Markopekka Niinimäki

Conceptual Modelling Languages

DEPARTMENT OF COMPUTER SCIENCES
UNIVERSITY OF TAMPERE

A-2004-1

TAMPERE 2004

Markopekka Niinimäki

Conceptual Modelling Languages

ACADEMIC DISSERTATION

DEPARTMENT OF COMPUTER SCIENCES
UNIVERSITY OF TAMPERE

A-2004-1

TAMPERE 2004

Opponent: Prof. Dr. Bernhard Thalheim
Christian-Albrechts-University Kiel

Reviewers: Prof. Emer. Veikko Rantala
University of Tampere

Docent Jari Palomäki
Massey University

Electronic dissertation

Acta Electronica Universitatis Tamperensis 326
ISBN 951-44-5910-5
ISSN 1456-954X
http://acta.uta.fi

Department of Computer Sciences
FIN-33014 UNIVERSITY OF TAMPERE
Finland

ISBN 951-44-5846-X
ISSN 1459-6903

Tampereen yliopistopaino Oy
Tampere 2004

Abstract

Conceptual modelling is needed to form a description of the domain of application at hand. In order to express the result of conceptual modelling, we need a modelling language. First order predicate logic (FOPL) or its variants (like Horn-clauses and Description Logics) and extensions (second order logics) can form a backbone of such a language, but some criteria is needed to define the fruitfulness or suitability of a modelling language, given the modelling task. The modelling language together with its methodology constitute a modelling perspective.

Many accounts of conceptual modelling emphasise an intensional perspective. This means that the starting point in conceptual modelling is the contents of the concepts that subsist in the domain of application (as opposed to “things” in the domain of application – they belong to the extensions of these concepts). If an intensional perspective is used, it should be visible in the modelling language as well; therefore we divide modelling languages into intensional and extensional languages, and “hybrid” languages that combine some aspects of these two.

First order predicate logic has often been used as an example of a language with a well established extensional semantics. We examine FOPL as a modelling language in connection with Sowa’s Conceptual Graphs (CGs). It can be demonstrated that a limited version of the language of CGs is equal (in expressive power) to that of FOPL with unary and binary predicates. However, contrary to the claims of proponents of CGs, CG presentations are not necessarily easier to read or understand than the same presentations expressed in FOPL (as can be demonstrated by comparing “typical” but complex FOPL formulas and their CG counterparts) .

Kauppi’s concept calculus is based on concepts and the relation of intensional containment. An approach where a modelling language is based completely on concept calculus is presented in this thesis. This approach has the advantage that the user can apply the operations of the calculus when designing a conceptual schema of the domain of application. However, this sort of modelling can be restrictive and impractical in many cases, since it enforces rather strict concept structures. CONCEPT D, a modelling language, can be seen as a less restrictive alternative.

Using CONCEPT D, the modeller reports the results of the modelling task in the form of concept diagrams. But we often need to ask “is this concept diagram correct” or “does it correspond well to the domain of application”. Without semantics (which connects the diagrams to something extra-linguistic) we can only answer these questions on the basis of our intuitions. We address these questions from two different angles. First we demonstrate how to map (simplified) CONCEPT D concept diagrams into IFO schemata that have well-defined semantics. Then we study what kind of semantical theory (e.g. possible world semantics, situation semantics, HIT-semantics) would capture the features that we want to express in concept diagrams. CONCEPT D has been rarely used in applications where it would be important to make a distinction between, for example, **prime number less than one** and **round square**, but is has the capability of making these distinctions. Therefore, it seems that it needs semantics “fine-grained” enough. Finally, we discuss, based on the previous chapters, on what premises HIT-semantics would serve as the semantical background of conceptual modelling.

Acknowledgments

The author wishes to thank Marie Duzi, Ari-Pekka Hameri, Matti Heikkurinen, Marko Junkkari, Hannu Kangassalo, Makiko Matsumoto Niinimäki, Erkki Mäkinen, Tapio Niemi, Timo Niemi, Jyrki Nummenmaa, Veikko Rantala, Vesa Sivunen, Stephen Slampyak, Ari Virtanen, and John White for their valuable advice.

This research has been partially supported by TISE travel grants, a grant by Helsinki Institute of Physics and a grant by Frenckell foundation.

Geneva, December 2003
Marko Niinimäki

Contents

Preface

Chapters 2 and 7 of this thesis are based on articles published in the series Information Modelling and Knowledge Bases.[1]

Chapters 4 and 5 are revised versions of papers written jointly with Marko Junkkari and published in the same series.[2] An earlier version of Chapter 3 has been published as a Technical Report.[3] Both the authors have had equal and inseparable contributions in these papers.

Chapter 6 has been revised from an article published in the "Filosofia" series of the Academy of Sciences of the Czech Republic.[4]

All contents reprinted by permission of the publishers, where required.

[1]Marko Niinimäki: Intensional and extensional languages in conceptual modelling, in H. Jaakkola, H. Kangassalo, and E. Kawaguchi, editors, Information Modelling and Knowledge Bases XII, IOS Press, 2001 and Marko Niinimäki: Semantics and conceptual modelling – Explicating the semantics of concept diagrams, in H. Kangassalo, H. Jaakkola, E. Kawaguchi, and T. Welzer, editors, Information Modelling and Knowledge Bases XIII, IOS Press, 2002.

[2]Marko Junkkari and Marko Niinimäki: An algebraic approach to Kauppi's concept theory in H. Jaakkola, H. Kangassalo, and E. Kawaguchi, editors, Information Modelling and Knowledge Bases X, IOS Press, 1999 and Marko Junkkari and Marko Niinimäki: An algebraic approach to Kauppi's concept theory II: Concept operations and associations in E. Kawaguchi, H. Kangassalo, H. Jaakkola, and I.A. Hamid, editors, Information Modelling and Knowledge Bases XI, IOS Press, 2000.

[3]Marko Niinimäki: Understanding the semantics of conceptual graphs. University of Tampere, Department of Computer Science, Technical Report A-1999-4.

[4]Marko Niinimäki: Semantic Data Models and Concepts – A Comparison of IFO and COMIC, in O. Majer, editor, Topics in Conceptual Analysis and Modelling, Institute of Philosophy, Academy of Sciences of the Czech Republic, 2000.

Chapter 1

Introduction

1.1 Research problems

This thesis addresses problems ("What are intensions?", "What is the semantics of a concept diagram?") and proposes solutions concerning the role of intensions and formalisms in conceptual modelling. In order to make these problems understandable, in this introduction we first define some terminology used throughout the thesis. As a preview, the research problems of the thesis are as follows (for details, see Section 1.6):

- Are there notable differences in conceptual modelling languages? If that is the case, in what way and based on what principles, can different modelling languages (formalisms) be classified? Tentatively, are there differences between "extensional" and "intensional" approaches and languages?
- If we take an extensional approach, where First Order Predicate Logic is often used, do we gain any benefits using a formalism like Conceptual Graphs instead of the traditional First Order Predicate Logics?
- What are the possibilities and limitations of an intensionally based approach? What is the role of concept theory like Kauppi's (see [Kau67]) in this kind of approach?
- If we are to utilise a modelling language based on the intensional approach, how does it compare to more conventional languages like well-known IFO, and what kind of semantics would it have?
- What kind of semantical background theory could be seen as useful from the point of view of conceptual modelling?

1.2 Basic terminology

Modelling can be seen as an activity consisting of (i) an object to be modelled, (ii) a model, (iii) a modelling relationship between these two, and (iv) someone – a modeller – conducting this activity.[1] In this thesis, we call the object of the model the *domain of application*. The domain

[1] This is loosely based on [Pal94]. Some authors, including [Kan00] emphasize that the relationship is affected by factors like information available about the domain of application, the ontology used as the basis of the conceptualisation process, the purpose of the modelling, etc.

of application is simply something the modeller is interested in, and we do not assume it has a specific structure – it may have, but in many cases the structure is imposed later by the modeller. We assume that the conceptualisation of the domain of application is carried out by means of some language. Once we have some language by which we can approach the domain of application, we can define the language's semantics. At that stage, we consider relations between linguistic entities and "non-linguistic phenomena" in the Universe of Discourse (UoD), which is a set whose elements have been "selected" from the domain of application.

In a model, irrelevant details are not taken into account, thus allowing the user of the model to examine and manipulate the objects of interest in the model.[2] Data modelling has a long tradition in computer science; it is concerned with representing the complexity of the domain of application in a structure that can be manipulated by a computer.

Data models are specific languages[3] for describing the structure of the data stored and operations for manipulating it (see [Bor91]). Among them, the relational model is probably the best known. Since the late 1970's, several *semantic data models* (SDMs) have been suggested. Instead of relying on relations and their operations (such as projection and union), in SDMs the language in question contains terminology that can be more directly related to the domain of application (like IS-A and has-attribute).[4]

In this thesis, we analyse modelling languages, i.e., languages that have been designed for the purposes of conceptual modelling. Not all of them come from the tradition of semantic data models, but all of them aim at creating a model of the domain of application. In the model, we can recognise (i) language primitives that have counterparts in the domain of application or some other realm (like natural numbers) and (ii) language constructs by which these items can be combined.[5]

In general, *conceptual modelling* is a special case of modelling where the model is not something physical (e.g., a miniature model of a future building) but conceptual. According to Marjomaa in [Mar02], conceptual modelling is the description of information systems on the meta-level, where conceptual processes, model constructions and knowledge representations play an essential role. On the other hand, conceptual modelling can be seen as an activity the goal of which is to develop high level concepts, tools and techniques for all areas in computer science. In [KKJH00], Kawaguchi et al. define information modelling as "structuring originally unstructured or ill-structured information by applying various types of abstract models and principles, for different purposes." Theory of science, organisational knowledge management, database design and software development are mentioned among the application areas of information modelling, and conceptual modelling is seen as one of its most important sub-areas in [KKJH00].

The goal of a conceptual modelling process is to develop a *conceptual schema* of the domain of application. In the construction of the conceptual schema, two principles serve as guidelines (see [TCI87]), the "conceptualization principle" (only conceptual aspects should be taken into account when constructing the conceptual schema) and the 100% principle (all the relevant aspects of the domain should be described in the conceptual schema).

The status of the conceptual schema is not entirely clear. On one hand, the working group of Technical Committee ISO/TC 97 in [TCI87] emphasises the conceptual nature of the conceptual schema. This would mean that the conceptual schema is something identical to conceptual model.

[2]For further discussion about models, see e.g. [MM01].

[3]For a precise definition of languages, see later in this chapter. Here, it is sufficient to state that languages consist of primitives and ways of combining them.

[4]In this chapter, we use the terms "IS-A" and "has-attribute" in an intuitive and practical sense. For discussion about the IS-A relation in knowledge representation, see [Bra83].

[5](i) and (ii) correspond to terminal alphabet and production rules of Section 1.7.

On the other hand, in [TCI87], a conceptual schema is defined as a formal description of a domain of application (called Universe of Discourse, UoD in [TCI87]). It uses some (normative) formalism and it allows a formal description of entities contained in a domain of application, along with properties and relationships between those entities. Moreover, it allows the description of formal rules, constraints, events, processes and other features of semantics. In order to avoid ambiguity, we emphasise that:

- A conceptual schema does not necessarily need to be implemented in a database, but it can be manifested for instance in a diagram;
- A conceptual schema is something conceptual (e.g. even in the sense of platonic realism, where concepts have their own mode of existence), but for the purpose of communication it has to be written down or otherwise presented using some conventions of a modelling language, i.e. a formalism. Following Kawaguchi et al. in [KKJH00], we call a conceptual schema in this form an *externalised conceptual schema*. It is expressed using the expressions of the modelling language.

In order to discuss conceptual modelling, models, schemata, and modelling languages, we make a distinction between things in the real world, things in the language, and concepts. More precise definitions can be found in Section 2.2.

1.3 Preliminaries: Approaches to conceptual modelling

In this thesis we concentrate on different approaches to conceptual modelling. These will be discussed in connection with structural aspects in the description of the domain of application. The motivation for this is that in practical terms, the modeller needs some sort of a framework for his modelling task. This can contain the classification of reality as objects, roles, and relationships, as in many popular modelling approaches. On the other hand, the framework can include identification of different knowledge primitives, too, as in the COMIC approach discussed in [Kan93].

It is mostly out of the scope of this thesis to discuss the process of identifying and classifying real world phenomena in the process of conceptual modelling. This process has been emphasised in the intensional approach to conceptual modelling, discussed in Chapter 4, and is quite crucial in the COMIC methodology. However, we make a distinction between *semiotic* (meaning in cultural contexts and minds[6]) phenomena that take place during this process, and *semantic* phenomena. The semantic theory employed here is more formal (see Section 1.7) and its relationship to conceptual modelling can be explained as follows:

The modeller reports the results in the form of an externalised conceptual schema, using a modelling language. An externalised conceptual schema is a linguistic construct, though it represents (probably) conceptual things. Externalised conceptual schemata are expression sets, linguistic level objects, composed of alphabets according to rules of a grammar. Thus, it is possible to explicate the semantics of externalised conceptual schemata. A clear semantics is a primary concern of the people designing a modelling language, since an externalised conceptual schema can be used as a tool of communication in the user community. It must be emphasised that the principles of

[6]Nauta, in [Nau72], defines semiotics as the study of "semiosis", which is a sign process, described as a five-term relation S(s,i,e,d,c). S stands here for the semiotic relation; s for 'sign'; i for 'interpreter'; e for 'effect'; d for 'denotatum' and c for 'context'. See [Nau72], p.36 and p.28.

semantics are not bound to any particular language: any language with symbols and ways of combining them should have semantics. However, in different kinds of modelling approaches, different kinds of languages are employed and they are supposed to differ in terms of semantics as well.

The notion of *intensionality* in conceptual modelling has been emphasised in many accounts, e.g. [Sow84], [DLNN97], [Woo91], [Kan92] and [Kan93]. In philosophy, intensionality is often understood through a distinction between intensions and extensions, on one hand, and intensional and extensional contexts, on the other hand.[7] If we think of the intension of a concept as the internal contents of the concept, we are often interested in the elements and relationships within these contents. The latter, naturally, we call intensional relationships.

It is an open question (discussed in Chapter 2) what the role of intensional relationships is in a conceptual schema. What is meant by the term "intensionality" in general is discussed from the perspective of semantics in Chapter 7, but the following will serve as a short summary:

In general, concepts are relatively independent of minds and things in the world. There is a relation Z (in German "zukommen") that connects a thing in the world with a concept. The *extension* of a concept in the world is a set of things for which this relation applies. Naturally, this set can be empty as well: in the actual world, there are probably no things in Z-relation with the concept of **ghost**.

The *intension* of a concept, on the other hand, is on the conceptual level. In the context of this thesis, we adopt the view that the intension of concept **a** contains the concepts that are contained in **a**.

- According to Kauppi in [Kau67], the intension of a concept is based on the relation of intensional containment. Kauppi's concept theory in based on Leibniz's (1646 - 1716) philosophy, where truth is analysed as "the subject concept containing the predicate concept" (see [Zal00]).[8] Kangassalo has given interpretations of Kauppi's theory that relate it to conceptual modelling. According to Kangassalo in [Kan96], the intension of a concept is the "knowledge contents" of the concept and the relation of intensional containment covers those of IS-A, contains (part-of), and has-attribute. To sum up, according to this view, "something

[7]Quoting [Bla96], "The extension of a predicate is the class of objects that it describes: the extension of 'red' is the class of red things. The intension is the principle under which it picks them out or in other words the condition a thing must satisfy to be truly described by the predicate. Two predicates ('...is a rational animal', '...is a naturally featherless biped') might pick out the same class but they do so by a different condition. If the notions are extended to other items, then the extension of a sentence is its truth-value, and its intension a thought or proposition; and the extension of a singular term is the object referred to by it, if it so refers, and its intension is the concept by means of which the object is packed out. A sentence puts a predicate or other term in an extensional context if any other other predicate or term with the same extension can be substituted without it being possible that the truth-value changes: if John is a rational being, and we substitute the co-extensive 'is a naturally featherless biped', the John is a naturally featherless biped. Other contexts such as 'Mary believes that John is a rational animal', may not allow the substitution, and are called intensional contexts."

Encyclopaedia Britannica [eb-94b] states the difference of intension and extension as "'intension' indicates the internal contents of a term or concept that constitutes its formal definition; and 'extension' indicates its range of applicability by naming the particular objects it denotes."

Moreover, in philosophy, intensional logics are maybe the best examples of studies of intensionality. Contrary to many first order predicate logics "intensional logics allow one to develop theories of properties that have the same extension but differ in intension" ([eb-94a]). A variant of intensional logic, Transparent Intensional Logic (TIL) is introduced in Section 7.3.2. For a more detailed discussion of intensions in philosophy and logic, see e.g. [vB88].

[8]"The subject concept containing the predicate concept"we, naturally, interpret as intensional containment. From today's point of view, Leibniz's theory is peculiar, since it analyses, e.g. the statement "Alexander is a king" in terms of the concept of Alexander containing the concept of king. Thus, is appears necessary that the historical person Alexander was indeed a king though we would rather see it as a contingent fact (for details, see [Zal00]).

is intensional" means that something belongs to the knowledge contents of a given concept.[9] This view can be criticised on the basis of the properties of the relation of intensional containment. We would like to maintain (based on Kauppi) that the relation is reflexive, transitive and antisymmetric, and this is the case if we equate intensional containment with IS-A relationships or necessary attributes (see Section 2.2). However, if intensional containment is to cover all of IS-A, part-of and has-attribute relationships, this seems unlikely. Let us suppose a modeller who has no background information about concept theory wants to model gardens and stones. A garden may have a stone as its parts (part-of relation), a stone may have an attribute "survives in totally dry places" (a has-attribute relation). If we only have the relation of intensional containment, **garden** intensionally contains **stone**, **stone** intensionally contains **survives-in-dryness**. But for a garden survives-in-dryness would not apply, so (i) either the relation cannot be transitive in a case like this or (ii) the relation of intensional containment cannot be always applied to cases where there are many different relations to be covered by it.[10]

- In artificial intelligence related studies of computer science, intensionality is usually seen from a different angle. The research is motivated by clarity of the background theory. In the background theory (and thus in the artificial intelligence applications as well), it must hold for instance that co-extensional concepts are not automatically identified with each other, and that belief worlds can be asserted (see Chapter 2). Possible world semantics and situation semantics (discussed in Chapter 7) have usually been employed as background theories.

  In possible world semantics, the interpretations of some expressions (individual constants, predicates and functions) are studied in the context of a collection of possible worlds. For instance, a predicate (say, *red*) maps to sets (of red objects) in each of the worlds. Now, the intension of a concept that corresponds to the predicate is the intersection of these sets in all the accessible worlds (see [BS79]). The theory gives an explanation (semantics) of intensional containment, in the case that it is identified with IS-A relationship.

- HIT semantics (Homogeneous Integrated Type-Oriented data model semantics) can be seen as a detailed extension of the possible worlds theory. The background of HIT semantics relies on Tichy's analysis of Frege's philosophy and, as a result, his Transparent Intensional Logic (TIL, see [Mat00]). Frege's distinction between Sinn and Bedeutung is apparent in his famous star example. There, the expressions "Morning star" and "Evening star" denote to the same object (Bedeutung – planet Venus) but they have different "modes of presentation" of it, that is, different Sinns. Though it is suspicious to equate Sinns with concepts, it is obvious that there is something between the expressions and their denotations. These can justly be called concepts. According to Materna in [Mat00], Frege disliked the idea that concepts could be denoted in a same way as (other) objects, but this idea is utilised in TIL, where many different kinds of objects (including intensions) can be denoted. In TIL, this means that we use constructions to reach objects.

  In HIT semantics, intensions are "empirical functions" (they actualise in an empirical manner in different worlds, unlike analytical functions) and intensional containment can be explained in terms of subconstructions (for details, see Chapter 7).

[9] Analyti et al. have applied this kind of an view by the name of "real world intension" in [ASCD99].

[10] In the latter case, we can still assume that part-of and has-attribute relations *as such* are reflexive, transitive and antisymmetric.

As we adopt the view "intensional = based on intensional containment", we can trivially define an intensional modelling language; it is a modelling language where intensional containment is utilised. On the other hand, an extensional modelling language uses terminology that refers directly to the objects, relations, attributes, etc., in the domain of application. With a hybrid language, one can postulate both intensional and extensional phenomena. It is likely that most conceptual modelling languages have hybrid properties. However, a pure intensional language (as presented in Chapter 4) is an intensional language that has no "direct" mechanism to refer to directly to the objects, their relations etc.[11] A pure extensional language has no mechanism to assert concepts, classes or relationships among concepts or classes (like IS-A).[12]

An intensional approach to conceptual modelling can be seen as utilising an intensional modelling language and methodology in the modelling process. This approach has its theoretical advantages (see Chapter 2), but in practical terms we can see the connection between an extensional and intensional approach as follows:

Let us suppose a modeller models a "new" domain of application with an extensional modelling language. It is likely that many objects (e.g. a new company with its clients, suppliers, etc.) asserted in the model do not exist yet.

Strictly, if we consider "a company that does not exist yet" and "clients of a company that does not exist yet" as sets, we notice that they are empty. Since these "concepts" have the same extension, they should be considered identical according to a strictly extensional view. Naturally, for the modeller, this would hardly mean that as concepts **a company that does not exist yet** and **clients of a company that does not exist yet** are identical, rather, he would think the company and its clients as possible objects. Another way of seeing this is that the intensions of **a company that does not exist yet** and **clients of a company that does not exist yet** are different. We maintain that it is meaningful to elaborate these intensions and that the intensional approach can give us tools for that.

1.4 The contents of the thesis

Several formalisms (modelling languages) have been developed for the purpose of expressing a conceptual schema. In Chapter 2, we roughly divide these languages in three categories; extensional, intensional and hybrid languages. An extensional modelling language uses terminology of (extensional) entities and relationships in the description of the domain of application. In an intensional language, concepts have a central role. Hybrid languages combine both intensional and extensional features.

We also present the semantic background for a language of each category. We study how suitable these languages are for the purposes of conceptual modelling: the method for that is to present a set of everyday conceptual modelling needs that can be expressed using a simplified natural language description. We inspect how the languages of each of the categories can meet these needs and what kinds of other features they offer in addition to that.

Among languages that we call extensional, first order predicate logic (FOPL) has a long tradition and a lot of the terminology used in this thesis (the distinction between syntax and semantics, issues of computability etc.) comes from that tradition. A logical system consists of a syntactic

[11] There is, of course, the "indirect" way, as J. Palomäki mentioned in a private conversation: the intension dominates the extension, i.e. using the Z relation, we can refer to objects. However, this does not implicate that our modelling language would employ this kind of mechanism.

[12] Here, we do not consider the possibility of using second order predicate logic where it is possible to quantify over predicates. If concepts are seen as predicates, this would enable us to state relationships among them.

proof theory and a semantical model theory. In the standard system of FOPL, the deductive power is rather weak ("semi-decidable"). Because of that, some more limited logic systems have been proposed, among them Horn-clauses and languages like Prolog and Datalog based on them (see [EN94]).[13] In Chapter 3, we discuss a prominent modelling language, Conceptual Graphs, which has been proposed as an alternative of FOPL. Though the background theory of conceptual graphs enables us to add intensional features to the formalism, it is natural to use a restricted version of the language as an example of an extensional modelling language. We demonstrate that this restricted version is equivalent to a restricted version of FOPL. We maintain, too, that no apparent advantage is gained using the conceptual graph formalism instead of FOPL.

In Chapters 4 and 5, we discuss a concept-theoretical (intensional) approach to conceptual modelling, and a language based on it. In the concept-theoretical approach discussed here, there are only two kinds of primitives in the language: concepts and the relation of intensional containment on a set of concepts. We present functional methods for expressing and analysing concept systems (axiomatic systems stating what kinds of relationships are possible among concepts) and outline a scheme to utilise concept systems in conceptual modelling.

CONCEPT D is a modelling language that has been inspired by the concept-theoretical approach (see e.g. [Kan83]). Using CONCEPT D, the modeller expresses the concepts that are relevant to the domain of application (Universe of Discourse in the COMIC terminology), and the relationships between these concepts. Most of the relationships are considered to be variants of the relation of intensional containment. This approach is most economical, but it may present problems concerning the semantics of the relations between concepts. We approach the problem from two different angles: in Chapter 6 by comparing CONCEPT D with another modelling language and in Chapter 7 by constructing a semantics of the structures presented in CONCEPT D using a semantic background theory. Both of these approaches enable us to use some of the tools developed in Chapters 4 and 5 in combination with the established semantics.

In many other modelling languages, the basis of the modelling is the extensions of some concepts in the domain of application and various relationships between these extensions (sets). This starting point has its limitations, but it provides clear semantics for the modelling language. In Chapter 6, we discuss a mapping between a limited version of CONCEPT D and a well-known and semantically well founded modelling language, IFO (see [AH87]).

In Chapter 7, we discuss several approaches to concept research, semantics and their relevance to conceptual modelling. The approaches include the theory of possible worlds, situation semantics, theories of predication and transparent intensional logic. The HIT data model, based on transparent intensional logic, is discussed as well, and it is applied as the semantics of a limited CONCEPT D language.

1.5 Related research

Previous research to be mentioned in this context concerns semantics, data models, modelling languages and intensional features in conceptual modelling. Duzi in [Duz00b] discusses the criteria for conceptual modelling languages and explicitly mentions expressibility, clarity, semantic stability, semantic relevance, validation mechanisms, abstraction mechanisms and formal foundation. Hausser's work on computational linguistics in [Hau01a] contains an extensive study of semantics in the context of natural languages and the possibilities of human-computer communication. Data

[13]In addition to limited logics, there are logic systems that limit the syntax of FOPL in some ways and expand it in some other ways. One of the best known examples of these is KIF [GF92].

models in general are discussed by Elmasri and Navathe in [EN94]. The Entity-Relationship formalism (The ER Model, or simply ER), presented by Chen in [Che76] is an influential semantic data model that popularised many notions (entities, attributes, relationships,..) that are still commonly used.[14] Several extensions to ER were suggested; they include e.g. Extended Entity Relationship formalism (EER), discussed in [BCN92], where subclass-superclass relationships can be expressed; and various temporal extensions (see e.g. [Tau91]). Though famous, ER is not the only semantic data model. Other merited ones include IFO (see [AH87]), whose semantics is based on database updates and is thus very well defined; SDM (see [HM81]) for its relative "semantical relativism", where it is not very important if something is a relationship or an attribute[15]; NIAM (see [VvB82]), and more recently its successor Object Role Modelling (ORM) [NH89]. In Chapter 2, we further discuss modelling using an ER-like formalism and compare it to other kinds of formalisms as well.

Alongside semantic data models, knowledge representation languages have gained ground since the 1980's. According to Borgida in [Bor91], knowledge representation languages (such as KRL [BW77] and later KL-ONE [BS85]) have similarities with SDMs; both of them aim at the construction and use of a database/knowledge base. However, the motivation for their use is different, the users of SDMs are humans; software developers/maintainers or end-users. The normal user of a knowledge representation system is a program, which tries to perform some task by using the knowledge base as a "server". The program normally carries out some inferences based on FOPL (or a limited version of it) and much of the post-1980's research of knowledge representation languages has concentrated on the computability of such languages. Papers by Donini et al. ([DLNN97], [DLN$^+$92]) are good examples of this research. These languages have become known as "concept languages" or description logics (DLs), and are further discussed in Chapter 2.

Given the similarities of SDMs and knowledge representation languages, many languages have features of both of them. Sowa's Conceptual Graphs [Sow84] is a rich formalism, where concepts form a subconcept-superconcept hierarchy (a lattice). However, Sowa makes a difference between reasoning about the hierarchy (intensional features) and things expressed by Conceptual Graphs themselves (extensional features).[16] Conceptual graphs and their relationship to FOPL are considered in Chapter 3.

Formal ontologies (see e.g. [Gua97], [GG95]) are another area of computer science where concepts and their relationships are discussed. We observe much similarity between the intensional approach (especially of the kind in Chapters 4 and 5) of conceptual modelling and the terminology in formal ontologies.[17]

The perspective of intensionality in conceptual modelling has been discussed in many occasions, but probably [Kan92] is the most precise in formulating the view of intensionality as the knowledge contents of concepts. Much of the related work has been done in the University of Tampere, including studies by Junkkari (see [Jun98]) and Niemi (see [Nie00]). Moreover, Berztiss in [Ber99] discusses intensional conceptual modelling in a more general sense; Motro in [Mot94] and Falquet et al. in [FLS94] apply some intensional features to databases; and in [NP98] Nilsson and Palomäki define a logic-based language to compute intensions and extensions. This is,

[14]For instance, the popular "Unified Modelling Language", UML, employs some ER conventions. Here, we do not cover UML extensively due to its emphasis of software engineering. For details, see e.g. [JBR99].

