (n>2) parameters are *color*, *shape* and *phase* (solid/liquid/gas). Multi-dimensional or non-bipolar parameters make reasoning very complex and are beyond the scope of this document . Therefore in this document focus is on bipolar parameters. Another criterion to classify adjectives is the *gradual* nature of the involved parameter. Parameters like age, weight and temperature are gradual, while a parameter like *gender* and the parameter involved in adjectives *open / closed* is not.

The first type of antonym-relation to be discussed is the relation between contradictory terms. Contradictory terms originate from Aristotle's logic.

**Definition 5.6** CONTRADICTORY PROPOSITIONS
*Two propositions P1 and P2 are contradictory if the truth of one of these implies the falsity of the other, and vice versa: the falsity of one of these implies the truth of the other:*

$$\text{contradictory}(P1, P2) \equiv (P1 \rightarrow \neg P2) \wedge (P2 \rightarrow \neg P1) \wedge (\neg P1 \rightarrow P2) \wedge (\neg P2 \rightarrow P1)$$

□ **End Definition**

**Corollary 5.1** CONTRADICTORY PROPOSITIONS
By the definition of contradictory terms, the following inferences can be made:

$$\text{contradictory}(P1, P2) \vdash P1 \rightarrow \neg P2$$
$$\text{contradictory}(P1, P2) \vdash \neg P1 \rightarrow P2$$
$$\text{contradictory}(P1, P2) \vdash P2 \rightarrow \neg P1$$
$$\text{contradictory}(P1, P2) \vdash \neg P2 \rightarrow P1$$

□ **End Corollary**

**Definition 5.7** CONTRADICTORY TERMS
*A Contradictory-relation is introduced to capture contradictory term-pairs 'Term$_1$' and 'Term$_2$' of a given property 'Property' of an object type 'ObjectType':*
Contradictory(ObjectType, Property, Term$_1$, Term$_2$).

□ **End Definition**

The definitions above show that contradictory terms operate on a bipolar, non-gradual parameter. Contradictory terms have a typical logical behavior illustrated in (25)-(28).

(25) If a person is male, it is not female.
(26) If a person is female, it is not male.
(27) If a person is not male, it is female.
(28) If a person is not female, it is male.

The logical behavior of contradictory terms can be simulated by our prototype PSM$^{2+}$-expert system by introducing inference rules (29) and (30). Rule (29) handles inferences (25) and (26), while rule (30) deals with the suggested inferences of (27) and (28).

(29) if Property(ObjectType, Instance, Property, Value)
and (Contradictory(ObjectType, Property, Value, Antonym)
or Contradictory(ObjectType, Property, Antonym, Value))
then Property(ObjectType, Instance, Property, not(Antonym)).
(30) if Property(ObjectType, Instance, Property, not(Value))
and (Contradictory(ObjectType, Property, Value, Antonym)
or Contradictory(ObjectType, Property, Antonym, Value))
then Property(ObjectType, Instance, Property, Antonym).

While these inference rules are very straightforward, applications of rule (30) may result into difficulties. An example of such an application comes into being when the rule is applied on inappropriate object types. In such cases the parameter is not applicable at that specific object type. Consider the following statement: *The rock is not alive.* Although being dead or alive is normally not predicated about rocks, the statement is factually true. However, if rule (30) is applied on this true statement, the result proves to be unsatisfactory: *The rock is dead.*

As illustrated in (29) and (30) contradictory terms have a dual nature. The truth of a terms implies the falsity of the other, and the falsity of a term implies the truth of the other. This dual character is a result of the fact that contradictory terms operate on non-gradual parameters. However, when studying gradual parameters like *age* the inference of (30) is not allowed, as illustrated in (31).

(31) from:
*The man is not old*
it may **not** be concluded:
*The man is young.*

Obviously, parameters with a gradual nature have more than one possible value, making the inference (30) and (31) illegal. The inference mentioned in (29) however, is legal. Adjectives with a gradual nature constitute the antonym-type *contrary terms*, which can be defined as follows.

**Definition 5.8** CONTRARY PROPOSITIONS
*Two propositions P1 and P2 are contrary if although they can* **not** *both be true, they can be both false at the same time.*
contrary(P1, P2) $\equiv$ (P1 $\rightarrow \neg$P2) $\wedge$(P2 $\rightarrow \neg$P1)
□ **End Definition**

**Corollary 5.2** CONTRARY PROPOSITIONS
By the definition of contrary terms, the following inferences can be made:
contrary(P1, P2) $\vdash$ P1 $\rightarrow \neg$P2
contrary(P1, P2) $\vdash$ P2 $\rightarrow \neg$P1

□ **End Corollary**

**Definition 5.9** CONTRARY TERMS
*A Contrary-relation is introduced to capture contrary term-pairs 'Term$_1$' and 'Term$_2$' of a given property 'Property' of an object type 'ObjectType':*
Contrary(ObjectType, Property, Term$_1$, Term$_2$).

□ **End Definition**

This definition shows that contrary terms are terms on a bipolar gradual parameter. The inference of (29) also holds for contrary terms and is modified for this purpose (32). However, as stated before the inference of (30) is illegal in the scope of contrary terms.

(32) if Property(ObjectType, Instance, Property, Value)
and (Contrary(ObjectType, Property, Value, Antonym)
or Contrary(ObjectType, Property, Antonym, Value))
then Property(ObjectType, Instance, Property, not(Antonym)).

The next antonym-type to be discussed are pairs of *reversal terms*. Reversal terms have a semantical load that slightly differs from the antonym types mentioned above. While contradictory and contrary terms are terms describing a state, reversal terms are applied on actions and processes (see section 4.2.2). Reversal terms are pairs, such that one of them expresses the progress and the other the decline of an action or process. Reversal terms can be verbs like *build-pull down*, but in this section focus will be on adjectives. Examples of reversal adjectives are *constructive-destructive* and *dressed-naked*. Because reversal terms have an underlying action or process that normally elapses gradually, they have the same properties as contrary terms. Therefore their logical behavior is analogically defined as follows.

**Definition 5.10** REVERSAL PROPOSITIONS
*Two propositions P1 and P2 are reversal if their parameter has an underlying action or process and although P1 and P2 can* **not** *both be true, they can be both false at the same time.*
reversal(P1, P2) $\equiv$ (P1 $\rightarrow \neg$P2) $\wedge$(P2 $\rightarrow \neg$P1)

By the definition of reversal terms, the following inferences can be made:
reversal(P1, P2) $\vdash$ P1 $\rightarrow \neg$P2
reversal(P1, P2) $\vdash$ P2 $\rightarrow \neg$P1

□ **End Definition**

**Definition 5.11** REVERSAL TERMS
*A Reversal-relation is introduced to capture reversal term-pairs 'Term$_1$' and 'Term$_2$' of a given property 'Property' of an object type 'ObjectType':*
Reversal(ObjectType, Property, Term$_1$, Term$_2$).

□ **End Definition**

The last antonym-type to be discussed is the set of *contrasting* term pairs. Reconsider the following examples of contrasting terms: *dry-damp*, *rich-needy*. Although terms like these describe a certain contrast in a way that something that is dry is not damp and someone who is rich not needy, these words are not directly recognized as antonym pairs. The obvious (direct) antonym of *dry* seems to be *wet* and *poor* is mostly recognized as the direct antonym of *rich*. On their turn, *needy* and *damp* are recognized as partial synonyms of *poor* and *wet* respectively. The direct (contrary) antonym-relation between *wet* and *dry* forms a basis for indirect contrasting terms as *dry-damp*, *wet-barren* etc. This situation is depicted in figure 5.10. While contradictory, contrary and reversal terms are direct antonym-relations, contrasting terms constitute an indirect antonym-relation. The indirect antonym-relation of contrasting terms is based on a single direct antonym relation(contradictory, contrary or reversal) and one or more partial synonym relations.

**Definition 5.12** CONTRASTING TERMS
*Given a direct antonym relation (contradictory, contrary or reversal) between two terms 'Term$_1$' and 'Term$_2$': if 'Syn$_1$ is a partial synonym of 'Term$_1$' then 'Syn$_1$' is a contrasting term of 'Term$_2$', or - vice versa - if 'Syn$_2$ is a partial synonym of 'Term$_2$' then 'Syn$_2$' is a contrasting term of 'Term$_1$'.*

□ **End Definition**

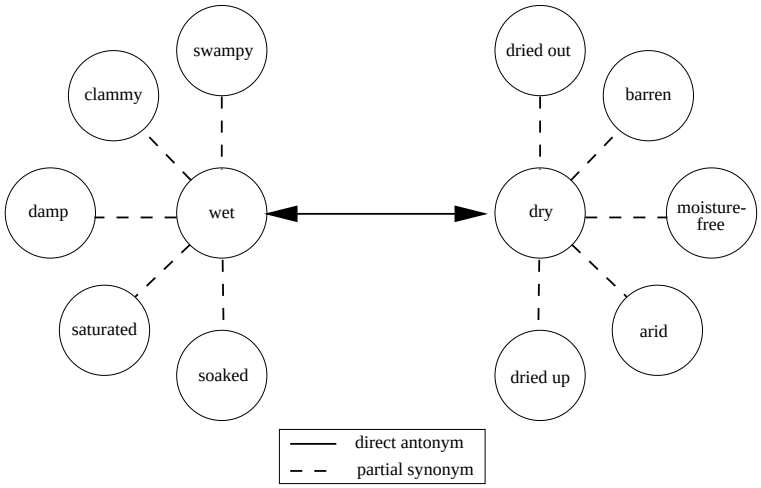


Figure 5.10: Contrasting terms based on a single direct antonym relation.

It follows from the definition above that, a contrasting antonym relation is not elementary but formed by combining a direct antonym relation and partial synonym relation. Therefore the logical behavior of contrasting terms is completely determined by the logical behavior of its composing elementary semantical relations. The knowledge base of our prototype system can be extended with two inference rules. Rule (33) is introduced to define direct antonym relations which is used by rule (34) to define contrasting terms. Table 5.3 provides an overview of the different direct antonym-relation types discussed in this section.

(33) if Contradictory(ObjectType, Property, Term1, Term2)
or Contrary(ObjectType, Property, Term1, Term2)
or Reversal(ObjectType, Property, Term1, Term2)
then DirectAntonym(ObjectType, Property, Term1, Term2).

(34) if DirectAntonym(ObjectType, Property, Term1, Term2)
and PartialSynonym(ObjectType, Syn1, Term1)
then Contrasting(ObjectType, Property, Syn1, Term2).

| Relation-type | Gradual | Inferences P1 $\rightarrow \neg$P2 | P2 $\rightarrow \neg$P1 | $\neg$P1 $\rightarrow$ P2 | $\neg$P2$\rightarrow$ P1 | SoA-type |
|---|---|---|---|---|---|---|
| Contradictory | | * | * | * | * | State |
| Contrary | * | * | * | | | State |
| Reversal | * | * | * | | | Process/Action |

Table 5.3: Overview of direct antonym-relations

### 5.3.7 Semantic consequence relations operating on verbs

Hypernym and meronym relations discussed in one of the previous sections have been applied on nouns representing object types. These hypernym and meronym relations can therefore be classified as class (A) semantic relations. In this section it is studied how the concepts of hypernym and meronym can be applied on verbs, representing actions (class (B)). This constitutes a new class of semantic consequence relations, that strictly operates on verbs. The examples below show the existence of semantic relations between verbs.

(35a) Someone who strolls, also walks.
(35b) Someone who walks, also moves.
(36) Someone who snores, also sleeps.
(37) Someone who is being released, must have been arrested before.

Strolling, walking and moving in (35) are actions that are performed simultaneously. Someone who strolls also walks and moves at the same time. Sentences (35a) and (35b) suggest a kind of

hypernym relation between verbs. Linguists have actually recognized this relation and called a *troponym*-relation. A troponym relation is a partial ordering relation and thus constitutes a hierarchy of verbs. A troponym relation between a $verb_1$ and a $verb_2$ exists if the following statement holds:

(38) $<Verb_1>$*-ing is a specific way of* $<Verb_2>$*-ing.*

This statement applied to (35a) and (35b) yields the following (obviously) truly results: *Strolling is a specific way of walking* and *Walking is a specific way of moving.* The characteristic property of troponym-relations is simultaneity: if someone strolls he walks at the same time and if he stops strolling he also stops walking. Therefore a special relation Simultaneity is introduced to capture troponym relations. See figure 5.11(a) for a visualization.

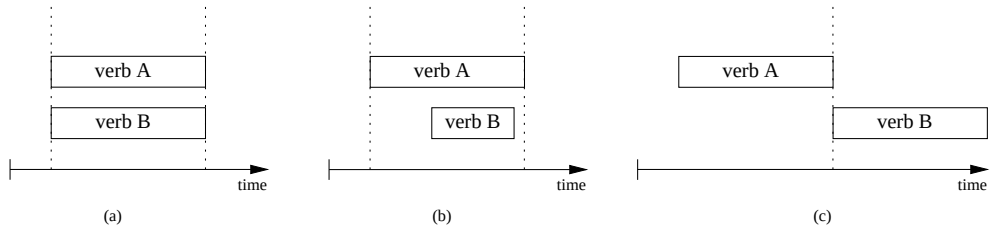


Figure 5.11: Semantical consequence relations between verbs: (a) simultaneity, (b) proper inclusion and (c) presupposition

**Definition 5.13** SIMULTANEITY
*Iff there is a time span in which action A and B (denoted by 'verbA' and verbB' respectively) are performed completely simultaneously, i.e. there is no moment in time at which A is performed and B is not performed or at which B is performed and A is not performed, then the following statement holds:*
Simultaneity(verbA, verbB).

□ **End Definition**

**Example 5.10** *This example shows some applications of a semantic consequence relation called Simultaneity.*

- Simultaneity(stroll, walk): *Strolling is a specific way of walking.*
- Simultaneity(walk, move): *Walking is a specific way of moving.*
- Simultaneity(hit, touch): *Hitting is a specific way of touching.*

□ **End Example**

**Axiom 5.7** TRANSITIVE NATURE OF SIMULTANEITY
if Simultaneity(VerbA, VerbB)
and Simultaneity(VerbB, VerbC)
then Simultaneity(VerbA, VerbC)

□ **End Axiom**

**Axiom 5.8** INFERENCES INCLUDING SIMULTANEITY
if Relation(Modality, Tense, VerbA, Args)
and Simultaneity(VerbA, VerbB)
then Relation(Modality, Tense, VerbB, Args).

□ **End Axiom**

**Lemma 5.2**
ObjectPart (Modality, Tense, VA, Args, ObjectType, Instance, Role)
$\wedge$ Simultaneity(VA, VB)
$\vdash$ ObjectPart (Modality, Tense, VB, Args, ObjectType, Instance, Role)

$$\dfrac{\dfrac{\dfrac{\text{ObjectPart(M, T, Va, A, O, I, R)}}{\text{Relation(M, T, Va, A)}}\ \text{Def. 5.2} \qquad \text{Simultaneity(Va, Vb)}}{\text{Relation(M, T, Vb, A)}}\ \text{Ax. 5.8} \qquad \dfrac{\text{ObjectPart(M, T, Va, A, O, I, R)}}{\text{member((R, instance(I, O)), A)}}\ \text{Def. 5.2}}{\text{ObjectPart(M, T, Vb, A, O, I, R)}}\ \text{Def. 5.2}$$

**Proof**

□ **End Lemma**

**Example 5.11** *This example shows some applications of the transitivity of simultaneity. Consider a world in which the following statement holds* Person 'John' hits the ball 'MyBall'*, formally defined as:*

Relation(FACTUAL, PERF, [(ag, person('John')), (go, ball('MyBall'))]),
Simultaneity(hit, touch) ⊢

- ObjectPart (FACTUAL, PERF, hit, [(ag, person('John')), (go, ball('MyBall'))], person, 'John', ag): *'John hits 'MyBall'.*
- ObjectPart (FACTUAL, PERF, hit, [(ag, person('John')), (go, ball('MyBall'))], ball, MyBall, go): *'MyBall' is hit by 'John'.*
- ObjectPart (FACTUAL, PERF, touch, [(ag, person('John')), (go, ball('MyBall'))], person, 'John', ag): *'John touches 'MyBall'.*
- ObjectPart (FACTUAL, PERF, touch, [(ag, person('John')), (go, ball('MyBall'))], ball, 'MyBall', go): *'MyBall' is touched by 'John'.*

□ **End Example**

The type of relation illustrated in (36) does not have this typical simultaneous character of Simultaneity. Neither *Snoring is a specific way of sleeping* nor *Sleeping is a specific way of snoring* is a true statement. This litmus test shows that the semantic load of this relation differs from that of (35). Sleeping seems to be an 'activity' that is sometimes assisted by another activity, called *snoring*. If someone stops snoring, he may be still sleeping. Although snoring is temporally included by sleeping, 'actions' like these are not performed completely simultaneously: someone can be sleeping without snoring. Snoring can be viewed as a part of sleeping, and can therefore be seen as a kind of meronym of sleeping (or more precisely of *to sleep*). A special relation called ProperInclusion is introduced to avoid confusion with real class (A) meronym relations. See figure 5.11(b) for a visualization. In this case 'verb A' represents the 'activity' *to sleep* and activity *to snore* is represented by 'verb B'.

**Definition 5.14** Proper Inclusion
*Iff there is a time span in which action A and B (denoted by 'verbA' and verbB' respectively) are performed simultaneously, such that there is no moment in time at which B is performed and A is not performed and there are one or more moments in time at which A is performed and B is not performed, then the following statement holds:*
ProperInclusion(verbB, verbA).