[15]Hammer and McLeod in [HM81] use the term "relative viewpoint". It should be noticed that SDM here is a proper name and does not refer to semantic data models in general.

[16]The same kind of distinction has been applied to many description logics, including KL-ONE's [BS85] distinction of T-Box (concept definitions) and A-Box ("rules" and other extensional statements).

[17]Some of the similarities of the intensional approach and formal ontologies were pointed out by H. Kangassalo in a private conversation.

however, different from our approach in Chapters 4 and 5, where an abstract implementation of managing concept structures developed on the basis of the intensional containment relation.

COMIC methodology with its CONCEPT D modelling language emphasises the intensional perspective and semantical relativism. Instead of having separate notions of entities, relationships, attributes or roles like in ER, there are only concepts. COMIC and CONCEPT D are introduced in [Kan83], [KV90], [Kan92] and [Kan93].

Several objections to COMIC methodology and CONCEPT D can been raised, like those of Duzi in [Duz00c]. These objections can concern both the philosophical background of COMIC (notions of concepts and intensions) and the semantics of concepts diagrams (externalised conceptual schemata expressed with CONCEPT D). The philosophical background is analysed in Chapter 6. Both Chapters 6 and 7 address the question of semantics, too. However, the focus of Chapter 7 is to analyse the semantic theories that would serve as a background for conceptual modelling. This kind of survey is rather unusual, but inspired by [HLvR96], [Duz01] and [Hau01b].

1.6 The purpose and the results

The purpose of this thesis is to:

- Construct a feasible categorisation of different kinds of modelling approaches and languages and evaluate them based on that;
- Evaluate and study prominent modelling languages in each of the categories. Especially evaluate the pros and cons of Conceptual Graphs, a language proposed as an alternative to first order predicate logic in knowledge representation and conceptual modelling;
- In the category of intensional languages, analyse and demonstrate the possibilities of a language based on a purely intensional description of the domain of application;
- Emphasise simplicity and well-defined semantics as virtues of any modelling approach.

The main results are:

- Creating a general framework and clarifying the terminology of modelling languages by means of the categorisation (Chapter 2);
- Among extensional languages, comparing first order predicate logic and conceptual graphs and pointing out the limitations of Conceptual Graphs in practice;
- Constructing an abstract implementation of a modelling language that is based purely on concept theory, and its theoretical background. On this theoretical basis, however, it appears that only quite limited (and strict) conceptual schemata can be created using the language, and it may be difficult to relate the language to everyday modelling tasks. These difficulties are related to semantics and the following items.
- Comparing CONCEPT D with a semantically well-defined conventional modelling language, IFO, in order to clarify its semantics;
- Based on the above, proposing a variant of CONCEPT D so that the semantic properties of CONCEPT D diagrams could be better understood.
- Studying different semantics as background theory and clarifying their positions to conceptual modelling.

1.7 Terminology and conventions

In this thesis, a *set* is loosely defined as follows: "By a set we mean any collection of entities of any sort" and that the "members of a set [..] belong to the set" [Sup57]. Following Suppes in [Sup57] we say, too, that "x belongs to set A" is denoted by $x \in A$, and "x does not belong to A" by $x \notin A$. The *empty set*, denoted by $\{\}$, is the set such that for every x, $x \notin \{\}$.

The members of a set are also often called the *elements* of the set.

In order to give a set a precise definition, and to avoid the so-called Russell's paradox, axiomatic set theory can be used.[18]

By convention, in computer science, *classes* are collections of objects of the same type. In mathematics, however, classes are collections that can be formed arbitrarily (thus, sets are classes). Logical notions like "predicate", "quantifier", "L-model" etc., are presented in Chapter 3, and some mathematical notions in Chapter 4. Other notions related to theory of computation and set theory are discussed below.

Two sets, A and B are identical (denoted by $A = B$) if and only if they have the same members. If A and B are sets such that every member of A is also a member of B, then we call A a subset of B [Sup57], denoted by $A \subseteq B$. If A and B are sets, then by the *union* of A and B (in symbols $A \cup B$) we mean the set of all things which belong to at least one of the sets A and B [Sup57]. Likewise, if A and B are sets, then by the *intersection* of A and B (in symbols $A \cap B$) we mean the set of all things which belong to both A and B. If A and B are two sets, then by the *difference* of A and B (in symbols $A - B$) we mean the set of all things which belong to A but not B.

Given a set A, the power set of A, denoted by $P(A)$ is the set of all subsets of A.

If x and y are two objects, we can connect them together explicitly by the ordered pair $\langle x, y \rangle$.

We define ordered n-tuples as follows (see [Sup57]):

$$\langle x_1, x_2, .., x_n \rangle = \langle \langle x_1, x_2, .., x_{n-1} \rangle, x_n \rangle.$$

The *Cartesian product* of two set A and B (in symbols $A \times B$) is the set of all ordered pairs $\langle x, y \rangle$ such that $x \in A$ and $y \in B$ (see [Sup57]).

If A and B are sets, then any subset of $A \times B$ is a *relation* from A to B. If the ordered pair $\langle x, y \rangle$ is a member of R, we use notations $\langle x, y \rangle \in R$ or xRy, alternatively. A two-placed relation (like R above) is also called a binary relation.

If R is a binary relation, then the *domain* of R is the set of all things x such that, for some y, $\langle x, y \rangle \in R$. The *counterdomain* of R is the set of all things such that, for some x, $\langle x, y \rangle \in R$ (see

[18]For conciseness, we quote the axioms of set theory, as presented in [Jec78]:

- I Axiom of Extensionality: if X and Y have the same elements, then $X = Y$.
- II Axiom of pairing: for a and b there exists a set $\{a, b\}$ that contains exactly a and b.
- III Axiom schema of separation: If ϕ is a property with parameter p, then for X and p there exists a set $Y = \{u \in X : \phi(u, p)\}$ that contains all those $u \in X$ that have the property ϕ.
- IV Axiom of Union: for any X there exists a set $Y = \cup X$, the union of all elements of X.
- V Axiom of power set: for any X there exists a set $Y = P(X)$, the set of all subsets of X.
- VI Axiom of infinity: there exists an infinite set.
- VII Axiom of schema replacement: If F is a function, then for any X there exists a set $Y = F[X] = \{F(x) : x \in X\}$.
- VIII Axiom of regularity: Every nonempty set has an $\in$ -minimal element.

[Sup57]).

Following Suppes in [Sup57], we define the notions of reflexivity, antisymmetricity, transitivity and partial ordering as follows.

A binary relation R is reflexive in the set A if for every $x \in A$, xRx.

A relation R is antisymmetric in the set A if for every x and y in A, whenever xRy and yRx, then $x = y$.

A relation R is transitive in the set A if for every x, y and z in A, whenever xRy and yRz, then xRz.

A relation R is a partial ordering of the set A if and only if R is reflexive, antisymmetric, and transitive in A.

A *function* R is a binary relation such that if xRy and xRz then $y = z$ (see [Sup57]).

By definition, a function from A to B is a relation from A to B. Let f be a function. If $a \in A$, then the member $b \in B$ such that afb is called the *value* of f at a, and is designated by $f(a)$ (definition adapted from [Lip76]). The set of all such values is called the *image* of f and is denoted by $Im(f)$ (see [Lip76]).

We denote function f from A to B as $f : A \longrightarrow B$. There, we call A the domain of the function and B its *range*. $f : A \longrightarrow B$ is often called the *signature* of the function. Now, $Im(f)$ is a subset of B.

If $b = f(a)$, we call b the *result* of f for *argument* a. We say, too, that the function *returns* b for argument a.

Informally, the *cardinal* (cardinality) of a set is the number of its members. We define this notion more precisely following Zhongwan in [Zho98]. First, two sets S and T are said to be equipotent, written as $S \sim T$ if and only if there is a one-one function from S to T. Now, a cardinal of a set S, denoted by $|S|$, is associated with S in such a way that $|S| = |T|$ if and only if $S \sim T$. A finite set S is equipotent to $\{0, .., n-1\}$ for some natural number n.

Informally, an algorithm is a sequence of elementary operations (steps) required to carry out a computational task. The *complexity* of an algorithm is relative to the size of its input. Intuitively, if the size of the input is n and the algorithm computes the output using n^k elementary operations, the complexity of the algorithm is *polynomial* ("tractable"). If the number of operations needed is 2^n, for example, the complexity is *exponential*.[19]

A set is *decidable* if there is an algorithm for determining for any x if x is a member of the set. In general, a *decision problem* is a problem with a "yes" or "no" answer. A famous complexity class of decision problems is called NP (Nondeterministic Polynomial), for which answers can be checked by an algorithm whose "run time" (number of steps) is polynomial to the size of the input.[20]

A *language* is a set of sequences of alphabet (strings) that is composed of *terminal alphabet* using the *production rules* of a grammar. We say, too, that the grammar defines the *syntax* of the language. Let A be a finite set of alphabet. We denote by A^* (reflexive transitive closure of A) all the strings (sequences of alphabet) that can be constructed from A. If s is atring and $s \in A^*$, we say that s is an *expression* of the language.

Formally, a grammar is a 4-tuple $\langle T, N, R, S \rangle$. There, T is a finite nonempty set called the terminal alphabet. The members of T are called terminals or terminal symbols. N is a finite nonempty set disjoint from T. The members of N are called the nonterminals or auxiliary symbols.

[19] A proper definition of complexity would require a lengthy discussion of Turing machines. For details, see [Ata99].

[20] According to [Joh90], the class "NP" is defined to be the set of all decision problems solvable by NDTMs in polynomial-bounded time. There, NDTM (Nondeterministic Turing Machine) is a Turing machine which at each step (of calculation) has several choices as to its next move.

R is a finite set of productions (see below) and $S \in T$ is a distinguished nonterminal called the start symbol (or starting symbol).[21]

R consists of 2-tuples (ordered pairs) $\langle a, b \rangle$, where a is a string of terminals and nonterminals containing at least one nonterminal and b is a string of terminals and nonterminals. A popular way of presenting these ordered pairs is to omit the angle brackets and insert a production symbol ($\mapsto$) between them, i.e. $a \mapsto b$.

In Chapter 3, we use a very traditional form of a grammar, where all the auxiliary symbols used are represented by a single alphabet and not a string. In Chapter 2, to conform with Lambrix's (see [Lam96]) conventions, we apply a form that is more often used in computer science. There, "::=" is used as the production symbol, auxiliary symbols are strings surrounded by "<>", and "+" is used as a symbol of repeating the previous symbol 1,..,n times.

In Section 6, we apply some basic terminology of graph theory. Informally, a graph is a finite set of dots called *nodes* (or vertices) connected by links called *edges* (or arcs). An edge connecting node A to node B can easily be presented as an ordered pair $\langle A, B \rangle$. If the direction of the edge is important, we talk about *directed edges*, otherwise undirected edges. A graph with directed edges is called a directed graph.

Horn clauses are defined as follows (please see Chapter 3 for the definition of atomic formulas):

A literal is either an atomic formula or an atomic formula preceded by a negation symbol ($\neg$). A literal preceded with $\neg$ is called a negative literal. Otherwise, it is called a positive literal. Clausals are positive or negative literals, connected with $\vee$-connectives. A sequence of clausals, connected with $\wedge$-connectives, form the clausal form of the formula. A Horn clause is a clausal form that has maximally one positive literal.

When we use language L to describe something specific, the result is a subset of L, since it uses a specific set of the terminal alphabet. However, it is more natural to call the result an expression set.

In modelling, the notion of something being or not being *printable* (representable by a string) often occurs. We use the following definition: Let A be a finite set of alphabet. A class is printable if there is a recursive injection[22] from its members to A^*.[23] If something is not printable, we call it *abstract*. In the NIAM tradition, printable types are called lexical object types (LOTs) and abstract types are called nonlexical object types (NOLOTs) [NH89].

Semantics connects some of the strings with *meanings*. We identify these meanings with concepts, though there can be concepts that have no linguistic expression (a string) related with them. However, these concepts are not accessible to us until we find a way (a string) to refer to them. If there is a semantic connection between a string and a concept, we call the string a *name of the concept*.

A *modelling language* or a *formalism* is a well-defined technique (language with possibly some guidelines of how to apply the language, and often some form of graphical visualisation) used in modelling for expressing something about the domain of application. A modelling language necessarily has a syntax, but we assume that semantics can be defined for the language, too, thus allowing us to talk about "the semantics of modelling language X". In the case of modelling languages, we often call the terminal alphabet *language primitives*.

A *conceptual model* is on the level of concepts, but an *externalised conceptual schema* is a linguistic level object, an expression set, created using a modelling language.

[21] The definition of grammar has been adopted from [Ata99] with minor changes in the terminology and the naming of sets T, N and R.

[22] A function f is an injection if and only if $f(x) = f(y)$ implies that $x = y$ for all x and y in the domain of f.

[23] "Printable type" is defined accordingly, since types are sets (and thus classes) given by a predicate.

A concept has an intension and an extension, linguistic expressions (strings) have meanings and references. We use the expression "*contents* of a concept" instead of "content of a concept".

"Domain of application" refers to the object of modelling. "Universe of Discourse" is reserved for a more technical use.

Similarly, "a relationship" is a generic non-technical name (like "relationships between objects". "Relation" is reserved for more technical use in mathematics, concept theory ("the relation of intensional containment"), etc.

A database is "a structured collection of data held in computer" [SW98].

According to [Yov93], "information is data which is used in decision-making". Here, we consider the notions of information and knowledge to be largely synonymous.

In this thesis, conceptual entities are presented in **bold typeface** (unless otherwise indicated) and linguistic entities are written in *italics*. For emphasis, too, we use *italics*.

A short arrow ($\rightarrow$) is used for the sign of logical implication. In the signatures of functions, for instance $f_s : \mathbb{N} \longrightarrow \mathbb{N}$, a long arrow is used.

Chapter 2

Intensional and Extensional Modelling Languages

2.1 Introduction

Data models are abstractions that hide the technical details of storing data. Generally, there are three different types of data models: high level, implementational and physical data models [EN94]. High level data models are close to the way the users see the information. In information system design, high level data models have gained more and more attention as the applications get more complex (see [Lan78]). It can be said that with high level data models, there is a trend towards a conceptual description of the domain of application (apparent, for example, in [TCI87], [Che76], [HK87]).

In conceptual modelling, the role of *intensionality* has been emphasised in e.g. [Woo91] and [Kan96]. In philosophy, intensionality is understood as a distinction between denotation and meaning; that is, the word “stone” denotes a concrete stone, but means the concept of **stone** (see [Bla96]). Yet, it is unclear what exactly is meant by intensionality in conceptual modelling and how this intensionality can be reflected in conceptual modelling languages. We clarify this by (i) considering what kind of semantics can be called intensional and (ii) classifying the modelling languages into three different categories: extensional, intensional and hybrid. An extensional language uses terminology of extensional entities (things in the domain of application) and relationships among them. In an intensional language, concepts have a central role and the language has means of expressing relationships between concepts. Hybrid languages combine both intensional and extensional features.[1] Each of the categories presents a different approach to conceptual modelling. The following features can be seen as representative among them:

- In Section 2.3, we present an extensional language, influenced by the Entity-Relationship Model (ER, [Che76]). Extensionality is reflected in the terminology of the language (“sets of objects”, “relations”), and its simple semantics. The semantics is based on the idea that sets of objects actually consist of objects in the domain of application. The relationships are subsets of Cartesian products of the sets of objects, so eventually the relationships also accommodate objects of the domain of application. While this kind of semantics is very easy to understand, it does not provide a natural way of describing why some relationships are intensional in nature. For instance, there is no way to make a difference in notation (or semantics) between the WINE-TASTING-SOCIETY - SOCIETY relationship and the “buys”

[1]For a proper definition of the basis of the classification, see Section 2.2.

relationship (between WINE-TASTING-SOCIETY and WINE-BOTTLE) of our example in Figure 2.1.

- Many papers by Kangassalo, especially [Kan93] and [Kan83], promote an interesting idea of conceptual modelling based almost entirely on intensional relationships (mainly intensional containment) between concepts. In this approach, concepts represent anything that is of interest in the domain of application; for instance an entity, a relationship, an attribute, or a process.

  Since concepts are relatively independent of the domain of application, this approach can avoid some problems of (too) simple extensional semantics. For instance, we can talk about concepts of **female president of the US in the 20th century** and **ghost** without assuming that they are identical, though they are co-extensional (in the actual world).

  While the extensional approach is given an explicit form in Section 2.3, the intensional approach is explained in Chapters 4 and 5. There, we only discuss a formalism based on concepts and the relation of intensional containment in the set of concepts. The relation is not given any specific interpretation, but in the examples we consider it similar to IS-A relationship. However, in [Kan93] the relation of intensional containment is applied to cover several kinds of relationships that are in other approaches considered to be semantically different from each other, like IS-A and has-attribute.

- Since KL-ONE (see [BS85]) and Classic (see [B$^+$89]), Description Logics (DLs) have played an important role in knowledge representation. In DLs, intensional relations are represented as a taxonomy using a so-called concept language as a representation method [DLNN97]. A concept language is in fact a limited variant of the First Order Predicate Logic.[2] In a typical DL, there are only two types of expressions: subsumption expressions and role expressions. A subsumption expression states a subconcept-superconcept relationship in the set of concepts. A role expression states that some attributes are linked with some concepts; or that some concepts must match other concept's attributes. Using these means, a concept language can be used to describe concepts and to derive new concepts from the existing ones. The core of a concept language system is a classifier that organises concepts into a hierarchy according to their specificity (see [DLNN97]). A concept hierarchy is a partial ordering of the set of concepts, based on the subconcept-superconcept relationship (see [HT00]). The more expressive the concept language is, the more difficult this classification gets. Much work has been devoted to the analysis of the complexity of different concept languages as in [DLNN97], [DLN$^+$92] and [Gre91].

In this chapter, we discuss each of these approaches using a simple modelling example. We evaluate the approaches from the perspective of conceptual modelling. To do so, we present some general notions about conceptual modelling, the example and some basic terminology in Section 2.2. In Section 2.3, we describe an extensional modelling language and present the example using it. In Section 2.4 we discuss concept systems and an intensional modelling language based on them. "Hybrid languages" that combine both intensional and extensional features are discussed in Section 2.5. A discussion and a summary can be found in Section 2.6.

[2]In essence, according to Donini et al. [DLN$^+$92], the First Order Predicate Logic language that corresponds with a concept language has only unary and binary predicates, and each concept language expression can be translated into a predicate logic formula which has one free variable.

2.2 Conceptual modelling and modelling languages

In the process of conceptual modelling, the modeller creates a conceptual description of the object to be modelled and expresses it in some formalism, a modelling language (see [Kan83]). We call the result of the process an externalised conceptual schema. In order for the schema to be understandable, the language in question should be suited both to the needs of the domain of application and the user. Moreover, the schema should meet some technical requirements, like illustrativeness and unambiguity (see [TCI87], [Mar97]). As discussed for example in [LB85] and [Nii98], it is fruitful to consider the expressiveness and efficiency of the languages as well: a language is useless for the purpose if one cannot express enough with it, or if it is so expressive that its use becomes computably too demanding.

In this chapter, we discuss three semantically different types of languages that can be used to express the externalised conceptual schema. We use a simple example in order to compare features of these languages. We see *extensional* modelling languages as a semantically common sense approach to modelling problems. Though sufficient for many everyday needs, the extensional semantics of the languages present some theoretical difficulties. These can be summarised as follows:

- In simple extensional semantics, the notion of meaning can only be defined in terms of references. Therefore, language expressions that have the same referent automatically have the same meaning. This is often expressed by saying "co-extensional concepts are identical". This applies to empty referents as well. Thus, semantically, no distinction can be made between the expressions **female president of the US in the 20th century** and **ghost**, since they have an empty extension in the actual world.

- The "belief worlds" are missing; let us suppose John just moved into the flat where Mary lives and Sarah does not know that. Sarah knows Mary's phone number, so against our intuition we should deduce that Sarah knows John's phone number as well.

To get over these difficulties, we should prefer semantics where the expressions and referents are not bound together monolithically. This kind of semantics would establish the meaning of the expression independently of the referent. We call this meaning a concept, and this kind of semantics *intensional*.

It is out of the scope of this chapter to discuss actual semantic theories. However, the features listed above are used as the background theory of our classification of modelling languages, as follows:

- If the semantics of the language forces us to identify meanings with referents, we call the language extensional;

- If the semantics of the language explicitly postulates concepts, and the relationship of intensional containment, we call the language intensional;

- If the language is intensional, but allows us to refer to "things in the world" in the domain of application, we call it a hybrid language.

In this chapter, we employ the following terminology, partially adopted from [Jun01]. Here, we make a difference between the representation language level, where there are strings composed of terminal alphabet using production rules of a grammar; object level, where there are things in the domain of application; and conceptual level, where there are concepts. Abstraction methods (like grouping, aggregation and taxonomy) are needed to create the structure (structural relationships) for our representation of the domain of application.

- In general, everything at the representation language level is called an *L-item*.
- In the domain of application, there are *objects*. We do not assume that an object has any metaphysical characteristics, but only that an object is uniquely distinguishable from other objects.[3]
- Objects have *properties* that are intrinsic to them. Properties that describe functional capabilities of objects are called *behavioural properties*.
- Some objects are *printable*, i.e. representable by an alphanumeric string. Objects that are not printable are called *abstract*.
- Similar objects share an *object type*. Collections of objects of the same type are called *classes*.[4] At the conceptual level, there are *concepts*. Some classes are *extensions* of some concepts. At the representation language level, *L-entities* are L-items that represent classes or concepts. These we refer to with capital letter strings, like STUDENT.
- In the domain of application there are *relationships* between entities. Each relationship can be considered a tuple of entities. In the process of modelling, we are interested in presenting *relationship types* that we can name, and where we state what are the classes of elements in this relationship type. For instance, STUDIES-AT-DEPARTMENT is a (binary) relationship type between STUDENTs and DEPARTMENTs. At the representation language level we call these relationship types *L-links*. In what follows, we present structural relationships indicated by abstraction methods. For this reason, we reserve the term "L-link" to be used for non-structural relationships, i.e. domain specific relationships, like STUDIES-AT-DEPARTMENT.
- There is an abstraction method of *grouping*, sometimes called iteration, by which we can form member-of relationships. By grouping, we form finite sets whose members belong to a certain class. For instance, a STUDENTS-ATTENDING-A-COURSE has members all of which are STUDENTs. Thus, STUDENTS-ATTENDING-A-COURSE is a grouping of STUDENTs. At the representation language level, this abstraction method is presented as *L-grouping*.
- There is an abstraction method of *aggregation* by which properties are introduced. Using aggregation, the modeller can present L-entities as compounds of other L-items that are called the *attributes* of the entity. For instance, a STUDENT can be modelled consisting of STUDENT-NUMBER, NAME, SEX and AGE; a MOTOR-BOAT consisting of MOTOR and HULL. At the representation language level, this abstraction method is presented as *L-aggregation* and the attributes are called *L-attributes*. L-attributes can generally be referred to by their names in the aggreagate, like MOTOR-BOAT's HULL.

[3]We may consider "tropic realism" as a plausible background theory of objects. This, however, is outside the scope of our research. For details, see Niiniluoto in [Nii99].

[4]For discussion of sets and classes, see Section 1.7.

- Some attributes are *necessary*, in a way that the entity could not exist without them. Other attributes are contingent.
- *Part-of-aggregation* is a special case of aggregation.[5] At the representation language level, we call this *L-part-of-aggregation* and these kind of attributes are called *L-part-of-attributes*.
- Some of the attributes are *identifying*: they unambiguously identify a particular entity. Their value sets are printable scalars and we call them identifier value sets. A member of that set is called an identifier and, at the representation language level, *L-identifier*.
- There is an abstraction method of *taxonomy* for expressing subclass-superclass relationships, by which we express IS-A relationships. At the representation language level they are called *L-isa-links*. In many modelling methodologies (see e.g. [AH87]), IS-A comes in the form of either specialisation or generalisation: an L-entity can be depicted as a generalisation or specialisation of another. If these variants of IS-A differ in the level of language, they will be called L-specialisation-links and L-generalisation-link. Here, however, we only use L-isa-links in general.
- There is a method of generating derived information: we can define functions that operate on entities, attributes or relationships and define their values. Traditionally, some modelling languages provide support for this only by derived attributes (see e.g. [Che76]).
- There are methods of expressing *constraints*, that reflect the constraints of known to apply to the domain of application. For instance, we know that person's age never decreases. Here, we consider constraints that can be expressed using relationships. On one hand, a relationship can directly state a constraint (e.g. "the EXPENSES of a DEPARTMENT may not exceed the BUDGET of the DEPARTMENT"). On the other hand, a constraint can be an *cardinality constraint*, (e.g. "a STUDENT must have exactly one MAJOR").

If a simple natural language based depiction of the domain of application uses the above terminology, we call it a *simplified natural language description*. As an example of this, let us consider the following depiction of the yearly activities of a wine tasting society, as in Figure 2.1. There, the wine tasting society in question is a society; it is a group of persons, and has a budget. A person has a name, an id number, an address and a phone number. A wine tasting society buys wine in wine bottles, but it cannot spend more than its budget for buying wine. A society has a name and a society id. An address consists of a street address, a zip code and a city name. A wine bottle has a name, a year, a type (one of red, white or rosé) a label and a price. A label has text and an image.

2.3 A simple extensional modelling language

In this section, we represent the example of Figure 2.1 using a formalism that has been influenced by the ER model, introduced in [Che76].

There are two basic L-items, *named sets of objects* and *named relations* between named sets of objects. Named sets of objects are L-entities and named relations cover L-aggregation, L-part-of-aggregation, L-links and L-isa-links.

[5]It is difficult to define precisely which attributes are "part-of" -attributes and which ones are not. Winston et al [WCH87] simply call "attributes" those properties of which are not intuitively "part-of". For instance, in their discussion, the height of a tower is an attribute but not a part.

Attributes of a SOCIETY: name (necessary), society-id (necessary, identifying).
Relationship between SOCIETY and WINE-TASTING-SOCIETY: WINE-TASTING-SOCIETY is a SOCIETY.
Attributes of WINE-TASTING-SOCIETY: budget (necessary).
Relationship between WINE-TASTING-SOCIETY and PERSON: WINE-TASTING-SOCIETY built out of PERSONS by grouping.
Attributes of PERSON: name (necessary), membership-id (necessary, identifying), phone number (not necessary), address (necessary).
Attributes of address: Street address (necessary), zip code (necessary), city name (necessary).
Relationship between WINE-TASTING-SOCIETY and WINE-BOTTLE:
WINE-TASTING-SOCIETY buys WINE-BOTTLEs.
Attributes of WINE-BOTTLE: name (necessary), year (necessary), type (necessary, one of red, white, rosé), price (necessary).
WINE-BOTTLEs have LABELs as a parts of them. Relationship between WINE-BOTTLE and LABEL: LABEL is a part of WINE-BOTTLE.
Attributes of LABEL: text, image.
There is a cumulative price CUMULATIVE-PRICE of all the prices of the wine-bottles. It is calculated as: the sum of WINE-BOTTLEs' attribute price.
CUMULATIVE-PRICE must be less than the budget of WINE-TASTING-SOCIETY.

Figure 2.1: Example: a simplified natural language description.

A named set of objects can be printable like ZIP (a set of zip codes) or abstract, like PERSON (a set of persons).

Named relations persist among two or more named sets of objects. Given n named sets $N_1..N_n$, a named relation is a subset of the Cartesian product $N_1 \times .. \times N_n$. For instance, an ADDRESS is a named relation of STREET-ADDRESS $\times$ ZIP $\times$ CITY_NAME.

In order to keep the language simple and consistent, it is reasonable to allow named relations only between named sets (and not between a named set and a named relation, for example). However, we introduce the usage of a named relation as an abstraction mechanism (Hull and King would call this a type constructor, see [HK87]). This is actually a function $f(N_1,..,N_n) \longrightarrow N$, where N, N_i are named sets, and whose name (f) is the same as the respective named relation.

The graphical conventions for the language are presented in Figure 2.2 and the example in Figure 2.4. In the syntax in Figure 2.3, the following things should be noticed:

- An expression is either a named set (with a type), a named relation, or a named relation as abstraction;
- "with a type" means that it is indicated with a letter (A/P) if the named set is abstract or printable;
- A named relation is just a list of two or more named sets with a type,
- A named relation as abstraction is represented by a named relation, an arrow $\longrightarrow$, and a named set with a type.

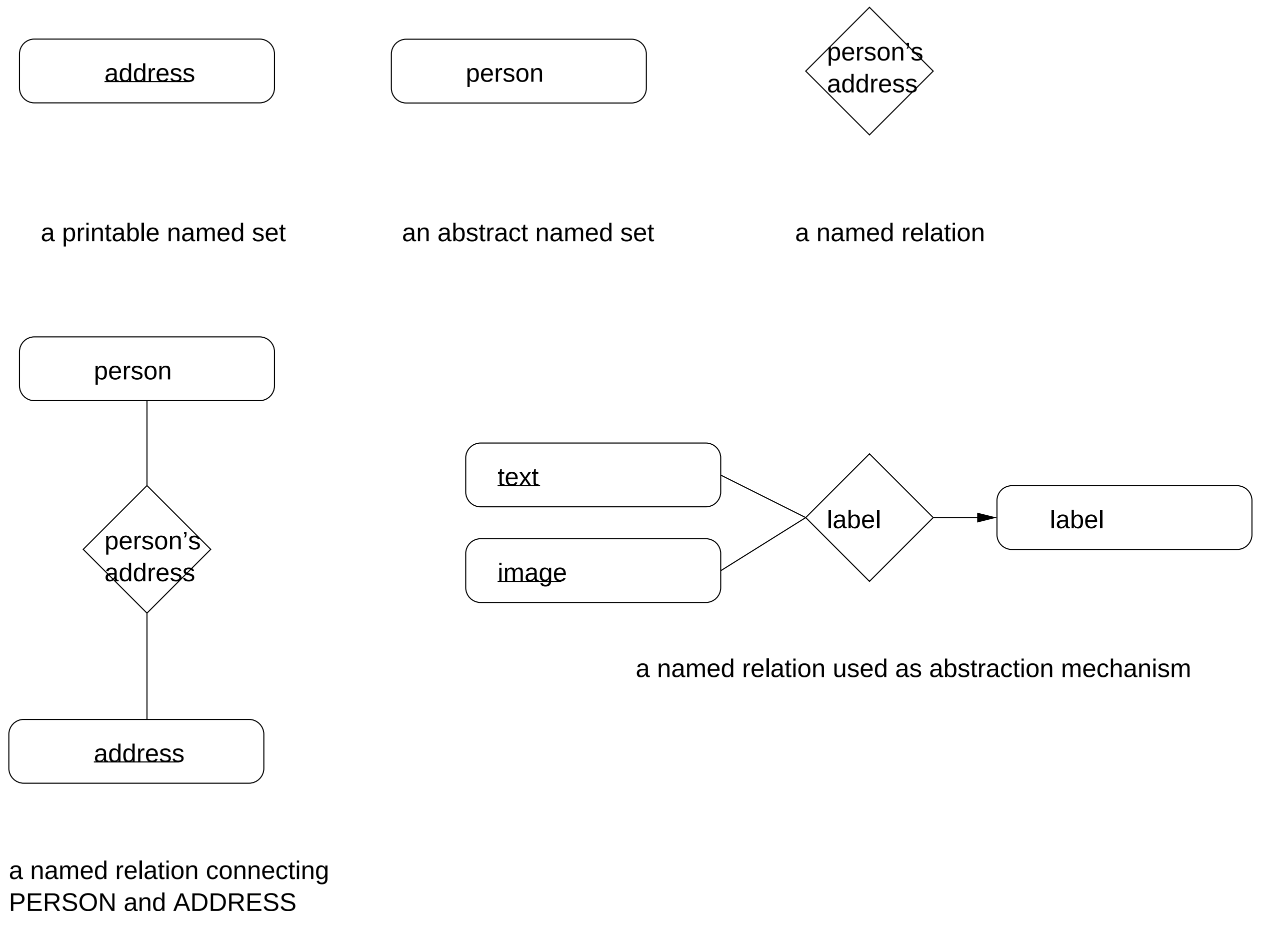

Figure 2.2: The language items of a simple extensional language

$< expression >::=$
$\quad < named-set-with-type > | < named-relation > | < named-relation-as-abstraction >$
$< named-set-with-type >::=$
$\quad < named-set > A | < named-set > P$
$< named-relation >::=$
$\quad < named-set-with-type >< named-set-with-type >^{+}$
$< named-relation-as-abstraction >::=$
$\quad < named-relation > \longrightarrow < named-set-with-type >$

Figure 2.3: The syntax of a simple extensional language

In the spirit of ER, L-grouping, constraints, distinction between necessary/contingent attributes, and derived attributes are not included in the language, though it would be possible.

We can observe that this representation language is appropriate for the purposes of the example. There are, however, some cumbersome details:

- A statement like “a wine tasting society is a society” requires a repetition of society-id and society-name in both society and wine tasting society;
- It cannot be expressed that wine tasting society members are a grouping of persons;
- It cannot be expressed that there is a difference between part-of and other kinds of named

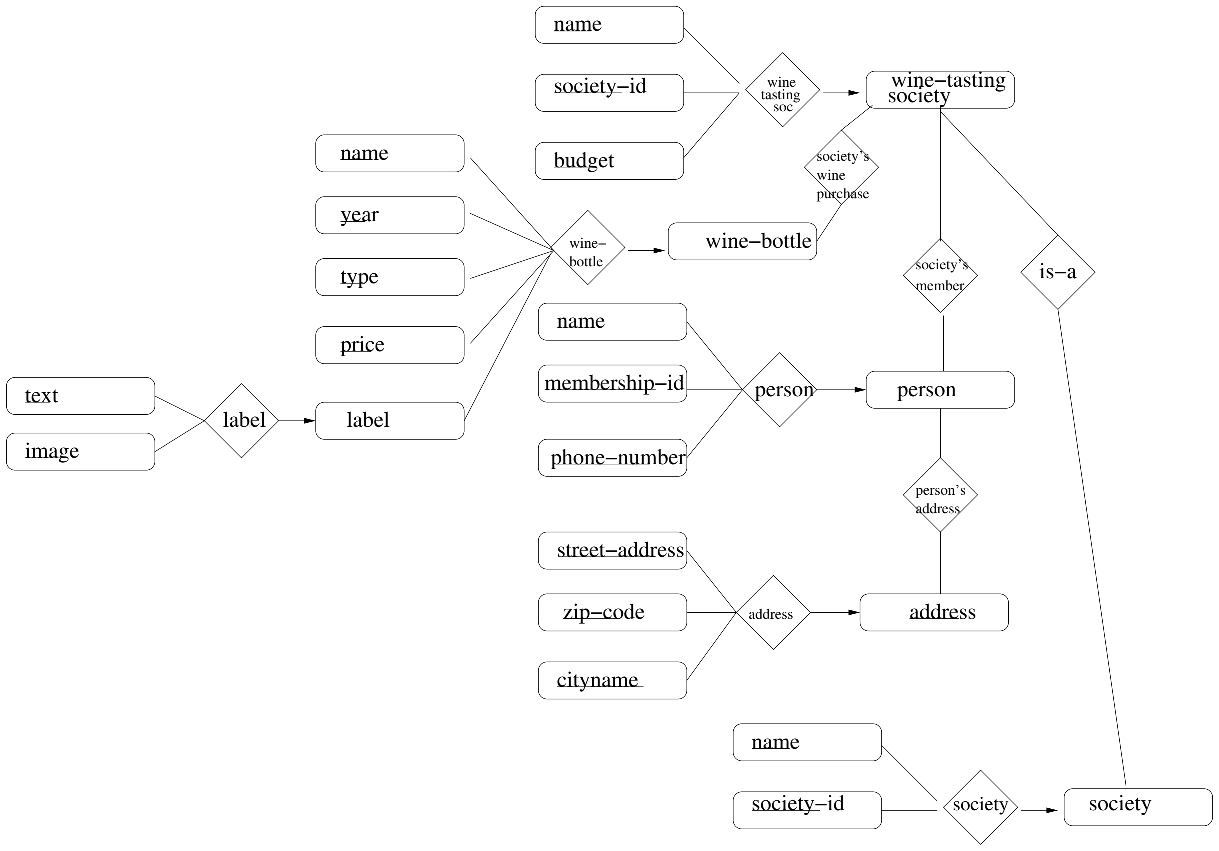

Figure 2.4: The natural language description represented by using an extensional language.

relations;

- The cumulative sum of wine bought by the wine tasting society cannot be expressed at all; neither can its comparison with the budget of the wine tasting society be expressed.

Most of these shortcomings, however, do not follow from the language being extensional. It would be possible to design a language with a similar kind of semantics but expressive enough to express all the required details.

2.4 Languages based on intensional approaches

To our knowledge, in computer science there have been two different attempts to establish a totally intensional approach to conceptual modelling through a concept language. The first one was manifested in the early semantic networks (see [Qui68], [Win75], [Ran88]), where links directly connected nodes that obviously presented concepts. The second one is Kangassalo's COMIC methodology with its CONCEPT D language (see [Kan83], [Kan93], [Kan92]) influenced by Bunge's philosophy of science (as in [Bun67]) and Kauppi's concept calculus (see [Kau67]). Our example of an intensional modelling language is largely similar to CONCEPT D.[6]

[6]It should be noticed, however, that CONCEPT D as presented in [Kan83] is not a purely intensional language; one can state facts about extensions using CONCEPT D.

In the COMIC methodology, conceptual modelling is based on describing concepts and the relation of intensional containment that subsists in the set of concepts. Informally, we can say that concept **a** intensionally contains concept **b** if the knowledge content of concept **b** is a part of the knowledge contents of concept **a**. Using COMIC, the modeller creates a description or a theory of the domain of application and represents it in the form of an externalised conceptual schema. This is normally a hierarchical CONCEPT D diagram, where the topmost concept has the largest knowledge content.

The method of using intensional containment in conceptual modelling has been used elsewhere, as well. Papers by Junkkari and Niemi ([Jun98], [Nie00]) utilise many ideas first expressed by Kangassalo, but in a more formal manner. Similarly, Berztiss in [Ber99] makes a distinction between IS-A intensional containment and "part-of" intensional containment and gives them different formulations.

It can be said that in both early semantic networks and COMIC-style modelling, some notions have semantically several interpretations and sometimes the interpretation is not clear:

- As an example, the early semantic networks had some limitations both in the semantics of the links and in the semantics nodes. In nodes, the same type of node was used to express a class (type), an instance, an event and a relationship. Moreover, with links, the networks often failed to make a distinction between structural relationships and domain specific relationships (see [Ran88]).

- Similarly, according to Kangassalo in [Kan93], the notion of intensional containment in CONCEPT D is considered to include (at least) IS-A and aggregation ("has" in [Kan96]). This is natural if we consider both IS-A and aggregation to be reflexive, transitive and antisymmetric. However, in other modelling approaches IS-A and aggregation are different relationships and have different semantics.

It seems, as well, that these approaches have a view of intensionality that is somewhat similar to the idea of the "database intension". There, the database schema contains knowledge about the domain of application (in the structure of its tables and constraints, see [EN94]) and this knowledge can be seen as intensional. However, this view of intensionality is different from the view of Description Logics, where intensionality is more based on possible worlds semantics or situation semantics, as in [Woo91].

The rest of this section discusses a simple intensional modelling language, and the observations that we present do not necessarily apply to other languages of a similar kind (unless otherwise indicated).

Our intensional modelling language can be characterised as follows:

- L-entities are called concepts. Concepts do not only cover L-entities, but some L-links as well. For instance, the relationship of a wine tasting society buying bottles of wine is expressed here as the concept of WINE-BUYING that contains the concepts of WINE-TASTING-SOCIETY and WINE-BOTTLE, since WINE-TASTING-SOCIETY and WINE-BOTTLE are parts of the knowledge contents of WINE-BUYING.

  Concepts are presented simply as underlined uppercase words, which are names of the concepts.

- The principal relationship in the set of concepts is intensional containment. This is represented by a line between the contained and the containing concept. The containing concept is above the contained one, unless otherwise indicated.

The intensional containment links cover those of L-grouping, L-aggregation and L-links in other languages. With necessary attributes, the intensional containment is of the type 1:1, i.e., each occurrence of the containing concept is related to exactly one occurrence of the contained concept. This is expressed by placing "1" next to the both ends of the line.

The grammar of the language is very simple; it is sufficient to say that an expression is a concept statement or an intensional containment statement connecting to concepts (with possibly a 1:1 constraint).

A fragment of the statements of the simplified natural language description and its graphical representation can be seen in Figure 2.5.

concept(wine-buying)
concept(wine-tasting-society)
concept(wine-bottle)
concept(total-price-limitation)
...
intensionally-contains(wine-buying,wine-tasting-society)
intensionally-contains(wine-buying,wine-bottle)
intensionally-contains(wine-buying,total-price-limitation)
...

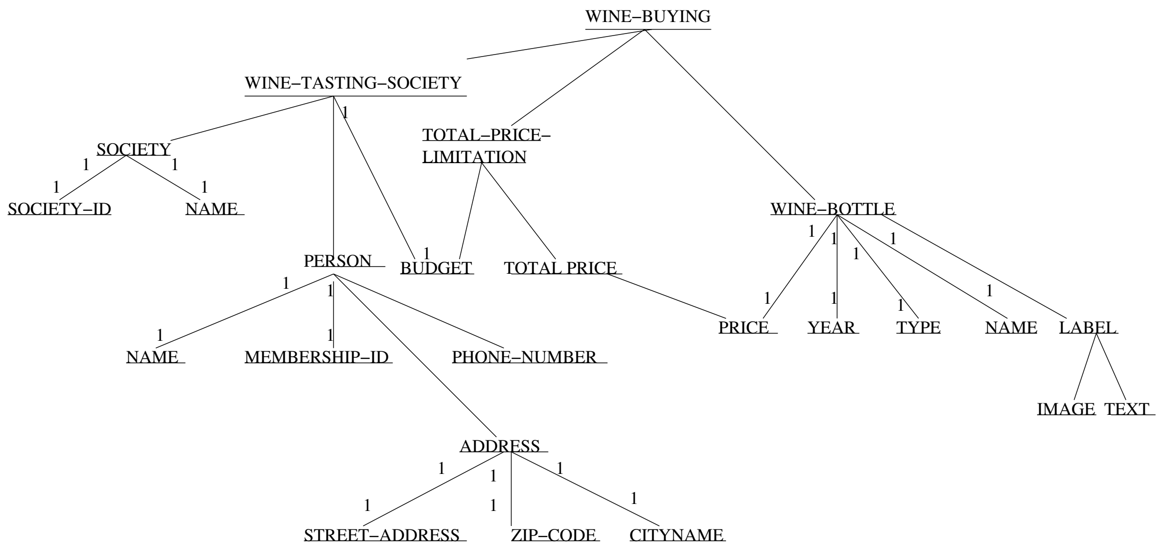

Figure 2.5: The natural language description represented by using an intensional language.

We can observe that the graphical representation is at least equally simple as in the extensional language. However, there are some cumbersome details in the design. We shall first discuss the one related to the representation and then those more theoretically based:

- The method of expressing functions (total price being a cumulative sum of bought wine bottles) and constraints (total price must not exceed the budget) relies more on intuition than on precise formulas.
- It is not obvious what the different forms of intensional containment (L-isa-links, L-aggregation) mean in each case in Figure 2.5.

The latter item has been discussed in detail by Duzi in [Duz00c], where the theoretical background of intensional containment as such was challenged. Two of the main points in Duzi's criticism are mentioned in the context of attributes (L-aggregation, "has-a"):

- If the relation is contingent ("a person has a phone number", but not "a person necessarily has a phone number") then the relation cannot be intensional in a real sense – intensional containment is supposed to cover only the necessary relations.[7]
- As intensional containment is applied in our example language, it is unlikely that the relation can really be reflexive, transitive and antisymmetric as in Kauppi's theory (see Chapter 1).

2.5 Hybrid languages

Although both approaches described above seemed to be suited to conceptual descriptions, they had their shortcomings as well. Description logics (DLs) or, as Woods in [Woo91] calls them, concept languages combine both intensional and extensional features. The intensional approach becomes apparent in the fact that concept descriptions are an important part of DLs. The extensional part is demonstrated through the use of rules ([B$^+$89], not discussed in this chapter) and by the fact that concept definitions may contain individual names – symbols refering to extensional things.

In a DL, *terminological axioms* are used to express concept names and concept definitions. Terminological axioms express either sufficient and necessary ($\doteq$) or just necessary ($\dot{\leq}$) forms of definitions (see [Lam96]; not to be confused with necessary attributes of Section 2.2). Since terminological axioms are used to express IS-A relationships, they correspond to L-isa-links.

For relationships other than IS-A, *roles* are used in DLs. To assert that WINE-TASTING-SOCIETY buys WINE-BOTTLEs, we would state that there is a concept (L-entity in our terminology) that has the role bought-wine and that WINE-BOTTLEs can be the fillers of that role. Roles provide us with the means of L-aggregation.

To meet the need for expressing "part-of" relationships, we employ a description logic designed by Lambrix in [Lam96]. In addition to normal L-aggregation constructs, this language has special "part-of" aggregation constructs in order to make (syntactic) difference between basic aggregation and part-of aggregation. The syntax of the language is shown in Figure 2.6. Terminological axioms are the expressions of this language.

To describe this DL briefly and informally, let us state that $\top$ is the concept that is on the top of the IS-A hierarchy in a way that all other concepts are either directly or indirectly sub concepts (subclasses) of $\top$. $\perp$ is at the bottom of the hierarchy.

Here, we employ the following conventions:

- To introduce names for L-entities or L-attributes, we use necessary definitions: every L-entity or L-attribute is necessarily a subclass of $\top$;
- L-part-of and L-attribute constructions are created using roles. The fact that some attribute ("role name") is necessary can be expressed using cardinality constraints (atleast 1 role-name);
- Identifying attributes are necessary attributes and, further, have cardinality constraints (atmost 1 role-name);

[7]However, with CONCEPT D's additional features it is possible to state "contingent intensional containment".

$< terminological-axiom >::=$

$\quad < concept-name > \doteq < concept-descr > \mid < concept-name > \dot{\leq} < concept-descr >$

$< concept-descr >::=$

$\quad \top$

$\quad \mid \bot$

$\quad \mid < concept-name >$

$\quad \mid (and < concept-descr >^{+})$

$\quad \mid (all < role-name >< concept-descr >)$

$\quad \mid (atleast < positive-integer >< role-name >)$

$\quad \mid (atmost < non-negative-integer >< role-name >)$

$\quad \mid (fills < role-name >< individual-name >^{+})$

$\quad \mid (allp < part-name >< concept-descr >)$

$\quad \mid (atleastp < positive-integer >< part-name >)$

$\quad \mid (atmostp < non-negative-integer >< part-name >)$

$\quad \mid (part-fills < part-name >< individual-name >^{+})$

$\quad \mid (pp-constraint < role-name >< part-name >< part-name >)$

Figure 2.6: The syntax of the DL used here.

- Names such as “red”, “white” and “rosé” form an enumeration that can fill the role of wine type;
- In Figure 2.7, comments are preceded by a #-sign.

# introduce names for L-items that will not have other definitions

name $\dot{\leq}\top$

society-id $\dot{\leq}\top$

budget $\dot{\leq}\top$

membership-id $\dot{\leq}\top$

phone-number $\dot{\leq}\top$

street-address $\dot{\leq}\top$

zip-code $\dot{\leq}\top$

cityname $\dot{\leq}\top$

year $\dot{\leq}\top$

type $\dot{\leq}\top$

price $\dot{\leq}\top$

text $\dot{\leq}\top$

image $\dot{\leq}\top$

#wine-bottle is an aggregate of name, year, type and price

#all of them are necessary, type is one of red, white, rose

wine-bottle $\doteq$

    (and (all name-slot name)
    (atleast 1 name-slot)
    (all year-slot year)
    (atleast 1 year-slot)
    (all type-slot type)
    (atleast 1 type-slot)
    (all price-slot price)
    (atleast 1 price-slot)

```
    (fills type-slot red white rosé)
    (allp label-slot label))
#labels are aggregates of text and image
label ≐
    (and (all text-slot text)
    (all image-slot image))
#address is an aggregate of street address, zip code and cityname,
#all of them necessary
address ≐
    (and (all street-address-slot street-address)
    (atleast 1 street-address-slot)
    (all zip-code-slot zip-code)
    (atleast 1 zip-code-slot)
    (all cityname-slot cityname)
    (atleast 1 cityname-slot))
#person is an aggregate of name, membership id, phone number
#and address. Name, address and membership id are necessary,
#membership-id is identifying.
person ≐      (and (all name-slot name)
    (atleast 1 name-slot)
    (all membership-id-slot membership-id)
    (atleast 1 membership-id-slot)
    (atmost 1 membership-id-slot)
    (all phone-number-slot phone-number)
    (all address-slot address)
    (atleast 1 address-slot))
#society is an aggregate of name and society-id,
#both necessary. Society-id is identifying.
society ≐      (and (all name-slot name)
    (atleast 1 name-slot)
    (all society-id-slot society-id)
    (atleast 1 society-id-slot)
    (atmost 1 society-id-slot))
#bought wine consists of wine bottles
#and has attribute price
bought-wine ≐      (and (allp wine-bottle-slot wine-bottle)
    (all price-slot price))
#wine-tasting-society is a society, buys wine
#and consists of persons
wine-tasting-society ≐      (and society
    (allp person-slot person)
    (all bought-wine-slot bought-wine))
```

Figure 2.7: The natural language description represented using a description logic.

It is evident that a DL language is not at its best for a task like this; the DL representation of the example is long and not exactly visually illustrative. After all, DLs were originally designed for a more demanding task of classifying concept expressions into a taxonomy; some forms of aggregation with DLs has been discussed only recently (e.g. in [Lam96]). Since classification can be a demanding (NP-complete[8]) task even with a limited number of syntactic elements in the language, researchers have been reluctant to add extra features to DL languages. Partly because of that, in our example, several features were omitted, including:

- There is no simple way of adding functions in the language; for instance, in order to calculate the cumulative price of wine bought by the wine tasting society;[9]
- The same applies to constraints (the price of bought wine must be less than the budget);
- There is no easy way of expressing grouping. In the example, part-of has been used instead.

2.6 Summary and discussion: languages in conceptual modelling

In this chapter, we have discussed three different approaches to conceptual modelling based on the languages used in modelling: extensional, intensional and hybrid languages. The choice of language in conceptual modelling reflects rather well the underlying commitments in ontology and semantics. It is out of the scope of this chapter to give exact definitions of different types of semantics on which each of these languages rely.[10]

We have evaluated a prototypical language of each kind by a real world example. By presenting a typical real world example, we hope to have avoided elliptical reasoning ("this example requires a certain type of language", "it can be presented using only this type of language", "this language is certainly most suited for conceptual modelling"). Any language that has been designed to represent something that exists will naturally have some kinds of ontological commitments. However, we have tried first to introduce the example in a language as "ontologically neutral" as possible, by a simplified natural language description. This description expresses the most often used abstraction methods, like taxonomy, aggregation and grouping.

Based on the representation of the example, we have observed some shortcomings in all the approaches. Our extensional representation language is influenced by the ER model (see [Che76]). Mainly, the shortcomings of our extensional language were only difficulties in expressing all the details of the example. This is because our example did not deal with "belief-worlds" or other things where the extensional approach cannot be applied at all.

In the intensional approach, we utilised a language influenced by CONCEPT D of [Kan83]. It seems that this kind of language is an able tool to present everything that was needed by the simplified natural language description. The diagrams seem to be appealingly simple, in their basic form, where they only contain concept nodes and intensional containment links. Yet the semantics of the links can be difficult to specify (see Section 7.4.3 for a detailed study).

As an example of a hybrid language, i.e., a language by which one can state expressions about intensions and expressions about extensions, we used a description logic influenced by Lambrix's

[8]NP-complete set of decision problems is a subset of NP, see Section 1.7 and [Ata99].

[9]There is a DL with this feature as well; see [BS98]. Naturally, in some languages like KL-ONE, these functions could be embedded using a "host language connection", i.e., by programming the function in the language which KL-ONE has been implemented in (see [BS85], [Ran88]).

[10]It can be noticed, however, that Chen in [Che76] describes the semantics of the ER model by a set theory based semantics, giving the normal interpretations to relations as tuples of entities and attributes as functions.

Concerning the intensional languages, according to Duzi in [Duz00a], intensional containment can be defined using HIT semantics, where it corresponds to being member of the contents of the concept. This will be further discussed in Chapter 7.

formalism in [Lam96]. The background theory (representations using terminological axioms and concept descriptions) seems suitable to conceptual representation and the semantics is well defined. However, there were several difficulties in expressing the example in a description logic formalism. We see that developing a description logic designed for conceptual modelling seems to be a promising item for future research (see [CLN98]). This could include the development of a modelling methodology as well, in the guidelines of the COMIC methodology of CONCEPT D.

Chapter 3

Logic and Conceptual Graphs

3.1 Introduction

Conceptual graphs have been proposed as an alternative to First Order Predicate Logic (FOPL) in knowledge representation ([Sow84], [Sow00]; see also [JP89]). FOPL and its derivatives (subsets), like Horn clauses and logic programming languages, are perhaps the best known and established formalisms for knowledge representation. To gain acceptance, the alternative formalisms should offer advantages over FOPL. The advantages may be:

- Simplicity: formulas expressed by an alternative formalism may be easier to read and to write (or closer to natural language) than formulas of FOPL and its derivatives;
- Deductive power, i.e. how efficient it is to calculate if some formula can be deduced from a set of other formulas. FOPL is known to be semi-decidable; more limited formalisms can have a proof theory that is decidable, and even tractable[1];
- Expressive power, i.e. what can be expressed using the formalism at hand. With traditional FOPL, uncertainty or default reasoning, for example, cannot be handled.

In order to see if conceptual graphs are simpler, have more deductive power or have more expressive power than FOPL, we shall carry out an extensive comparison between FOPL and conceptual graphs in Sections 3.3 and 3.4.

To make the comparison easier, we use slightly simplified forms of both FOPL and conceptual graphs.[2] This is explained in detail in Sections 3.2.1 and 3.3.1. Section 3.2.1 contains a standard presentation of a slightly limited FOPL and Section 3.2.2 its semantics. We shall use the term "FOPL" to refer to this limited version. The same kind of presentation of conceptual graphs (limited syntax) and their semantics will follow in Sections 3.3.1 and 3.3.2. Here, the features included in the discussion only cover the formalism of conceptual graphs, not the background theory (concept hierarchies; see [Sow84]).

The basis for our comparison is the following simple notion: If any expression of a language L_1 can be translated into an expression of a language L_2 and any expression of a language L_2 can

[1]It can be shown that the set of theorems of FOPL is indeed decidable but the set of non-theorems is not. Therefore, a procedure that tries to determine if some FOPL formula is a theorem would run endlessly if the formula is a non-theorem.

[2]Basically, we only limit the arity of predicates. This enables us to handle relations in a more mechanical manner. This limitation can be removed but it would make the translations more difficult.

be translated into an expression of a language L_1, then the languages have equal expressive power. The translation is justified by the semantics of the languages L_1 and L_2; i.e., if expression e_1 of L_1 is true in the same models as expression e_2 of L_2, then e_1 can be translated into e_2 and e_2 into e_1 (we assume that L_1 and L_2 have similar models). Naturally, if any expression of L_1 can be translated into L_2, but there are expressions of L_2 that cannot be translated into L_1, we say that the expressive power of L_2 is greater than that of L_1.

Sowa has presented in [Sow84] an algorithm ("operator ϕ") that translates a conceptual graph into a FOPL formula. Sowa states, however, that the expressive power of conceptual graphs may exceed the expressive power of FOPL: "When the referents of concepts are limited to single individuals, conceptual graphs cannot go beyond first-order logic." ([Sow84], p. 116). Later Sowa extends the notation so that a referent of a concept can be a set of individuals. Sowa's algorithm obviously applies only to conceptual graphs where the referents of concepts are single individuals. This is the limited syntax of conceptual graphs defined in Section 3.3.1.

Assuming that Sowa's algorithm translates an arbitrary conceptual graph into a formula of FOPL, it follows that FOPL is at least as expressive as conceptual graphs. In Section 3.4 we outline an algorithm for translating closed formulas of FOPL into conceptual graphs. On the basis of these two algorithms it can be claimed that conceptual graphs (with limited syntax) and (our version of) FOPL equal in expressive power.

3.2 First Order Predicate Logic

Here we present FOPL in a simplified form, but much of the formalism has been adopted from [vD97] and [Zho98].