□ **End Definition**

**Axiom 5.9** Transitive nature of Proper inclusion
if ProperInclusion(VerbA, VerbB)
and ProperInclusion(VerbB, VerbC)
then ProperInclusion(VerbA, VerbC)

□ **End Axiom**

**Axiom 5.10** Inferences including Proper Inclusion
if Relation(Modality, Tense, VerbB, Args)
and ProperInclusion(VerbB, VerbA)
then Relation(Modality, Tense, VerbA, Args).

□ **End Axiom**

The semantic load of (37) differs both from that of (35) and (36), because these actions are not performed at the same time. In (37) an action is necessarily preceded by another action. Another example of such a relation taken from the library case is *If someone returns a book, he should have borrowed that book before.* Their is no possible sequence of actions in which the returning of a book is not preceding by the borrowing of the same book by the same person. A special relation Presupposition is introduced for this purpose. See figure 5.11(c) for a visualization. In this case 'verb A' is used as denotation of *to borrow* and 'verb B' as denotation of *to return.*

**Definition 5.15** PRESUPPOSITION
*Iff action B (denoted by VerbB) is necessarily preceded by action A (denoted by verbA) then the following statement holds:*
Presupposition(VerbB, VerbA).

□ **End Definition**

**Axiom 5.11** TRANSITIVE NATURE OF PRESUPPOSITION
if Presupposition(VerbA, VerbB)
and Presupposition(VerbB, VerbC)
then Presupposition(VerbA, VerbC)

□ **End Axiom**

**Lemma 5.3**
ObjectPart (Modality, Tense, VerbA, Args, ObjectType, Instance, Role)
∧ ProperInclusion(VerbA, VerbB)
⊢ ObjectPart (Modality, Tense, VerbB, Args, ObjectType, Instance, Role)

**Proof** See proof lemma 5.2.

□ **End Lemma**

While simultaneity and proper inclusion are based on a single action, presupposition deals with two distinct actions. In a KISS-model ([Kri94]) this would lead to the introduction of a so-called *weak object type.* Presupposition defines an analytical constraint on the time order of two different actions, resulting into additional complexity:

- two different actions are involved with possibly different object types in different roles
- a formal notion of time is needed, to define that action A should be performed *before* action B is being performed.

Due to this additional complexity presupposition is more subtle than simultaneity and proper inclusion, making a general inference rule for presupposition (like axiom 5.8 and 5.10) impossible. Consider for example the time order between two actions *load* and *unload* that are used to denote loading and unloading of containers on / from ships. See figure 5.12 below. In this situation the following inference is possible, if a container C is unloaded from ship S, then C should have been loaded on S beforehand. This analytical inference is expressed in (38). Note that an extension to PSM$^{2+}$ is needed: the inclusion of satellites like time.

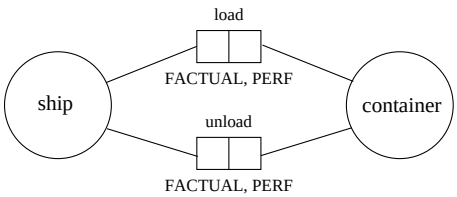


Figure 5.12: Illustration of presupposition

(38) if Relation(FACTUAL, ACTION, unload, [(ag, ship(S)), (go, container(C))]) [(time, T1)]
then Relation(NEC, PERF, load, [(ag, ship(S)), (go, container(C))]) [(time, T2)]
and (T2 before T1)

In addition to these semantic consequence relations, a more complicated relation between verbs can be recognized. Consider the visualization of a gift in figure 5.13.

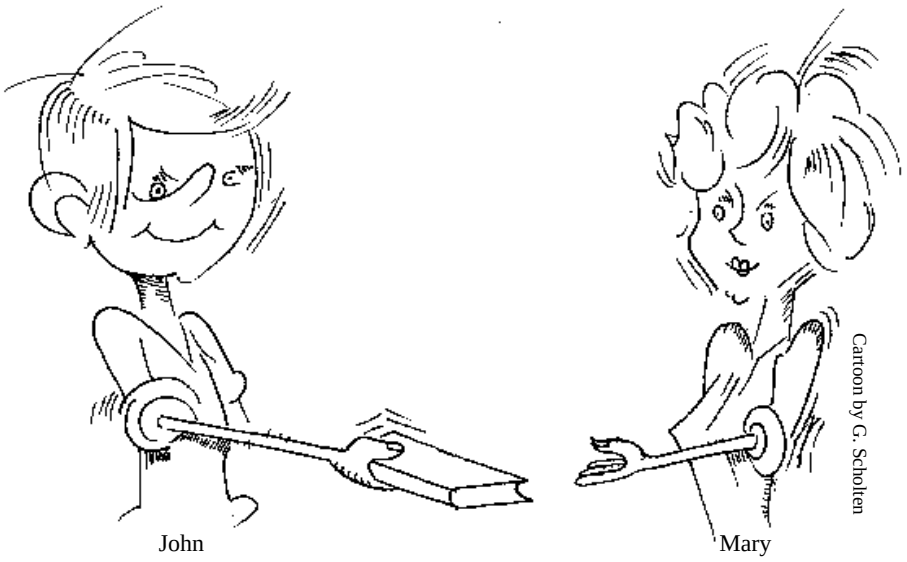


Figure 5.13: Visualization of a 'gift'.

Obviously, this miniature drama can be expressed by essentially different predications.

(39a) Relation(FACTUAL, ACTION, give, [(ag, person('John')), (go, book('the book')), (rec, person('Mary').)]).
(39b) *'John' gives the book to 'Mary'.*
(40a) Relation(FACTUAL, ACTION, receive, [(proc, person('Mary')), (go, book('the book')), (so, person('John'))]).
(40b) *'Mary' receives the book from 'John'*

These different predications are based on different predicates *to give* and *to receive* but describe the same conception depicted in figure 5.13. This demonstrates the semantic relation between the two verbs (or predicates). This relation can be defined as follows:

(41) Relation(FACTUAL, ACTION, give, [(ag, Agent), (go, Goal), (rec, Recipient)]).
≡ Relation(FACTUAL, ACTION, receive, [(proc, Recipient), (go, Goal), (so, Agent)]).

### 5.3.8 Function relations

While perceptual properties - modeled by meronym relations - have been studied by psychologists for years, relatively few attention is given to another distinctive property, called *function*-relation. While meronym-relations can often be found by visual observation, the function-relation - stating for what purpose the object is made - belongs to the domain of general world knowledge. The function of a pen e.g. is not to use it as a murder weapon but to use it a as tool for writing. Therefore a function-relation should relate an object *pen* to its characteristic function *to write*. Obviously function relations are class (D) relations. The American philosopher Jerold J. Katz believes that functional information is recorded for the greater part of objects ([Mil93]). With this statement it is possible to explain the meaning of *good* in a sentence like *This is a good pen*: the pen is good because it performs its characteristic function (*to write*) well. The word-association test mentioned in table 5.1 shows the existence of functional relations in the mental lexicon. The distinctive function of *chair* seems to be *to sit* and even belongs to the top three. The importance of functional relations in human mind (read mental lexicon) is also reflected in dictionaries; substantives are defined by means of their typical functions. Alternatively formulated, substantives are often defined by answering one of the following questions:

- For what purpose is the the substantive typically used?

- What is the typical ability of the substantive involved?

Apparently, answers to these questions provide important clues for humans to associate the right concept with the right word. This can be illustrated with some examples retrieved from Webster's dictionary:

**pen** an implement for *writing* or *drawing* with ink or a similar fluid

**finch** any of numerous small *songbirds* with short stout bills adapted for crushing seeds

The definition of *pen* shows that the first question formulated above is directly answered. The definition of *finch* answers the second question, however the typical ability *to sing* of the finch is packed in the word *songbirds*. See the following definition below:

**songbird** a bird that utters a succession of musical tones

On base of this discussion two different types of functional relations can be distinguished that correspond to the two questions posed above: *instrument* and *ability*. The first type, reflected in (42), is a semantic relation between an object and an action in which the object is used as an *instrument* by an autonomous agent ('John'). Note that this type of functional relation correspond with the notion of *Instrument* recognized by functional grammar as a satellite (see table 4.2). A special relation Instrument is introduced for this purpose:

(42) John writes a letter with a pen.
(43) The finch sings lovely.

**Definition 5.16** INSTRUMENT RELATIONS
*Iff action 'Verb' denotes a typical action that can be performed by an agent with the help of an object 'Instrument', then the following statement holds:*
Instrument(Instrument, Verb).

□ **End Definition**

**Example 5.12** *This example shows some applications of the functional instrument relation*

- Instrument(chair, sit)
- Instrument(pen, write)
- Instrument(pen, draw)
- Instrument(hammer, hammer)

□ **End Example**

In sentence (43) a different type of functional relation can be observed. While in (42) the object was used for a certain purpose, in (43) the object has complete control over the action expressed by the predication. Sentence (43) describes a so called *ability* of an autonomous object. A special relation Ability is introduced to capture this type of semantic relation.

**Definition 5.17** ABILITY RELATIONS
*Iff an object type 'ObjectType' is typically able to perform an certain action (denoted by 'Verb') then the following statement holds:*
Ability(ObjectType, Verb).

□ **End Definition Example 5.13**

*This example shows some applications of the functional ability relation*

- Ability(musician, play)
- Ability(bird, fly)

- Ability(airplane, fly)
- Ability(singer, sing)

□ **End Example**

Note that with the introduction of the Ability-relation a sound semantic foundation is found for the relation ActionAgent, which has been introduced in section 5.3.3. The Ability-relation can now be used for this purpose, because the following evidently relation holds:

(44) ActionAgent(Relation, Agent) ≡ Ability(Agent, Relation)

### 5.3.9 Gerund relations

A gerund is a substantive noun that is derived from the infinitive form of a verb by suffixing this initially with *-ing* ([Kri94]). Examples of gerunds are *to record-recording*, *to insure-insuring*, *to pay - paying*, etc. In everyday English however it has become common to replace the *-ing* suffix with a number of other suffixes such as *-ance*, *-age*, *-ment* and so on. The commonly-used form of the above examples therefore becomes: *to insure-insurance* and *to pay-payment*. A gerund is a denotation of objects resulting from objectification of fact types. Reconsider the UoD of figure 5.14 formally defined as:

- person(p1), producer(p2), song(s1), Hypernym(producer, person).
- Relation(FACTUAL, PERF, record, [(ag, person(p1)), (go, song(s1))]).
- Relation(FACTUAL, PERF, produce, [(ag, producer(p2)),
  (go, Relation(FACTUAL, PERF, record, [(ag, person(p1)), (go, song(s1))])]).

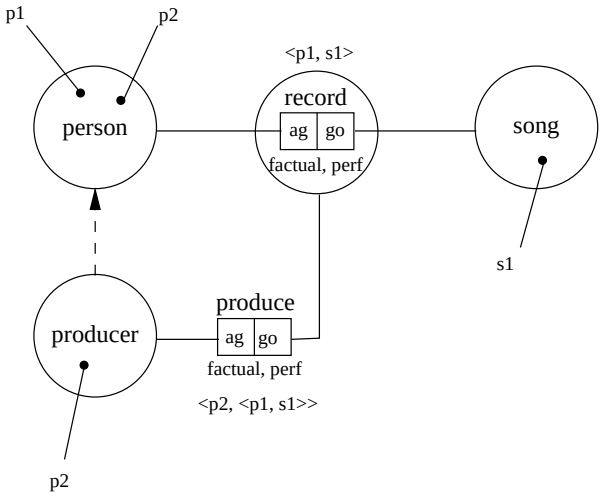


Figure 5.14: Gerund as denotation for objectified fact-types.

The objectification of fact type *record* results into a new object type *recording*. A gerund-relation exists between *record* and *recording*. Therefore the following statement holds: the instance <p1, s1> of fact type *to record* is an instance of object type *recording* (45). The same analogy holds for *to produce* and *production* (46).

(45) recording([(ag, person(p1)), (go, song(s1))]).
(46) production([(ag, producer(p2)),
(go, [(ag, person(p1)), (go, song(s1))])]).

Note that all the examples given above deal with transitive verbs. However, strictly spoken, gerunds can also be applied on intransitive verbs (*sleep*, *walk*, (*to sit*), etc.) or ditransitive verbs (*to give*, *to sell*, etc.). Sentence (47) and (48) shows an example of each category.

(47) walking([(ag, person('John'))]).
(*The walking of 'John'*)
(48) gift([(ag, person('John')), (go, book('the Bible')), (rec, person('Mary'))]).
(*The gift of 'the Bible' from 'John' to 'Mary'*)

**Definition 5.18** GERUND RELATION
*Iff noun 'Gerund' is a gerund of a certain relation (denoted by verb 'Verb') then the following statement holds:*
Gerund(Verb, Gerund)

□ **End Definition**

**Axiom 5.12** INFERENCES INCLUDING GERUND RELATIONS
*Iff there is a relation 'Relation' with arguments 'Args' and gerund 'Gerund' then 'Args' is to be considered an instance of the object type denoted by 'Gerund':*
if Relation(_Modality, _Tense, Relation, Args)
and Gerund(Relation, Gerund)
then Instance(Args, Gerund).

□ **End Axiom**

### 5.3.10 Transmission relations

The last semantic relation to be discussed is the *transmission*-relation . A transmission-relation between two objects exists if one object functions as a sender and the other as a receiver. The object to be transmitted can be a concrete or abstract object. Sentence (49) provides and example of transmission of a concrete object *book*, while (50) shows the transmission of an abstract object *apology*.

(49) John gives the book to Mary.
(50) John apologized to Mary.

In both sentences a transmission relations exists between 'John' and 'Mary' in which 'John' participates in the role of *sender* and 'Mary' in the role of *receiver*. Note that the role of receiver is recognized by *functional grammar* as the semantic role of *recipient* (see table 4.1). These roles are not influenced when (49) is alternatively formulated as done in (51), because the underlying predication remains the same.

(51a) John gives the book to Mary.
(51b) The book is given to Mary by John.
(51c) Mary is given the book by John.

Although these sentences describe the same predication $give(John)_{Ag}(the\ book)_{Go}(to\ Mary)_{Rec}$ from different perspectives, they may have a different pragmatic impact. The speaker may use sentence (51a) to realize a focus of attention on term *John*, or use (51b) or (51c) to focus on the terms *The book* and *Mary* respectively. This process is known as *pragmatic foregrounding* (([Dik89, page 218]). In the three different sentences of (51) 'John' remains 'giver' and 'Mary' remains 'taker' in the action. This firmness can be expressed by a transmission relation, in which *agent* and *recipient* respectively correspond to *giver* and *taker*. The term *giver* can be retrieved by consultation of the functional relation Ability, as discussed in the previous section, which relates *to give* to *giver*. On the other hand the term *taker* can be retrieved by consultation of the Antonym-relation which relates *to give* to *to take* followed by consultation of the Ability-relation, relating *to take-taker*. See figure 5.15 for a visualization. A special symmetric Transmission-relation is introduced in the following lines.

**Definition 5.19** TRANSMISSION RELATION
*Iff two object types correlate with each other such that one of them function as a 'Sender' and the other as a 'Receiver' then the following statement holds:*
Transmission(Sender, Receiver).

This statement can be defined as follows:
Antonym(VerbA, and VerbB)

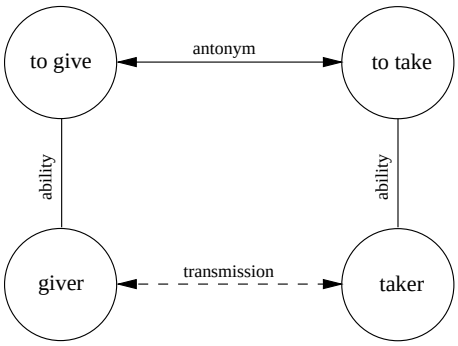


Figure 5.15: Derivation of the transmission relation.

$\wedge$ Ability(Sender, VerbA)
$\wedge$ Ability(Receiver, VerbB)
$\equiv$ Transmission(Sender, Receiver).

□ **End Definition**

**Example 5.14** *This example shows some applications of the transmission relation*

- Transmission(sender, receiver).
- Transmission(producer, consumer).
- Transmission(speaker, listener).
- Transmission(giver, taker).

□ **End Example**

**Axiom 5.13** INFERENCES INCLUDING TRANSMISSION RELATION
*If an object 'Instance' participates in a relation 'Relation' as receiver called 'Receiver' and there exists a transmission relation between the Agent and Instance then Instance is called an instance of 'Receiver':*

if ObjectPart (_Modality, _Tense, Relation, _Args, _ObjectType, Instance, rec)
and Ability(Agent, Relation)
and Transmission(Agent, Receiver)
then Instance(Instance, Receiver)

□ **End Axiom**

**Example 5.15** *This example shows some applications of inferences including the transmission relation.*

Relation(FACTUAL, PERF, give, [(ag, person('John')), (go, book('Bible')),
(rec, person(Mary))])
$\wedge$ Ability(giver, give)
$\wedge$ Transmission(giver, taker)
$\vdash$ taker('Mary').