3.2.1 The language of FOPL, L_{FOPL}

A language is a set expressions (in the case of logic, well-formed formulas) composed of a (finite) set of terminal symbols by rules of a grammar. The set T_{FOPL} contains the terminal symbols of our language L_{FOPL}. Elements of T_{FOPL} are a constant symbol c, a variable symbol x, a unary predicate symbol P, a binary predicate symbol R, connectives ($\neg, \wedge, \vee, \rightarrow, \leftrightarrow$), quantifiers ($\exists, \forall$), parenthesis, comma and prime.[3]

Constants are sequences of the constant symbol and an arbitrary number of primes: $c', c'', ...$ Variables ($x', x'', ..$), unary predicates ($P', P'', ..$) and binary predicates ($R', R'', ..$) are formed in the same way.

In a more formal approach, predicates can have any arity and the information of a predicate's arity must be given. In this chapter we use only unary and binary predicates.

In a metalanguage, i.e. when describing expressions of L_{FOPL}, we use symbols x, y, z to denote variables; a, b, c to denote constants, A, B, C to denote formulas; P, Q to denote unary predicates and H when a predicate (no matter if its arity is 1 or 2) is discussed.[4]

In an informal approach (like in Section 3.3.1), we also use lower case natural language words instead of predicates.

Here, T_{FOPL} does not include the identity symbol $=$ and there are no functions in the language.

[3]The inverse binary predicate symbol (see section 3.2.3) can be considered as a special case of binary predicate symbol.

[4]It should be noticed that P, R, x and c are symbols of both the metalanguage and T_{FOPL}. However, in what follows, it should be obvious when they are used as T_{FOPL} symbols or in the metalanguage.

Variables and constants are jointly called *terms*. Terms are written after the predicate, between the parentheses. The number of these terms is determined by the arity of the predicate. These terms are called the *arguments*. A predicate followed by its arguments (in parenthesis) is called an *atomic formula*. An atomic formula is called a *ground formula* if all the arguments are constants.

A quantifier *binds* the variable x in formulas $\exists x(A)$ and $\forall x(A)$. A variable in formula A is bound if it is bound by a quantifier. Otherwise it is free. If there are no free variables in a formula, then the formula is called a *closed formula*. A ground formula is an example of a closed formula.

A *replacement* of a variable x with a variable y or a constant c in formula A is denoted with *A(x/y)* and *A(x/c)*, respectively.

Using replacements, it is possible to rename variables in a formula. As an example, let formula A be $(\forall x(P(x)) \wedge \forall x(Q(x)))$, which we can express in the form $(B \wedge C)$. Now, let us perform the replacement $B(x/y)$ resulting in $(\forall y(P(y)) \wedge \forall x(Q(x)))$. If, as in the example, the result of renaming of variables is a formula where independent variables do not have the same name, we call this process an *unambiguous naming of variables.*

Gr_{FOPL} is a grammar that generates all formulas of L_{FOPL}. The terminal symbols of the grammar are elements of T_{FOPL} and the symbol "|" in the production rules means "or". The auxiliary symbols are $\{B,X,C,U,Q,T,F\}$ and F is the starting symbol of the grammar. The production rules of the grammar are:

1. $B \mapsto '$
2. $B \mapsto B'$ (a sequence of primes)
3. $X \mapsto xB$ (variables)
4. $C \mapsto cB$ (constants)
5. $U \mapsto X|C$ (constants and variables are terms)
6. $Q \mapsto PB$ (unary predicates)
7. $T \mapsto RB$ (binary predicates)
8. $F \mapsto Q(U)|T(U,U)$ (atomic formulas are formulas)
9. $F \mapsto \neg(F)$ (a negation of a formula)
10. $F \mapsto (F \wedge F)$ (two formulas connected with $\wedge$)
11. $F \mapsto (F \vee F)$ (two formulas connected with $\vee$)
12. $F \mapsto (F \rightarrow F)$ (two formulas connected with $\rightarrow$)
13. $F \mapsto (F \leftrightarrow F)$ (two formulas connected with $\leftrightarrow$)
14. $F \mapsto \exists X(F)$
15. $F \mapsto \forall X(F)$ (a quantifier, a variable and a formula together form a formula.)

It is worth noticing that the grammar Gr_{FOPL} allows formulas with bound variables (like $\exists x'(P'(x'))$ and $\exists x'(\exists x''(R'(x',x''))))$, and formulas with free variables: $\exists x'(P'(x''))$.

3.2.2 Semantics of L_{FOPL} (model theory)

In this section we shall give a short account of the semantics of FOPL. We shall discuss model theory, satisfiability and validity (truth in all L-models). The formal definition of L-model is presented after these notions.

A formula is said to be *satisfiable* if it can be true in some situation, i.e. in some L-model. For instance, the formula $\exists x'(P'(x'))$ is satisfiable. If the predicate P' is interpreted "to be an even natural number", the formula is satisfied when x' equals 2,4,6,.. In this case, the set $\{2,4,6,..\}$ is also called the extension of the predicate P'.

Similarly, a set of formulas is satisfiable (or consistent), if there is an L-model, where all the formulas of the set are (simultaneously) true.

It is evident that the formula $\exists x'((P'(x') \wedge \neg(P'(x'))))$ cannot be true in any L-model. This kind of formula is *unsatisfiable*. A set of formulas is unsatisfiable (or inconsistent), if there is no L-model, where all the formulas of the set can be (simultaneously) true.

A formula is *valid* if it is true in all L-models. $\neg(\exists x'((P'(x') \wedge \neg(P'(x')))))$ is an example of a valid formula.

The formal definitions of an L-model and truth in L-model are as follows (c.f. [SV93]).

Let L be a set of terminal symbols. An L-model is a structure (tuple) M, with the following parts:

- A non-empty set of elements, a domain $dom(M)$.
- A (constant) element of $dom(M)$, c^M, for each constant c of L.
- A subset of $dom(M)$, P^M for each unary predicate P of L.
- A subset of $dom(M) \times dom(x)$, R^M for each binary predicate R of L.

An interpretation function is a function that interprets ("maps") variables into elements of $dom(M)$.

An interpretation *in* an L-model is an interpretation function such that $v(x) \in dom(M)$ for each variable x of the language.

Let v be an interpretation in L-model M. The value of term t, $t^{M,v}$ in interpretation v is:

$$\begin{cases} v(x), \text{ if } t \text{ is variable } x \\ c^M, \text{ if } t \text{ is constant } c \end{cases}$$

Let v be an interpretation function in M and $a \in dom(M)$. We define a replacement of variable in an interpretation $v(x/a)$ as

$v(x/a)(y) =$

$$\begin{cases} v(x), \text{ if } x \neq y \\ a, \text{ if } x = y \end{cases}$$

Let M be an L-model and v an interpretation function. The following set of conditions defines when v satisfies a formula in M:

1. Let t be a term. If the formula is of form $P(t)$, v satisfies the formula if and only if $t^{M,v} \in P^M$.
2. Let t and u be terms. If the formula is of form $R(t,u)$, v satisfies the formula if and only if $(t^{M,v}, u^{M,v}) \in R^M$.

3. If the formula is of form $\neg(B)$, v satisfies the formula if and only if v does not satisfy B.
4. If the formula is of form $(B \vee C)$, v satisfies the formula if and only if v satisfies B or C.
5. If the formula is of form $(B \wedge C)$, v satisfies the formula if and only if v satisfies B and C.
6. If the formula is of form $(B \rightarrow C)$, v satisfies the formula if and only if it is *not* the case that v satisfies B and v does not satisfy C.
7. If the formula is of form $(B \leftrightarrow C)$, v satisfies the formula if and only if
   - v satisfies B and C.
     or
   - v does not satisfy B and C.
8. If the formula is of form $\exists x(B)$, v satisfies the formula if and only if $v(x/a)$ satisfies B for some $a \in dom(M)$.
9. If the formula is of form $\forall x(B)$, v satisfies the formula if and only if $v(x/a)$ satisfies B for all $a \in dom(M)$.

Truth in a model is defined only for closed formulas, as follows:

A closed formula is true in L-model M if all valuation functions in M satisfy it.

If a formula A is true in an L-model when formulas of a set S are true in that L-model too, then A is a *logical consequence* of the formulas of S. This is denoted by $S \models A$.

In the above case, if S in an empty set, then A is a valid formula. This is denoted by $\models A$.

If formulas A and B are satisfied in the same L-models, then they are *logically equivalent*. This is denoted by $A \Leftrightarrow B$.

The following "shortcut rules" S1 - S20 are examples of logical equivalences:

S1. $(A \vee B) \Leftrightarrow \neg((\neg(A) \wedge \neg(B)))$,

S2. $\neg(\neg(A)) \Leftrightarrow A$,

S3. $((A \wedge B) \vee C) \Leftrightarrow ((A \vee C) \wedge (B \vee C))$,

S4. $\neg((A \wedge B)) \Leftrightarrow (\neg(A) \vee \neg(B))$,

S5. $\neg((A \vee B)) \Leftrightarrow (\neg(A) \wedge \neg(B))$,

S6. $(A \rightarrow B) \Leftrightarrow \neg((A \wedge \neg(B)))$,

S7. $(A \rightarrow B) \Leftrightarrow (\neg(A) \vee B)$,

S8. $(A \leftrightarrow B) \Leftrightarrow (\neg((A \wedge \neg(B))) \wedge \neg((B \wedge \neg(A))))$,

S9. $(A \leftrightarrow B) \Leftrightarrow ((A \rightarrow B) \wedge (A \rightarrow B))$,

S10. $\forall x(H(x..)) \Leftrightarrow \neg(\exists x(\neg(H(x..))))$,
where H is a predicate and $x..$ stands for its arguments.

S11. $\neg(\forall x(H(x..))) \Leftrightarrow \exists x(\neg(H(x..)))$, S12. $\neg(\exists x(H(x..))) \Leftrightarrow \forall x(\neg(H(x..)))$,

S13. $(\forall x(P(x)) \wedge \forall x Q(x)) \Leftrightarrow \forall x((P(x)) \wedge Q(x)))$,

S14. $(\exists x(P(x)) \vee \forall x Q(x)) \Leftrightarrow \exists x((P(x)) \vee Q(x)))$,

S15. $(K_1 x(P(x)) \wedge K_2 x(Q(y))) \Leftrightarrow K_1 x(K_2 y((P(x) \wedge Q(y))))$, where K_1, K_2 are quantifiers and x does not occur in $Q(y)$ and y does not occur in $P(x)$,

S16. $(K_1 x(P(x)) \vee K_2 x(Q(y))) \Leftrightarrow K_1 x(K_2 y((P(x) \vee Q(y))))$, where K_1, K_2 are quantifiers and x does not occur in $Q(y)$ and y does not occur in $P(x)$,

S17. $((A \wedge B) \wedge C) \Leftrightarrow ((A \wedge (B \wedge C)))$,

S18. $((A \vee B) \vee C) \Leftrightarrow ((A \vee (B \vee C)))$,

S19. $(A \wedge B) \Leftrightarrow (B \wedge A)$,

S20. $(A \vee B) \Leftrightarrow (B \vee A)$.

These shortcut rules will be utilised when FOPL formulas are transformed into a form that is easy to process and understand. This form will be used when FOPL formulas are compared with conceptual graphs.

Here, we do not present a proof theory for FOPL. Instead, we state the following short remarks on semantics and proof theory.

A proof theory for any language consists of a set of axioms and a set of deduction rules. Let A be a formula and S a set of formulas. If a A can be deduced from the formulas S, then A is called a theorem. This is denoted by $S \vdash A$. Suppose there is a proof theory PT_{FOPL}. Language L_{FOPL}, with semantics as defined here, and the proof theory PT_{FOPL} are jointly called an interpreted system of logic. In this system, if every theorem is a valid formula, then the system is *sound*. If the system is sound and every valid formula is also a theorem, then the system is *complete*. It is possible to construct a proof theory so that the system is complete (see e.g. [vD97]). Thus, all the semantic shortcut rules are syntactic, too. For instance, since it is known that the shortcut $(A \vee B)$ is logically equivalent to $(B \vee A)$, we can also deduce $(B \vee A)$ from the set $\{(A \vee B)\}$.

3.2.3 Quantifiers and inverse relations

To support the discussion in Sections 3.2.4 and 3.3.1, we present a short remark on the relationship of quantifiers and relations.

The inverse relation R^{-1} of R consists of the pairs (y,x) for each pair $(x,y) \in R$. Let R be a binary predicate. We shall call R^{-1} the *inverse predicate* of R.

The following *inverse predicate formula* relates the inverse relations with quantifier sequences:

$$Ky(Kx(R(x,y))) \Leftrightarrow Ky(Kx(R^{-1}(y,x))),$$

where K's are (possibly different) quantifiers.

To compare FOPL formulas with conceptual graphs, it is convenient to arrange the variables so that they appear in the same order in the relations as in the quantifier sequence. For binary predicates this is easy based on the inverse predicate formula; for example, $\exists y(\forall x(R(x,y))) = \exists y(\forall x(R^{-1}(y,x)))$.

3.2.4 The translation form

Any FOPL formula can be transformed into a *translation form* (as defined by the following algorithm) by using the shortcut rules S1 - S20. The translation form makes FOPL formulas more uniform and thus easier to process. A closed formula in a translation form can, in principle, be translated into a conceptual graph (see Section 3.4 and Appendix A).

An algorithm that transforms an FOPL formula A into a corresponding translation form formula is as follows:

REPEAT

1. Use shortcut rule S9 to replace equivalences with implications and conjunctions. Use shortcut rule S6 to replace implications $(B \rightarrow C)$ with $\neg((B \wedge \neg(C)))$.
2. If there are binary predicates in the formula, use commutativity- and associativity shortcuts (S17 - S20) so that the binary predicates appear as early (left hand side) as possible in the formula.
3. Limit the scope of the connective $\neg$ by applying the following shortcuts until none of them can be applied.
   - shortcut S2: replace $\neg(\neg(B))$ with B,
   - shortcut S5: replace $\neg((B \vee C))$ with $(\neg(B) \wedge \neg(C))$,
   - shortcut S4: replace $\neg((B \wedge C))$ with $(\neg(B) \vee \neg(C))$.
4. Perform an unambiguous naming of the variables.
5. Move all the quantifier - variable pairs to the left hand side of the formula by applying rules S15 - S16.
6. Using the inverse predicate formula of Section 3.2.3, transform the binary predicates into their inverse predicates if necessary. That is, maintain the same order of variables in binary predicates as in the quantifier sequence.

UNTIL none of the rules above can be applied.

3.3 Conceptual graphs

Conceptual graphs are a general, large formalism for knowledge representation developed by John F. Sowa ([Sow84]). The term "general" means that the formalism is not designed for e.g. physical modelling, but can serve as a similar paradigm as FOPL. Sowa mentions that so-called existential graphs by Peirce (see [Rob73]) have served as an archetype for conceptual graphs.

Sowa's system of conceptual graphs includes mechanisms to implement deductions using a concept hierarchy (information based on the intensions of concepts), but we shall omit these features in this discussion.

In addition to these features, our discussion of conceptual graphs in this section differs from Sowa's presentation as follows:

1. In Sowa's account, a concept node can refer to a set of terms (Sowa uses the term "referent"; see [Sow84], p. 116). For the definition of concept node, see Section 3.3.1 page 38.

2. Sowa employs relations whose arity is greater than 2. Here, the maximum arity is limited to 2.

3. Sowa presents various "shortcuts" in the language for proper names, measures and quantities.

4. Sowa uses concept nodes that have no referent. However, this is a shortcut for the representation of a general concept (see below).

5. Equally, Sowa uses the universal quantifier without an explicit variable. In this chapter, the variables are explicit.

6. Sowa uses a formalism "#" for a specified referent. This is replaced with a constant in this chapter.

7. The connective $\wedge$ does not appear in Sowa's (or Peirce's, [Rob73]) accounts, but graphs are written next to each other to represent conjunction. In this chapter, the $\wedge$ -connective is explicit.

8. Sowa represents most of the connectives and the universal quantifier as shortcuts. Here they are included in the terminal symbols of the language.

9. Sowa suggests that the set of relation nodes consists solely of symbols corresponding to "cases" of case grammars, i.e. relation names correspond to AGNT (agent of doing something) or OBJ (object of doing something). In this chapter the set of relation symbols is a matter of choice.

10. Sowa (and Peirce) use a line of identity to denote the same referent in two or more concepts. In this chapter, referents with the same variable or constant "automatically" refer to same individuals. The same convention is used in logic programming.

It is evident that the differences 3 - 10 are only notational and have no effect to the expressive power of the language. The first two ones, however, limit the expressive power to a great extent. Therefore, in this chapter, the version of the conceptual graphs that we use will be called conceptual graphs with a limited syntax. This version is a subset of Sowa's original conceptual graph language.

3.3.1 The language of conceptual graphs (limited syntax), L_{CG}

Introduction

As in FOPL, normal connectives are employed in the set of terminal symbols of conceptual graphs (limited syntax): $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$.

The existential quantifier is not needed in the language, since all variables are assumed to be existentially quantified, unless they are universally quantified. Thus, only the universal quantifier ($\forall$) is in the set of terminal symbols.

Variables x', x'',.. and constants c', c'',.. will be included, as in FOPL. In the metalanguage, variables are referred by *x, y* and *z* and constants by *a, b* and *c*.

Instead of unary and binary predicates, in conceptual graphs (with limited syntax) there are *concept nodes* and *relation nodes*. A concept node is constructed of the left bracket ([), the concept type symbol (P', P'',..), the colon (:), a term (a constant or a variable) and the right bracket (]). In a concept node, the concept type symbol (P', P'',..) is called the *type* of the term. In the metalanguage, *P* and *Q* stand for concept type symbols, and informally uppercase natural language words are used instead of predicates. A concept node $[P : x]$ is interpreted the same way as the expression $\exists x(P(x))$ in FOPL. A concept node may also include an occurrence of the universal quantifier. A special concept type $\top_c$ is included in terminal symbols. This is discussed in example 3.3.

Relation nodes connect concept nodes to each other in a conceptual graph. A relation node consists of an arrow ($\leftarrowtail$ or $\rightarrowtail$) that is connected with a concept node, a relation $(R', R'', ..)$ in parenthesis and another arrow ($\leftarrowtail$ or $\rightarrowtail$) that is connected with another concept node.[5] Informally, uppercase natural language words stand for relations, too.

A concept node alone is a conceptual graph, but a relation node must be connected with two[6] concept nodes. A simple example of a conceptual graph with a relation node is:

$$[P : x] \rightarrowtail (R) \rightarrowtail [Q : y].$$

Intuitively, a conceptual graph (or a graph, for short) is:

- A single concept node, or
- two concept nodes, connected with a relation node, or
- a construction, where two or more conceptual graphs are joined by connectives.

The scopes of quantifiers in conceptual graphs are discussed on page 41.

Simple examples

A concept node with a variable is called a *generic concept*. A concept node with a constant is called an *instantiated concept*.

$$[PAINTING : x]\ ,\ [PAINTING : a] \tag{3.1}$$

A generic concept and an instantiated concept

or, more formally,

$$[P' : x']\ ,\ [P' : c']. \tag{3.2}$$

The relationship “owns” (i.e. “x owns y”) is presented by:

$$\rightarrowtail (OWN) \rightarrowtail,$$

where the left $\rightarrowtail$ connects the relation node with a concept node with the term x and the right $\rightarrowtail$ connects the relation node with a concept node with the term y.

The inverse relation “is owned by” can be presented simply by reversing the direction of the arrows, that is:

$$\leftarrowtail (OWN) \leftarrowtail .$$

In FOPL, $\exists x(\exists y(R(x,y)))$ is a well-formed closed formula. In conceptual graphs, it seems that the types of x and y should be represented by a concept node. For this purpose, the symbol $\top_c$

[5]The question whether to use left or right directed arrows is related with the semantics of the relation. This will be discussed on page 43.

[6]This is because in this chapter, no other arities than 2 are allowed for relations.

was introduced in the set of terminal symbols. $\top_c$ is needed to represent an unknown type. The conceptual graph that conveys the information "there is some x that owns some y" is

$$[\top_c : x] \rightarrowtail (OWN) \rightarrowtail [\top_c : y]. \tag{3.3}$$

Using connectives, one can construct conceptual graphs that have *subgraphs* as in Graphs 3.4 and 3.5.

$$\begin{gathered}[[ARTIST : a] \rightarrowtail (PAINT) \rightarrowtail [PAINTING : b] \\ \wedge \\ [COLLECTOR : c] \rightarrowtail (BUY) \rightarrowtail [PAINTING : b]].\end{gathered} \tag{3.4}$$

Graph 3.4, that conveys the information "A certain artist paints a certain painting that is bought by a certain collector".

$$\begin{gathered}[[ARTIST : a] \rightarrowtail (PAINT) \rightarrowtail [PAINTING : b] \\ \wedge \\ \neg[[COLLECTOR : c] \rightarrowtail (BUY) \rightarrowtail [PAINTING : b]]].\end{gathered} \tag{3.5}$$

Graph 3.5 that conveys the information "A certain artist paints a certain painting and a certain collector does not buy it".

In graph 3.5 the brackets following the connective $\neg$ enclose the subgraph

[COLLECTOR: c] $\rightarrowtail$ (BUY) $\rightarrowtail$ [PAINTING: b].

The following conventions are employed for negations:

- $\neg$ in front of a graph or subgraph G simply means that G is negated, as in the lower subgraph in Graph 3.5.
- $\neg$ in the concept node, like $[\neg P : x]$ means that there exists a variable x whose type is not P.
- The $\sim$ symbol is reserved to represent negation in relations. For example, the graph

  $$[P : x] \rightarrowtail (\sim R) \rightarrowtail [Q : y]$$

  means that there exists a variable x of type P and a variable y of type Q and they are not related by R.

In Appendix A, we present some more complicated examples of knowledge representation with conceptual graphs. The same information will be represented in FOPL, too, to enable a comparison of "readability" of FOPL formulas and conceptual graphs. Moreover, in Appendix A, the translation of a FOPL formula into a conceptual graph is represented using the method of Section 3.4.

The grammar of conceptual graphs (limited syntax)

The set of terminal symbols, T_{CG}, for the language of conceptual graphs (limited syntax) is

$$T_{CG} = \{\wedge, \neg, \vee, \rightarrow, \leftrightarrow, \forall, (,), [,], x, c, P, R, ', \top_c, \sim, \leftarrowtail, \rightarrowtail\}.$$

The auxiliary symbols of the grammar are $\{B,U,C,Q,A,F\}$, the starting symbol is F, and the production rules are:

1. $B \mapsto '$
2. $B \mapsto B'$ (Sequences of primes)
3. $U \mapsto xB$ (Variables are terms.)
4. $U \mapsto cB$ (Constants are terms)
5. $C \mapsto \top_c | PB$ (Concept type symbols, like P',P",..)
6. $Q \mapsto RB$ (Relation type symbols, like R',R",..)
7. $A \mapsto [C : U] | [\forall C : U] | [\neg C : U]$ (Concept nodes)
8. $F \mapsto A$ (Concept nodes are conceptual graphs)
9. $F \mapsto A \rightarrowtail (Q) \rightarrowtail A | A \leftarrowtail (Q) \leftarrowtail A$ (Concept nodes, joined by a relation, are conceptual graphs)
10. $F \mapsto A \rightarrowtail (\sim Q) \rightarrowtail A | A \leftarrowtail (\sim Q) \leftarrowtail A$ (Negated relation)
11. $F \mapsto \neg[F]$
12. $F \mapsto [F \wedge F]$
13. $F \mapsto [F \vee F]$
14. $F \mapsto [F \rightarrow F]$
15. $F \mapsto [F \leftrightarrow F]$ (Connectives)

A graph is called *simple* if it has no connectives, whereas a graph with connectives is called *complex*. With complex graphs we shall call the parts (that correspond to each F in the grammar) *subgraphs*.

Quantifiers – The scopes

Previously, we stated that there are no free variables in a conceptual graph. In this section, the scope of quantifiers in graphs is discussed.

For example, Graph 3.6

$$[[ARTIST : x] \wedge [RICH : x]] \tag{3.6}$$

could be interpreted in two ways:

a) In the same way as the FOPL formula $(\exists x(artist(x)) \wedge \exists x(rich(x)))$, i.e. “there exists someone who is an artist and someone who is rich”.

b) In the same way as the FOPL formula $\exists x((artist(x) \wedge rich(x))$, i.e. “there exists someone who is an artist and rich”.

Like Sowa, we maintain the *latter* interpretation. In practice this means that *if there are variables that share a name in a graph, the variables are meant to refer to the same individual.*

The variable x in Graph 3.7 (“there is a rich artist x, he does not paint painting z”) refers to the same individual.

$$[[RICH : x] \wedge \neg[[ARTIST : x] \rightarrowtail (PAINT) \rightarrowtail [PAINTING : z]]] \tag{3.7}$$

This interpretation will unfortunately have an unpleasant consequence: the renaming of the variables will not be trivial. Let us consider this with the universal quantifier, as in graphs 3.8 (“all the collectors buy all the paintings”) and 3.9.

$$[COLLECTOR : \forall x] \rightarrowtail (BUY) \rightarrowtail [PAINTING : \forall y] \tag{3.8}$$

$$[COLLECTOR : \forall x] \rightarrowtail (BUY) \rightarrowtail [PAINTING : \forall x] \tag{3.9}$$

It is obvious that we do not intend to state that the collector and painting in question are the same individual; i.e. 3.8 is the graph we prefer.

Quantifiers – The universal quantifier

Let us consider universal quantifiers in graphs using the examples of 3.10 and 3.11.

$$[PAINTING : \forall x] \tag{3.10}$$

$$[PAINTING : \forall x] \rightarrowtail (PAINTEDBY) \rightarrowtail [ARTIST : y] \tag{3.11}$$

Graph 3.10 expresses that “every object in the domain is a painting”. Graph 3.11 does certainly not claim that, but it expresses that all those objects in the domain that are paintings have an artist. Using the connective $\rightarrow$ (and applying the semantics that we shall define in Section 3.3.2), we can see that Graph 3.11 could be presented in the form of Graph 3.11’:

$$[[PAINTING : \forall x] \rightarrow [PAINTING : x] \rightarrowtail (PAINTEDBY) \rightarrowtail [ARTIST : y]]. \tag{3.11'}$$

With Graph 3.11’, it is easier to see the scope of the universal quantifier: in addition to node $[PAINTING : \forall x]$, it also covers the subgraph $[PAINTING : x] \rightarrowtail (PAINTEDBY) \rightarrowtail [ARTIST : y]$.

Quantifiers – Quantifier sequences

As explained in Section 3.2.3, for the sake of clarity it is sometimes profitable to use inverse relations. In conceptual graphs, the inverse relations can be presented in an easy and intuitive way: by reversing the direction of $\rightarrowtail$ or $\leftarrowtail$ symbols connecting the relation with the concept nodes.

Concerning FOPL and conceptual graphs, the following analogies seem obvious (for translation details, see section 3.4):

For the FOPL formula $\forall x(\exists y(R(x,y)))$, the graph counterpart is $[\top_c : \forall x] \rightarrowtail (R) \rightarrowtail [\top_c : y]$.

For the formula $\forall y(\exists x(R(x,y)))$ that equals $\forall y(\exists x(R^{-1}(y,x)))$ the counterpart is

$$[\top_c : \forall y] \leftarrowtail (R) \leftarrowtail [\top_c : x].$$

Using the inverse relation, the order of variables in quantifier sequences can be preserved, which is an advantage in the translation process.

Quantifiers – Summary and the explicit quantifier form

The scopes of the quantifiers are essential in terms of semantics. Since the scopes of quantifiers in graphs are more implicit than in FOPL, we present a simple explication method. With the method, the quantifier - variable pairs are presented in front of the subgraph that is their scope. The resulting form, *the explicit quantifier form*, is utilised when discussing the semantics of graphs. It should be noticed that the explicit quantifier form is not the same as the so-called prenex form; instead, the explicit quantifier form repeats the existing quantifiers in the front of the formula.

Creating the explicit quantifier form is a mechanical tranformation based on the idea of expressing, for each variable in the graph and for each subgraph of the graph, whether or not the variable occurs in the subgraph. "Whether or not variable x occurs in the subgraph" is referred to as *variable occurence value* of x below.

As an example, let us consider Graph 3.6:

	[	$[ARTIST : x] \wedge$	$[RICH : x]$	]
x:	false	true	true	false

It is easy to notice that the quantifier variable pair should occur in the subgraph that has the variable occurence value `false` just outside of the outermost subgraph that has the variable occurence value `true`.[7] Naturally, for a simple graph like [ARTIST:x], there is no subgraph with the variable occurence value `false`. In that case, the quantifier variable pair should occur in front of the graph. If the quantifier that binds any occurence of variable x is $\forall$, then the quantifier variable pair will be $\forall x$, otherwise; $\exists x$.

The explicit quantifier form of 3.6 is thus

$$\exists x[[ARTIST : x] \wedge [RICH : x]].$$

We shall apply the method to the rest of the examples above. The variable occurence value of each subgraph for each variable is shown below it.

3.7':

$\exists x$	[	$[RICH : x] \wedge \neg\exists z$	[	$[ARTIST : x] \rightarrowtail (PAINT) \rightarrowtail [PAINTING : z]$	]	]
x:	f	t	f	t	f	f
z:	f	f	f	t	f	f

[7]Please notice that this corresponds to normal scopes of variables in FOPL.

3.8’:

$\forall x \forall y$	[	[	$[COLLECTOR : \forall x] \wedge$	$[PAINTING : \forall y]$	$] \rightarrow$	$[COLLECTOR : x] \rightarrowtail (BUY) \rightarrowtail [PAINTING : y]$	]
x:	f	f	t	f	f	t	f
y:	f	f	f	t	f	t	f

Here, the outermost subgraph that has the variable occurence value true (for both x and y) is $[COLLECTOR : \forall x] \rightarrowtail (BUY) \rightarrowtail [PAINTING : \forall y]$.

3.9’:

$\forall x \forall y$	[	[	$[COLLECTOR : \forall x] \wedge$	$[PAINTING : \forall x]$	$] \rightarrow$	$[COLLECTOR : x] \rightarrowtail (BUY) \rightarrowtail [PAINTING : x]$	]
x:	f	f	t	t	f	t	f

3.3.2 Semantics of L_{CG}

The semantics for the language L_{CG} is based on the same principles as the semantics for the language L_{FOPL}. We assume the notions of the value of a term and a replacement of variable in an interpretation to apply to conceptual graphs too.

Let M be a model, v an interpretation function and EA the explicit quantifier form of a graph A. The following set of conditions (CGM) defines when v satisfies a graph in M:

1. Concept nodes:

   Let t be a constant. If EA is of form $[P : t]$, v satisfies the graph if and only if $t^{M,v} \in P^M$.

   If EA is of form $[\neg P : t]$, v satisfies the graph if and only if $t^{M,v} \notin P^M$.

2. Let B and C be concept nodes and t_B and t_C terms such that t_B is the term of B and t_C is the term of C.

   If EA is of form $B \rightarrowtail (R) \rightarrowtail C$, then v satisfies the graph if and only if

   - v satisfies B and
   - v satisfies C and
   - $\langle t_B^{M,v}, t_C^{M,v} \rangle \in R^M$.

3. If EA is of form $B \rightarrowtail (\sim R) \rightarrowtail C$, then v satisfies the graph if and only if

   - v satisfies B and
   - v satisfies C and
   - $(t_B^{M,v}, t_C^{M,v}) \notin R^M$.

4. If EA is of form $\neg[B]$, v satisfies the graph if and only if v does not satisfy B.

5. If EA is of form $[B \vee C]$, v satisfies the graph if and only if v satisfies B or C.

6. If EA is of form $[B \wedge C]$, v satisfies the graph if and only if v satisfies B and C.

7. If EA is of form $(B \rightarrow C)$, v satisfies the graph if and only if it is *not* the case that v satisfies B and v does not satisfy C.

8. If EA is of form $(B \leftrightarrow C)$, v satisfies the graph if and only if

- v satisfies B and C.

  or

- v does not satisfy B and C.

9. If EA is of form $\exists x..$, and there is a node $[B : x]$ in the graph, then v satisfies the graph if and only if $v(x/a)$ satisfies B for some $a \in dom(M)$.

10. If EA is of form $\forall x..$, and there is a node $[B : x]$ in the graph, then v satisfies the graph if and only if $v(x/a)$ satisfies B for all $a \in dom(M)$.

As with FOPL, we define truth in model for a graph as follows: A graph is true in model M if all valuation functions in M satisfy it.

On the basis of these conditions, one can easily observe that the conditions for connectives in conceptual graphs are equal to the conditions for connectives in FOPL. Therefore, the shortcut rules (S1 - S9, S17 - S20) that are related to connectives can be applied to conceptual graphs, too.

3.4 Translations, discussion and summary

In [Sow84], Sowa presents "operator ϕ" that he uses to map conceptual graphs into formulas of FOPL. Here, we consider the possibility of translating (limited) FOPL formulas into conceptual graphs.

On the basis of the conditions and the definition of truth in a model for a graph in Section 3.3.2, it is easy to justify the following analogies between conceptual graphs and FOPL:

- The FOPL formula $P(t)$ is semantically identical with the conceptual graph $[P : t]$, since $P(t)$ and $[P : t]$ are true in M under the same conditions.

- Analogously, the FOPL formula $R(t,u)$ is semantically identical with the conceptual graph $[\top_c : t] \rightarrowtail (R) \rightarrowtail [\top_c : u]$.

Based on these analogies, it is possible to design an algorithm that translates a given FOPL closed formula into a conceptual graph that (according to the conditions) has the same meaning as the formula. The principle of the algorithm is as follows:

Let A be a closed FOPL formula where all the predicates are either unary or binary. Using the inverse predicate formula of Section 3.2.3, it is possible to generate a form of A where the variables in predicates are in the same order as in the quantifier sequence. Moreover, A can be processed into the translation form represented in Section 3.2.4. This form of A can be translated into an explicit quantifier form of a conceptual graph. The resulting conceptual graph can be constructed simply by erasing the quantifier sequences.

Examples of applications of this method are shown in Appendix A.

Given that FOPL formulas with only unary and binary predicates on the one hand and conceptual graphs (with the limited syntax) on the other hand can be translated into each other, it can be claimed that these languages have equal expressive power. This fact has many interesting consequences. It is possible to design a proof theory for conceptual graphs and study the language, its semantics and its proof theory. Together, the syntax, proof theory and semantics form an interpreted system of conceptual graphs in the same manner as there are interpreted systems of FOPL. This kind of system of conceptual graphs was suggested by Sowa in [Sow84] and he maintains that the system is sound and complete.

Moreover, the expressive power of conceptual graphs with limited syntax is greater than that of logic programming, a subset of FOPL.

In the case of equal expressive powers, the use of conceptual graphs could be justified by their "elegance", or ease of use. Appendix A features some typical FOPL formulas (including formulas that contain quantifiers, connectives, etc.) and their Conceptual Graph counterparts. As some of the examples in the appendix demonstrate, the representation using conceptual graphs is sometimes more complex than the one using FOPL.

In an "unlimited" language of conceptual graphs (as presented by Sowa), the term in a concept node can be a set instead of a single term. It is not obvious how these kinds of graphs could be translated into FOPL formulas. Therefore, it is difficult to estimate the properties of an interpreted system that is based on it.

Chapter 4

Concept Calculus, a Functional Approach I

4.1 Introduction

In this chapter, by using set theory we present association relations and operations of concept theory in a well-known and established manner. This chapter brings together two research traditions: concept analysis and conceptual modelling ([Kau67], [Kan83], [Kan93]) and algebraic database definition languages ([TJ92a], [TJ92b], [JN93]). Kauppi's concept theory is concerned with concepts as purely intensional objects. Intensional representation of concepts is essential (see, e.g. [Woo91]) and it has been utilised in Knowledge Representation, especially in Description Logics (see [Bor95], [B$^+$89]). Recently, in Knowledge Representation, the processing of transitivity, and the problems related to it, have become an important field of study (see [Sat95], [AFGP96]). Niemi and Järvelin have presented in [TJ92a] a general framework for processing transitive relations based on set theory. Considering concepts, this kind of a framework can be applied to relationships in the set of concepts. They may be IS-A -relationships as well as transitive part-whole relationships. Our contribution here is to apply main features of this set-theoretical method to the field of concept systems.

According to Bunge in [Bun67], the intension of a concept can be defined as a (possibly infinite) set of properties and relations contained in the concept. In Bunge's (intensional) view, the modeller discovers the relevant part of these properties and relations by abstracting his or her observations in a Universe of Discourse. The problem in Bunge's approach is that the structure of the intension is not explicit.

The containment relation between two concepts can be studied according to Bunge's idea, i.e. the generality of concept is defined by "subsumption" between the intensions of these two concepts. For instance, the concept of **centaur** should be more general than the concept of **Greek centaur**, provided that all the properties of **centaur** are properties of **Greek centaur**, too. This is the case, of course, even though the extensions of both **centaur** and **Greek centaur** are empty.

In this chapter, we do not adopt Bunge's idea of infinite intension. Bunge, however, presents a notion of finite core intension that consists of the characteristics (earmarks) of the concept. We assume the characteristics of a concept to be the concepts that are contained in it. For instance, characteristics of the concept of **dog** could be **mammal**, **quadruped** and **carnivore**. But in addition to these, the concept of **dog** would include the characteristics of **mammal**, **quadruped** and **carnivore**, like **living being**, **thing**, etc.

Bunge discusses the containment among intensions, but Kauppi in [Kau67] bases the intension of a concept on the relation of intensional containment. Kauppi's notion of intensional containment is different from Bunge's; for Kauppi the relationship of intensional containment is a basic relationship in the set of concepts.

We observe, too, that Bunge's "generality" between concepts is very close to the well-known IS-A-relationship (see [Bra83]). Nilsson and Palomäki in [NP98] seem to identify intensional containment with IS-A relationship. According to Kangassalo in [Kan96], this is, however, only a "special case" of intensional containment.[1]

While intensional containment as such is significant, our concern is the explicit representation of concept systems that are based on this relation. In our study, some limitations are enforced on the relation of intensional containment on a finite set of concepts. These limitations are imposed by a concept theory, like a simplified version of Kauppi's axiomatised concept theory that is discussed in this chapter. In this way, the concept theory forms a firm basis for discussion of concept systems. A concept system can be utilised in the process of conceptual modelling. Finally, the set-theoretical formulation that is presented in this chapter makes the representation and processing of these concept systems explicit.

In the following sections we first introduce some basic mathematical conventions for representing a concept theory. In Section 4.3, a simple concept calculus (in the sense of Kauppi [Kau67]) will be outlined. In Section 4.4, we introduce our definition for the intension of concept. We discuss how the concept system can be assumed to imitate a Universe of Discourse (UoD). Section 4.5 gives a more formal treatment for concepts and concept algebra and defines the concept system on its basis. Finally, Section 4.6 contains a short summary and items for further study.

4.2 Mathematical notational conventions

For basic definitions that are not presented here, see Section 1.7. Here, we repeat the most essential ones.

A binary relation R is reflexive in the set A if for every $x \in A$, xRx (c.f. [Sup57]). In other words, the ordered pair $\langle x,x\rangle \in R$. The relation R is irreflexive in the set A, if for every $x \in A$, it is not the case that xRx.

We now define the reflexive relation Ref_A in the set A as follows:

$$Ref_A = \{\langle x,x\rangle | x \in A\}.$$

If R is a relation in A then $R - Ref_A$ does not contain those ordered pairs of R in which the first and the second elements are same (i.e. reflexive ordered pairs).

A relation R is symmetric in the set A if for every x and y in A, whenever xRy, then yRx (c.f. [Sup57]). In other words, if for every $\langle x,y\rangle \in R$, $\langle y,x\rangle \in R$.

A relation R is antisymmetric in the set A if for every x and y in A, whenever xRy and yRx, then $x = y$ (c.f. [Sup57]). In other words, $\langle x,y\rangle \in R$ and $\langle y,x\rangle \in R$ implies $x = y$.

A relation R is transitive in the set A if for every x,y and z in A, whenever xRy and yRz, then xRz (c.f. [Sup57]). In other words, if $\langle x,y\rangle \in R$ and $\langle y,z\rangle \in R$, then $\langle x,z\rangle \in R$.

A relation R is a partial ordering of the set A if and only if R is reflexive, antisymmetric, and transitive in A (c.f. [Sup57]).

If R is a partial ordering of A, then A is called a partially ordered set, or a poset.

[1]As Hautamäki points out in [Hau86], Kauppi does not give a model to her concept calculus (in the sense of L-model of Section 3). Therefore, we probably cannot be sure if the models or interpretations provided by Nilsson and Palomäki, on the one hand, and this section, on the other hand, correspond to Kauppi's intentions of the scope of her calculus. However, Kauppi's theory appears to be fruitfull and flexible enough to allow interesting models and interpretations. Hautamäki provides one model by means of sets of determinables and their values (see [Hau86]).

We define the composition of two relations R_1 and R_2, denoted by $R_1 \circ R_2$, as follows:

Let R_1 be a relation from A to B, and let R_2 be a relation form B to C. Then $R_1 \circ R_2$ is the relation from A to C such that if $\langle x,y\rangle \in R_1$ and $\langle y,z\rangle \in R_2$, then $\langle x,z\rangle \in R_1 \circ R_2$.

The composed relation $R \circ R$, is denoted by R^2, $R \circ R \circ R$, is denoted by R^3 etc. For simplicity, we define $R^1 = R$.

Generally, we denote by $\cup_{i=1}^{n} R^i$ the union $R^1 \cup R^2 \cup .. \cup R^n$. The union $\cup_{i=1}^{\infty} R^i$ is known as the transitive closure of R. This is often denoted as R^+. Intuitively, if $\langle a,b\rangle \in R^+$ then an immediate or indirect relationship exists between a and b in R.

The reflexive transitive closure of the relation R in the set A is the union of the transitive closure of R and the reflexive relation in the set A,

$$R^* = R^+ \cup Ref_A.$$

4.3 Intensional containment and concept systems

In this section, Kauppi's theory is outlined in a very restricted form. Kauppi designed her theory to apply to infinite concept systems as well as finite ones. Therefore, some definitions and axioms of this section can be a bit overlapping. For a more detailed explanation of Kauppi's theory, see [Pal94].

There are only two ontological primitives in the theory: concept and the *relation of intensional containment*. This basic relation is denoted by

$$\mathbf{a} \geqslant \mathbf{b},$$

and reads "concept **a** intensionally contains concept **b**" or "concept **b** is intensionally contained in concept **a**". We can also say that "**b** is a characteristic of **a**" or "**a** has a characteristic **b**".

The first axiom of the theory is reflexivity, so every concept contains intensionally itself:

$$Ax_{ref} \quad \forall \mathbf{x}(\mathbf{x} \geqslant \mathbf{x}).$$

The second axiom is transitivity, i.e. if **x** contains intensionally **y** and **y** contains intensionally **z**, then **x** contains intensionally **z**,

$$Ax_{tr} \quad \forall \mathbf{x} \forall \mathbf{y} \forall \mathbf{z}(\mathbf{x} \geqslant \mathbf{y} \wedge \mathbf{y} \geqslant \mathbf{z} \rightarrow \mathbf{x} \geqslant \mathbf{z}).$$

The intensional equivalence means conceptual equivalence and it is defined as:

$$Df_{\equiv} \quad \mathbf{a} \equiv \mathbf{b} =_{df} \mathbf{a} \geqslant \mathbf{b} \wedge \mathbf{b} \geqslant \mathbf{a}.$$

The definition $Df_{\equiv}$ states that two concepts **a** and **b** are equivalent if and only if **a** contains intensionally **b** and **b** contains intensionally **a**. The definition $Df_{\equiv}$ is not given as an axiom in Kauppi's theory. In spite of that, we assume that the relation of intensional containment is anti-symmetric. Therefore, the relation of intensional containment forms a partial ordering in a set of concepts.

With intensional containment Kauppi defines the relation of *strict containment*. Concept **a** contains intensionally strictly concept **b**, if and only if **a** contains intensionally the concept **b** and they are not equivalent:

$$Df_{>} \quad \mathbf{a} > \mathbf{b} =_{df} \mathbf{a} \geqslant \mathbf{b} \wedge \neg(\mathbf{b} \equiv \mathbf{a}).$$

Kauppi does not present any definition for *immediate containment*, since her study applies to any infinite concept system. In our study, only finite concept systems are considered, and for them the following definition applies. Concept **a** contains immediately concept **b** ($\mathbf{a} \succ_{ic} \mathbf{b}$), if and only if **a** contains intensionally strictly **b**, and if there is no concept "between" **a** and **b**,

$$Df_{\succ_{ic}} \quad \mathbf{a} \succ_{ic} \mathbf{b} =_{df} \mathbf{a} > \mathbf{b} \wedge \neg\exists \mathbf{x}(\mathbf{a} > \mathbf{x} \wedge \mathbf{x} > \mathbf{b}).$$

Based on the relation of intensional containment among concepts, some further concept association relations between concepts have been defined. Given two concepts, if there is a concept that is intensionally contained in both of them we call these concepts *comparable*. This is refered to by symbol H, and the definition is

$$Df_H \quad \mathbf{a}H\mathbf{b} =_{df} \exists \mathbf{x}(\mathbf{a} \geqslant \mathbf{x} \wedge \mathbf{b} \geqslant \mathbf{x}).$$

Concepts **a** and **b** are *incomparable* if there is no concept that is intensionally contained in both of them. It is refered to by symbol $\overset{\top}{\bot}$, and the definition is

$$Df_{\overset{\top}{\bot}} \quad \mathbf{a}\overset{\top}{\bot}\mathbf{b} =_{df} \neg\exists \mathbf{x}(\mathbf{a} \geqslant \mathbf{x} \wedge \mathbf{b} \geqslant \mathbf{x}).$$

Another way to observe indirect connection between concepts **a** and **b** is compatibility. If there exists a concept, **x**, that intensionally contains both **a** and **b**, it is said that these concepts are *compatible*. It is refered to by symbol $\underset{\wedge}{|}$ and its definition is

$$Df_{\underset{\wedge}{|}} \quad \mathbf{a} \underset{\wedge}{|} \mathbf{b} =_{df} \exists \mathbf{x}(\mathbf{x} \geqslant \mathbf{a} \wedge \mathbf{x} \geqslant \mathbf{b}).$$

The opposite of compatibility is incompatibility. Concepts **a** and **b** are *incompatible* if there is no concept that could make **a** and **b** compatible. It is refered to by symbol $\overset{\vee}{|}$, and the definition is

$$Df_{\overset{\vee}{|}} \quad \mathbf{a} \overset{\vee}{|} \mathbf{b} =_{df} \neg\exists \mathbf{x}(\mathbf{x} \geqslant \mathbf{a} \wedge \mathbf{x} \geqslant \mathbf{b}).$$

If concept **a** is contained intensionally in every concept, then **a** is equivalent with the so called *general concept* **G**:

$$Df_G \quad \mathbf{a} \equiv \mathbf{G} =_{df} \forall \mathbf{x}(\mathbf{x} \geqslant \mathbf{a}).$$

According to Kauppi's theory there exists a concept that is contained intensionally in every concept,

$$Ax_G \quad \exists \mathbf{x} \forall \mathbf{y}(\mathbf{y} \geqslant \mathbf{x}).$$

From the axiom Ax_G it follows that any two concepts are comparable.

A *special concept* is a concept that is not contained intensionally in any other concept than itself. There can be several special concepts in Kauppi's system. The property of a concept being a special concept is denoted by S. Concept **a** being a special concept is defined as follows:

$$Df_S \quad S(\mathbf{a}) =_{df} \forall \mathbf{x}(\mathbf{x} \geqslant \mathbf{a} \rightarrow \mathbf{a} \geqslant \mathbf{x}).$$

For every concept **y** there must be one or several special concepts so that **y** is contained in them:

$$Ax_S \quad \forall \mathbf{y} \exists \mathbf{x}(S(\mathbf{x}) \wedge \mathbf{x} \geqslant \mathbf{y}).$$

As indicated earlier, Kauppi's system includes exactly one general concept and one or several special concepts.

The *concept operations* in the theory are intensional sum, intensional product, intensional negation, intensional reciprocal, intensional difference and intensional quotient. In the scope of this chapter, we introduce only intensional product and intensional sum. In standard lattice theory, the counterpart for intensional product is the greatest lower bound (glb) operation and the counterpart for intensional sum is the least upper bound (lub) operation.

Concept **c** is equivalent with the *intensional product* of the concepts **a** and **b**, if and only if **c** contains intensionally every concept that **a** and **b** contain.

$$Df_{\otimes} \quad \mathbf{c} \equiv \mathbf{a} \otimes \mathbf{b} =_{df} \forall \mathbf{x}(\mathbf{c} \geqslant \mathbf{x} \leftrightarrow \mathbf{a} \geqslant \mathbf{x} \wedge \mathbf{b} \geqslant \mathbf{x}).$$

The axiom of intensional product states that if concepts **a** and **b** are comparable, then their product exists:

$$Ax_{\otimes} \quad \mathbf{a} H \mathbf{b} \rightarrow \exists \mathbf{x}(\mathbf{x} \equiv \mathbf{a} \otimes \mathbf{b}).$$

Because we have here assumed the existence of the general concept, the intensional product between any two concepts naturally exists in any valid concept system.

Another intensional operation considered here is intensional sum. Concept **c** is equivalent with the *intensional sum* of the concepts **a** and **b**, if for every concept **x** it holds that **x** contains intensionally **c** if and only if **x** contains intensionally **a** and **b**.

$$Df_{\oplus} \quad \mathbf{c} \equiv \mathbf{a} \oplus \mathbf{b} =_{df} \forall \mathbf{x}(\mathbf{x} \geqslant \mathbf{c} \leftrightarrow \mathbf{x} \geqslant \mathbf{a} \wedge \mathbf{x} \geqslant \mathbf{b}).$$

The axiom of intensional sum states that if concepts **a** and **b** are compatible, then their intensional sum exists:

$$Ax_{\oplus} \quad \mathbf{a} \mathrel{\overset{\vert}{\scriptstyle\wedge}} \mathbf{b} \rightarrow \exists \mathbf{x}(\mathbf{x} \equiv \mathbf{a} \oplus \mathbf{b}).$$

The axioms $Ax_{\otimes}$ and $Ax_{\oplus}$ bring some demands to the form of a concept system. For example, the form of Figure 4.1 is forbidden, because there is no intensional sum for **a** and **b**, and no intensional product for **c** and **d**. In Section 4.6, we discuss more closely the example in Figure 4.1.

In Figures 4.1 and 4.2, any upper level concept contains intensionally the lower level concepts.

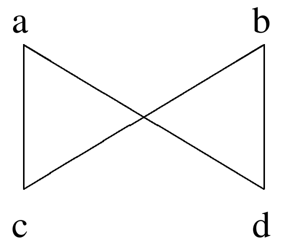

Figure 4.1: A concept structure that conflicts with $Ax_{\otimes}$.

There is also an axiom related to distributivity in Kauppi's theory. We do not, however, assume the axiom in question, because we want to accept structures like the one in Figure 4.2.

With some additions (further axioms), Palomäki in [Pal94] has demonstrated that Kauppi's concept theory forms a distributive structure, a complete semilattice. The system we consider is not as strong as a pure Kauppi's system. It is worth noticing that the set of axioms presented here does not require the concept system to be a lattice, but a semilattice.

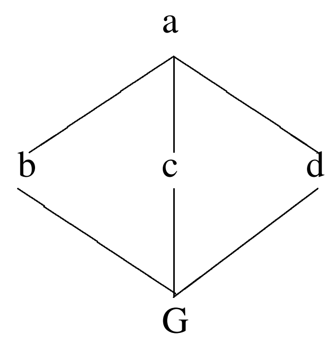

Figure 4.2: A non-distributive structure.

4.4 The intension of a concept

In this section, we define the intension of a concept as a structure based on the intensional containment relation. The starting point is to bring together Kauppi's and Bunge's views of characteristics for concepts.

Bunge in [Bun67] defines the intension of a concept as "the set of properties and relations P_i subsumed under the concept or which the concept, so to speak, synthesizes." This is possibly an infinite set. Bunge proposes, however, that we can study a finite subset of this set. This relevant subset, the core intension of concept **c**, $I_{core}(\mathbf{c})$, is a finite set of the characteristics contained in **c**,

$$I_{core}(\mathbf{c}) = \{P_1, P_2, .., P_n\}.$$

We assume in this chapter that these characteristics are concepts, too. We shall also assume that the concept **c** belongs to the set of its characteristics.

The set of characteristics is finite, but the problem for conceptual modelling is that it is uncertain if all the characteristics can be found. A characteristic of concept **c** is, for example, a property or a concept that is more general than **c**. Thus, being a **mammal** is a characteristic for the concept of **dog**. In this way, there should be a structure in the set of characteristics, but by using a simple set of characteristics we cannot present this. Next we describe the structure for the characteristics of a concept.

According to Kauppi's approach in [Kau67], concept **c** intensionally contains concepts and we assume that these contained concepts correspond to Bunge's characteristics. The characteristics are thus the concepts intensionally contained in **c**. Contrary to Bunge, our definition of intension is based on the relation of intensional containment within concepts. We express it as *a set of ordered pairs* of concepts. In this respect, the approach is more accurate than Bunge's. It also incorporates the view of Kauppi's axiom Ax_{ref}, i.e. the intension of the concept containing the concept itself (here, of course, in the form of an ordered pair).

Here, $I(\mathbf{c})$ denotes a set of ordered pairs $\langle \mathbf{x}, \mathbf{y} \rangle$ such that the intensional containment relation holds among **x** and **y** and they belong to the set of characteristics of concept **c**. $I(\mathbf{c})$ of concept **c** is thus the relation $\geqslant$ in the core intension, $I_{core}(\mathbf{c})$ for **c**,

$$I(\mathbf{c}) = \{\langle \mathbf{x}, \mathbf{y} \rangle | \mathbf{x}, \mathbf{y} \in I_{core}(\mathbf{c}) \land \mathbf{x} \geqslant \mathbf{y}\}.$$

$I(\mathbf{c})$ of concept **c** is finite and it is a subset of a Cartesian product $I_{core}(\mathbf{c}) \times I_{core}(\mathbf{c})$. Kangassalo in [Kan82] suggests a related kind of definition for the intension of concept. His definition includes other relations in addition to intensional containment.

In conceptual modelling, a limited part of the real world is relevant from the view point at hand. This relevant part is called the Universe of Discourse (UoD). UoD is often understood simply as a collection of individuals, but we prefer Kauppi's and Palomäki's more explicit approach (in [Kau67] and [Pal94], respectively).

Applying a modified version of Palomäki's notation, a UoD is a sequence $\langle V, X, F \rangle$, where V is a universe of individuals, X is a universe of concepts and F is a binary "falls under" -relation that is a subset of $V \times X$. We say that the individual i belongs to the extension of concept $\mathbf{c}$, if i falls under concept $\mathbf{c}$, i.e. if $\langle i, \mathbf{c} \rangle$ belongs to F. Moreover, the intersection $V \cap X$ is not empty. According to Palomäki in [Pal94], the fact that the intersection of V and X is not empty "follows from the Platonic assumption that some concepts can be considered as individuals, since, for instance, we can talk about them".

The intension of concept $\mathbf{c}$ in the UoD, denoted by $I_{UoD}(\mathbf{c})$, can now be expressed using ordered pairs as follows:

$$I_{UoD}(\mathbf{c}) = \{\langle \mathbf{x}, \mathbf{y} \rangle | \mathbf{x}, \mathbf{y} \in X \wedge \langle \mathbf{x}, \mathbf{y} \rangle \in I(\mathbf{c})\}.$$

Axiomatised concept systems are abstract structures that impose some constraints on the relationships between concepts. However, there is a connection between concept systems and concepts that are abstracted from the UoD. A concept system can be a conceptual model of the UoD. Suppose we have a UoD called UoD_1 and a concept system CS_1 that contains concepts abstracted from UoD_1. As we have introduced above, we have a strong metaphysical approach that there really are concepts in a UoD and that they subsist relatively independently of us and can be modelled (see [Kau67]). Having this approach, it is natural to think that the concepts of UoD_1 form a (possibly infinite or very large) concept system.

For practical reasons we cannot handle concept systems that large. But let us assume that the modeller has discovered a set of concepts, X_{CS1} from UoD_1; that is, $X_{CS1} \subset X$. It is reasonable to assume that the intensional containment relation in X_{CS1} is a subset of the intensional containment relation on the concepts of UoD_1 in general.

Following this idea, the aim of conceptual modelling is to find the most relevant concepts from the UoD at hand and to present intensional containment between them. The relevant concepts form a set (in what follows, C-set) and intensional containment is a partial ordering in this set (in what follows, I-rel).