□ **End Example**

### 5.3.11 Overview of Semantical relations

This section provides an overview of the semantic relations discussed in the previous section. The semantic relations are listed in table 5.4. Additional columns are provided in order to describe the domain of the relations and to illustrate the use of the relation by means of an example. Special domains are introduced that contain all possible object types, relations, adjectives and instances.

| | | | |
|---|---|---|---|
| MERONYM | | | |
| | ObjectComp | Objt× Objt | *tree - branch* |
| | SetElement | Objt× Objt | *forest -tree* |
| | QuantPort | Objt× Objt | *cake - portion* |
| | ObjectMat | Objt× Objt | *airplane - aluminum* |
| | Regplace | Objt× Objt | *Holland - Amsterdam* |
| (PARTIAL) SYNONYM | | | |
| | Synonym | Objt× Objt | *bike - bicycle* |
| | PartialSynonym | Objt× Adj× Adj | *bread - bad - stale* |
| ANTONYM | | | |
| | Contradictory | Adj× Adj | *complete - incomplete* |
| | Contrary | Adj× Adj | *big - small* |
| | Reversal | Adj× Adj | *constructive - destructive* |
| | Contrasting | Adj× Adj | *dry - damp* |
| SEMANTICAL CONSEQUENCE | | | |
| | Simultaneity | Rel× Rel | *stroll - walk* |
| | ProperInclusion | Rel× Rel | *snore - sleep* |
| | Presupposition | Rel× Rel | *load - unload* |
| FUNCTION | | | |
| | Instrument | Objt× Rel | *pen - write* |
| | Ability | Objt× Rel | *airplane - fly* |
| MISCELLANEOUS | | | |
| | Instance | Inst× Objt | *'John' - person* |
| | Hypernym | Objt× Objt | *producer - person* |
| | Gerund | Rel× Objt | *record - recording* |
| | Transmission | Objt× Objt | *giver - taker* |

Table 5.4: Overview of semantic relations.

## 5.4 Evaluation

In the previous chapters it has been shown that communication is actually about meanings instead of words or sentences. The words and sentences however are used to express the intended meanings. The structure of sentences correspond to the syntactical level of natural language, while meanings correspond to information at the semantical level. In this chapter it has been argued that intelligent communication between user and information system is complicated by the following points:

1. the information analyst has the difficult task to define precise but universal (i.e. comprehensible for all users of the information system) names of predicators that describe the meaning of the participation of an object type within a relation in an unambiguous way.

2. the information model, as a base of the information system, contains semantic ambiguities

3. the information system does not have disposal of a *(mental) lexicon* as humans do

4. the information system is not able to relate stored information to *common sense* knowledge, stored in the auxiliary lexicon.

The artificial intelligence approach to information systems, i.e. the combination of PSM$^{2+}$-models with semantic relations and inference rules, solves the above stated problems as follows:

1. The set of semantic roles recognized by FG, and used in PSM$^{2+}$-models, describe the meaning of participation of an object type within a relation unambiguously. Information can be retrieved from the information system without precise knowledge of the used names in the information model that were put in by the information analyst.

2. The $PSM^2$ model is extended with a.o. semantical roles, conform the semantic linguistic theory of Functional Grammar. The extension of $PSM^2$ models has resulted into the $PSM^{2+}$-modelling technique. This $PSM^{2+}$-modelling technique resolves the ambiguities of $PSM^2$ models.

3. The word-association test made the existence of semantic relations in human minds plausible. Semantic relations in the mind of NLUs are stored in a so called mental lexicon. This mental lexicon contains a set of analytical (i.e. domain independent) facts and rules describing common sense knowledge. The analytical inferences made by humans, are based on the relation between meanings of words and sentences. This chapter has shown, that formal definitions with similar semantic load can be provided. A set of these definitions has been used to simulate consultation of the mental lexicon by information systems.

4. The introduction of semantic roles, has led - in addition to resolving ambiguities - to a natural integration of common sense knowledge and stored information. The universal definitions stored in the lexicon can be combined with stored information to realize intelligent communication.

With respect to applicability of the approach sketched above, the following remarks can be made:

- Obviously $PSM^{2+}$-models can be applied in the same application domains as $PSM^2$- models.
- $PSM^{2+}$-models are not only valuable in combination with a lexicon and an inference engine. $PSM^{2+}$-models on their own, can be used as an extension of $PSM^2$-models to avoid semantic ambiguities. The following chapter will show that the increment of expressive power, realized by this simple extension, even facilitates verbalization, and thereby validation, of information models.
- Because the lexicon is constituted by *if-then* rules, it has the following desirable properties:

  **incrementability** New rules can be added to the knowledge base relatively independently of the other rules.

  **modifiability** Old rules can be changed relatively independently of other rules.

  **comprehensibility** *If-then* rules are not very difficult to write, read and understand.

  Together these properties restrict the complexity of the involved lexicon. The backward chaining strategy of the inference engine can even be used to explain each conclusion derived from the lexicon.

# Chapter 6

# Supporting Information Analyst and Domain Experts

## 6.1 Introduction

The previous chapters 4 and 5 have provided a discussion on semantics of natural language and its applications in the field of information modelling. While chapter 4 discussed a *way of thinking* (or philosophy) and chapter 5 a *way of modelling*, this chapter focuses on the *way of supporting* ([Wij91]). The *way of supporting* encompasses a set of tools. Tools are *supporting means* for performing and facilitating system development tasks ([Hof93]). The following sections provide a discussion on supporting the information modelling task of the information analyst (in cooperation with domain-experts) and describe how interaction with the inference engine can be realized to constitute the desired information model for the intended information system. This chapter is closed with a discussion on automatic verbalization of the information model.

Figure 6.1 shows the communication between information analyst, domain experts and the various supporting tools. The main task of the information analyst is to model the UoD by using the *information modelling* tool. He performs this task in cooperation with the domain experts, see information flow (a). With the information modelling tool, the information analyst, defines a set of predications (b) that together describe the UoD. The information modelling tool offers support in this task by suggesting appropriate terms to constitute a predication (c). The verbalization mechanism provides a natural language description of the information model (d). Domain experts are asked to validate this description (e). Domain experts and other users, use the forward chaining strategy of the inference engine to derive new information (f) or use the backward strategy to test a certain hypothesis (g).

## 6.2 PSM$^{2+}$ Meta model revisited

The task of the information analyst, in cooperation with domain-experts (or *informed users* ([FKW95])), is to model the relevant information of the UoD. Communication between information analyst and domain-experts is realized by means of natural language. A PSM$^{2+}$-model is based on a natural language description of the UoD, referred to as *initial specification*. Reconsider, the meta model of PSM$^{2+}$, as depicted in figure 5.5. This meta schema can be detailed and modified according to the linguistic theory of *Functional Grammar* (FG) terminology. Conform FG terminology, a PSM$^{2+}$-relation corresponds to the notion of *predication*. As we have seen in section 4.2 a predication is constructed by combining a single predicate with one, two or three terms. The PSM$^{2+}$ meta schema can therefore be adapted as follows:

- a predication consists of one, two or three participations
- a participation is a predication (recursive) or a term

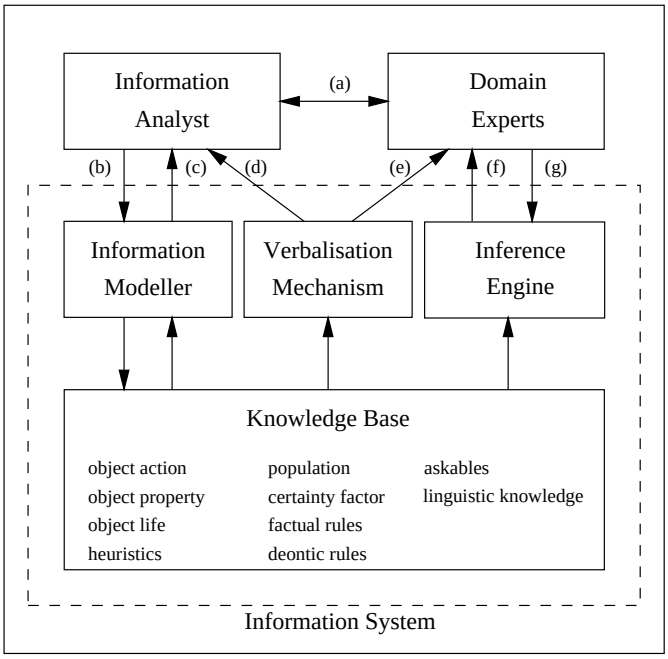


Figure 6.1: Various ways of supporting

- a term is a unique combination of an object type and an instance
- a predicate consists of one, two or three arguments
- an argument performs a certain (semantical) role in a predicate
- a constraint can be defined on an argument which restricts the possible object type of terms that fill the argument positions

Figure 6.2 shows the resulting PSM$^{2+}$ meta model. The relational database schema corresponding to this meta model is given in appendix C.

In this new PSM$^{2+}$ terminology constituting an information model corresponds to defining a set of predications that together describe the *Object Action Involvement* and *Object Property Model* perspective of the UoD. As stated before, defining a predication equals selecting a predicate from the lexicon and fill its arguments with appropriate terms. Figure 6.3 shows an example of the construction of the predication

record(ag, person('John')) (go, song('Kayleigh')).

which can be verbalized as:

*Person 'John' records the song 'Kayleigh'.*

## 6.3 Formulation by Navigation

Predicates are lexical items of the language, and thus are contained in the lexicon ([Dik89, page 68]). As stated in section 4.2, a predicate can be *basic* or *derived* . In the former case they are contained in the lexicon, in the latter they are produced through *predicate formation rules*. Here, our attention is restricted to basic predicates. Basic predicates have to be learned and memorized by NLUs in order to be used correctly. This does **not** imply that the predicate has no internal semantical structure. Indeed the meaning of most predicate is such that it can be analyzed in terms of combinations of the meanings of semantically simpler predicates. Figure figure 6.4 shows a dialog which is used by the information analyst to define predications on base of simple predicates. The dialog shows the definition of the following predication:

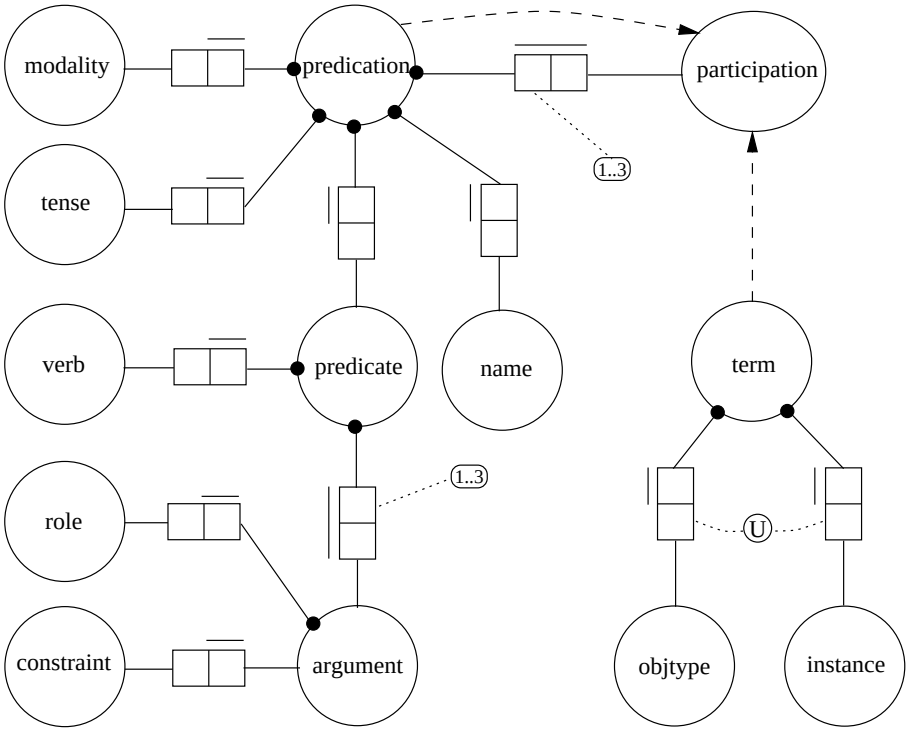


Figure 6.2: The adapted version of the PSM$^{2+}$meta schema

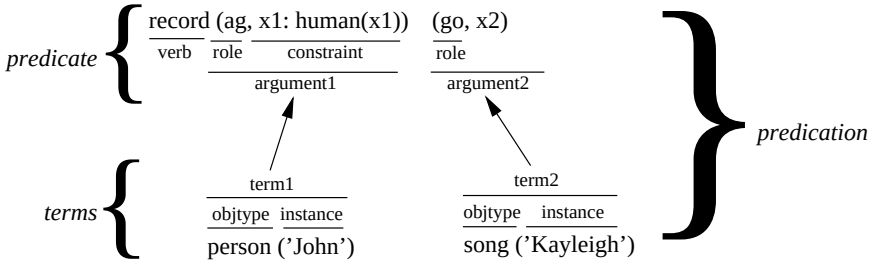


Figure 6.3: Construction of a PSM$^{2+}$-predication.

(1) Relation(FACTUAL, PERF, record, [(ag, person('John')), (go, song('Kayleigh'))]).

Definition of predications is supported with this information modelling tool. The information analyst selects appropriate predicates and terms from the lexicon, then defines modality and tense and a new predication has been defined. A predication is defined by means of the following information:

**Predication** The first frame in the dialogue below shows the name and the identifier of the predication being defined. In this case, the name of the predication is *recording1* and its id number is *#1*. Note that the name is chosen by the information analyst (if desired), while the identification is a generated number (i.e. counter).

**Predicate** The following frame enables selection of a predicate from a list of predicates contained in the lexicon. Selection of a certain predicate determines the role of the terms and restricts the possible object types of the terms. In this case the *agent*-argument is restricted to object type *person*, stating that the object filling this argument position should be a subtype of the object type *person*. Note that when the information analyst selects a predicate from the list, the appropriate semantic roles are retrieved from the lexicon and set automatically for each argument.

**Arguments** An argument position can be filled by a term or by a complete predication, in the case of complex predications. For each term the information analyst selects the type of participation. If the participation type is 'term', he is enabled to select an object type and an instance from a list of all possible object types and instances in the UoD. In this case, the first argument position is filled by term *person('John')*, the second argument by term *song('Kayleigh')*. Note that the object type of term *person('John')* is indeed a subtype of object type *person* and thus fulfills the constraint mentioned above.

**Modality / Tense** Modality and tense are defined by selecting one of the appropriate option buttons.

**Satellites** Optionally, additional like *time* and *location* (see section 4.2) can be defined.

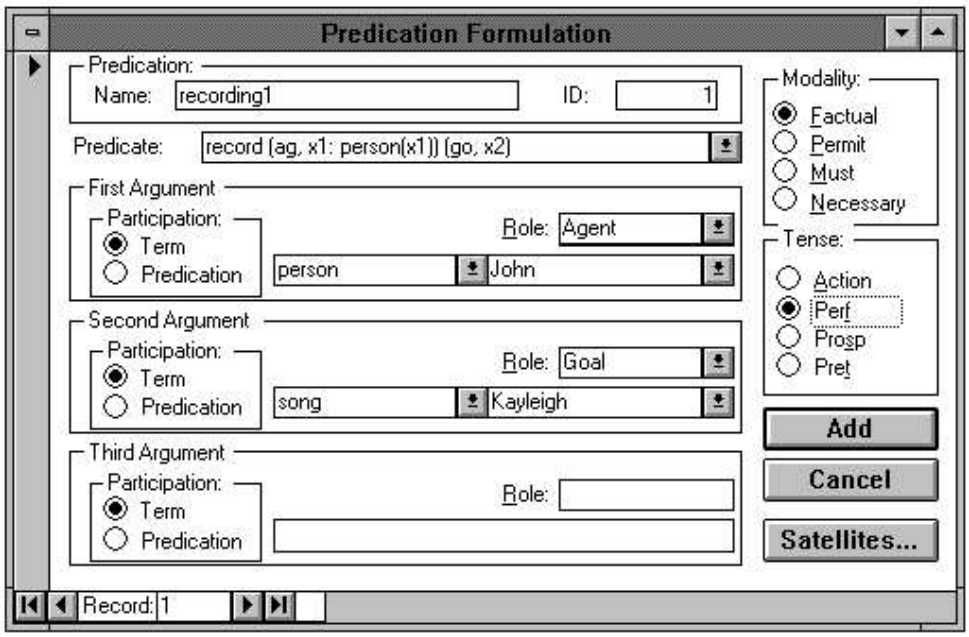


Figure 6.4: Predication formulation dialog of prototype Information Modeler.

The definition of the simple predication (1) is used to constitute complex predication (2), stating *The recording of (1), is produced by producer 'Mary'.*

(2) Relation(FACTUAL, PERF, produce, [(ag, person('Mary')),
(go, Relation(FACTUAL, PERF, record, [(ag, person('John')), (go, song('Kayleigh'))]))]).

Figure 6.5 shows the how the information analyst is enabled to define complex predication (2). He selects option button *predication* as participation type of the second argument, and selects predication (1) referred to as *recording1* by selecting it from a list of defined predications.

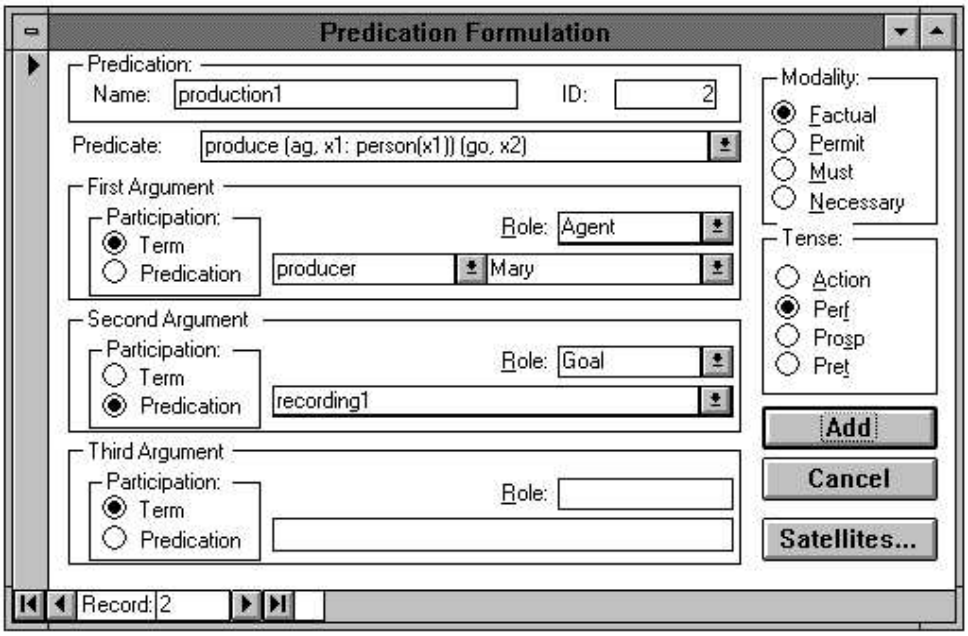


Figure 6.5: Defining a complex predication (2) based on (1).

This information modelling tool enables adequate semantics-oriented information modelling of the UoD. The information analyst defines a set of predications that together describe the UoD. Definition of a simple or complex predication is realized on a *step by step* principle. The information analyst interacts with the lexicon by selecting the appropriate predicates and terms. Constraints on arguments of the predicate avoid definition of semantical anomalous predications. The formulation of the UoD can be regarded as navigation through the lexicon and selection of predicates and terms to constitute a predication definition. This process can be called *Formulation by Navigation*.

## 6.4 Interaction with the Inference Engine

This section focuses on interaction with the inference engine. As stated in chapter 2 the inference engine uses two different reasoning strategies: *forward* and *backward chaining* . The forward chaining strategy uses the knowledge base to derive new information, which should be validated by domain experts. If this process produces improper information, the information analyst uses the backward chaining strategy which explains *how* the information was derived. The importance of such explanation facilities increases if the number of facts and rules in the knowledge base increases. The backward chaining strategy thus plays a crucial role in the validation of $PSM^{2+}$-models. Therefore optimizing the interaction with the backward chaining strategy is studied in more detail.

During backward inference the inference engine attempts to prove propositions stated by the user. If the engine succeeds, it is able to generate a proof tree which shows a sequence of applications of rules and facts, that together yield the user query. Below the proof tree of the proposition

Property(person, John, age, 24).

is provided, which has been generated by our $PSM^{2+}$-expert. For the sake of clarity, names have been added to inference rules. The engine uses these names to refer to the inference rules defined

in the previous chapter. The inference rule *objectparticipation_rule1* refers to definition 5.2, while inference rule *property_rule1* refers to definition 5.3.

```
Property(person, 'John', age, 24)
     was derived by property_rule1 from
     Relation(FACTUAL, PERF, have_prop, [(zero, person('John')), (go, age(24))])
          Which was found as a fact!
     and
     ObjectPart(FACTUAL, PERF, have_prop, [(zero, person('John')), (go, age(24))],
     person, 'John', zero)
          was derived by objectparticipation_rule1 from
          Relation(FACTUAL, PERF, have_prop, [(zero, person('John')),
          (go, age(24))])
               Which was found as a fact!
          and
          member((zero, person('John')), [(zero, person('John')), (go, age(24))])
               Which was found by execution!
     and
     ObjectPart(FACTUAL, PERF, have_prop, [(zero, person('John')), (go, age(24))],
     age, 24, go)
          was derived by objectparticipation_rule1 from
          Relation(FACTUAL, PERF, have_prop, [(zero, person('John')),
          (go, age(24))])
               Which was found as a fact!
          and
          member((go, age(24)), [(zero, person('John')), (go, age(24))])
               Which was found by execution!
```

During backward inference, the inference engine attempts to find facts to fulfill the antecedent of inference rules. Not all needed facts may be available to the inference engine, but some can be asked of the user. Such facts are called *askables*. An askable is a (simple) proposition which can be asked of the user. An example of an askable is:

(3) askable(instance(Instance, ObjectType), 'Instance' is a 'ObjectType').

The first argument in (3) is the fact that is attempted to be proven by the system , the second argument describes the external format of this facts when it is asked of the user. The external format defines *how* the questioning about the fact is presented to the user. With this simple extension the interaction provided below between user and system can be realized (user's responses are in boldface and PSM$^{2+}$-expert output is in italics). The following contents of the knowledge base is assumed: Hypernym(person, causal_agent), Hypernym(causal_agent, entity) and inference rules that define the transitive nature of Hypernym and Instance relations as defined in the previous chapter.

*Question, please:*
**'John' is a entity**.
*Is it true: 'John' is a causal_agent?*
**Why are you interested?**
*Well, I can use this to investigate ('John' is a entity) by applying hypernym_rule2*
*Is it true: 'John' is a causal_agent?*
**I don't know.**
*Is it true: 'John' is a person?*
**yes.**
*('John' is a entity) is true*
*Would you like to see how?*
**yes.**
*Instance('John', entity)*
*was derived by hypernym_rule2 from*

*Instance('John', person)*
*Which was found as a fact!*
*and*
*Hypernym(person, entity)*
*was derived by hypernym_rule1 from*
*Hypernym(person, causal_agent)*
*Which was found as a fact!*
*and*
*Hypernym(causal_agent, entity)*
*Which was found as a fact!*

## 6.5 Verbalization

In this section focus is on verbalization of the PSM$^{2+}$-model. Verbalization can be defined as a transformation of predications into linguistic expressions. A linguistic expression is a sentence of the object language. The verbalization is controlled by a set of rules, called the *expression rules*. Expression rules thus perform the transformation of an underlying clause structure into a corresponding linguistic expression. This situation has been be visualized in figure 4.1.

### 6.5.1 Introduction of Syntax

In this document our attention has been primary focussed on semantics instead of syntax. However, in the context of verbalization, syntax is necessary as well. In PSM$^{2+}$-terminology verbalization is the transformation of an expression at the semantic level (predication) to an expression at the syntactic level (sentence). The introduction of syntax brings an essential problem: it is at this point that *language specific* knowledge is involved in our discussion. While semantic (and pragmatic) functions are universally relevant to natural language, syntactic functions strongly depend on specific properties of a certain language. In this document the language chosen for verbalization is English (in a few cases even some comparisons with Dutch are made). As an example of the language specific nature of syntax, consider the linguistic notion of *definiteness*. For example in the English (or Dutch) language definiteness is often expressed in a definite article *the*. Take for example the linguistic expression *the house*. In Chomskian transformational generative grammar theory this may be coded as a grammar rule, denoted by the following (parse) tree.

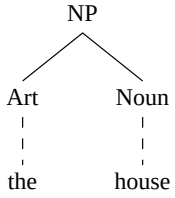


Figure 6.6: Definiteness expressed by a definite article

The syntactical (parse) tree of figure 6.6 shows that definiteness is coded by a definite article *the* preceding the noun *house*. The problem with this syntactic approach is that it will be very difficult to generalize about definiteness, both within and across languages, because in Danish definiteness is coded by a suffix instead of a preceding definite article. The Danish term *hus-* can be translated as *house*, while *the house* is represented by the Danish term *hus-et*. Obviously, the linguistic notion *definite article* is too concrete, i.e. too close to the actual linguistic expression - or even alternatively formulated too 'syntax-oriented' - to yield a typologically adequate account of definiteness ([Dik89]). Therefore a more abstract notion of definiteness is needed which can be universally applied, i.e. *language unspecific*. In FG, a definiteness operator *d[]* is introduced for this purpose. In (4) applications of the definite operator are provided. Naturally, proper nouns and pronouns are not affected by the definite operator[1].

[1] The symbol ∅ is introduced to denote the empty linguistic expression.

(4) d[house]= the house (*English*)
d[John]= ∅ John
d[hus-]= hus-et (*Danish*)

The general applicability of semantic functions, in contrast with that of syntactic functions, is put into words in the following quotation from [Dik89, page 25]:

> "...there is reason to believe that semantic and pragmatic functions are universally relevant to natural languages, although not all make the same distinctions within the general domain of these functions. Syntactic functions however are not universal in this sense."

Consider the semantic expression (or predication) in (5). Transforming this expression into a linguistic expression at the syntactical level, shows that verbalization constitutes a one to many relation between the semantic and syntactical level. All sentences in (6a)-(6c) describe the same underlying semantic expression from a different perspective.

(5) Relation(FACTUAL, PERF, give, [(ag, person('John')), (go, book('the Bible')), (rec, person('Mary'))]).
(6a) John gives the Bible to Mary.
(6b) The Bible is given to Mary by John.
(6c) Mary is given the Bible by John.

The syntactical expressions (6a)-(6c) describe (5) from different *points of view*. A point of view is sometimes also called a *perspective* or *vantage point* [Dik89, page 213]. Sentence (6a) describes (5) from the point of view of *John*, while (6b) and (6c) describe it from the perspective of *The Bible* or *Mary* respectively. Presentation (6a) is often known as the *active* form, (6b) as *passive* form and (6c) is sometimes called *secondary* or *derivative pseudo passive* form. Sentences (7a)-(7c) show that, with exception of the *pseudo-passive* form, equivalent expressions can be formulated in Dutch.

(7a) Jan geeft de Bijbel aan Mary. (*Dutch*)
(7b) De Bijbel wordt Mary gegeven door Jan. (*Dutch*).
∗(7c) Mary wordt gegeven de Bijbel door Jan. (*Dutch*)

The production of these different verbalizations of (6a)-(6c) and (7a)-(7c) can be explained by the application of the syntactical functions *Subject* and *Object* as introduced in section 4.2. Subject and Object functions are introduced in FG to capture the different perspectives (or points of view). More formally formulated, different perspectives are coded through different assignment of Subject and Object functions to the terms of a certain predication. In (6a) the Subject function is assigned to the term *John*, giving *John* the status of subject. In (6b) and (6c) the the syntactical function Object is assigned to the two terms *the Bible* and *Mary* respectively. Generalizing from this example: in (6a) the Subject status is assigned to the term playing the semantic role of *agent* (see (5)), in (6b) to the term playing the role of *goal* and in (6c) to the term representing the role of *recipient*. The subject represents the first term in all these three different verbalizations. The Subject function provides the primary vantage point. Application of the Object function, as a secondary vantage point, is illustrated in (8). The active construction of (6a), repeated in (8a), is modified in a subtle way in (8b).

(8a) John gives the Bible to Mary.
(8b) John gives Mary the Bible.

In both (8a) and (8b) the subject status is distributed to the *agent*-term *John*. So they share the same primary vantage point. Their secondary vantage point however differs: in (8a) object status is assigned to *goal*-term *the Bible*, and in (8b) it is assigned to the *recipient*-term *Mary*. The different verbalizations can now be characterized by their distribution of syntactical functions Subject and Object to semantic functions Agent, Goal and Recipient. Table 6.1 provides an overview of verbalizations of predication (5).

| Distribution | | Verbalization | |
|---|---|---|---|
| Subj | Obj | Ref. | |
| Ag | Go | (6a) | *John gives the Bible to Mary* |
| Go | | (6b) | *The Bible is given to Mary by John.* |
| Rec | | (6c) | *Mary is given the Bible by John.* |
| Ag | Rec | (8b) | *John gives Mary the Bible.* |

Table 6.1: Different verbalizations caused by distribution of Subject and Object

### 6.5.2 A Verbalization Algorithm

verbalization!algorithm As stated before, a predication consists of a *modality*, *tense*, *predicate* and a set of *terms* filling the predicate frame. Verbalizing a simple predication thus implies globally taking the steps mentioned below. Note that in passive verbalizations, step two and five can be omitted because the notion of object is not relevant in the scope of passive verbalization.

1. Select a term from the set of terms, assign it the status of subject and and remove it from the set.
2. Select a term from the set of terms remaining from the previous step, assign it the status of object and remove it from the set.
3. Verbalize the subject.
4. Verbalize the predicate.
5. Verbalize the object.
6. Verbalize each permutation of remaining terms.

This stepwise procedure is far too abstract, and needs to be studied in more detail. The following questions need to be answered in order to tackle the problem of predication-verbalization.

1. Which terms can obtain the status of subject?
2. Which terms can obtain the status of object?
3. How to verbalize terms?
4. How to verbalize the predicate?
5. How to verbalize simple predications?
6. How to verbalize complex predications?

These questions will be answered in the following subsections.

#### 6.5.2.1 Potential subject status

The first question to be answered deals with potential application of the Subject function. This discussion is restricted to a set of three semantic roles and two semantic satellites: *agent*, *goal* , *recipient*, *beneficery* and *instrument* (see section 4.2). Sentences (6a)-(6c) have shown that subject status can be assigned to terms playing the role of *agent*, *goal* and *recipient* in a certain underlying predication. That these are the only terms is illustrated in (9) and (10).

(9a) [John]$_{AgSubj}$ bought some flowers for [Mary]$_{Ben}$.
*(9b) [Mary]$_{BenSubj}$ has been bought some flowers (for) [by John]$_{Ag}$.
(10a) [John]$_{AgSubj}$ broke the window [with a brick]$_{Instr}$.
*(10b) [The brick]$_{InstrSubj}$ broke the window [by John]$_{Ag}$.

In (9) the term *for Mary* functions as a beneficery satellite. Sentence (9b) shows that in English such terms cannot obtain the status of subject. Sentence (10b) shows that the *instrument* term *with a brick* can neither be used a subject. Therefore *agent*, *goal* and *recipient* terms are the only terms which can obtain subject-status. With exception of the *recipient* term, the same holds for Dutch as well. Table 6.2 , adapted from [Dik89], lists possible subject assignments in various languages of the world.

| | Semantical notion | | | | |
|---|---|---|---|---|---|
| Language | Ag | Go | Rec | Ben | Instr |
| Dutch | * | * | | | |
| Melanau | * | * | | | |
| English | * | * | * | | |
| Japanese | * | * | * | | |
| Sanskrit | * | * | (*) | | |
| Sundanese | * | * | * | * | |
| Malagasy | * | * | * | * | * |

Table 6.2: Potential subject assignment in various languages of the world

#### 6.5.2.2 Potential object status

The second question to be answered implies a discussion on potential object assignment. Likewise subject assignment is not restricted to agents and goals, object assignment has a broader range than goal (8a) and recipient (8b) terms. Even in (a highly structured language like) English there is a marginal possibility of assigning object status to *beneficery* terms. Consider (11a) in which *Peter* participates as a beneficery term in a *buy*-predicate. Sentence (11b) shows that this beneficery term can indeed obtain the status of object.

(11a) [John]$_{AgSubj}$ bought [the book]$_{GoObj}$ [for Peter]$_{Ben}$.
(11b) [John]$_{AgSubj}$ bought [Peter]$_{BenObj}$ [the book]$_{Go}$.

An overview of potential object assignments in various languages is provided in table 6.3.

| | Semantical notion | | | | |
|---|---|---|---|---|---|
| Language | Ag | Go | Rec | Ben | Instr |
| Dutch | | * | * | | |
| English | | * | * | * | |
| Bahasa Ind. | | * | * | * | |
| Sundanese | | * | * | * | |
| Chimwi:ni | | * | * | * | * |
| Swahili | | * | * | * | * |

Table 6.3: Potential object assignment in various languages of the world

#### 6.5.2.3 Verbalization of terms

Now we are able to select the appropriate subject and object terms from the set of terms, the third question is about verbalization of these terms. Again reconsider the same *give*-predicate repeated below:

(12a) [John]$_{AgSubj}$ gives [the Bible]$_{GoObj}$ [**to** Mary]$_{Rec}$.
(12b) [John]$_{AgSubj}$ gives [Mary]$_{RecObj}$ [the Bible]$_{Go}$.

In sentence (12a) the *recipient*-term *Mary* is preceded by the linguistic expression *to*. In (12b) the recipient *Mary* is **not** preceded by the term *to*. This behavior has a general nature and can be described by the distribution of the Subject and Object functions. In an active sentence like (12a) verbalization of a recipient term is preceded by the term *to*. In (12b) the recipient term *Mary* has obtained object status and therefore the preceding term *to* is no longer needed. The fact that this reasoning has a more general nature is illustrated below. Verbalization of an beneficery term in a

sentence like (13a) is preceded by the term *for*, however in (13b) it is illustrated that this term is omitted if a beneficery term obtains the status of object.

(13a) [John]$_{AgSubj}$ bought [some flowers]$_{GoObj}$ [**for** Mary]$_{Rec}$.
(13b) [John]$_{AgSubj}$ bought [Mary]$_{RecObj}$ [some flowers]$_{Go}$.

In passive constructions verbalization of an agent term is realized by a preceding term *by*. See sentence (14) and (15).

(14) [The Bible]$_{GoSubj}$ is given [to Mary]$_{Rec}$ [**by** John]$_{Ag}$.
(15) [Mary]$_{RecSubj}$ is given [the Bible]$_{Go}$ [**by** John]$_{Ag}$.

As illustrated in sentence (6b) and (6c), the term *by* can be omitted when the *agent* term obtains the status of subject. Verbalizations of terms can be summarized as listed in table 6.4. Note that terms which have **not** been assigned the status of subject or object will usually be expressed through a characteristic case form or adposition, as can be observed in the fourth column. This additional expression is "neutralized" by application of the Subject or Object function. This is denoted by the empty linguistic expression symbol ∅ in the second and third column.

| | Subject and Object distribution | | |
|---|---|---|---|
| Sem. notion | Subject | Object | neither |
| Agent | ∅ <Term> | N.A. | by <Term> |
| Goal | ∅ <Term> | ∅ <Term> | ∅ <Term> |
| Recipient | ∅ <Term> | ∅ <Term> | to <Term> |
| Beneficery | N.A. | ∅ <Term> | for <Term> |
| Instrument | N.A. | N.A. | with <Term> |

Table 6.4: Term-verbalization inherent to distribution of subject and object assignment.

Table 6.4 shows all possible combinations of semantic notions (agent, goal, recipient, beneficery, instrument) with subject and object distribution (subject, object, neither). Four of these fifteen combinations (=5 x 3) are illegal, resulting into eleven possible verbalizations. Besides semantic notion and subject/object distribution, term verbalization depends on two other parameters: *definiteness* and *number*. Definiteness is available in two gradations: *definite (d)* or *indefinite (i)*. The other parameter *number* has also two possible values: *singular (sg)* and *plural (pl)*. The behavior of these parameters is illustrated below.