In the next section, we present tools to check the form (validity) of such a concept system. If the concept system is not valid, the modeller can add new concepts and relations to the system until it becomes valid. In this way, the functions we define can help the modeller to find the relevant concepts and relations. The concepts the modeller thus finds may be such that they do not appear explicitly in a preliminary modelling process.

4.5 A functional representation of concept systems

In this section, we first present the most important concept association relations and operations for a concept system. After that we shall define the legal check function that expresses the same axiomatisation that was given in Section 4.3.

Let *C-set* be a subset of concepts of a specific UoD.

Since the wine and art examples of Chapters 2 and 3 are too large for the scope of this chapter, we will consider the following set:

$$C\text{-}set_1 = \{G, gadget, radio, clock, clock\_radio\}.$$

In each concept system there is an intensional relation *I-rel* in the set *C-set*. This relation is a binary relation that corresponds to immediate containing of concepts presented in Section 4.3.

Formally, *I-rel* is a subset of the Cartesian product of *C-set*, $I\text{-}rel \subseteq C\text{-}set \times C\text{-}set$. The interpretation of *I-rel* is: if $\langle a,b\rangle \in I\text{-}rel$, then the concept **a** immediately contains the concept **b**, $a \succ_{ic} b$.[2]

In our example, the relation $I\text{-}rel_1$ in the set $C\text{-}set_1$ is the following binary relation:

$$I\text{-}rel_1 = \{\langle gadget, G\rangle, \langle radio, gadget\rangle, \langle clock, gadget\rangle, \langle clock\_radio, radio\rangle, \\ \langle clock\_radio, clock\rangle\}.$$

Given an *I-rel*, we can compute the corresponding $I\text{-}rel^*$, which is the reflexive transitive closure of *I-rel*. The relation $I\text{-}rel^*$ corresponds to the intensional containment relation considered in Section 4.4. In this approach we define a concept system to be an *I-rel* in a *C-set*.

The reflexive transitive closure of $I\text{-}rel_1$ is $I\text{-}rel_1^* = \{\langle gadget, G\rangle, \langle radio, G\rangle, \langle clock, G\rangle, \langle clock\_radio, G\rangle, \langle radio, gadget\rangle, \langle clock, gadget\rangle, \langle clock\_radio, gadget\rangle, \langle clock\_radio, radio\rangle, \langle clock\_radio, clock\rangle, \langle G, G\rangle, \langle gadget, gadget\rangle, \langle clock, clock\rangle, \langle radio, radio\rangle, \langle clock\_radio, clock\_radio\rangle\}$.

In the following functions, *I-rel* and *C-set* are considered global when they are not given explicitly.

The function *genuses* computes the set of all such concepts in which no other concepts are contained immediately or indirectly. *Genuses* is a function from the power set of the Cartesian product $C\text{-}set \times C\text{-}set$ to the power set of *C-set*.

$genuses : P(C\text{-}set \times C\text{-}set) \to P(C\text{-}set)$
$genuses(I\text{-}rel) = \{x \mid \neg\exists y \in C\text{-}set : \langle x,y\rangle \in I\text{-}rel\}.$

In our example, $genuses(I\text{-}rel_1)$ produces the set $\{G\}$.

Respectively, the function *specie* computes the set of all those concepts that are contained in no other concepts than themselves in a given concept system.

$specie : P(C\text{-}set \times C\text{-}set) \to P(C\text{-}set)$
$specie(I\text{-}rel) = \{x \mid \neg\exists y \in C\text{-}set : \langle y,x\rangle \in I\text{-}rel\}.$

Related to our example, $specie(I\text{-}rel_1) = \{clock\_radio\}$.

The Boolean functions *compatible* and *comparable* check whether two concepts are compatible or comparable (see Section 4.3). We define functions *incompatible* and *incomparable* on their basis.

$compatible : C\text{-}set \times C\text{-}set \to \{false, true\}$
$compatible(a,b) =$

$$\begin{cases} \text{true, if } |\{x \mid \langle x,a\rangle \in I\text{-}rel^* \wedge \langle x,b\rangle \in I\text{-}rel^*\}| > 0, \\ \text{false, otherwise.} \end{cases}$$

In our example, $compatible(clock, radio)$ is true, since *clock_radio* intensionally contains both *clock* and *radio*.

$incompatible : C\text{-}set \times C\text{-}set \to \{false, true\}$

[2] In the following examples, the names of concepts, as well as the names of the functions, will be written in italics, not in boldface. This is due to the convention that names are (terminal) symbols of a given language.

$incompatible(a,b) =$

$$\begin{cases} \text{true, if not } compatible(a,b), \\ \text{false, otherwise.} \end{cases}$$

$comparable : C\text{-}set \times C\text{-}set \rightarrow \{false, true\}$
$comparable(a,b) =$

$$\begin{cases} \text{true, if } |\{x | \langle a,x \rangle \in I\text{-}rel^* \wedge \langle b,x \rangle \in I\text{-}rel^*\}| > 0, \\ \text{false, otherwise.} \end{cases}$$

In our example, *comparable(clock,radio)* is true, since *G* and *gadget* are intensionally contained in both *clock* and *radio*.

$incomparable : C\text{-}set \times C\text{-}set \rightarrow \{false, true\}$
$incomparable(a,b) =$

$$\begin{cases} \text{true, if not } comparable(a,b), \\ \text{false, otherwise.} \end{cases}$$

In $I\text{-}rel_1$, no two concepts are incomparable.

Intensional product and intensional sum for any two concepts in a *I-rel* can be expressed in the following two functions:

$int\_sum : C\text{-}set \times C\text{-}set \rightarrow P(C\text{-}set)$
$int\_sum(a,b) =$
$\{c | \forall x \in C\text{-}set : (\langle x,a \rangle \in I\text{-}rel^* \wedge \langle x,b \rangle \in I\text{-}rel^* \leftrightarrow \langle x,c \rangle \in I\text{-}rel^*)\}.$

$int\_prod : C\text{-}set \times C\text{-}set \rightarrow P(C\text{-}set)$
$int\_prod(a,b) =$
$\{c | \forall x \in C\text{-}set : (\langle a,x \rangle \in I\text{-}rel^* \wedge \langle b,x \rangle \in I\text{-}rel^* \leftrightarrow \langle c,x \rangle \in I\text{-}rel^*)\}.$

Here, the functions return sets instead of single concepts. In what follows, we check if the functions return a set of only one element. If we want to ensure that the intensional sum and product correspond to Kauppi's definitions, this element is "the" sum or "the" product.

The concept system is said to be operation-legal, if and only if the following points hold:

- For any two concepts *a* and *b* in the *C-set*, if the concepts are compatible, there is exactly one intensional sum of *a* and *b*.

- For any two concepts *a* and *b* in the *C-set*, if the concepts are comparable, there is exactly one intensional product of *a* and *b*.

The function *legal_check_operations* checks if the concept system is operation-legal for given two concepts.

$legal\_check\_operations : C\text{-}set \times C\text{-}set \rightarrow \{false, true\}$
$legal\_check\_operations(a,b) =$

$$\begin{cases} \text{true, if } compatible(a,b) \rightarrow |int\_sum(a,b)| = 1 \wedge \\ \qquad comparable(a,b) \rightarrow |int\_prod(a,b)| = 1, \\ \text{false, otherwise.} \end{cases}$$

Now we can construct the legal check for a concept system. Having the axioms we presented in Section 4.3, there can be only one member in the set generated by the function *genuses* in the concept system. From that it follows also that if the concept system is seen as a graph, it must be connected. Moreover, there must be checking for concept operations in the legal check function.

Function *legal_check* checks if the concept system given by its *I-rel* is legal.

$legal\_check : C\text{-}set \times C\text{-}set \to \{false, true\}$ $legal\_check(I\text{-}rel) =$

$$\begin{cases} \text{true, if } \forall a \in C\text{-}set, \forall b \in C\text{-}set : legal\_check\_operations(a,b) \wedge \\ \qquad |genuses(I\text{-}rel)| = 1, \\ \text{false, otherwise.} \end{cases}$$

Applying the function *legal_check* we can see that $I\text{-}rel_1$ is legal.

When the legality of a concept system has been inspected, we can utilise, for example, the following simple method for computing the intensional product for the concepts **a** and **b**:

$$a \otimes b = \{x | \langle a, x \rangle\} \cap \{y | \langle b, y \rangle\}.$$

In this section, we have presented an abstract implementation for a part of Kauppi's theory. We have done that in order to give principles of implementations of concept systems. In the next section, we discuss its relevance to conceptual modelling and concept detection.

4.6 Summary and discussion: Concept calculi in conceptual modelling

In this chapter, we have discussed connections between conceptual modelling, Kauppi's concept theory and its abstract implementation. Here, we have presented a part (the core) of Kauppi's theory. In our approach, *C-set* is a finite set of concepts and *I-rel* is a binary relation on this set. Together *C-set* and *I-rel* can be interpreted to represent a relevant part of the Universe of Discourse. Since *I-rel* should reflect a UoD, it is not an arbitrary relation in *C-set*. We have provided functions that check the validity of a given *I-rel*. We have explicitly presented some association relations (e.g. compatibility) and concept operations of Kauppi's concept theory in a well-known and established formalism. Based on the defined functions, we have formed a validity check for an axiomatised concept system. The contribution of this validity check is in providing us a tool that helps in concept detection in conceptual modelling. In a limited case this would mean that the modeller realises that the model should be revised e.g. by adding new concepts, if it is not valid. For the actual process of detecting what these missing concepts are, see [Kan92].

In this chapter, we have presented only a skeleton model of Kauppi's concept theory. To serve the purposes of conceptual modelling, our version is simpler than Kauppi's. In fact, Kauppi in [Kau67] presented several axiomatisations, each of which of course resulted in different requirements for a concept system. She also presented more operations than intensional sum and intensional product. The other operations, discussed in Chapter 5, are intensional difference, quotient, reciprocal and negation. The axiomatisations that make use of intensional reciprocal and difference omit the axiom Ax_G (general concept).

Järvelin and Niemi have shown in [JN93] that an algebraic approach is suitable for management of relationships which are very close to IS-A relationship. We assume IS-A relationship to be a clear example of intensional containment. It is useful to study what kind of a set of axioms would be fruitful in practical conceptual modelling. In this chapter, the validity check has been done to check the consistency between a proposed concept system and an adaptation of Kauppi's concept theory.

According to the theory, for any two comparable concepts there is exactly one intensional product, and for any two compatible concepts, one intensional sum. These axioms make concept structures like the one in Figure 4.1 illegal. In many cases, this is fruitful for concept detection. For instance, in Figure 4.1, we would easily find concepts that make it legal, when added to the approppriate places in the concept system.

It can be the case that there are axiomatised concept structures that are more fruitful for different purposes in conceptual modelling than the one presented in this chapter. For instance, handling intensional negation and the relations incomparability and incompatibility may be applicable in cases where a modeller needs ways to incorporate new knowledge into his concept system, especially in the case where the new knowledge is inconsistent with the existing one. Moreover, using a proper set of axioms, we can provide the users with a structure that is useful in many activities of conceptual modelling. These include e.g. finding new concepts, locating concepts and linking together various concept systems.

Chapter 5

Concept Calculus, a Functional Approach II: Concept Association Relations and Operations

5.1 Introduction

In order to consider a structural framework for knowledge representation, finding the basic elements of a knowledge structure, i.e. concepts, is needed. In several areas of research, the conceptual framework is the basis of presentation of knowledge. For example, in DL (description logics) (see e.g. [BS92], [Bor95]) and in semantic networks (see e.g. [BS85]), one primary issue of the research has been to find a framework for representation of knowledge. A recent contribution to knowledge representation has been formal ontology and different ontological systems. An ontology forms the basis for an explicit presentation of knowledge. In general, according to Guarino in [Gua97], an ontology is a meta-level description of knowledge representation.[1] In a DL, the knowledge structure is generated by classification of individuals and by forming IS-A relationships and roles between class concepts. The part-whole relationships have been a recent object of interest and dispute in DL (see e.g. [AFGP96], [Lam96]). According to Kangassalo in [Kan96], intensional containment can cover IS-A relationships as well as part-whole ("has a component") relationships.

Our contribution here is to form an abstract implementation based on a large part of Kauppi's concept theory by extending the discussion of Chapter 4 to cover Kauppi's concept association relations and operations.

Basically, this means finding and representing those parts of Kauppi's theory that can be utilised in knowledge representation and for analysing concept structures. For its representation we use standard set theory and our formalism is based on that used by Niemi and Järvelin in [TJ92a]. There, and in [JN93], Järvelin and Niemi have demonstrated that this kind of an approach is indeed suitable for the management of those relationships that are very close to the IS-A relationship.

In the following version of concept theory there are three axioms to start our system with. We assume that the relation of the intensional containment is reflexive, transitive and antisymmetric. The other axioms of Kauppi's concept theory determine the structure of the concept system further (see [Pal94]). We do not adopt all those axioms, because we prefer to allow a more opulent

[1]It should be noticed that in computer science term "ontology" is used in a limited sense, whereas in philosophy ontology is understood as "the theory or study of being as such; i.e., of the basic characteristics of all reality" (c.f. [eb-94c]).

structure in the set of concepts. Some of these axioms, as well as functions to check the legality of a concept system that is based on them, have been presented in Chapter 4. In this chapter, we present some extensions to managing concept structures, namely that the results of the functions can be a set of concepts instead of a unique concept. According to our approach, we get some tangible results of the operations, although the structure of the concept system, at hand, does not comply with such a strict axiomatisation as in the original theory.

In Kauppi's terminology, the notions of the greatest (Ger.: grösste) and the least (Ger.: kleinste) concept appear frequently. The greatest concept is the concept which contains intensionally all the other concepts in the collection (structure) that is created using some relevant principles. Because our approach is set-oriented, we call a *maximal* concept any concept that is not contained in any other concept, apart from itself, in a given set of concepts. Analogously, we say that a concept is *minimal* if there does not exist any concept that contains intensionally this concept in a given set. The sets of minimal concepts and maximal concepts play a significant role in our representation.

By the relation of intensional containment, further relations between two concepts are defined. We call these *concept association relations*, meaning that there are indirect relationships between two concepts. When we consider the relationships between some given concepts, and their associations to the whole concept system, we refer to those relationships as concept operations. Intuitively, the principle behind the formation of concept operations is to restrict the set of concepts using some relevant principles, such as the intensional containment relation and the concept association relations, and consider the maximal or the minimal concepts under these restrictions. According to this method, we first form the "restricted" set of concepts, and then we find the maximal or minimal concepts in this set. If the concept system satisfies Kauppi's axioms as presented in [Kau67], then our functions return a set where there is exactly one concept.

By our set oriented extension of the concept theory, some decisive advances in the analysis of concept structures are gained. We can handle structures that are based on defective knowledge and, respectively, structures that are not based on a strict axiomatisation, like lattice theory.

In the following example, the concept of **electric bulb** contains intensionally three other concepts: **electrical equipment, glass** and **luminous**. Similarly, the concept of **glow-worm** contains intensionally two other concepts: **alive** and **luminous**. This is presented in Figure 5.1, which uses the illustration based on CONCEPT D [Kan83], where an upper level concept contains lower level concepts intensionally. It is evident that this structure is far from being a lattice, but our approach gives tools to analyse structures like this.

We assumed, for example, that electric bulb contains both electrical equipment and luminous as characteristics. ing luminous. We say, therefore, that the concepts **electrical equipment** and **luminous** are compatible. As an example of concept operations, we consider the intensional negation (see Section 5.7), by which we can study the remoteness of concepts by means of analysing concepts that are not compatible. The intensional negation of **electric bulb** is **living**, but there is no unique negation for **glow-worm** or **living**. According to our approach, the intensional negation of these concepts is the following set $\{$**electrical equipment, glass**$\}$. In the concept system of this example these are just the basic primitives that share no features with the concept **living**.

By using functions and set theory as tools of representation, we present in an explicit and well-known manner the formal concept theory, which we see as having the potential to manage knowledge and to describe the structure of knowledge. In this chapter, we generally consider concept theory at a rather formal level and without any inherent ontological or semantic attachments.

In Section 5.2 we present our notations, and in Section 5.3 we introduce the basics of our functions. Simple intensional relations are introduced in Section 5.4, and the functions of association relations are presented in Section 5.5. Concept operations are introduced in Sections 5.6-5.8.

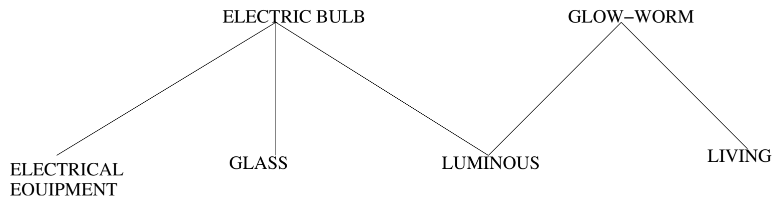

Figure 5.1: Intensional containment

In Section 5.9 we shortly discuss how our system can be applied to different fields of knowledge representation. Kauppi's original definitions of association relations and operations, as discussed in this chapter, are presented in Appendix B.

5.2 Basic notations

In addition to the conventions presented in Section 1.7 and Section 4.2, the following will be used in this chapter:

1. We call ordered n-tuples *tuples* for brevity. The empty tuple (0-tuple) is denoted by $\langle\rangle$.
2. The tuple set of a set S is denoted by $T(S)$. A tuple is an element of $T(S)$ if all elements of the tuple are different and they are members in the set S.

   For example, if $S = \{a,b,c\}$ then $T(S)$ is $\{\langle\rangle$, $\langle a\rangle$, $\langle b\rangle$, $\langle c\rangle$, $\langle a,b\rangle$, $\langle b,a\rangle$, $\langle a,c\rangle$, $\langle c,a\rangle$, $\langle b,c\rangle$, $\langle c,b\rangle$, $\langle a,b,c\rangle$, $\langle a,c,b\rangle$, $\langle b,a,c\rangle$, $\langle b,c,a\rangle$, $\langle c,a,b\rangle$, $\langle c,b,a\rangle\}$.

5.3 Concept systems and primary functions

C-set is the finite set of concepts of a specific concept system. To be exact, C-set consists of the names of the concepts that subsist in the Universe of Discourse. However, to avoid confusion, we call the elements of C-set concepts. As an example, we consider the set *C-set1* $= \{G, gadget, radio, clock, clockradio\}$, where G is the so-called general concept that is contained intensionally in every concept. Except in *C-set1*, we do not assume the existence of a single general (minimal) concept in a set of concepts.

In each concept system, there is an intensional binary relation *I-rel* in C-set. This corresponds to immediate containment among two concepts (see Chapter 4), i.e. if $< a,b > \in$ *I-rel*, then concept **a** immediately contains concept **b** intensionally.

As in Chapter 4 , the relation *I-rel1* in *C-set1* is the following binary relation:

I-rel1 $= \{\langle clockradio, radio\rangle, \langle clockradio, clock\rangle, \langle radio, gadget\rangle, \langle clock, gadget\rangle, \langle gadget, G\rangle\}$.

We compute transitive and reflexive relationships using paths. A simple directed path (or a path for short) is presented as a tuple of concepts $< a_1, a_2, .., a_n >$, where for every $i \in \{1,..,n-1\}$, it holds that $< a_i, a_{i+1} > \in$ *I-rel*.

The function *path_set* generates all paths from concept **a** to concept **b** in the given concept system. The function takes three arguments: two concepts, a and b, and the two place relation *I-rel*. On the basis of the ordered pairs which are included in the *I-rel*, we construct all the paths from **a** to **b** (see operation 15 in [TJ92a]).

path_set : *C-set* $\times$ *C-set* $\times P($*C-set* $\times$ *C-set*$) \rightarrow P(T($*C-set*$))$

$path\_set(a,b,I\text{-}rel) =$

$$\begin{cases} \{< a_1,..a_n > \mid < a_1,..,a_n > \in T(C\text{-}set) : a_1 = a \wedge a_n = b \wedge \\ \forall i \in \{1,..,n-1\} :< a_i, a_{i+1} > \in I\text{-}rel\}, if\, a \neq b \\ \{< a >\}, if\, a = b \end{cases}$$

Now in the example $path\_set(clockradio, G, I\text{-}rel1)$ is the set

$$\{\langle clockradio, clock, gadget, G\rangle, \langle clockradio, radio, gadget, G\rangle\}.$$

For the sake of future use, we define two auxiliary functions, *maximal_set* and *minimal_set* for finding the intensionally maximal and the intensionally minimal concepts from a given subset of all the concepts in a concept system. Given a set of concepts *C-set'*($\subseteq$ *C-set*), an intensionally maximal concept is such that in $C\text{-}set'$ there is no concept that intensionally contains it. Respectively, an intensionally minimal concept is such that in $C\text{-}set'$ there is no other concept that is intensionally contained in it. The defined functions return the set of maximal concepts and minimal concepts from the given set of concepts $C\text{-}set'$. The signatures of both the functions are of the form $P(C\text{-}set) \times P(C\text{-}set \times C\text{-}set) \to P(C\text{-}set)$.

$maximal\_set(C\text{-}set', I\text{-}rel) = \{x \in C\text{-}set' | \neg\exists y \in C\text{-}set' :< y, x > \in I\text{-}rel\}$

$minimal\_set(C\text{-}set', I\text{-}rel) = \{x \in C\text{-}set' | \neg\exists y \in C\text{-}set' :< x, y > \in I\text{-}rel\}$

In the case of *I-rel1*, $maximal\_set(\{clock, radio, gadget\}, I\text{-}rel1)$ returns the set $\{clock, radio\}$, whereas $minimal\_set(\{clock, radio, gadget\}, I\text{-}rel1)$ returns $\{gadget\}$.

5.4 Elementary relations

If concept **a** contains intensionally concept **b**, there is a path from **a** to **b**. The Boolean function $contains(a, b, I\text{-}rel)$ returns *true* if there is at least one path from **a** to **b**. The function $is\_contained(a, b, I\text{-}rel)$ is based on the inverse relation of intensional containment. The function returns true, if and only if concept **a** is intensionally contained in concept **b**. The signatures of the functions are of the form: $C\text{-}set \times C\text{-}set \times P(C\text{-}set \times C\text{-}set) \to \{false, true\}$.

$$contains(a, b, I\text{-}rel) = \begin{cases} \text{true, if path\_set(a,b,I-rel)} \neq \{\} \\ \text{otherwise false.} \end{cases} \tag{5.1}$$

$$is\_contained(a, b, I\text{-}rel) = \begin{cases} \text{true, if contains(b,a,I-rel)} \\ \text{otherwise false.} \end{cases} \tag{5.2}$$

We get the set of concepts that are contained intensionally in concept **a** by using the function *contains_set(a,I-rel)* in the given relation *I-rel*. Respectively, the function *is_contained_set(a,I-rel)* returns all concepts whose characteristic concept is **a**, i.e. the concepts in which a is contained. The signatures of the functions are of the form:

$C\text{-}set \times P(C\text{-}set \times C\text{-}set) \to P(C\text{-}set)$.

$$\text{contains\_set(a,I-rel)} = \{x \in C\text{-}set | \text{contains(a,x,I-rel)}\} \tag{5.3}$$

$$\text{is\_contained\_set(a,I-rel)} = \{x \in C\text{-}set | \text{contains(x,a,I-rel)}\} \tag{5.4}$$

For example, *contains_set(clock,I-rel1)* returns the set $\{G, gadget, clock\}$ and *is_contained_-set(clock, I-rel1)* returns $\{clock, clock\ radio\}$.

In any non-empty concept system, there is at least one concept which does not have any other characteristic but itself, and at least one concept which is not contained in any other concept. Function *bottom_concepts* returns the set of minimal concepts of the concept system, i.e. all such concepts which do not have any successors in the concept relation *I-rel*. Respectively, the function *top_concepts* computes the set of maximal concepts in the concept system, i.e. all the concepts that do not have any predecessors in *I-rel*. The signatures of the functions are of the form: $P(\textit{C-set} \times \textit{C-set}) \rightarrow P(\textit{C-set})$.

$$\textit{bottom\_conceps(I-rel)} = \textit{minimal\_set(C-set,I-rel)} \tag{5.5}$$

$$\textit{top\_concepts(I-rel)} = \textit{maximal\_set(C-set,I-rel)} \tag{5.6}$$

Clearly, in the example, $\textit{bottom\_concepts(I-rel1)} = \{G\}$ and $\textit{top\_concepts(I-rel1)} = \{clockradio\}$.

5.5 Concept association relations

We relate two concepts **a** and **b** to a third concept **c** in two ways. Either **c** contains intensionally both **a** and **b**, or **c** is contained in both the concepts **a** and **b**. By constituting converses of these situations and by composing all meaningful combinations, we get eight Boolean functions altogether. Respectively, we get eight functions that return the sets of concepts which are in a certain association relation with concept **a**.

We define compatibility, incompatibility, comparability and incomparability as in Chapter 4, i.e. as follows:

$$\textit{compatible(a,b,I-rel)} = \begin{cases} \textit{true, if } \exists x : x \in \textit{C-set} \land \textit{contains(x,a,I-rel)} \land \textit{contains(x,b,I-rel)} \\ \textit{otherwise false.} \end{cases} \tag{5.7}$$

$$\textit{incompatible(a,b,I-rel)} = \begin{cases} \textit{true, if } \neg compatible(a,b,\textit{I-rel}) \\ \textit{otherwise false.} \end{cases} \tag{5.8}$$

$$\textit{comparable(a,b,I-rel)} = \begin{cases} \textit{true, if } \exists x : x \in \textit{C-set} \land \textit{contains(a,x,I-rel)} \land \textit{contains(b,x,I-rel)} \\ \textit{otherwise false.} \end{cases} \tag{5.9}$$

$$\textit{incomparable(a,b,I-rel)} = \begin{cases} \textit{true, if } \neg comparable(a,b,\textit{I-rel}) \\ \textit{otherwise false.} \end{cases} \tag{5.10}$$

By using the above Boolean functions, we define the functions that return the set of those concepts that are in a respective association relation with a given concept **a**. The signatures of these functions are of the form $\textit{C-set} \times P(\textit{C-set} \times \textit{C-set}) \rightarrow P\textit{(C-set)}$.

$$\textit{compatible\_set(a,I-rel)} = \{x \in \textit{C-set} | \textit{compatible(a,x,I-rel)}\} \tag{5.11}$$

$$\textit{incompatible\_set(a,I-rel)} = \{x \in \textit{C-set} | \textit{incompatible(a,x,I-rel)}\} \tag{5.12}$$

$$\textit{comparable\_set(a,I-rel)} = \{x \in \textit{C-set} | \textit{comparable(a,x,I-rel)}\} \tag{5.13}$$

$$\textit{incomparable\_set(a,I-rel)} = \{x \in \textit{C-set} | \textit{incomparable(a,x,I-rel)}\} \quad (5.14)$$

By combination of the above association relations we get four new situations. For brevity, we present these relations, the name of the respective Boolean function and the name of the respective set function, in the form of table 5.1. The Boolean functions would be defined simply by the conjunction of partial conditions, and analogously set functions would be defined by the intersection of relevant sets. For example, the function *homogen_compatible(a,b,I-rel)* returns true, if and only if **a** and **b** are compatible and comparable, and the function *homogen_compatible_set(a,I-rel)* returns the intersection of comparable and compatible concepts of **a**.

Association relation	Combination	Boolean function	Set function
Homogen compatibility	Comparable, compatible	homogen_compatible (5.15)	homogen_compatible_set (5.16)
Heterogen compatibility	Incomparable, compatible	heterogen_compatible (5.17)	heterogen_compatible_set (5.18)
Opposition	Comparable, incompatible	opposition (5.19)	opposition_set (5.20)
Isolation	Incomparable incompatible	isolated (5.21)	isolated_set (5.22)

Table 5.1: Combined association relations.

Let us take an example. $I-rel2 = \{\langle c1,c2\rangle, \langle c1,c8\rangle, \langle c2,c6\rangle, \langle c3,c4\rangle, \langle c3,c5\rangle, \langle c4,c6\rangle, \langle c4,c7\rangle, \langle c5,c7\rangle, \langle c5,c8\rangle\}$ (Figure 5.2) is a relation in the concept set $\{c1,..,c8\}$, and we consider association relations under it. There are two members in the set that *top_concepts(I-rel2)* returns; *c1* and *c3*. Respectively, the function *bottom_concepts(I-rel2)* returns the set $\{c6,c7,c8\}$.