**Example 6.1** Application of Definiteness and Number parameters
*This example shows the effect of the parameters Definiteness and Number on the verbalization of a term. Note that plural form verbalization is simplified by suffixing the term with '-s'.*

| d/i | sg/pl | Verbalization |
|---|---|---|
| d | sg | the <Term>) |
| d | pl | the <Term>-s) |
| i | sg | a <Term>) |
| i | pl | ∅ <Term>-s |

Table 6.5: Definiteness and Number properties affect term verbalization.

□ **End Example**

Application of these two additional parameters results into four new combinations. These four combinations can be combined with the previous eleven combinations, giving 44 possible term verbalizations determined by four parameters. A special relation Verbalize is introduced to capture verbalization of terms:

**Definition 6.1** Verbalization of terms
*Verbalization of a term depends on the following parameters.*

- semantic notion (agent | goal | recipient | beneficery | instrument)

- subject/object distribution (subject | object | neither)
- definiteness (definite | indefinite)
- number (singular | plural)

*Iff 'VTerm' is the verbalization of a term 'Term' with semantic notion 'S', subject/object assignment 'A', definiteness 'D' and number 'N' then the following statement holds:*
Verbalize((S, A, D, N), Term, VTerm)

□ **End Definition**

**Example 6.2** VERBALIZATION OF TERMS
*This example shows some applications of the definition of term verbalization.*

- Verbalize((ag, subject, i, sg), 'person', 'A person').
- Verbalize((ag, neither, d, sg), 'dog', 'by the dog').
- Verbalize((rec, neither, d, pl), 'student', 'to the students').

□ **End Example**

The verbalization of the plural form of a term is not always realized by addition of a simple suffix '-s'. The plural form of e.g. *man* and *knife* is *men* and *knives* respectively instead of *mans* and *knifes*. Plural forms of these specific cases can therefore **not** not be derived by inference but need to be recorded in the lexicon explicitly. The following definition yields the correct plural form. Application of specific cases takes priority over application of the general inference rule. This principle of lexical priority can be formulated as follows ([Dik89, page 294]):

> "Whenever a rule is encountered of the general form M[x]=y first check the lexicon whether the y form of x is stored there ready-made. If so, select this form; otherwise apply the rule."

**Definition 6.2** MORPHOLOGICAL MODIFICATIONS OF TERMS
*A finite number of specific cases of plural forms are stored explicitly in the lexicon, while in all other cases the plural form can be derived by application of a general rule that simply adds a suffix '-s'. See table 6.6.*

| Term | Plural Form |
|---|---|
| abacus | abaci |
| ability | abilities |
| ... | ... |
| zany | zanies |
| zucchini | zucchini |
| *<Term>* | *<Term>-s* |

Table 6.6: Plural form of terms

□ **End Definition**

#### 6.5.2.4 Verbalization of predicates

The fourth question to be answered deals with verbalization of the predicate in a predication. Consider e.g. (16)-() that show different verbalizations of the predicate *walk*.

(16a) I walk.
(16b) You walk.
(16c) He/she/it walk-s.
(16d) We walk.
(16e) You walk.
(16f) They walk.

In a language like English (or Dutch) the form of the predicate is only sensitive to subject assignment, i.e. the appropriate form of the predicate is completely determined by the type of subject. Certain languages however are not only sensitive to subject assignment but to object assignment as well. An example of such a language is Bahasa Indonesia. The morphological variation of the predicate in (17b) in comparison to (17a) is caused by assigning object status to *recipient*-term *Ali*.

(17a) saya mem-bawa surat itu kepada Ali.
"I bring letter this to Ali"
$[\text{I}]_{AgSubj}$ bring $[\text{this letter}]_{GoObj}$ $[\text{to Ali}]_{Rec}$.
(17b) saya mem-bawa-**kan** Ali surat itu.
"I bring Ali letter this"
$[\text{I}]_{AgSubj}$ bring $[\text{Ali}]_{RecObj}$ $[\text{this letter}]_{Go}$.

Similar object-sensitive verb marking is found in Bantu languages. In a number of these languages, not only *recipient* and *beneficery* but also *instrument* and sometimes even *location*-terms can obtain object function [Dik89, page 225]. Again it is stressed that attention is focussed on English and therefore morphological changes of predicates, caused by object assignment, can be ignored. Verbalization of predicates is affected by four parameters:

- subject property *number*
- subject property *semantic notion*.
- predication property *modality*
- predication property *tense*.

Sentence (16c) shows that the morphological change of predicate *walk*, is affected by the subject in third person singular form. In the scope of $\text{PSM}^{2+}$-models only third person form of subjects is relevant, because terms always represent an object type or instance (of an object type). The distinction between singular and plural third person can be made by the *number* parameter. Verbalization of predicate *walk* in the context of third person singular (16c) is defined in (18a), while a third person plural form subject context is provided in (18b).

(18a) Verbalize((sg), walk, walks).
(18b) Verbalize((pl), walk, walk).

Obviously, not all these Verbalize relations as mentioned in (18) are explicitly stored in the the lexicon, most of them can be formed by inference (see definition 6.3).

The following parameter affecting predicate verbalization is *semantic notion*. The semantic notion determines whether the predicate is represented in an *active* or *passive* form. If the semantic notion of the subject belongs to the set of first argument position roles (i.e. the set $\text{A}^1 \equiv \{$*agent, positioner, force, processed, zero*$\}$), then the predicate must be expressed in active form (6a). Otherwise the predicate must be expressed in passive form ((6b), (6c)). The verbalization relation Verbalize can be extended with an additional argument specifying the semantic notion played the subject. While sentences in (19) provide example involving the predicate *to walk*, sentence (20) shows the necessary morphological changes of predicate *to hit*. The predicate *to walk* is used in the sense of *John walks* and the predicate *hit* has the meaning as in *John hits the ball*.

(19a) $\text{Sem} \in \text{A}^1 \vdash$ Verbalize((sg, Sem)), 'walk', 'walks').
(19b) $\text{Sem} \in \text{A}^1 \vdash$ Verbalize((pl, Sem)), 'walk', 'walk').
(20a) $\text{Sem} \in \text{A}^1 \vdash$ Verbalize((sg, Sem)), 'hit', 'hits').
(20b) $\text{Sem} \in \text{A}^1 \vdash$ Verbalize((pl, Sem)), 'hit', 'hit').
(20c) $\text{Sem} \notin \text{A}^1 \vdash$ Verbalize((sg, Sem)), 'hit', 'is hit').
(20d) $\text{Sem} \notin \text{A}^1 \vdash$ Verbalize((pl, Sem)), 'hit', 'are hit').

Verbalization of predicates is not only determined by subject properties, but also by predication properties containing the predicate: *tense* and *modality*. As stated in section 4.4, verbalization of the predicate *paint* is influenced by predication property *tense* as follows.

(21a) John paint-s the portrait. (*ACTION*)
(21b) John has painted the portrait. (*PERF*)
(21c) John painted the portrait. (*PAST*)
(21d) John will paint the portrait. (*PROSP*)

The perfect tense in (21b) is realized by the introduction of the appropriate form of the predicate *have*, while in (21d) future tense is verbalized by a form of *to will*. Predicate verbalizations as shown in (21a)-(21d) can be formally defined as given in (22a)-(22d) respectively. The Verbalize-relation is extended with an additional argument containing the *tense* of the predication.

(22a) Verbalize(((sg, ag), (ACTION)), 'paint', 'paint-s')
(22b) Verbalize(((sg, ag), (PERF)), 'paint', 'has paint-ed')
(22c) Verbalize(((sg, ag), (PAST)), 'paint', 'paint-ed')
(22d) Verbalize(((sg, ag), (PROSP)), 'paint', 'will paint')

Again we stress that these Verbalize-relations can be derived by inference and therefore need **not** to be stored in the lexicon explicitly.

The last parameter, which affects predicate verbalization, is predication property *modality*. As stated in section 4.4, verbalization of the predicate *paint* is influenced by predication property *modality* as follows.

(23a) John paint-s the portrait. (*FACTUAL*)
(23b) John is permitted to paint the portrait. (*PERMIT*)
(23c) John has to paint the portrait. (*MUST*)
(23d) John is obliged to paint the portrait. (*NEC*)

Predicate verbalizations shown in (23a)-(23d) can be formally defined as given in (24a)-(24d) respectively. The Verbalize-relation is extended again with an additional argument containing the *modality* of the predication.

(24a) Verbalize(((sg, ag), (ACTION, FACTUAL)), 'paint', 'paints')
(24b) Verbalize(((sg, ag), (ACTION, PERMIT)), 'paint', 'is permitted to paint')
(24c) Verbalize(((sg, ag), (ACTION, MUST)), 'paint', 'has to paint')
(24d) Verbalize(((sg, ag), (ACTION, NEC)), 'paint', 'is obliged to paint')

Verbalization of predicates can now be defined in terms of *number*, *semantic notion*, *modality* and *tense*. Table 6.7 and 6.8 show the possible predicate verbalizations of the singular subject type *Monet* involved in a predication with modality FACTUAL or PERMIT, respectively. Predicate verbalizations in the scope of modality MUST and NEC are provided in appendix E. All these tables are based on the following underlying predication:

Relation(Modality, Tense, paint, [(ag, painter('Monet')), (go, painting('Le parc Monceau'))]).

Linguistic knowledge of predicates is stored in the lexicon and can be retrieved to realize the appropriate morphological modifications. Definition of these modifications are stored as described by the the following definition.

**Definition 6.3** MORPHOLOGICAL MODIFICATIONS OF PREDICATES
*In English the list of about 200 irregular verbs is stored explicitly in the lexicon. All other cases can be derived by application of a general rule. All empty cells in the table, representing a part of the lexicon, can be filled by application of this general rule. See table 6.9*

□ **End Definition**

#### 6.5.2.5 Verbalization of simple predications

The verbalization of a complete verbalization can be defined with the help of the previously introduced verbalization relations of terms and predicates. Below the necessary PROLOG main clauses are provided that realize active verbalization of predications.

| Modality | SemN. | Tense | Verbalization / General format |
|---|---|---|---|
| FACTUAL | ag | ACTION | 'Monet' paint-s 'Le Parc Monceau' |
| | | | Present(sg, <pred>) |
| FACTUAL | ag | PERF | 'Monet' has paint-ed 'Le Parc Monceau' |
| | | | Present(sg, 'have'), PaP(<pred>) |
| FACTUAL | ag | PAST | 'Monet' paint-ed 'Le Parc Monceau' |
| | | | PAST(sg, <pred>) |
| FACTUAL | ag | PROSP | 'Monet' will paint 'Le Parc Monceau' |
| | | | 'will', <pred> |
| FACTUAL | go | ACTION | 'Le Parc Monceau' is paint-ed by 'Monet' |
| | | | Present(sg, 'be'), PaP(<pred>) |
| FACTUAL | go | PERF | 'Le Parc Monceau' has been paint-ed by 'Monet' |
| | | | Present(sg, 'have'), PaP('be'), PaP(<pred>) |
| FACTUAL | go | PAST | 'Le Parc Monceau' was paint-ed by 'Monet' |
| | | | PAST(sg, 'be'), PaP(<pred>) |
| FACTUAL | go | PROSP | 'Le Parc Monceau' will be paint-ed by 'Monet' |
| | | | 'will', 'be', PaP(<pred>) |

Table 6.7: Possible predicate verbalizations for modality FACTUAL

| Modality | SemN. | Tense | Verbalization / General format |
|---|---|---|---|
| PERMIT | ag | ACTION | 'Monet' is permitted to paint 'Le Parc Monceau' |
| | | | Present(sg, 'be'), 'permitted', 'to', <pred> |
| PERMIT | ag | PERF | 'Monet' has been permitted to paint 'Le Parc Monceau' |
| | | | Present(sg, 'have'), PaP('be'), 'permitted', 'to', <pred> |
| PERMIT | ag | PAST | 'Monet' was permitted to paint 'Le Parc Monceau' |
| | | | PAST(sg, 'be'), 'permitted,' 'to', <pred> |
| PERMIT | ag | PROSP | 'Monet' will be permitted to paint 'Le Parc Monceau' |
| | | | 'will', 'be', 'permitted', 'to', <pred> |
| PERMIT | go | ACTION | 'Le Parc Monceau' is permitted to be paint-ed by 'Monet' |
| | | | Present(sg, 'be'), 'permitted', 'to', 'be', PaP(<pred>) |
| PERMIT | go | PERF | 'Le Parc Monceau' has been permitted to be paint-ed by 'Monet' |
| | | | Present(sg, 'have'), PaP('be'), 'permitted', 'to', 'be', PaP(<pred>) |
| PERMIT | go | PAST | 'Le Parc Monceau' was permitted to be paint-ed by 'Monet' |
| | | | PAST(sg, 'be'), 'permitted', 'to', 'be', PaP(<pred>) |
| PERMIT | go | PROSP | 'Le Parc Monceau' will be permitted to be paint-ed by 'Monet' |
| | | | 'will', 'be', 'permitted', 'to', 'be', PaP(<pred>) |

Table 6.8: Possible predicate verbalizations for modality PERMIT

| Verb | Tense | *Number* sg | pl |
|---|---|---|---|
| abide | present | | |
| | past | abode | abode |
| | pap | abode | |
| ... | ... | ... | ... |
| be | present | is | are |
| | past | was | were |
| | pap | been | |
| ... | ... | ... | ... |
| have | present | has | have |
| | past | had | had |
| | pap | had | |
| ... | ... | ... | ... |
| write | present | | |
| | past | wrote | wrote |
| | pap | written | |
| *<pred>* | present | *<pred>-s* | *<pred>* |
| | past | *<pred>-ed* | *<pred>-ed* |
| | pap | *<pred>-ed* | |

Table 6.9: Morphological modifications of predicates

```
%
% active verbalization of predication
%

verbalize(relation(Mod, Tense, Predicate, Args), VPredication):-
   % Select Subject
        selectsubject(Args, Subject, Rest1),
        Subject = (SemNotion1, Term1),
   % Check whether dealing with active construction
        member(SemNotion1, [ag, pos, for, pro, zero]),
   % Select Object.
        selectobject(Rest1, Object, Rest2),
        Object = (SemNotion2, Term2),
   % Verbalize subject
        verbalize((SemNotion1, subject, d, sg), Term1, VTerm1),
   % Verbalize predicate
        verbalize(((sg, SemNotion1),(Mod, Tense)), Predicate, VPredicate),
   % Verbalize object
        verbalize((SemNotion2, object, d, sg), Term2, VTerm2),
   % Verbalize each permutation of remaining terms
        permutation(Rest2, PRest2),
        verbalize(PRest2, VRest2),
        VPredication = [VTerm1, VPredicate, VTerm2, VRest2].
```

**Example 6.3** ACTIVE VERBALIZATION OF PREDICATIONS
*This example shows the active verbalizations produced by the algorithm presented above. Consider a UoD as depicted in figure 6.7, and defined as follows:*

Verbalize(Relation(FACTUAL, ACTION, give, [(ag, person('John')),
(go, book('Bible')), (rec, person('Mary'))]), VPredication).

Active verbalization of this PSM$^{2+}$-model yields the following results:

```
VPredication =
```

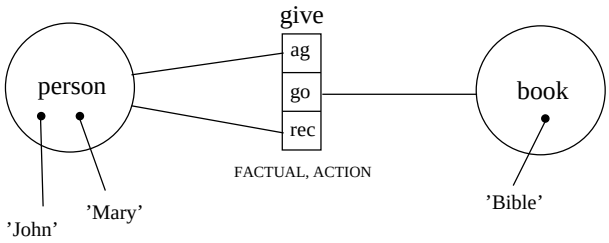


Figure 6.7: PSM$^{2+}$-model to be verbalized

```
[the, person, John, give - s, the, book, Bible, to, the, person, Mary];
[the, person, John, give - s, the, person, Mary, the, book, Bible]
```

□ **End Example**

The notion of object is not relevant in the scope of non-active constructions. Therefore the algorithm above can be simplified as follows in order to realize non-active verbalizations.

```
%
% passive verbalization of predication
%
verbalize(relation(Mod, Tense, Predicate, Args), VPredication):-
   % Select Subject
        selectsubject(Args, Subject, Rest),
        Subject = (SemNotion1, Term1),
   % Check whether dealing with non-active construction
        not member(SemNotion1, [ag, pos, for, pro, zero]),
   % Verbalize subject
        verbalize((SemNotion1, subject, d, sg), Term1, VTerm1),
   % Verbalize predicate
        verbalize(((sg, SemNotion1),(Mod, Tense)), Predicate, VPredicate),
   % Verbalize each permutation of remaining terms
        permutation(Rest, PRest),
        verbalize(PRest, VRest),
        VPredication = [VTerm1, VPredicate, VRest].
```

**Example 6.4** PASSIVE VERBALIZATION OF PREDICATIONS
*This example shows the passive verbalizations produced by the algorithm presented above. Again the UoD of figure 6.7 is assumed, as defined below:*

Verbalize(Relation(FACTUAL, ACTION, give, [(ag, person('John')),
(go, book('Bible')), (rec, person('Mary'))]), VPredication).

yields the following results:

```
VPredication =
[the, book, Bible, is, given, by, the, person, John, to, the, person, Mary] ;
[the, book, Bible, is, given, to, the, person, Mary, by, the, person, John];
[the, person, Mary, is, given, by, the, person, John, the, book, Bible];
[the, person, Mary, is, given, the, book, Bible, by, the, person, John]
```

□ **End Example**

#### 6.5.2.6 Verbalization of complex predications

The last question to be answered, studies the verbalization of complex predications. Remind that a complex predication is a predication containing a complete predication in one or more of its

arguments. An example of a complex predication was given in (2). Active verbalization of complex predication (2) can be realized as follows:

(23) The producer 'Mary' produces the recording "The person 'John' records the song 'Kayleigh'".

Sentence (23) is a direct result of treating an instance of a fact type *to record* as an instance of the objectified fact type *recording*. Actually, the previous chapter has shown that the *gerund*-relation can be used to objectify each fact type. Therefore verbalization of complex relations introduces the need of gerund relations. Although (23) is a correct verbalization of the underlying predication (2), the verbalization can be modified slightly to result into a more "natural" sentence. Consider the following verbalization as given in (24) and suggested in [FKW95].

(24) The producer 'Mary' produces the recording of the song 'Kayleigh' by person 'John'.

This construction will be studied in more detail. The construction *...the recording of the song 'Kayleigh' by person 'John'* does raise the following questions.

- Does a linguistic foundation exist for such a construction?
- And if so, how can it be described conform FG terminology?

The first question can indeed be answered positively. The mentioned construction is recognized in FG as *predicate nominalization* [Dik89, page 164]. To answer the second question, more examples of predicate normalization have to be studied. These examples are used to investigate constituent ordering in the scope of nominalization. Again focus is restricted to terms with semantic notions belonging to the set *agent, goal, recipient, beneficery* and *instrument*.

(25a) The walking [of 'John']$_{Ag}$.
(25b) The kissing [of 'Mary']$_{Go}$ by ['John']$_{Ag}$.
(25c1) The gift [of the 'Bible']$_{Go}$ [to 'Mary']$_{Rec}$ [by 'John']$_{Ag}$.
(25c2) The gift [of the 'Bible']$_{Go}$ [by 'John']$_{Ag}$ [to 'Mary']$_{Rec}$.
(25d1) The buy [of some flowers]$_{Go}$ [for 'Mary']$_{Ben}$ [by 'John']$_{Ag}$.
(25d2) The buy [of some flowers]$_{Go}$ [by 'John']$_{Ag}$ [for 'Mary']$_{Ben}$.
(25e1) The breaking [of the window]$_{Go}$ [with a brick]$_{Instr}$ [by 'John']$_{Ag}$.
(25e2) The breaking [of the window]$_{Go}$ [by 'John']$_{Ag}$ [with a brick]$_{Instr}$.

In (25a) *The walking* is regarded as a kind of possession of *John* ([Dik89, page 366]). This kind of possession is expressed by the term *of*. In this case *John* is the agent of the underlying predication. The other cases (25b)-(25e) show that if more arguments are available, as in two or three place predicate frames, the nominalized predicate is regarded as being 'possessed' by the second argument, represented by the semantic function *goal*. The verbalizations of (25) correspond to the patterns of (26), respectively.

(26a) <nom. predicate> of <agent>.
(26b) <nom. predicate> of <goal> <agent>.
(26c1) <nom. predicate> of <goal> <recipient> <agent>.
(26c2) <nom. predicate> of <goal> <agent> <recipient>.
(26d1) <nom. predicate> of <goal> <beneficery> <agent>.
(26d2) <nom. predicate> of <goal> <agent> <beneficery>.
(26e1) <nom. predicate> of <goal> <instrument> <agent>.
(26e2) <nom. predicate> of <goal> <agent> <instrument>.

These verbalization patterns can be summarized as follows: firstly find the gerund of the predication, secondly verbalize the *goal* of the predication (if available), and finally verbalize each permutation of remaining terms. The following clauses perform the verbalizations of (25).

```
%
% Verbalize complex predication without goal-term
%
verbalize((SemNotion, Distr, Def, Num), relation(Mod, Ten, Pred, Args), VTerm):-
```

```
   % Select Agent-term
        selectterm(Args, (ag, Agent), Rest), !,
   % Retrieve appropriate prefixes from table 6.4 and table 6.5
        table64(SemNotion, Distr, Prefix1),
        table65(Def, Num, Prefix2),
   % Get gerund
        fact : gerund(Pred, Gerund), !,
   % Verbalize predication
        verbalize((ag, subject, d, sg), Agent, VAgent),
        VTerm = [Prefix1, Prefix2, Gerund, of, VAgent].

%
% Verbalize complex predication containing goal-term
%
verbalize((SemNotion, Distr, Def, Num), relation(Mod, Ten, Pred, Args), VTerm):-
   % Select Goal-term
        selectterm(Args, (go, Goalterm), Rest),!,
   % Retrieve appropriate prefixes from table 6.4 and table 6.5
        table64(SemNotion, Distr, Prefix1),
        table65(Def, Num, Prefix2),
   %Get gerund
        fact : gerund(Pred, Gerund), !,
   % Verbalize goal term.
        verbalize((go, subject, d, sg), Goalterm, VGoal),
   % Verbalize each permutation of remaining terms
        permutation(Rest, PRest),
        verbalize(PRest, VRest),
        VTerm = [Prefix1, Prefix2, Gerund, of, VGoal, VRest].
```

**Example 6.5** VERBALIZATION OF COMPLEX PREDICATIONS
*This example shows some applications of the verbalization algorithm presented above.*

- Verbalize(Relation(FACTUAL, ACTION, see, [(ag, person('Mary')),
  (go, Relation(FACTUAL, ACTION, walk, [(ag, person('John'))]))]), Vterm).

  yields the results:

  ```
  VTerm =
  [the, person, Mary, see - s, the, walk - ing, of, the, person, John];
  [the, walk - ing, of, the, person, John, is, seen, by, the, person, Mary]
  ```

- Verbalize(Relation(FACTUAL, ACTION, appreciate, [(ag, person('Peter')),
  (go, Relation(FACTUAL, ACTION, give, [(ag, person('John')),
  (go, book('Bible')), (rec, person('Mary', person))])), VTerm).

  yields the following results:

  ```
  Vterm =
  [the, person, Peter, appreciate - s, the, gift, of, the, book, Bible,
  by, the, person, John, to, the, person, Mary];
  [the, person, Peter, appreciate - s, the, gift, of, the, book, Bible,
  to, the, person, Mary, by, the, person, John];
  [the, gift, of, the, book, Bible, by, the, person, John, to, the, person,
  Mary, is, appreciate - ed, by, the, person, Peter];
  [the, gift, of, the, book, Bible, to, the, person, Mary, by, the, person,
  John, is, appreciate - ed, by, the, person, Peter]
  ```

□ **End Example**

Although the verbalization tends to become more and more complex, the above suggested algorithm can be applied recursively again and again, as illustrated in the following example.

**Example 6.6** VERBALIZATION OF EVEN MORE NESTED PREDICATIONS
*This example shows the verbalization of a even more nested (complex) predications. The relation is defined below, and visualized in the PSM$^{2+}$-model of figure 6.8.*

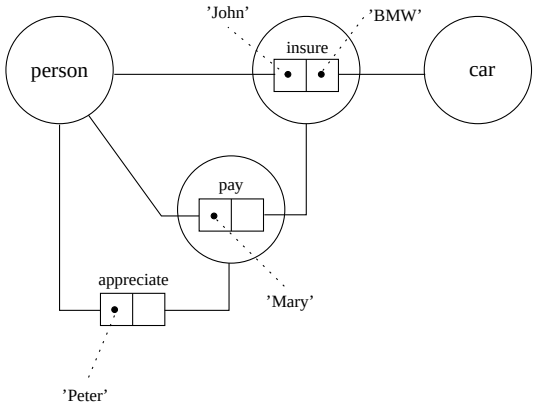


Figure 6.8: Verbalization of a nested (complex) predication

Relation(FACTUAL, ACTION, appreciate, [(ag, person('Peter')),
(go, Relation(FACTUAL, ACTION, pay, [(ag, person('Mary')),
(go, Relation(FACTUAL, ACTION, insure, [(ag, person('John')),
(go, Instance('BMW', car))])])])]), VTerm).

producing the verbalizations:

```
[the, person, Peter, appreciate - s, the, payment, of, the, insurance, of,
the, car, BMW, by, the, person, John, by, the, person, Mary];
[the, payment, of, the, insurance, of, the, car, BMW, by, the, person, John,
by, the, person, Mary, is, appreciate - ed, by, the, person, Peter]
```

□ **End Example**

The complete verbalization algorithm is given in appendix D. In this appendix verbalization rules are available for predications with modality FACTUAL. The predicate verbalization rules in this appendix are based on table 6.7.

## 6.6 Evaluation

This chapter has shown that the way of thinking and the way of modelling can be supported by various tools, which support the information modelling task and enhance communication between man and machine.

- The information modelling task of the information analyst is supported in different ways.
  - with *Formulation by Navigation*, modelling the UoD is simplified to understanding the natural language description delivered by domain experts and then selecting the corresponding predicates and terms. The predicates and terms are selected from a predefined set recorded in the the lexicon. When the information analyst selects a predicate from the list, the appropriate semantic roles are retrieved from the lexicon and set automatically for each argument.
  - construction of semantic anomalous predications is prevented by the information modelling tool on base of the knowledge available in the lexicon.

  - with the use of semantic roles (1) the task of the information analyst with respect to definition of meaningful, precise and unambiguous names of predicators is considerably simplified (2) query formulation is not restricted to the names as used in the information model (3) addition of common sense knowledge can be defined universally, i.e. domain independent.

- The interaction with the inference engine can be extended to realize a 'natural dialogue' in which both man and machine may ask questions or provide answers.

- Validation of the information model can be supported by verbalization. The verbalization algorithm introduced here, has shown the following desirable properties:

  - the verbalization algorithm is able to describe the same semantic load from different perspectives.
  - the verbalization algorithm can - without any modifications - be applied universally (i.e. domain independent).

# Chapter 7

# Conclusions and Further Research

## 7.1 Summary

**Chapter 2** has shown that the evolution of information systems has led to systems that focus on communication and intelligent decision making support. Especially, rule-based expert systems have to be mentioned in this context. The inference engine - one of the important components of an expert system - is able to simulate human reasoning by performing a forward or backward reasoning strategy. Verbalization of the information that directly or indirectly follows from the knowledge in the knowledge base - containing all the relevant information about the UoD - can be assisted by the forward chaining strategy. When verbalization shows a (potential) design flaw, backward chaining can be used to provide more insight in the information model and knowledge-base.

In **chapter 3**, the different abstraction-levels of natural language have been introduced, with an important separation between syntax (read structure of natural language) and semantics (read meaning of natural language). It has been argued that language is just a means to communicate ideas and conceptions in someone's mind to others. An idea is represented by a single word, while a conception is represented by a complete sentence. The semantic relations between ideas and conceptions are carried in a so-called mental lexicon. This mental lexicon enables *Natural Language Users* (NLUs) to understand words and sentences and to make semantic inferences. A semantic inference is an inference based on the meaning of words and sentences in the antecedent of the inference rule. In addition to this, this chapter has shown that the action-oriented approach in which verbs are treated as basic frames, enables precise definition of the meaning of a sentence.

The discussion on the action-oriented approach is detailed in **chapter 4**, by introduction of the linguistic theory of *Functional Grammar* (FG). One of the basic assumptions of FG is the idea of verb-frames (or so-called *predicates*) partially corresponding the the action-oriented approach. The exact meaning of a sentence can be given by filling this frame with terms representing objects. The role played by an object in this frame is denoted by a so-called *semantic role*. This promising semantics-oriented linguistic theory forms the foundation of a *Knowledge Representation Language*, called *Conceptual Prototyping Language* (CPL). The application of CPL in the field of information modelling has resulted into more expressive power and resolving of ambiguities.

The approach followed by CPL has been used in **chapter 5** as a source of inspiration for the introduction of a new information modelling technique based on the combination of $PSM^2$ with *if-then* rules, called $PSM^{2+}$. Actually, the $PSM^{2+}$ formalism can be regarded as a syntactical variant of CPL. With $PSM^{2+}$ the semantic inferences performed by NLUs on base of their mental lexicon can be simulated. The expressive power of $PSM^{2+}$ enables the establishment of a natural integration of *common sense* knowledge with data in order to realize intelligent, effective and efficient communication between man and machine.

The way of modelling introduced in the previous chapter, has been extended in **chapter 6** with various ways of supporting. In this chapter an information modelling tool is introduced by which the information analyst navigates through the lexicon to model the UoD. In addition, the ease of

automatic verbalization of PSM$^{2+}$-models is demonstrated. Again based on the solid foundation offered by the linguistic theory of FG. Finally, the interaction with the inference engine is optimized. The inference engine is extended with additional capabilities like the possibility to ask information of the user when needed during inference. On the other hand, the user may ask, *why* certain information is needed by the inference engine.

## 7.2 Evaluation

To recapitalize, this study has shown the following profits of semantics in comparison to syntax-oriented approaches:

**Way of Thinking**

- The way of thinking is not considerably affected by the introduction of semantics in information modelling: an elementary sentence is considered as a verb-frame (representing a relation) filled by nouns (representing objects or object types).

**Way of Modelling**

- *formal foundation*: The formal semantics of PSM$^{2+}$ is analogous to the sound formal foundation of CPL. The formal foundation of CPL is discussed in detail in [Dig89].
- *expressive power*: Because PSM$^{2+}$(just like CPL) is based on the linguistic theory of FG, in principle every sentence of NL can be expressed in PSM$^{2+}$. PSM$^{2+}$(just like CPL) exceeds the expressive power of PSM$^{2}$.
- *comprehensibility*: The PSM$^{2+}$-formalism has a negative influence on comprehensibility. However, the information modelling tool, as demonstrated in chapter 6, supports the information analyst with *Formulation by Navigation*. Additional enhancement of comprehensibility is achieved by the introduction of a graphical layer on top of the formalism.
- *executability*: The PSM$^{2+}$-formalism is based on predicate logic and *if-then* rules and is therefore without any modifications suitable for automatic processing by a PROLOG interpreter. This also holds for the semantic inference rules.
- *conceptual*: With exception of the introduction of properties of object types, no direct implementation decisions are taken. Predicate logic is accepted as a formal theoretical formulation language, without any implementation implications.

**Way of Supporting**

- *Information modelling*: The information modelling process is supported in different ways: *Formulation by Navigation*, construction of semantic anomalous predications is prevented and semantic roles have been introduced that simplify information modelling unambiguously.
- *Interaction*: Forward and backward reasoning strategies provide more insight in the information model. The information analyst may ask *how* conclusions are derived and *why* certain information is needed by the inference engine.
- *Verbalization*: The lexicon is used for automatic verbalization of the information model. Different verbalizations with the same semantic load can be generated without the necessity to define an explicit information grammar for each specific UoD.

## 7.3 Further Research

Many aspects of this document may trigger further researches to deepen or broaden the results of this study. In the following lines some of these are mentioned:

- In this document focus has been primarily on the *Object Action Involvement* and *Object Property* model. More research is needed on the *Object Life* model and the modelling of (complex) constraints.
- In this document our attention has been primarily focussed on a subset of set of available semantic roles and satellites recognized by *Functional Grammar*. The usefulness of other semantic roles and satellites however is not a priori denied.
- The backward chaining strategy is able to answer (and proof) user queries formulated in $PSM^{2+}$ terminology. However a natural language interface would be more appropriate, which translates user queries expressed in natural language to a corresponding query expressed in semantic notions. Information asked for by interrogative pronouns corresponds to objects playing a certain semantic roles: Who?-*agent*, When?-*tmp*, How?-*instr*, etc.
- The verbalization algorithm presented in this document has been restricted to a small set of semantic roles. More research is needed to discover and integrate the theoretical results of Functional Grammar with information modelling and verbalization of information models.
- In this document focus has been primarily on verbs (or predicates) expressing a process or action. More research is needed to fully understand the semantic load of so called *non-verbal* predicates, like *to be*.
- More research is needed to investigate the opportunities that bring the involvement of meaning of single words in the information modelling process. The definitional and prototype theory of meaning provide valuable clues for the incorporation of *if-then* rules expressing the semantic load of a single word.

I really do believe that with additional research the combination of the semantic approach to natural language and artificial intelligence will enable smooth natural language communication between users and information systems in the future. In addition to this, I hope that this study has contributed to the approach in which information systems try to understand users instead of users trying to understand information systems.