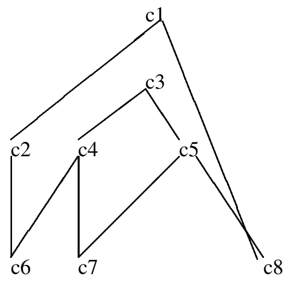

Figure 5.2: *I-rel2*

Let us take concept *c4* as an object for more circumstantial study. First, there are three characteristics of *c4* and *c4* is a characteristic of two concepts, i.e. *contains_set(c4,I-rel2)* $= \{c4,c6,c7\}$ and *is_contained_set(c4,I-rel2)* $= \{c3,c4\}$. There are six concepts that have a characteristic in common with concept *c4* and one concept that has no characteristics in common with concept *c4*. Functions of comparability and compatibility are applied to *c4* as follows: *compatible_set(c4,I-rel2) = {c3, c4, c5, c6, c7, c8}*, *incompatible_set(c4,I-rel2) = {c1, c2}*, *comparable_set(c4,I-rel2)* $= \{c1,c2,c3,c4,c5,c6,c7\}$ and *incomparable_set(c4,I-rel2)* $= \{c8\}$. The rest of the functions return the following sets: *homogen_compatible_set(c4,I-rel2)* $= \{c3,c4,c5,c6,c7\}$, *heterogen_compatible_set(c4,I-rel2)* $= \{c8\}$, *opposition_set(c4,I-rel2)* $= \{c1,c2\}$ and *isolated_set(c4,I-rel2)* $= \{\}$.

5.6 The intensional product and sum

According to Kauppi's theory, the intensional product is an unambiguous concept. In the theory, concept **x** is equivalent to the intensional product of the concepts **a** and **b**, if and only if **x** has every common characteristic of **a** and **b**, but not any other characteristics apart from itself. If we assume, as Kauppi did, that there is exactly one intensional product for any comparable pair of concepts in the concept system, we rule out some concept systems that could be wholly appropriate in everyday modelling. Here, we define intensional product in a more relaxed manner so that concept **x** is in the set of product concepts of **a** and **b** if the following conditions holds: 1. **x** is a characteristic of both **a** and **b**; 2. within these characteristics, **x** is among the maximal concepts.

$$\begin{aligned} &prod\_set : C\text{-}set \times C\text{-}set \times P(C\text{-}set \times C\text{-}set) \rightarrow P(C\text{-}set) \\ &prod\_set(a,b,I\text{-}rel) = maximal\_set(contains\_set(a,I\text{-}rel) \cap contains\_set(b,I\text{-}rel), I\text{-}rel) \end{aligned} \tag{5.23}$$

In the case of *I-rel1* intensional product between any two concepts is simple. For example, the function *prod_set(radio,clock,I-rel1)* returns the set, whose single member is *gadget*. By following the function *prod_set(radio,clock,I-rel1)*, the intersection of radios and clocks characteristics is $\{gadget, G\}$. The maximal of these is *gadget*.

Let us take a more complicated example. The relation of Figure 5.3, *I-rel3* = $\{\langle c1,c3\rangle, \langle c1,c4\rangle, \langle c2,c3\rangle, \langle c2,c4\rangle, \langle c4,c5\rangle, \langle c3,c5\rangle\}$ is the relation in the set *C-set3* = $\{c1,..,c5\}$. Now the function *prod_set(c1,c2,I-rel3)* returns the set $\{c3,c4\}$. Here the common characteristics of the concepts *c1* and *c2* are *c3*, *c4* and *c5*. Among these, *c3* and *c4* are the maximal ones.

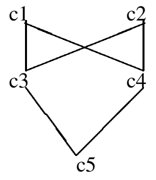

Figure 5.3: *I-rel3*

The intensional sum is considered analogously to the intensional product. In Kauppi's theory, concept **c** is equivalent to the intensional sum of the concepts **a** and **b**, if and only if for every **x** it holds: **x** contains **c** intensionally, if and only if **x** contains **a** and **b** intensionally. As with intensional product, we define the corresponding function to be more relaxed than the original definition. By our definition, concept **x** is in the set that the function sum_set(a,b,I-rel) returns, if: 1. **x** is among the concepts that contain both operand concepts **a** and **b**; and 2. **x** is among the minimal of these.

$$\begin{aligned} &sum\_set : C\text{-}set \times C\text{-}set \times P(C\text{-}set \times C\text{-}set) \rightarrow P(C\text{-}set) \\ &sum\_set(a,b,I\text{-}rel) = minimal\_set(is\_contained\_set(a,I\text{-}rel) \cap \\ &\qquad is\_contained\_set(b,I\text{-}rel), I\text{-}rel) \end{aligned} \tag{5.24}$$

The function *sum_set* works in the same way as *prod_set*, but inversely. For example, *sum_-set(clock, radio, I-rel1)* produces the set $\{clockradio\}$. Respectively, *sum_set(c3, c4, I-rel3)* returns the set $\{c1, c2\}$.

5.7 The intensional negation

In line with Kauppi's theory, we express the intensional negation **-a** of concept **a** as a concept which is: 1. incompatible with **a**; and 2. for every **x**: if **x** is incompatible with **a**, then **-a** is one of the characteristics of **x**.

If there are no concepts incompatible with concept **b**, there cannot be an intensional negation of concept **b**, either. In Kauppi's theory, there is an axiom according to which if there exists an incompatible concept with **b**, there must exist the intensional negation of concept **b**. Therefore, the structure of the concept system is very strict. On this account, we consider the intensional negation in a a "wider" way. That is, we consider the negation set instead of a single negation concept.

In our approach the negation of a concept is understood as follows: If concept **b** is incompatible with concept **a**, and if **b** does not have any such characteristics (apart from itself) that are incomparable with concept **a**, then concept **b** is in the *neg_set(a,I-rel)*. In the definition of the function, we first consider the set concepts that are incompatible with concept **a**, and then choose the minimal concepts from this set.

$$\begin{aligned}&\mathit{neg\_set} : \mathit{C\text{-}set} \times P(\mathit{C\text{-}set} \times \mathit{C\text{-}set}) \rightarrow P(\mathit{C\text{-}set}) \\ &\mathit{neg\_set}(a, \mathit{I\text{-}rel}) = \mathit{minimal\_set}(\mathit{incompatible\_set}(a, \mathit{I\text{-}rel}), \mathit{I\text{-}rel})\end{aligned} \tag{5.25}$$

Let us take a new relation *I-rel4* = $\{\langle c1,c4\rangle, \langle c1,c5\rangle, \langle c2,c4\rangle, \langle c2,c6\rangle, \langle c3,c5\rangle, \langle c3,c6\rangle, \langle c4,c7\rangle, \langle c5,c7\rangle, \langle c6,c7\rangle\}$ in the concept set $\{c1,..,c7\}$. Now, the results of the function in these different cases are as follows: *neg_set(c1,I-rel4)* = $\{c6\}$, *neg_set(c2, I-rel4)* = $\{c5\}$, *neg_set(c3,I-rel4)* = $\{c4\}$, *neg_set(c4, Irel4)* = $\{c3\}$, *neg_set(c5,I-rel4)* = $\{c2\}$, *neg_set(c6, I-rel4)* = $\{c1\}$ and *neg_set(c7, I-rel4)* = $\{\}$. In this example, if the negation of a concept exists, the concept is equivalent to its negation's negation.

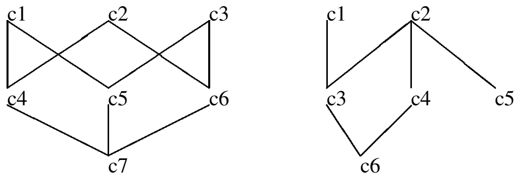

Figure 5.4: *I-rel4* and *I-rel5*

In relation *I-rel4* (Fig. 5.4), every negation set corresponds to Kauppi's definition of the intensional negation because there is maximally one concept in each negation set. The function *neg_set(c7,I-rel4)* is an empty set, because *c7* is compatible with every concept. Let us modify the situation. In *I-rel5* = $\{< c1,c3 >, < c2,c3 >, < c2,c4 >, < c2,c5 >, < c3,c6 >, < c4,c6 >\}$, there is no unequivocal intensional negation for every concept. The concepts *c3* and *c6* are compatible with every concept. The function *neg_set(_,I-rel5)* with one of the arguments *c2, c4* or *c5* returns the set $\{c1\}$. On the other hand, *neg_set(c1,I-rel5)* produces the set $\{c4,c5\}$.

In *I-rel5* there are no such concepts which would be equivalent with negation's negation of $c1$.

5.8 The intensional quotient and difference

According to Kauppi's theory, the intensional quotient from concept **a** to concept **b** is the concept that is a characteristic of every such concept that contains **a** intensionally and is incompatible with **b**.

As earlier in this chapter, we shall define a function that produces a set of concepts so that the definition preserves the most essential lineaments of Kauppi's one. If a concept **x** ($\neq$ **a**) is in the set that the function *quo_set(a,b,I-rel)* produces, it must hold that: 1. **x** contains **a** intensionally; 2. **x** is incompatible with **b**; and 3. **x** is among the minimal concepts in the set of concepts that satisfy the conditions 1 and 2. If concept **a** satisfies condition 2, i.e. **a** is incompatible with **b**, then the function returns $\{\mathbf{a}\}$.

$$\begin{aligned} &quo\_set : C\text{-}set \times C\text{-}set \times P(C\text{-}set \times C\text{-}set) \rightarrow P(C\text{-}set) \\ &quo\_set(a, b, I\text{-}rel) = minimal\_set(S, I\text{-}rel) \end{aligned} \tag{5.26}$$

where S = *is_contained_set(a,I-rel)* $\cap$ *incompatible_set(b,I-rel).*

The intensional difference from concept **a** to concept **b** is the concept that contains intensionally all those concepts that are the characteristics of concept **a** and are incomparable with **b**.

We define the function *diff_set* that produces the set of concepts which satisfy the following conditions. A concept **x** ($\neq$ **a**) is in the set *diff_set(a,b,I-rel)* if: 1. **a** contains **x** intensionally; 2. **x** is incomparable with b; 3. **x** is among the maximal in the intersection of sets that the conditions 1 and 2 indicate. If concept **a** satisfies condition 2, i.e. **a** is incomparable with **b**, the function returns the set $\{\mathbf{a}\}$.

$$\begin{aligned} &diff\_set : C\text{-}set \times C\text{-}set \times P(C\text{-}set \times C\text{-}set) \rightarrow P(C\text{-}set) \\ &diff\_set(a, b, I\text{-}rel) = maximal\_set(S, I\text{-}rel) \end{aligned} \tag{5.27}$$

where S = *contains_set(a,I-rel)* $\cap$ *incomparable_set(b,I-rel).*

In the example of Figure 5.5, *I-rel6* = $\{\langle c1, c3\rangle, \langle c1, c4\rangle, \langle c2, c4\rangle\}$, some non-empty *quo_set* sets can be produced. The functions *quo_set(c2,c1,I-rel6)* and *quo_set(c2,c3,I-rel6)* return the set $\{c2\}$. The functions *quo_set(c1,c2,I-rel6)* and *quo_set(c3,c2,I-rel6)* return the set $\{c1\}$.

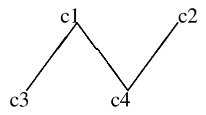

Figure 5.5: *I-rel6*

As with the above, the function *diff_set(c2,c3,I-rel6)* returns the set $\{c2\}$ and *diff_set(c4,c3,I-rel6)* returns $\{c4\}$. The functions *diff_set(c3,c2,I-rel6)* and *diff_set(c3, c4, I-rel6)* return the set $\{c3\}$.

5.9 Summary and discussion: Concept association relations and operations

In this chapter, we have represented the relations and the concept operations of Kauppi's theory without any reference to the semantics of the relation of intensional containment. Nilsson and Palomäki maintain in [NP98] that intensional containment is a certain kind of IS-A relationship. However, Kangassalo suggests in [Kan96] that the intensional containment covers the IS-A relationship, the aggregation and the component relationship. Whatever view is taken towards the semantics of intensional containment, the representation we employ here provides powerful tools for analysing different kinds of knowledge structures.

We can compare Kauppi's calculus with DL (see e.g. [BS92], [Bor95]), if we equate the relation of the intensional containment with the IS-A relationship. Then, for example, the operations of the intensional sum and the intensional product are intensional counterparts for the operations AND and OR in DL. Using the association relations, one can study various relationships (like comparability and compatibility) between concepts: if two concepts are compatible, there exists a concept that is a specialisation of both concepts. To our knowledge, an approach involving concept association relations has not been applied in DL. In many extensions of DL, there are the top-element (empty set) and the bottom-element (universal set) in the system. In the point of view of the association relations and operations presented here, these special elements make all the concepts both comparable and compatible with each other. Consequently, many of the operations of this chapter would not be usable in such a system. However, in some cases, ignoring (omitting) these special elements can provide a starting point for the intensional analysis presented above.

The component relationship (transitive part-of) can be presented similarly to the intensional relation *I-rel* here such that only immediate component relationships are explicitly presented, as in [Jun98]. If notation $\langle a, b \rangle \in$ *I-rel* is interpreted such that a has an immediate component b, comparability would mean that the given arguments have some common components. Analogously, *negation_set* generates these objects that are as simple as possible (undivided) and that are not components of any such objects that have a given argument as a component.

In an analysis of a process system (cf. [Lan78]) we can employ the representation of this chapter as follows: Let *I-rel* be a binary relation in a set of processes. *I-rel* contains all process pairs $\langle a, b \rangle$ such that a is an immediate successor process of b, i.e. process b is demanded for the production of process a. As an interesting application of concept theory, we can present a function that returns the least defined processes in the set of those processes that are not demanded by the given one. According to the tradition of concept theory, we prefer to call this the negation_set.

In this chapter, we have considered the association relations and concept operations of Kauppi's concept theory and presented an abstract implementation of them. We have extended the concept theory in two ways. First, we have presented operations that return sets of those concepts that are in some association relation with the given concept. Second, we have represented a wider interpretation for concept operations. Our functions do not return a single concept, but a set of concepts that satisfy the most essential conditions. Therefore, the functions provide a non-empty result, even in the case where there was not a unique concept that satisfied the original definition.

By using a set-oriented approach, we have presented in an explicit and implementable, albeit an implementation-independent, manner, the formal concept theory which we see as having the potential to manage and describe the structure of knowledge. This method forms the basis of an executable knowledge representation language and can be applied, for example, to the analysis of IS-A relationship, component structures and process structures.

Chapter 6

IFO and CONCEPT D – A Comparison of Modelling Languages

6.1 Introduction

The aim of conceptual modelling is to create models that describe the conceptual level of the domain of application and which are needed in designing and defining information systems.[1] In this modelling process, in most cases some relationships in the domain of application are given special consideration. As discussed in Chapter 2, these prevalent kinds of relationships (like IS-A, "has-attribute" or "contains intensionally") should be captured by the analysis and represented by some formalism as its result.

A large variety of Semantic Data Models (SDMs) have been proposed for this representation, e.g. in [Poh96]. In general, following [AH87], we can say that SDMs are based on the following principles:

- Data about objects and relationships between them should be modelled in a direct manner;
- many relationships (like "has-attribute") are functional in nature,
- IS-A relationships are significant;
- there are hierarchical mechanisms for building (object) types out of other (object) types.

As in many formalisms of the 1980's and the 1990's, all of these elements are present in the IFO model ([AH84], presented 1984 and [AH87] 1987 in the form discussed here) and COMIC methodology ([Kan93], presented 1993, whereas its modelling language CONCEPT D was introduced already in 1983 in [Kan83]). In addition, IFO and COMIC are similar in at least two respects. Both provide a rich modelling language and a graphical representation formalism. Likewise, for both there is a translation scheme by which the information represented in the modelling language can be used to build a relational database implementation (see [AH87], [Nie99]).

The fundamental differences between IFO and COMIC are that IFO concentrates in semantics and mathematical analysis, while COMIC focuses on supporting the process of conceptual modelling.

Naturally, we make a distinction between the *description* of the structure of the domain of application and the actual things (objects) that reside in the domain of application. Since the

[1]Adapted from [KKJH00].

description relies on a modelling language, we shall use the terminology of language items (L-items, for short; see Chapter 2). The names of L-items are written in capital letters in this chapter.

The goal of this chapter is to point out the beneficial aspects of both IFO and COMIC approaches and to discuss the possibility of transforming a CONCEPT D (externalised conceptual) schema into a corresponding IFO (externalised conceptual) schema. This would enable us to combine the benefits of both approaches: COMIC's support of the process of conceptual modelling and IFO's precise semantics.

In Chapter 2, we discussed intensional, extensional and hybrid languages in conceptual modelling. In Section 2.4, we used a language whose syntax resembled that of CONCEPT D as an example of an intensional modelling language. However, in what follows we use a "larger" CONCEPT D type language and map its expressions into IFO expressions. Though our CONCEPT D variant utilises intensional containment, the expressions will gain a perfectly extensional interpretations when they are translated into IFO.

This chapter is organised as follows. In Section 6.2, we describe the IFO approach and the L-items of an IFO (externalised conceptual) schema. In Section 6.3, we discuss the COMIC approach and CONCEPT D. Section 6.4 discusses the different semantical emphasis of IFO and CONCEPT D. However, we utilise a notion of "real world intension" to point out how the semantics of a CONCEPT D (externalised conceptual) schema can be understood in a way that resembles IFO's semantics. Based on this similarity, in Section 6.5, we identify similar components in CONCEPT D and IFO (externalised conceptual) schemata. Finally, in Section 6.6, we describe how a translation of a CONCEPT D schema into an IFO schema could be performed. The strengths and weaknesses of the intensional approach (and thus the value of combining it with IFO) are also discussed in Section 6.6.

6.2 The IFO approach

The background of IFO is in semantic data models. IFO was among the first models to introduce a well defined procedural semantics (based on database updates). In this way, the L-items of the IFO language are connected to the database in a rather unambiguous way. Here, like Abiteboul and Hull in [AH87], we do not make a distinction between IFO's methodology and modelling language; they are both called IFO.

In IFO, the L-items are called types (atomic or constructed[2]), fragments and schemata. Types model the structure of the objects in the domain of application and roughly correspond to L-entities in Chapter 2. Fragments contain types and represent functional relationships. IS-A relationships are defined in fragments by either generalisation or specialisation. IFO schemata are directed graphs that are built by combining fragments. We shall now shortly describe the use of L-items in the process of creating an externalised conceptual schema of the domain to be modelled. A summary of L-items of IFO, described below, is presented in Figure 6.1.

An atomic type can be printable, abstract or free. *Printable atomic types* correspond to objects of predefined types that serve as the basis for input and output [AH87]. SURNAME serves as an example of a printable atomic type; each SURNAME refers to a string (of characters). *Abstract atomic types* (like STUDENT) differ from printable ones. Each refers to a subset of a domain of this abstract type, like a set of particular persons. Finally, *free atomic types* are used in cases where there are IS-A relationships between them and other types. It can be said that free atomic types gain their actual type through the IS-A relationship from other types.

[2][AH87] employs the terms "derived", "more complicated" and "non-atomic" for constructed types.

Constructed types are built of atomic types by creating finite sets (like COURSE-PARTICIPANTS, a set of STUDENTs) or Cartesian products (aggregation, like forming the type CIGAR by combining the types GROUND TOBACCO and TOBACCO LEAF).

Fragments are simply functions ("mappings") whose domain and range are types. For instance, there may be a fragment THE GRADE OF A STUDENT that represents a function that maps each student (an abstract type) into his or her grade (a printable type). A fragment is a directed tree, whose root is called the primary node of the fragment.

In IFO, IS-A relationships are acquired by either specialisation or generalisation. In a specialisation, the possible roles of a given type are considered: a person might be a student. Specialisation can be overlapping: a person may be both an employee and a student. In a generalisation, distinct pre-existing types are combined into a new type. These types are not allowed to be overlapping.

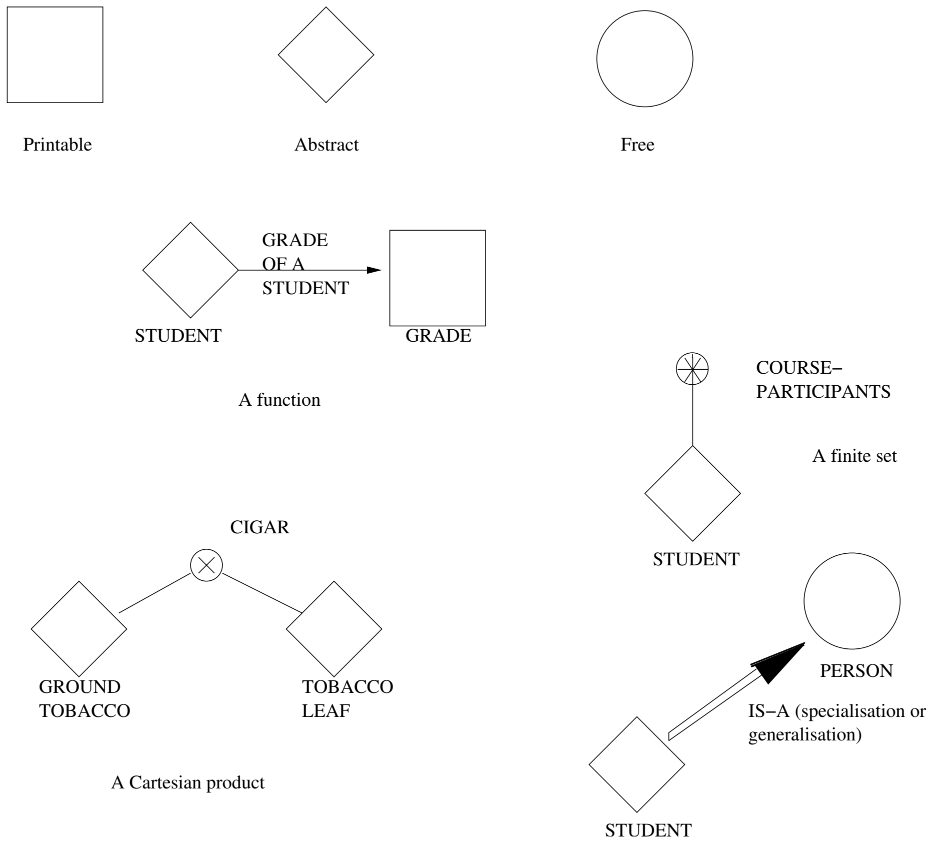

Figure 6.1: L-items of IFO

An IFO graph is a directed acyclic graph, where types are the nodes of the graph and IS-A relationships (specialisation and generalisation; see below) are represented as directed edges. The types (nodes) may, naturally, be parts of fragments as well. An IFO schema is an IFO graph that conforms to some requirements concerning IS-A relationships, as follows. Let us suppose that there exists an IS-A relationship between two IFO types, A and B, so that A is the first member of the IS-A relation (the head) and B is the second member (the tail). In this case, A is called the supertype of B and, respectively, B the subtype of A. The restrictions can be summarised as follows:

- In specialisation, the type of the node is gained top-down, from supertype to subtype. If both

A and B are constructed, non-free types, this can lead to conflicting situations; B is supposed to inherit its structure from A. Therefore, in specialisation, the tail must be of the free type.[3]

- In generalisation, the type of the node is gained bottom-up. As in the previous case, type conflicts may arise. Thus, the head must be a free type.

Other restrictions are related with the types that have some role in fragments. Roughly, in specialisation, the head is required to be a primary node of a fragment. In generalisation, the tail is a primary node. These restrictions form the basis of IFO's sophisticated update semantics that is discussed in Section 6.4.

6.3 The COMIC approach and CONCEPT D

COMIC, described in [Kan93], is closely connected to the intensional approach of conceptual modelling, whose background and presentation were discussed in Chapters 4 and 5. Recently, Niemi in [Nie99] and [Nie00] has presented methods to transform CONCEPT D diagrams into lattices, thus enabling the intensional operations and associations of Chapter 5.

In this chapter we discuss only the main features of COMIC and CONCEPT D.[4] More detailed accounts can be found in [Kan82] and [Kan92] where the background of the process of conceptual modelling and concept structures is discussed; in [Kan93] Kangassalo presents the COMIC methodology based on that background; and in [Kan83] he introduces CONCEPT D modelling language. In CONCEPT D, many different kinds of L-items of other modelling languages are covered by only two CONCEPT D L-items: concepts and intensional containment links. This means that intensional containment covers several different abstraction methods and their presentation on the level of language. Given this ambitious starting point, it is understandable that both the methodology and the language have been criticised. We take a closer look at this criticism in Section 6.6.

The background of COMIC lies in information systems development. The development starts solely on the concepts that the information system users have. Using these concepts, the modeller builds a theory of the domain to be modelled and expresses it with some formalism. Kangassalo in [Kan92] defines conceptual modelling as "a process of forming and collecting conceptual knowledge about the Universe of Discourse (UoD), and documenting the results in the form of a conceptual schema." We assume that conceptual knowledge consists of concepts and knowledge primitives, since Kangassalo and Viitanen characterise in [KV90] concepts and knowledge primitives as follows:

"A concept is defined to be an independently identifiable construct which has an internal structure and is composed of knowledge primitives and/or other concepts" and "knowledge primitives

[3]As an example of a conflicting situation, let us consider a case where the tail is not free, e.g. let us try to specialise CIGAR (as in Figure 6.1) from PRODUCT, a Cartesian product of MANUFACTURER and PRICE. Now, CIGAR has two identities, as two different Cartesian products, and its update semantics would not be unambiguous.

[4]More specifically, we omit the following important features of CONCEPT D:

- In the original definition of CONCEPT D, there are various different generalisation structures;
- The conditions and constraints can relate concepts using various different criteria (properties and relationships between their extensions, etc.). These have been omitted for simplicity;
- The role of the identifiers has been greatly simplified in this presentation.

are used as building blocks of concepts, possibly together with other concepts". Furthermore, as defined by Kangassalo in [Kan93], a *concept structure* is a diagram which represents a definition of a concept. In order to express a concept structure, a language is needed. Kangassalo and Viitanen state in [KV90] that language CONCEPT D/D "is used to describe the hierarchical concept structure of a single concept composed of other concepts and knowledge primitives". In addition to CONCEPT D/D, there is a language called CONCEPT D/CS but, in the scope of this chapter, we shall simply see "CONCEPT D/D" and "CONCEPT D" as synonyms. A graphical representation of the CONCEPT D L-items that are relevant in this discussion can be seen in Figure 6.2.

In [Kan96], Kangassalo mentions the following knowledge primitives:

- Name of the concept used to refer to the concept.
- Intensional relationship between two concepts.
- Extensional relationship between two occurrences of concepts.
- Identifying property is a property of concept B intensionally contained in concept A that enables an occurrence of concept B to be used to identify an occurrence of concept A.
- Condition schema.
- Constraint schema.
- Value set is a set of other concepts and their representations associated with a given concept to represent it.
- Function is a mapping from a value set to another.
- Semantic rule is a text explaining the concept.

Strictly, all of these knowledge primitives should have a presentation in a modelling language. Here, we simplify the discussion (and the graphical notation) and consider only the most frequently occuring knowledge primitives.

Together, concept structures build up a conceptual schema. This in turn is used as an input for a semi-automated (currently automated) support system that generates a database implementation of the schema, described in [Nie99]. The database implementation also includes update and query functionalities. There is a close relation between the database and the design formalism, and therefore it is not surprising that there is a graphical query tool as well. The graphical conventions of the query language resemble those of the design language (for details, see [KKP90]).

When expressed in a modelling language (CONCEPT D), a concept structure is a diagram which represents a definition of a concept. In a concept structure, concepts are primarily connected by directed edges, representing the *direct intensional containment* relationship between concepts (see Chapter 4), and we call them intensional containment edges. Some additional information (conditions, constraints) is provided by undirected edges as well, but their role is less important.

A concept structure consists of defining concepts and a defined concept, and forms a definition hierarchy (a directed, acyclic, connected graph with a root node; see [Kan93]). We equate the "concept **x** is defined by concept **y**" relation in the hierarchy with the intensional containment relation.[5] [6] For instance, if PERSON is (partially) defined by SURNAME, we can say that PERSON (directly) intensionally contains SURNAME, too; and that SURNAME is (directly) intensionally contained in PERSON. This is presented by an intensional containment edge from PERSON to SURNAME. It should be noticed, too, that the notion of intensional containment used in COMIC is "liberal" in a sense that it apparently does not imply that in every possible situation every person would have a surname, but that in the domain of our interest this is the case.

Naturally, there must be something with which to start the definition hierarchy: basic concepts. A basic concept is simply a concept not defined by any other concepts in the definition hierarchy. Concepts that are not basic are called defined (or derived) concepts. A concept structure has a unique root node, one that is defined by all the other concepts. The root node is a concept that contains all the knowledge concerning the domain of application (for details, see [Kan93], where the domain of application is called UoD).