# Appendix A

# Implementation Forward Chaining

This appendix provides a PROLOG implementation of a forward chaining algorithm.

```
%
% Family facts
%

fact : parent(pam, bob).
fact : parent(pam, liz).
fact : parent(tom, bob).
fact : parent(tom, liz).
fact : parent(liz, mathilde).
fact : parent(liz, simon).
fact : parent(peter, mathilde).
fact : parent(peter, simon).

fact : of_sex(simon, male).
fact : of_sex(peter, male).
fact : of_sex(bob, male).
fact : of_sex(tom, male).
fact : of_sex(pam, female).
fact : of_sex(liz, female).
fact : of_sex(mathilde, female).


%
% Analytical knowledge
%

analyt7a : if parent(X, Y)
                  then ancestor(X, Y).

analyt7b : if (parent(X, Y) and ancestor(Y, Z))
                  then ancestor(X, Z).

analyt7c : if (parent(X, Y) and of_sex(X, male))
                  then father(X, Y).

analyt7d : if (parent(X, Y) and of_sex(X, female))
                  then mother(X, Y).
```

```
analyt7e : if (father(X, Y) and (father(Y, Z) or mother(Y, Z)))
                  then grandfather(X, Z).


%
% Operator definitions
%

:-op(900, xfx, :).
:-op(870, fx, iff).
:-op(880, xfx, then).
:-op(550, xfy, or).
:-op(540, xfy, and).

:-op(800, xfx, was).
:-op(300, fx, 'derived by').
:-op(600, xfx, from).
:-op(600, xfx, by).


%
% Forward chaining algorithm
%
forward:-
        derive([]).

derive(Facts):-
        Rule : iff Cond then Concl,
        is_true(Cond, Facts),
        not member(Concl, Facts),!,
        verbalize(Concl), nl,
        derive([Concl|Facts]).

is_true(exec(Cond), Facts):-
        Cond.

is_true(Cond, Facts):-
        fact : Cond
        ;
        member(Cond, Facts).

is_true(Cond1 and Cond2, Facts):-
        is_true(Cond1, Facts),
        is_true(Cond2, Facts).

is_true(Cond1 or Cond2, Facts):-
        is_true(Cond1, Facts)
        ;
        is_true(Cond2, Facts).
```

# Appendix B

# Implementation Backward Chaining

This appendix provides a PROLOG implementation of a backward chaining algorithm that generates a *trace* (or *prove tree*) for explanation purposes. The generated trace can be used to facilitate answering *why* and *how* questions of the user of the expert system.

```
backward:-
        getquestion(Question),
        (
        answeryes(Question)
        ;
        answerno(Question)
        ).

answeryes(Question):-
        markstatus(negative),
        explore(Question, [], Answer),
        positive(Answer),
        markstatus(positive),
        present(0, Answer), nl,
        write('More solutions?'),
        getreply(Reply),
        Reply = no.

answerno(Question):-
        retract(no_positive_answer_yet), !,
        explore(Question, [], Answer),
        negative(Answer),
        present(0, Answer), nl,
        write('More negative solutions?'),
        getreply(Reply),
        Reply = no.

markstatus(negative):-
        assert(no_positive_answer_yet).

markstatus(positive):-
        retract(no_positive_answer_yet), !
        ;
```

```
        true.

getquestion(Question):-
        nl,
        write('Question, please'), nl,
        read(Question).

getreply(Reply):-
        read(Answer),
        means(Answer, Reply), !
        ;
        nl, write('Answer unknown, try again please'), nl,
        getreply(Reply).

means(yes, yes).
means(y, yes).
means(no, no).
means(n, no).
means(why, why).
means(w, why).

explore(Goal, Trace, Goal  is true was 'found as a fact'):-
        fact : Goal.

explore(exec(Goal), Trace, Goal is true was 'found by execution'):-
        Goal.

explore(Goal, Trace, Goal is TruthValue was 'derived by' Rule from Answer):-
        Rule : iff Condition then Goal,
        explore(Condition, [Goal by Rule | Trace], Answer),
        truth(Answer, TruthValue).

explore(Goal1 and Goal2, Trace, Answer):-
        explore(Goal1, Trace, Answer1),
        continue(Answer1, Goal1 and Goal2, Trace, Answer).

explore(Goal1 or Goal2, Trace, Answer):-
        exploreyes(Goal1, Trace, Answer);
        exploreyes(Goal2, Trace, Answer).

explore(Goal1 or Goal2, Trace, Answer1 and Answer2):-!,
        not exploreyes(Goal1, Trace, _),
        not exploreyes(Goal2, Trace, _),
        explore(Goal1, Trace, Answer1),
        explore(Goal2, Trace, Answer2).

explore(Goal, Trace, Goal is Answer was 'told'):-
        useranswer(Goal, Trace, Answer).

exploreyes(Goal, Trace, Answer):-
        explore(Goal, Trace, Answer),
        positive(Answer).

continue(Answer1, Goal1 and Goal2, Trace, Answer):-
        positive(Answer1),
        explore(Goal2, Trace, Answer2),
        (
```

```
                positive(Answer2),
                Answer = Answer1 and Answer2
        ;
                negative(Answer2),
                Answer = Answer2
        ).

continue(Answer1, Goal1 and Goal2, _, Answer1):-
        negative(Answer1).

positive(Answer):-
        truth(Answer, true).

negative(Answer):-
        truth(Answer, false).

truth(Question is TruthValue was Found, TruthValue):-!.

truth(Answer1 and Answer2, TruthValue):-
        truth(Answer1, true),
        truth(Answer2, true), !,
        TruthValue = true
        ;
        TruthValue = false.
```

# Appendix C

# RDB schema of Information Modelling tool

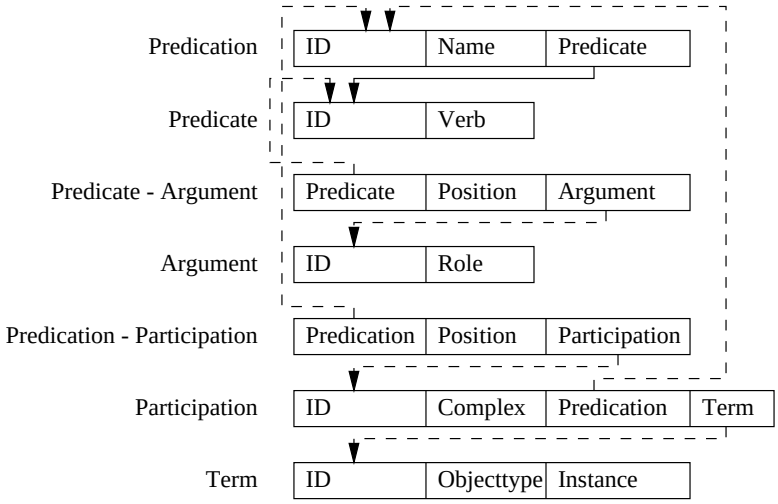


Figure C.1: Relational database schema of PSM$^{2+}$-tool

# Appendix D

# Implementation of Verbalisation algorithm

```
active([ag, pos, for, pro, zero]).

%
% Verbalisation rules
%

v:-
        gather([], Facts),!,
        member(Fact, Facts),
        verbalize(Fact, V),
        output(V),
        fail.

%
% active verbalisation of predication with only one argument
%
verbalize(relation(Mod, Tense, Predicate, [(SemNotion, Subject)]), VPredication):-
        active(Active),
        member(SemNotion, Active),

        verbalize((SemNotion, subject, d, sg), Subject, VSubject),
        verbalize(((sg, SemNotion),(Mod, Tense)), Predicate, VPredicate),
        VPredication = [VSubject, VPredicate].

%
% active verbalisation of predication
%

verbalize(relation(Mod, Tense, Predicate, Args), VPredication):-
        selectsubject(Args, Subject, Rest1),
        Subject = (SemNotion1, Term1),
        active(Active),
        member(SemNotion1, Active),

        selectobject(Rest1, Object, Rest2),
        Object = (SemNotion2, Term2),

        verbalize((SemNotion1, subject, d, sg), Term1, VTerm1),
```

```
        verbalize((SemNotion2, object, d, sg), Term2, VTerm2),
        verbalize(((sg, SemNotion1),(Mod, Tense)), Predicate, VPredicate),
        permutation(Rest2, PRest2),
        verbalize(PRest2, VRest2),
        VPredication = [VTerm1, VPredicate, VTerm2, VRest2].


%
% passive verbalisation of predication
%
verbalize(relation(Mod, Tense, Predicate, Args), VPredication):-
        selectsubject(Args, Subject, Rest),
        Subject = (SemNotion1, Term1),
        active(Active),
        not member(SemNotion1, Active),
        permutation(Rest, PRest),
        verbalize(PRest, VRest),
        verbalize((SemNotion1, subject, d, sg), Term1, VTerm1),
        verbalize(((sg, SemNotion1),(Mod, Tense)), Predicate, VPredicate),
        VPredication = [VTerm1, VPredicate, VRest].

%
% Verbalise a list of remaining terms
%
verbalize([], []):-!.

verbalize([(SemNotion, Term) |Terms], [VTerm|VTerms]):-
        verbalize((SemNotion, neither, d, sg), Term, VTerm),
        verbalize(Terms, VTerms).

%
% Verbalise simple term
%
verbalize((SemNotion, Distr, Def, Num), instance(Instance, Objecttype), VTerm):-
        table64(SemNotion, Distr, Prefix1),
        table65(Def, Num, Prefix2),!,
        VTerm = [Prefix1, Prefix2, Objecttype, Instance].

%
% Verbalise complex term containing a GOAL term
%
verbalize((SemNotion, Distr, Def, Num), relation(Mod, Ten, Pred, Args), VTerm):-
        selectterm(Args, (go, Goalterm), Rest),!,
        table64(SemNotion, Distr, Prefix1),
        table65(Def, Num, Prefix2),
        fact : gerund(Pred, Gerund),!,

        verbalize((go, subject, d, sg), Goalterm, VGoal),
        permutation(Rest, PRest),
        verbalize(PRest, VRest),
        VTerm = [Prefix1, Prefix2, Gerund, of, VGoal, VRest].

%
% Verbalise complex term without a GOAL term
%
verbalize((SemNotion, Distr, Def, Num), relation(Mod, Ten, Pred, Args), VTerm):-
selectterm(Args, (ag, Agent), Rest), !,
```

```
table64(SemNotion, Distr, Prefix1),
table65(Def, Num, Prefix2),
fact : gerund(Pred, Gerund), !,

verbalize((ag, subject, d, sg), Agent, VAgent),
VTerm = [Prefix1, Prefix2, Gerund, of, VAgent].

%
% Verbalise predicate
%
verbalize(((Num, Sem), (factual, action)), Predicate, Verbalisation):-
        active(Active),
        member(Sem, Active),
        present(Num, Predicate, Verbalisation).

verbalize(((Num, Sem), (factual, perf)), Predicate, Verbalisation):-
        active(Active),
        member(Sem, Active),
        present(Num, 'have', Prefix1),
        pap(Predicate, VPredicate),!,
        Verbalisation = [Prefix1, VPredicate].

verbalize(((Num, Sem), (factual, past)), Predicate, Verbalisation):-
        active(Active),
        member(Sem, Active),
        past(Num, Predicate, VPredicate),!,
        Verbalisation = [VPredicate].

verbalize(((Num, Sem), (factual, prosp)), Predicate, Verbalisation):-
        active(Active),
        member(Sem, Active),
        Verbalisation = ['will', Predicate].


verbalize(((Num, Sem), (factual, action)), Predicate, Verbalisation):-
        active(Active),
        not member(Sem, Active),
        present(Num, 'be', Prefix1),
        pap(Predicate, VPredicate), !,
        Verbalisation = [Prefix1, VPredicate].

verbalize(((Num, Sem), (factual, perf)), Predicate, Verbalisation):-
        active(Active),
        not member(Sem, Active),
        present(Num, 'have', Prefix1),
        pap('be', Prefix2),
        pap(Predicate, VPredicate),!,
        Verbalisation = [Prefix1, Prefix2, VPredicate].

verbalize(((Num, Sem), (factual, past)), Predicate, Verbalisation):-
        active(Active),
        not member(Sem, Active),
        past(Num, 'be', Prefix1),
        pap(Predicate, VPredicate),!,
        Verbalisation = [Prefix1, VPredicate].

verbalize(((Num, Sem), (factual, prosp)), Predicate, Verbalisation):-
```

```
        active(Active),
        not member(Sem, Active),
        pap(Predicate, VPredicate),!,
        Verbalisation = ['will', 'be', VPredicate].


%
% Verbalise utility rules
%

selectsubject(List, Subject, Rest):-
        selectterm(List, Subject, Rest),
        legalsubject(Subject).

selectobject(List, Object, Rest):-
        selectterm(List, Object, Rest),
        legalobject(Object).

legalsubject(Subject):-
        Subject = (SemNotion, _Term),
        member(SemNotion, [ag, go, rec]).

legalobject(Object):-
        Object = (SemNotion, _Term),
        member(SemNotion, [go, rec, ben]).

selectterm([X | Xs], Term, Rest):-
        member(Term, [X | Xs]),
        drop(Term, [X | Xs], Rest).


output(Structure):-
        flatten(Structure, Sentence),
        write(Sentence), nl.
```

# Appendix E

# Predicate verbalisations for modality MUSTand NEC

| Modality | SemN. | Tense | Verbalisation / General format |
|---|---|---|---|
| MUST | ag | ACTION | 'Monet' has to paint 'Le Parc Monceau' |
| | | | Present(sg, 'have'), 'to', <pred> |
| MUST | ag | PERF | 'Monet' has had to paint 'Le Parc Monceau' |
| | | | Present(sg, 'have'), PaP('have'), 'to', <pred> |
| MUST | ag | PAST | 'Monet' had to paint 'Le Parc Monceau' |
| | | | PAST(sg, 'have'), 'to', <pred> |
| MUST | ag | PROSP | 'Monet' will have to paint 'Le Parc Monceau' |
| | | | 'will', 'have', 'to', <pred> |
| MUST | go | ACTION | 'Le Parc Monceau' has to be paint-ed by 'Monet' |
| | | | Present(sg, 'have'), 'to', 'be', PaP(<pred>) |
| MUST | go | PERF | 'Le Parc Monceau' has had to be paint-ed by 'Monet' |
| | | | Present(sg, 'have'), PaP('have'), 'to', 'be', PaP(<pred>) |
| MUST | go | PAST | 'Le Parc Monceau' had to be paint-ed by 'Monet' |
| | | | PAST(sg, 'have'), 'to', 'be', PaP(<pred>) |
| MUST | go | PROSP | 'Le Parc Monceau' will have to be paint-ed by 'Monet' |
| | | | 'will', 'have', 'to', 'be', PaP(<pred>) |
| NEC | ag | ACTION | 'Monet' is obliged to paint 'Le Parc Monceau' |
| | | | Present(sg, 'be'), 'obliged', 'to', <pred> |
| NEC | ag | PERF | 'Monet' has been obliged to paint 'Le Parc Monceau' |
| | | | Present(sg, 'have'), PaP('be'), 'obliged', 'to', <pred> |
| NEC | ag | PAST | 'Monet' was obliged to paint 'Le Parc Monceau' |
| | | | PAST(sg, 'be'), 'obliged', 'to', <pred> |
| NEC | ag | PROSP | 'Monet' will be obliged to paint 'Le Parc Monceau' |
| | | | 'will', 'be', 'obliged', 'to', <pred> |
| NEC | go | ACTION | 'Le Parc Monceau' is obliged to be paint-ed by 'Monet' |
| | | | Present(sg, 'be'), 'obliged', 'to', 'be', PaP(<pred>) |
| NEC | go | PERF | 'Le Parc Monceau' has been obliged to be paint-ed by 'Monet' |
| | | | Present(sg, 'have'), PaP('be'), 'obliged', 'to', 'be', PaP(<pred>) |
| NEC | go | PAST | 'Le Parc Monceau' was obliged to be paint-ed by 'Monet' |
| | | | PAST(sg, 'be'), 'obliged', 'to', 'be', PaP(<pred>) |
| NEC | go | PROSP | 'Le Parc Monceau' will be obliged to be paint-ed by 'Monet' |
| | | | 'will', 'be', 'obliged', 'to', 'be', PaP(<pred>) |

Table E.1: Possible predicate verbalisations for modality MUST, NEC

# Appendix F

# PSM$^{2+}$-expert prototype

This appendix provides a PROLOG prototype implementation of a PSM$^{2+}$-expert. This prototype system is functions as the user interface to different supporting subsystems discussed in chapter 6:

- forward chaining algorithm (see appendix A)
- backward chaining algorithm (see appendix B)
- verbalisation mechanism (see appendix D).

```
expert:-
        welcome,
        start.

load:-
        consult(oper),
        consult(lib),
        write('Provide name of lexicon to be consulted:'),
        read(Lexicon),
        consult(Lexicon),
        write('Provide name of datamodel to be consulted:'),
        read(Datamodel),
        consult(Datamodel),
        consult(forward),
        consult(backward),
        consult(present),
        consult(useranswer),
        consult(verbalize).

welcome:-
        write('Welcome to prototype tool PSM2+-Expert, version 1.0'), nl,
        write('1996, University of Nymegen, Holland, E.S.C. van de Ven'), nl, nl.

start:-
welcome,
write('Please make your selection'), nl,
write('       (v)erbalize information model'), nl,
write('       (f)orward chaining'), nl,
write('       (b)ackward chaining'), nl, nl,
write('           Your answer:'), read(Answer),
startmodule(Answer).
```

```
startmodule(v):-
        v.

startmodule(b):-
        backward.

startmodule(f):-
        nl,
        write('Results of forward chaining process are:'), nl,
        write('-------------------------------------------------------'), nl,
        forward.

ss:-
        load,
        start.
```

# Bibliography


[AH87] S. Abiteboul and R. Hull. IFO: A Formal Semantic Database Model. *ACM Transactions on Database Systems*, 12(4):525–565, December 1987.

[AU77] Alfred V. Aho and Jeffrey D. Ullman. *Principles of compiler design.* Addison-Wesley, 1977.

[AW91] D.E. Avison and A.T. Wood-Harper. Information Systems Development Research: An Exploration of Ideas in Practice. *The Computer Journal*, 34(2):98–112, April 1991.

[Bem94] T.M.A. Bemelmans. *Bestuurlijke informatiesystemen en automatisering.* Kluwer Bedrijfswetenschappen, 1994. Dutch.

[BH89] H. Bennis and T. Hoekstra. *Generatieve Grammatica.* Foris Publications, Dordrecht - Holland, 1989. Dutch.

[Blo33] L. Bloomfield. *Language.* Holt and Company, 1933.

[BR95a] J.F.M. Burg and R.P. van de Riet. COLOR-X: Linguistically-based Event Modeling: A General Approach to Dynamic Modeling. In J. Iivari, K. Lyytinen, and M. Rossi, editors, *The Proceedings of the Seventh International Conference on Advanced Information System Engineering*, Lecture Notes in Computer Science, pages 26–39, Jyvaskyla, Finland, 1995. Springer-Verlag.

[BR95b] J.F.M. Burg and R.P. van de Riet. COLOR-X: Object Modeling profits from Linguistics. In *Proceedings of the KB&KS'95, the Second International Conference on Building and Sharing of Very Large-Scale Knowledge Bases*, Enschede, The Netherlands, 1995.

[Bra91] I. Bratko. *Prolog, programming for artificial intellegence.* Addison-Wesley, Reading, Massachusetts [etc.], 1991.

[Car56] Rudolf Carnap. *Meaning and necessity; a study in semantics and modal logic.* Phoenix, Chicago, 1956.

[Com87] B. Comrie. *The world's major languages.* Oxford University Press, 1987.

[Dig89] F.P.M. Dignum. *A Language for Modelling Knowledge Bases: Based on Linguistics and Founded in Logic.* Vrije Universiteit, 1989.

[Dik89] S.C. Dik. *The Theory of Functional Grammar. Part I: The Structure of the Clause.* Floris Publications, Dordrecht, The Netherlands, 1989.

[DKNZ92] C. Dekkers, C.H.A. Koster, M.-J. Nederhof, and A. van Zwol. Manual for Grammar WorkBench Version 1.5. Technical Report 92-14, Department of Computer Science, University of Nijmegen, Nijmegen, The Netherlands, July 1992.

[DR91] F.P.M. Dignum and R.P. van de Riet. Knowledge base modeling based on linguistics and founded in logic. *Data & Knowledge Engineering*, 7:1–34, 1991.

[Fah91] Chr. Fahner. *Taal voor Welzijn.* Number 9 in De vijverberg. Kok, Kampen, 1991. Dutch.

[FKW95] P.J.M. Frederiks, C.H.A. Koster, and Th.P. van der Weide. Object-Oriented Analysis using Informal Language. Technical Report CSI-R9516, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, December 1995.

[FW96] P.J.M. Frederiks and Th.P. van der Weide. From a File-Oriented View to an Object-Oriented View. Technical Report CSI-R9601, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, January 1996.

[Gle91] H. Gleitman. *Psychology.* W.W. Norton and Company, New York, London, 1991.

[Gri82] J.J. van Griethuysen, editor. *Concepts and Terminology for the Conceptual Schema and the Information Base.* Publ. nr. ISO/TC97/SC5-N695, 1982.

[Gri89] Ralph P. Grimaldi. *Discrete and Combinatorial Mathematics, an applied introduction, second edition.* Addison-Wesley, Reading, Massachusetts, Menlo Park/California [etc.], 1989.

[Hof93] A.H.M. ter Hofstede. *Information Modelling in Data Intensive Domains.* PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1993.

[HPW93] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.

[HPW94] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Grammar Based Information Modelling. Technical Report CSI-R9414, Submitted for publication, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, October 1994.

[HW92] A.H.M. ter Hofstede and Th.P. van der Weide. Formalisation of techniques: chopping down the methodology jungle. *Information and Software Technology*, 34(1):57–65, January 1992.

[HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.

[Jac83] M.A. Jackson. *System Development.* Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[Kos91] C.H.A. Koster. Affix Grammars for natural languages. In *Attribute Grammars, Applications and Systems, International Summer School SAGA*, volume 545 of *Lecture Notes in Computer Science*, pages 469–484. Springer-Verlag, Berlin, Germany, June 1991.

[Kri94] G. Kristen. *Object Orientation, the KISS Method: From Information Architecture to Information System.* Addison-Wesley, Reading, Massachusetts, 1994.

[Mil93] G. A. Miller. *De wetenschap van het woord, De hersenen als zetel van de taal.* Wetenschappelijke Bibliotheek, Natuur en Techniek, Maastricht/Brussel, 1993. Dutch.

[Mos79] Stephen T. Moskey. *Semantic Structures and Relations in Dutch.* Georgetown University, United States of America, 1979.

[NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach.* Prentice-Hall, Sydney, Australia, 1989.

[Rei86] R. Reiter. Foundations for knowlegde-based systems. *Information Processing*, 1986.

[RK91] E. Rich and K. Knight. *Artificial Intelligence Second Edition.* New York, 1991.

[Sch94] R. Schonfeldt. Mathematische eigenschaften fur thesaurirelationen. 1994.

[Sim60] H. Simon. *The new science of management decision.* Harper and Row, New York, 1960.

[Wij91] G.M. Wijers. *Modelling Support in Information Systems Development.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 1991.

[You89] E. Yourdon. *Modern Structured Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

# Author Index


A. van Zwol, 27, 123
A.H.M. ter Hofstede, 9–11, 19, 20, 38, 45, 62, 81, 124
A.T. Wood-Harper, 9, 123
Alfred V. Aho, 27, 123

B. Comrie, 32, 123

C. Dekkers, 27, 123
C.H.A. Koster, 10, 11, 18, 27, 40, 43, 44, 46, 57, 81, 98, 123, 124
Chr. Fahner, 23, 123

D.E. Avison, 9, 123

E. Rich, 14, 124
E. Yourdon, 10, 124

F.P.M. Dignum, 10, 11, 17, 18, 36, 38, 40, 45, 46, 63, 104, 123

G. A. Miller, 26, 29, 53, 62, 66, 74, 124
G. Kristen, 10, 11, 43, 50, 73, 76, 124
G.M. Nijssen, 16, 124
G.M. Wijers, 81, 124

H. Bennis, 23, 27, 28, 123
H. Gleitman, 14–16, 18, 23, 30, 34, 53, 124
H. Simon, 14, 124
H.A. Proper, 11, 20, 62, 124

I. Bratko, 15, 17, 18, 21, 123

J.F.M. Burg, 10, 11, 19, 36, 43, 123
J.J. van Griethuysen, 9, 124
Jeffrey D. Ullman, 27, 123

K. Knight, 14, 124

L. Bloomfield, 33, 123

M.-J. Nederhof, 27, 123
M.A. Jackson, 10, 124

P.J.M. Frederiks, 10, 11, 13, 18, 40, 43, 44, 46, 57, 81, 98, 124

R. Hull, 9, 123
R. Reiter, 15, 124
R. Schonfeldt, 63, 124
R.P. van de Riet, 10, 11, 17–19, 36, 38, 43, 45, 123
Ralph P. Grimaldi, 63, 124
Rudolf Carnap, 61, 123

S. Abiteboul, 9, 123
S.C. Dik, 30–33, 35, 77, 82, 87, 88, 90, 92, 93, 98, 123
Stephen T. Moskey, 32, 36, 124

T. Hoekstra, 23, 27, 28, 123
T.A. Halpin, 16, 124
T.M.A. Bemelmans, 14, 15, 123
Th.P. van der Weide, 9–11, 13, 18–20, 38, 40, 43–46, 57, 62, 81, 98, 124

# Index


action, 30, 37, 51, 56, 69, 70
    initialization, 51
    subtype defining -, 61
action-oriented, 30
adjective, 26, 66
    attributive, 66
    non-attributive, 66
    non-predicative, 66
    predicative, 66
adjectives, 27, 40
advantages, 11
adverb, 26
Agent, 33
aggregation, 39, 53
algorithms, 15
ambiguities, 46
ambiguity, 27, 45
ambiguous, 31, 36
analogical reasoning, 17
analytical rules, 20
antecedent, 86
antonym, 29
architectural evolution, 13
architecture, 13
argument, 82, 84
artificial intelligence, 14
askable, 86
Askables, 19

backward chaining, 21
base
    authorization, 13
    forms, 13
    information, 13
    procedures, 13
Beneficery, 36
bipolar, 67
brute force, 18

cardinality, 41
Case-based reasoning, 17
categorical notions, 32
category, 26
Cause, 36
Certainty factors, 18
chaining
    backward -, 85
    forward -, 85
clause, 35
collective, 41
common nouns, 26
common sense, 19
communication, 13, 53
communication-oriented, 13
Company, 36
competence, 23
components, 17
comprehensibility, 11, 31
comprehensible, 10
concept, 24, 29, 32, 53
concepts, 30
conceptual, 10
Conceptual Prototyping Language, 38
conclusions, 16
concrete objects, 25
conditional statements, 16
conjunction, 26
connectionist models, 17
constraint, 44, 82
    dynamic, 44
    occurrence frequency, 40
    permissive, 45
    prescriptive, 45
    static, 44
    total, 40
    uniqueness, 40
constraints, 18
context, 33, 65
context sensitive, 30
context-free grammar, 27
control, 37
countable nouns, 26
course of life, 18, 43
CPL, 38
creative, 23

data model, 9
data-oriented, 13
database systems, 15
decision
    making, 13
decision making, 13, 14
Decision Support Systems, 15
deduction, 16
Deductive reasoning, 16
deep structure, 28

definite (d), 91
definite article, 87
definiteness, 87, 91
definition, 29
definitional theory, 29
denotation, 25
Deontic rules, 18
Det, 27
determiner, 26, 27
direct object, 26
Direction, 33
distributive, 41
distributivity, 41
domain expert, 54
domain specific rules, 18
domain-expert, 9
domain-experts, 81
DSS, 15
dynamic behavior, 18, 43
dynamism, 37

ER, 55
ES, 15
executable, 10, 32
experience, 37, 38
experiential reasoning, 17
Expert System, 15
expert system, 16
explaining, 15
explanation, 17, 21, 85
exploration, 14
expression rules, 34, 87
expressive power, 10, 31, 38, 39, 46, 52
extension, 25

feedback, 14
FG, 38
file-oriented, 13
Force, 33
form, 34
formal, 10, 31
    semantics, 45
formal foundation, 9, 11
Formal specification, 10
Formulation by Navigation, 85
forward chaining, 21
Frame-based reasoning, 16
frames, 30
function
    Object -, 88
    pragmatic, 33
    semantic, 33
    semantic -, 88
    Subject -, 88
    syntactic, 33
    syntactic -, 88
Functional Grammar, 31, 32, 38, 81
functional grammar, 62

generalization, 39, 53, 59
Goal, 33
gradualness, 67
grammar based information modelling, 11

Heuristics, 18
heuristics, 15
hypothesis, 21

idea, 24, 29, 53
if-then, 16
implementation, 18
    decisions, 10
incompleteness, 15, 45
inconsistency, 45
inconsistent, 19
indefinite (i), 91
indirect object, 26
induction, 16
inductive reasoning, 16
inference, 16, 54
inference engine, 17, 21, 55, 85
infinitive, 34
information, 13
information analyst, 9, 54, 81, 84
information grammar, 13, 62
information model, 9
Information modelling, 9
information modelling, 81, 84
information processor, 13
information system, 13
informed users, 81
initial specification, 81
instance relation, 58
instantiation, 39
Instrument, 36, 75
intelligent, 52, 55
intension, 25
interaction, 52, 85
interjection, 26
interpersonal, 23

KBS, 15
knowledge, 13, 16
    analytical, 19, 61, 73
    common sense, 54, 61
    deontic, 18
    domain independent, 19
    factual, 18
    general world, 53, 55
    language specific, 87
    linguistic, 19, 55
    object action involvement, 18
    object life, 18
    object property, 18

population, 18, 55
knowledge base, 17, 18, 85
Knowledge base systems, 15
Knowledge Representation Language, 31, 38
knowledge representation language, 11
KRL, 31, 38

label type, 18, 39
language, 23
langue, 23
lexical object types, 18
lexicon, 19, 29, 30
linguistic, 38
linguistic expression, 34, 87
Location, 33, 36
logic, 11
deontic, 46
modal, 46
predicate, 46
temporal, 46
logic programming, 21

macro-definitions, 62
Manner, 36
meaning, 24, 25, 28, 55
definitional theory of, 29
prototype theory of, 29
meaning postulate, 61
meaningful, 23
meanings, 23
membership, 40
mental lexicon, 29, 53, 55, 67
mental vision, 62
meta schema, 57
method, 56
miniature drama, 30
modality, 45, 47, 84
model
constraint, 44
KISS, 50, 51
Object Action Involvement, 46, 82
Object Life, 46, 50
Object Property, 82
modelling
constraint, 38
data, 38
process, 38
momentaneous, 37
morpheme, 25
morphological modifications
of predicates, 93, 94
of terms, 92
morphology, 25

natural language, 23
Natural Language Users, 34
natural language users, 25
natural terminal point, 36
neural networks, 17
NIAM, 55
NLUs, 25, 34
non-countable nouns, 26
noun, 26
common, 26
countable, 26
non-countable, 26
proper, 26
noun phrase, 27
nouns, 26
NP, 27
number, 91
numeral, 26

object
participation, 61
property, 66
object assignment, 90
Object Life Model, 43
object life model, 43
Object oriented, 10
object type
weak -, 73
object types, 39
object-oriented, 13
Object-Participation, 59
objects, 56
OLM, 43
operators, 35
oriented
relation-, 32
over-specification, 45

parameter, 67
parole, 23
parse tree, 27
participation, 81, 84
pattern matching, 17
Pattern-based reasoning, 17
performance, 23
performers, 30
perspective, 88
data, 9
process, 9
Petri net, 10
philosophy, 81
phonemes, 25
phonetic alphabet, 25
phonology, 25
phrases, 25
plan, 13
planning, 13
plural (pl), 91
point of view, 88
polarity, 67

population, 42
Positioner, 33
possession, 40, 63, 98
PP, 27
pragmatic foregrounding, 77
precondition, 50, 51
predicate, 30, 82, 84
  basic-, 82
  derived -, 82
  formation rule, 82
  nominalization, 98
predication, 35, 81, 84
  complex-, 84
predication operator, 35
predication property
  modality, 93
  tense, 93
predicator, 62
preposition, 26, 27
prepositional phrase, 27
problem of, 24
problem-formulation, 15
problem-solving, 17
process, 37, 69
process model, 9
Processed, 33
production rules, 16, 27
PROLOG, 57
Prolog, 21
pronoun, 26
proper nouns, 26
properties, 18, 40
property, 56, 63
proposition, 30, 35
prototype theory, 29
PSM, 55
PSM$^{2+}$, 57
PSM$^{2+}$-Expert, 57

qualitative valency, 35
quality, 9
quantitative valency, 34

reasoning strategies
  see chaining, 85
reasoning strategy, 14, 16
Recipient, 33
redundancy, 45
Reference, 33
reference, 25, 29
referential, 23
relation, 56
  ActionAgent, 62
  antonym, 66
    complementary, 67
    contradictory, 67, 68
    contrary, 67, 69
    contrasting, 67, 69, 70
    direct, 69, 70
    indirect, 69
    relative, 67
    reversal, 67, 69
  function, 74
    ability, 75
    instrument, 75
  gerund, 76
  hypernym, 53, 60, 62, 70, 71
  instance, 58, 60
  is-a, 59
  many-to-many, 41
  meronym, 40, 53, 62, 63, 70, 72, 74
    activity-symptom, 63
    object-component, 63
    object-material, 63
    process-phase, 63
    quantity-portion, 63
    region-place, 63
    set-element, 63
  object-object, 63
  object-property, 63
  one-to-many, 41
  one-to-one, 41
  part-of, 62
  part-whole, 62
  partial-synonym, 65
  property, 64
  semantic consequence, 61, 70
    presupposition, 73
    proper inclusion, 72
    simultaneity, 71
  synonym, 53, 65
  transmission, 77
  troponym, 71
relations, 32, 39
requirements, 10
requirements engineering, 9
resource management, 13
resources, 13
retrieval language, 62
rewrite rules, 27
rewrite-rules, 30
role, 30
rule
  analytical, 44
  deontic, 44
Rule-based reasoning, 16

satellite
  time, 73
satellites, 35, 84
SDS, 15
selection restriction, 30, 35
semantic features, 29
semantic parameters, 37

semantic relation
see relation, 56
semantic relations, 53
semantic representation, 32
semantical
load, 46
semantical anomalies, 28
semantical approach, 10, 31, 52
semantical features, 53
semantical framework, 32
semantical relation, 29
semantical role, 47
semantical roles, 52
semantics, 11, 29, 87
semantics-oriented, 46
sentence, 87
active, 28
declarative, 28, 35
imperative, 35
interrogative, 28, 35
passive, 28
signs, 23
silver bullet, 10
singular (sg), 91
situation
desired, 44
possible, 44
SoA, 33, 35
Source, 33
space, 35
specialization, 39, 53, 59
speech act, 35
state, 37
State of Affairs, 33
structural principles, 24
structured, 23
Structured approaches, 10
Structured Decision Systems, 15
subject, 30, 33
subject assignment, 90
subject property
number, 93
semantic notion, 93
subordinate clause, 35
subtype defining rule, 20
support, 14
surface structure, 28
synonym, 29
syntactic category, 34
syntactical approach, 10, 11
syntactical categories, 26, 27, 30
syntactical level, 28
syntactical rules, 26, 27
syntax, 25, 87
syntax-oriented, 46, 52
system architecture, 17
system lexicon, 55
taxonomy, 62
telic, 37
tense, 47, 84
tense operator, 35
term, 82
TGG, 23
Time, 36
time, 35, 44, 73
time interval, 35
TPS, 15
Transaction Processing System, 15
transformation, 87
transformation rules, 28
transformational- generative grammar, 30
transformational-generative grammar, 23, 32
transitive nature
of properties, 64
transitivity
of hypernym relations, 59
of meronym relations, 63
of object-participation, 61
transparency, 17
troponym, 71
typology of predications, 36

uncertain, 15
uncertainty, 14
underlying clause, 34
Universe of Discourse, 9
user interface, 17, 22

validating, 9
Validation, 9
validation, 85
vantage point, 88
verb, 26
action, 57
auxiliary, 26
ditransitive, 26
intransitive, 26
lexical, 26
modality, 26
state, 57
transitive, 26
verb phrase, 27
verbalization, 22, 87
algorithm, 89
of predicates, 92, 94
of simple predications, 94, 97
of terms, 90, 91
passive -, 89
Verification, 9
verifying, 9
violation, 19
VP, 27

way

of modelling, 81
of supporting, 81
of thinking, 81
well-formedness, 28
word-association test, 53, 67, 74
words, 25

Zero, 33