Related to intensional containment between two concepts, there can be some additional information, i.e. information about the *numeric containment* (cardinality constraints in intensional containment edges) of the concepts. By default, it is 1:1 (as in the case of PERSON and SOCIAL SECURITY NUMBER; i.e. PERSON can intensionally contain exactly one SOCIAL SECURITY NUMBER), but can also be defined to be 1:n.

A constraint, for example, states that SENIOR CITIZEN's AGE must be greater than 64. In a CONCEPT D schema, conditions and constraints are expressed with (undirected) dotted lines that connect two concepts with each other, and a rule attached to the line. In our limited version of CONCEPT D, we only consider constraints where the constraining concept is directly or indirectly contained in the constrained concept.

The only additional relationship we consider in this chapter is the *cardinality constraint* (other than the numeric containment: intensional containment with cardinality constraints) and it is expressed by a dotted line between two concepts, and a cardinality ratio, like 1:n, attached to it.

There are three methods of defining concepts in concept structures; aggregation, generalisation, and value transformation. In the context of this chapter, we consider one additional one, specialisation, though in CONCEPT D it is a construct of aggregation and constraints. The concept definition created by these methods is evaluated in order to understand the intension of the defined concept. In Section 6.4 we suggest how this evaluation can be related to semantics.

Aggregation is the most basic form of concept definition. In aggregation, the defined concept is depicted mainly by intensional containment links that relate it to its defining concepts. For instance, the concept of PERSON can be defined by expressing that it intensionally contains the concepts SOCIAL SECURITY NUMBER, JOB and ADDRESS.

Using generalisation, a concept can be defined by extracting information from concepts that contain it intensionally (the opposite way of aggregation). For instance, PERSON can be defined as a generalisation of CHILD, ADOLESCENT, MIDDLE-AGED and SENIOR CITIZEN. Unlike in the original version of CONCEPT D, we do not consider generalisation a short-cut form of a complex aggregation, since for our purposes it is beneficial to reserve a specific notation for it. In the case of the example, the generalised concept (PERSON) would simply contain all the concepts

[5]In what follows, we shall use words written with capital letters to refer to CONCEPT D's L-items, too, although they should be interpreted to be concepts (and written in bold typeface).

[6]Kangassalo emphasises in [Kan93] and [Kan96] that the presentation of definitions is not solely based on intensional containment, but on definition types (aggregation, generalisation, value transformation) that utilise intensional containment. However, since these in turn are based on intensional containment, we consider our assumption to be valid.

that are intensionally contained in all of its defining concepts (like AGE).

Here, specialisation is a method of deriving a more specific concept from a more general one, like SENIOR CITIZEN from PERSON. This is done by constraints; in the case of the example, simply by stating that

- SENIOR CITIZEN intensionally contains PERSON, and
- there is a constraint between AGE and SENIOR CITIZEN so that the age of SENIOR CITIZEN must be greater than 64.

In what follows, we assume that in our CONCEPT D variant, all intensional containment structures that express IS-A type relationships are either generalisations or specialisations.

A value transformation represents a concept definition, wherein a transformation function is applied to some concept(s) to gain a new concept. For instance, the concept of AMOUNT OF SALARY can be defined by using a value transformation function of the concepts WORKING HOURS and HOURLY WAGE. In this case, the function is a simple multiplication applied to the values of WORKING HOURS and HOURLY WAGE. In a schema, a dotted arrow is used to represent a value transformation.

Apart from value transformation functions, some functional relationships are covered by intensional containment. This is apparent when we think about the relations between the concepts STUDENT, COURSE and THE GRADE OF A STUDENT. It is obvious that THE GRADE OF THE STUDENT intensionally contains STUDENT, since this information/knowledge is a prerequisite of THE GRADE OF THE STUDENT. However, the functional relationship between STUDENT and THE GRADE OF THE STUDENT is not a value transformation.

In addition to these simple structures, CONCEPT D contains a set of methods for more precise definition of concepts. These are related to the knowledge primitives mentioned above. The most important of these are identifiers and their scopes. An identifier is simply a unique string of characters that is attached to some concept. An identifier (K) primarily identifies the concept it is attached to (**a**) and the concept(s) (**b**) that directly intensionally contain **a**. Secondarily, it identifies the concepts that are directly or indirectly contained in **b**. The concepts that an identifier can identify are jointly called the scope of the identifier. Since it is often the case that some concepts contained in **b** can be identified by an identifier other than K, the scope of K can be limited. This limit is imposed by a "cut" of the scope, represented simply as a tag on some intensional containment edge with the name of the identifier next to it (for details, see [Kan83]).

Identifiers and cardinality constraints are important if we want to determine what kind of a variant of intensional containment (IS-A, "part-of" ...) each one of the intensional containment edges actually represents. To simplify this task, we propose a limited version of CONCEPT D, whose L-items are presented in Figure 6.2.

We emphasise that if we consider only intensional containment relations in a CONCEPT D schema, it is always a acyclic directed graph. This means that cyclic concept definitions are not allowed in CONCEPT D schemata.

Unlike in the original version of CONCEPT D, we introduce an addition by which the user can state if the concept in question is abstract or printable. In the graphical presentation, printable concepts are designated with the letter P, and abstract concepts do not have a designation. This addition will help the translation process.

In 6.2, *defined concepts* are presented as underlined uppercase words, *basic concepts* as presented as underlined uppercase words, *printable concepts* are designated by letter P, *intensional*

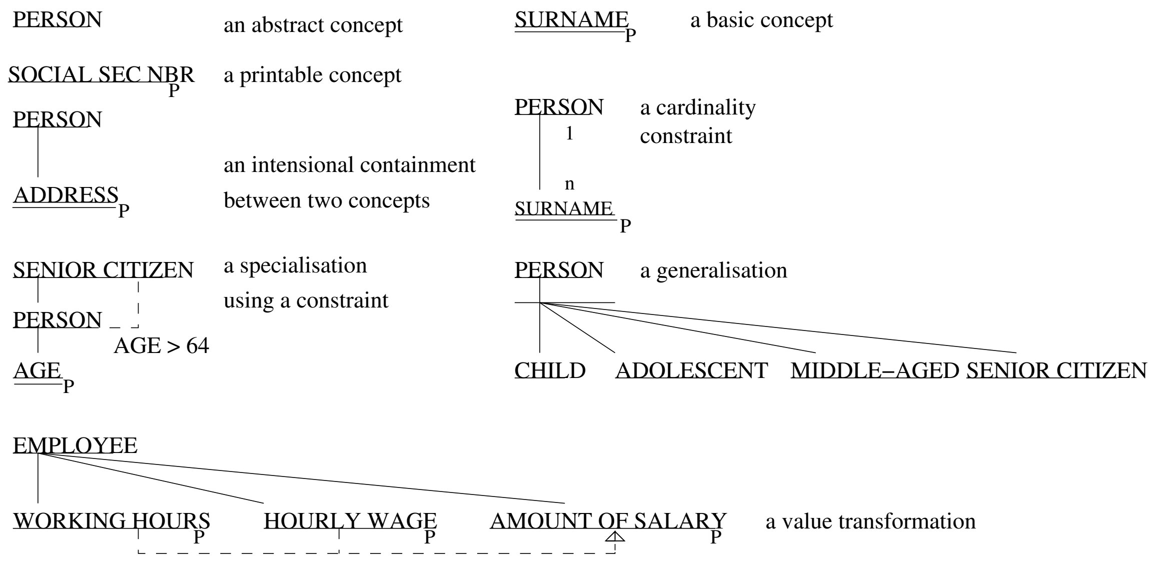

Figure 6.2: L-items of limited CONCEPT D

containment is simply a line connecting an upper (containing) concept to lower (contained) concepts, *a generalisation* is an intensional containment structure with a horizontal line, *cardinality constraints* are show next to concepts, *a specialisation* of a contained concept is expressed using a dotted line and an expression next to it, and a *value transformation* is a dotted arrow pointing from concepts by which the value is calculated to the concept that expresses the value of the calculation.

6.4 The notion of semantics

In IFO, there are operations that operate on the database, i.e. they add, delete or update some sets of values in it. We call these sets of values *instances* and they correspond to the IFO L-items. The semantics of IFO is based on the instances that would exist in the database after each update. This provides a strong basis for mathematical analysis of the update operations.

An IFO operation is a three place tuple (A,B,C), where A is a node, and B and C are sets of values, B representing an old set and C representing a new set. Thus, an operation is a modification of the values of an instance. An addition of a new instance is defined as $(A,\perp,C)$, and a deletion of an existing instance is defined as $(A,B,\perp)$.

Some update operations (more precisely those that apply to instances associated with a given primary node) lead to propagational effects – i.e. they change the instances residing in some other nodes. The propagational effect behaves differently for each of the different types of edges connecting the nodes. Abiteboul and Hull have designed sophisticated algorithms to carry out the propagational effect in any valid IFO schema (see [AH87]).

In principle, our view of the IFO update semantics is as follows:

An IFO instance can be given a normal set-theoretical interpretation, in the sense that the values in the database reflect some objects in a given domain of application. They have attributes (that would be defined as functions in the standard set-theoretical approach) and relationships with other objects of some type, the most important of these relationships being the IS-A relationship. In the set-theoretical approach, the behaviour of objects whose types are in an IS-A relationship is based on set inclusion. This kind of semantics is intuitive, but it is given its actual substance through update operations. The update algorithms effectively specify, in different situations, what it means at the extensional level that the type is in an IS-A relationship with another type. For instance, an update on STUDENT updates PERSON since STUDENT is a PERSON. The same principle applies to IFO functions as well.

In COMIC the notion of semantics relies on the semantic ideas of users or the semantics that the concepts and other elements have in the conceptual schema. Some parts of this probably cannot be analysed by conventional means. However, in this chapter, we utilise the notion of the "real world intension", as presented in [ASCD99]. There, each object is defined by a set of constraints called the real world intensions of the object. Using the notion of real world intension, we can make a distinction between the intension in general (the way concepts could be defined in some possible world) and the way we want concepts to be defined in the domain in which we are interested. We prefer to think that in COMIC's concept structures, the concept hierarchy as well as the constraints and conditions reflect those in the real world intension. Thus, we can say that the externalised conceptual schema represents the real world intension of the concept structure during some time span. In the next section, we consider how to compare the (externalised conceptual) schemata of IFO and CONCEPT D on this semantical basis.

6.5 Comparison

In Section 6.4 we discussed how the semantical basis of IFO differs from that of COMIC. In COMIC, the evaluation process of a concept structure leads to an understanding of the Universe of Discourse (the domain of application). When this evaluation is applied to some given state of the UoD, it actually describes the objects of the domain of application at that moment. In this way, the evaluation can be seen as a counterpart of the procedural semantics of IFO.

Based on this similarity, we now compare the elements of the modelling languages of IFO and COMIC. In the case of COMIC, the modelling language is, of course, CONCEPT D. In both of these formalisms (IFO and CONCEPT D), there are *nodes* and *directed edges*.

We shall now proceed to compare the L-items of the modelling languages. In IFO, *types* are the basic units of knowledge. Naturally, their counterparts in CONCEPT D are *concepts*. *Printable types* in IFO have simple semantics, and therefore they can be equated with *printable concepts* in our version of CONCEPT D. IFO's *abstract types* and *free types* correspond with the *defined concepts* of CONCEPT D.

Since there is only one kind of directed edge (intensional containment) connecting the nodes in CONCEPT D, it is obvious that all the *constructed types* of IFO are constructs created by intensional containment in CONCEPT D. Therefore, IFO's *finite sets* that are built by "iterating" (grouping) a type (like STUDENT) to form a set type (like STUDENTS ATTENDING A COURSE), have their counterpart in CONCEPT D: concept STUDENT ATTENDING A COURSE is defined as containing STUDENT. Likewise, *Cartesian products*, like CIGAR that has GROUND TOBACCO and TOBACCO LEAF, is simply an aggregation in CONCEPT D; the concept of CIGAR is an aggregation of GROUND TOBACCO and TOBACCO LEAF.

However, in the scope of this presentation, it seems impossible to have a semantic criteria in which CONCEPT D intensional containment structures could correspond to finite sets of IFO. Therefore, in the discussion of schema transformations in Section 6.6, we omit finite sets.

The *fragments* of IFO contain types and functions. A *function* of IFO is a mapping of a type to a set of values (a printable type). In CONCEPT D, a function is a *value transformation* that is attached between two concepts, but intensional containment sometimes covers functional relationships between concepts, as indicated earlier.

Finally, an IFO *schema* is a collection of fragments connected to each other by *IS-A links* (generalisation or specialisation). In this chapter, we consider CONCEPT D to have something that corresponds to specialisation and generalisation, but naturally they are both based on the intensional containment relation. In specialisation, the defined concept is a modified version of the concepts that define it. This modification is done by conditions and constraints. In a generalisation, the generalised concept simply gathers those concepts that are shared by the concepts included in it.

Figure 6.3 presents a summary of the counterparts of IFO and our version of CONCEPT D.

IFO	CONCEPT D variant
printable types	concepts (printable)
abstract types	concepts (abstract)
free types	concepts (defined)
finite sets	intensional containment
Cartesian products	intensional containment
fragments (functions)	intensional containment
specialisations	intensional containment with a constraint
generalisations	generalisations
schemata	schemata

Figure 6.3: The counterparts of IFO in our version of CONCEPT D

6.6 Summary and discussion: CONCEPT D and IFO

In this chapter, we have compared two modelling languages, CONCEPT D and IFO, both of which can be understood as semantic data models. The IFO approach concentrates on the mathematical analysis of semantic data models, whereas in COMIC, the conceptual modelling perspective is emphasised.

We have shown the similarities between CONCEPT D and IFO notations. We see these similarities as a justification for transforming a CONCEPT D schema into a IFO schema. This would provide the COMIC approach with conventional semantics (like that of IFO or predicate logic). The transformation procedure will follow the following principles:

- Create an IFO printable type for each CONCEPT D printable concept. If the printable concept at hand has an identifier, determine the range of the printable type to be that of the identifier.

- Recognise CONCEPT D L-items that correspond with IFO fragments. By analysing the structure of a simplified CONCEPT D schema (suggested in Section 6.3), the following steps can be taken:

  - Functions: Any CONCEPT D value transformation maps to an IFO function.
  - Cartesian products (intensional containment edges representing aggregation): if the CONCEPT D structure does not employ a constraint or a generalisation, it corresponds

to an IFO Cartesian product. However, if there is only one contained concept, it corresponds to a IFO function.

- Construct the corresponding IFO types based on each of the relationships identified above.
- The remaining CONCEPT D relations correspond with IS-A relationships (generalisations or specialisations) in IFO. They can be recognised by their graphical form (generalisations) and by the fact that specialisations employ constraints.

  To meet IFO's requirements:

  - for every generalisation, set the type of head node free,
  - for every specialisation, set the type of tail node free.

In our opinion, the benefits of using the COMIC methodology reside in its easiness (a very limited amount of L-items), semantic relativism (no need to make a difference if something is an entity, a named relationship, or an attribute – they are all concepts) and its support for the process of conceptual modelling. COMIC's first implementation in the 1980's was one of the first (possibly the first) data modelling packages in the world where the database design and queries were carried out completely on the level of user knowledge, that is with no visual or terminological reference to the underlying database layer.

However, some objections can be raised concerning both the philosophical background of COMIC and the formalism used.

The philosophical background relies heavily on the theoretical tradition of concepts developed by Leibniz and later axiomatised by Kauppi [Kau67]. This view can be characterised as being thoroughly Platonic, i.e. concepts subsist in their own world, independently of minds and extensions. Kangassalo [Kan92] combines this view with ideas of psychology of cognition (modeller's conceptions of the domain of application and construction of theories). This combination seems beneficial for the purposes of conceptual modelling, but its notion of concept seems a bit vague. It can be best described as "concepts are knowledge structures, build out of knowledge units (other concepts or knowledge primitives) and are independent of minds or extensions, but detected or constructed by modellers".[7]

In this process of detection or construction, intensional containment between concepts plays an important role. Kangassalo maintains in [Kan92] that the relation of intensional containment creates a partial ordering between knowledge units within a body of knowledge. This seems to suggest that all hierarchical structures that we normally use in modelling (the relation of something being something's part or attribute, a generalisation or a specialisation of something) are different kinds of applications of the relation of intensional containment (see, e.g. [Kan96] and [Kan00]). If we accept the Leibniz/Kauppi tradition this may be so, but the following objections should be observed:

- In real life, we are simply interested in whether something is an attribute, a part, or a specialisation of something else, since the nature of the relationship sometimes determines how we carry out reasoning. So there should be a way of including this information in the knowledge structure (perhaps by using knowledge primitives);
- We probably should not assume any particular properties (transitivity, antisymmetry) with this relation, since not all of the different applications have the same kinds of properties;

[7]This description is the author's own, but contains components of papers [Kan83], [Kan93] and [Kan00].

- While the concepts that are within the same knowledge structure can present "entities" (whose counterparts in predicate logic are unary predicates) and "relations" (n-ary predicates, n<1), it is not clear if the background philosophy agrees with this, as mentioned in [Duz00c]. For instance, Kauppi's work [Kau67] clearly makes a distinction between concepts (unary) and relations (n-ary), and they have axiomatic systems of their own.

Apart from the shortcomings mentioned above, CONCEPT D does have an interesting philosophical background. But there are some items that can be used to criticise CONCEPT D as well:

- A theoretical point: CONCEPT D is a language by which concepts and their intensional relationships are described. However, it does not guarantee that using this kind of language we would "automatically" have access to the conceptual domain and thus link the L-items of the schema with something conceptual. In our opinion, this link needs proper semantics.
- A practical point: the expressiveness of the original CONCEPT D is hard to define in terms of more conventional languages or modelling approaches. However, with description logics (DLs) or knowledge representation systems this is a crucial issue, since the reasoning carried out by these systems is dependent on: if the expressive power is too strong, the reasoning is computationally complex, possibly even undecidable. Likewise, if the language does not support L-items like explicit attributes, generalisations or specialisations, the reasoning based on this cannot be implemented at all.

To sum up, the semantic problems of CONCEPT D can be at least partially solved by mapping CONCEPT D diagrams into some other formalism (as in this chapter) or by designing a semantics that supports the intensional characters of the formalism (as in Chapter 7). The evaluation of COMIC methodology and CONCEPT D language depends thus on the view we have about the purpose of its use. If we consider the primary goal of conceptual modelling to be to create intensional descriptions of the domain of application, COMIC and CONCEPT D are most suited for the task. But if we want to make other kinds of descriptions, which emphasise several kinds of relationships (aggregations, attributes, time based relationships), we need to employ another methodology and formalism. The same applies, of course, if the goal is to create a logic based system that needs strict (traditional) semantics.

Chapter 7

Explicating the Semantics of Concept Diagrams

7.1 Introduction

In the previous chapters we discussed the role of language in conceptual modelling. We considered different approaches to conceptual modelling and, based on these approaches, divided the languages used in conceptual modelling (modelling languages) into three categories; extensional, intensional and hybrid languages. An extensional modelling language uses terminology of (extensional) entities and relationships in the description of the domain of application. In an intensional language, concepts have a central role and it is possible to asserts intensional relationships between them. Hybrid languages combine both intensional and extensional features. We used typical modelling constructs to demonstrate the syntactic conventions and "tricks of the trade" of a language from each category.

In this chapter the focus is on what can be seen as being behind the language. We concentrate on the following questions:

- What kind of semantics is useful in conceptual modelling?
- What kind of semantic theory would provide semantics for intensional and hybrid modelling languages?
- How can we apply a semantic background theory to explicate so-called concept diagrams (externalized conceptual schemata created using a formalism like CONCEPT D that employs symbols of concepts and intensional containment)?

If the semantics is not clear, the language used for conceptual modelling will run into obvious difficulties that can compromise the whole effort of conceptual modelling, as in early semantic networks, discussed in [Ran88]. More modern approaches to conceptual modelling, like Classic (presented by Bordiga et al. in [B$^+$89]) and later Description Logics (DLs) [Bor95] feature a much more precise semantics. Often, however, they neglect the intensional aspect of conceptual modelling. That aspect will be emphasised in this chapter.

The starting point of this chapter is to demonstrate how different theories postulate concepts – a key issue in conceptual modelling. We take a closer look at the following theories: possible worlds semantics and situation theory (situation semantics), theories of predication and HIT-semantics.

In Section 7.2, we take a general look at the possible roles of semantics in conceptual modelling and evaluate them on the basis of what they can contribute in conceptual modelling. In Section 7.3, we examine more closely the different semantic approaches. In Section 7.4, we concentrate on HIT-semantics that, based on the analysis, has the features needed to work as a semantics for conceptual modelling. Basic constructs of HIT-semantics are applied to explicate the semantics of CONCEPT D, a modelling language, whose background relies in the intensional containment of concepts. Conclusions are presented in Section 7.5.

7.2 Semantics and conceptual modelling

Conceptual modelling languages can be roughly divided into three categories; extensional languages, intensional languages and "hybrid" languages which utilise both intensional and extensional features. Since the semantics of extensional languages is quite traditional and the semantics of DLs is well established, we shall mostly concentrate on the semantics of intensional and hybrid modelling languages.

As an example, we shall use a variant of the CONCEPT D modelling language. CONCEPT D is based on a modelling methodology with intensional background (see [Kan83], [Kan93], [Kan92]). When CONCEPT D is used, the externalised conceptual schema is expressed by concept diagrams. The actual meaning of the diagrams is, however, sometimes unclear. We maintain that even though *concepts* do not have meanings (they *are* meanings of some expressions), concept *diagrams* exist on the level of language, they have a syntax and gain their meaning trough appropriate semantics. In the case of a language with an intensional background, the semantics clearly needs intensional features, which are studied later.

Intensional containment is an important relation in the set of concepts. In many modelling languages, intensional containment appears as the IS-A relationship. In the case of CONCEPT D, intensional containment covers many other relationships that traditionally have been treated with a different kind of semantics (see [Kan93]).

The following list presents the features needed in an appropriate semantics of concept diagrams:

- A clear distinction between the language (expressions), occurrences (things in the World), and conceptual levels,
- A capability to express what is intensional and what is extensional,
- A capability to express relations between concepts, as follows: (i) there should be a way to express subconcept/superconcept relationships, like that of intensional containment or IS-A. (ii) There should be means to map the different uses of intensional containment into different relationships in the domain of application. (iii) There should be a possibility to include concept-theoretical aspects (like those of Kauppi's concept theory in [Kau67]) into the theory.

Based on the above requirements, we shall evaluate the following semantic approaches of conceptual modelling: possible worlds semantics and situation semantics, theories of predication, and HIT-semantics. They will be introduced in the next section.

Previous research in the field of modelling and semantics is many-faceted and we limit the discussion to semantic data models and CONCEPT D. According to Abiteboul and Hull in [AH87], semantic data models emphasise that data about objects and relationships between them should be modelled in a direct manner. This is manifested in the fact that implementation-dependend features are omitted in the models of our interest. On the other hand (see [AH87]), semantic data models include tools to express functional relationships, IS-A relationships and building "object types" (in our case concepts) out of other object types. Considered this way, ER ([Che76]), EER ([EN94]) as well as IFO ([AH87]) and SDM ([HM81]), and even description logics can all be considered semantic data models. SDM is one of the few ones that support semantic relativity, where something can be equally seen as an entity, a relationship or an attribute. Semantic relativity is an important feature of CONCEPT D as well – what is represented as a concept in CONCEPT D can be seen as an entity, a relationship or an attribute in many other modelling languages. The principles of CONCEPT D have been presented in [Kan83] and [Kan93]. A thorough syntax of CONCEPT D is given in [Kan92]. In this chapter, like in Niemi's papers [Nie00] and [NN01], we use only a part of the original syntax of CONCEPT D. In Niemi's approach, generating a relational database schema and creating an intensional query language have been studied. Both of these involve a semantical point of view: the data in the database have a semantics and queries have a semantics corresponding to the results of each query. Our approach is more concerned with finding the semantics in the concept diagrams.

7.3 The different approaches

In this section, we briefly examine different philosophical theories that can serve as a semantic background for conceptual modelling. It is worth noticing that already in ER presented in [Che76], there was a semantic perspective mainly based on the semantics of first order predicate logic (FOPL). In ER, entities can have the same semantics as unary predicates in FOPL, relations can have the semantics of n-ary predicates, and attributes can have the semantics of functions. ER does not appear to have a possible world perspective, but concentrates on just one domain of application. On the other hand, Classic, a description logic, very clearly postulates possible worlds in its semantics for subtype/supertype definitions (for details, see [B^{+}89]).

Although the approaches taken in ER and Classic have historical importance, we are more interested in explicitly semantic theories.

Situation theory of Barwise and Perry ([BP85], see also [Bar89] and [Per99]) has been suggested to serve as a philosophical background of concept languages in [Woo91]. This theory is based on the logical tradition of Frege, model theory, and possible worlds semantics, but challenges it in many respects. The changes Barwise and Perry present to model theory make the theory work well with so-called opaque contexts, like attitude reports.

HIT semantics (Homogeneous Integrated Type-Oriented data model semantics) of Materna and Duzi (see [Duz92]) is based on the work of Tichy's Transparent Intensional Logic (see e.g. [Tic88]). HIT-semantics is a thorough theory, using a wide array of logical tools. As in situation semantics, the starting point lies in the analysis of natural language expressions. Yet the conclusions are very different.

Theories of predication, as studied by Palomäki in [Pal94], spring from another tradition of logics. Using second order logic, different formulations of ontological views of concepts (nominalism, conceptualism, realism) are considered.

We shall now take a look at how these different approaches formulate concepts, intensional containment, and other items that are important in conceptual modelling.

7.3.1 Possible worlds semantics and situation semantics

Possible world semantics has a close connection with modal logics, i.e. logics that formulate the notions of possibility and necessity. The idea is that a proposition is possible if it is true in some (accessible) world and necessary if it is true in all (accessible) worlds.

In possible world semantics, the interpretations of some expressions (individual constants, predicates and functions) are studied in the context of a collection of possible worlds instead of a single world (or "Universe of Discourse").

Let W be a non-empty (possibly infinite) set and R a binary relation on W. Intuitively, W is a set of possible worlds and R a relation that states what worlds are accessible from the given one.

If $w \in W, w' \in W$ and $\langle w, w' \rangle \in R$, we say that world w‘ is possible with respect to world w or that w‘ is accessible from w [BS79].

More formally, a *frame* is a tuple $\langle W, R \rangle$ consisting of a non-empty set of possible worlds W and R that is a relation on W. W is called a *domain* of $\langle W, R \rangle$, and R the *accessibility relation*. Let D be a valuation, and in this simple case, let it relate members of W to proposition symbols. Now, we call a tuple $\langle W, R, D \rangle$ a *model*, where $\langle W, R \rangle$ is a frame and D is a valuation on W.

We can inspect concept symbols (terms that correspond to concepts) in a model, so that for a symbol of an individual concept we interpret it to individuals in each accessible world; and for a symbol of a predicate or a relation we interpret it to a set of individuals or a set of tuples in each accessible world.

Another method is to use possible worlds to define a concept. Following Hintikka and Montague, Rantala in [Ran03] defines a concept as a function (possibly partial) from the possible worlds we are considering to extensions in those worlds. We can express this in a more formal manner if we categorise concepts as individual concepts (e.g. the concept of Thomas Jefferson), properties ("being a person") or relations ("someone being someone's child") and propositions ("It rained in Geneva on 3rd June 1999.").

In the following definition by Rantala in [Ran03], w refers to a possible world, and properties are considered 1-place relations.

- If c is an individual concept then $c(w)$ is an individual of w, that is, an element of the domain of w (if c is defined at w);
- If c is an n-ary relation concept, then $c(w)$ is a set of n-tuples of individuals of w;
- If c is a proposition, then $c(w)$ is a truth value at w.

As an example, we can assume that **person** is a 1-place relation concept. There, $c(w)$ for the actual world would be a set of persons. Naturally, this does not equate the extension of **person** with persons of the actual world, since other possible must be considered, too.

With the presented formalism, we can now express the connection between possible world semantics and intensional containment, as follows (see [Ran03]):

Let **c** and **d** be concepts. We use the notation "$\mathbf{c} \geqslant \mathbf{d}$" to denote that concept **c** intensionally contains concept **d**" as in Chapter 4. Now, for all concepts **c** and **d**,

$$\mathbf{c} \geqslant \mathbf{d} \Rightarrow \text{for all worlds } w, c(w) \subseteq d(w).$$

Since this is only an implication, we cannot say that we had defined intensional containment by means of possible worlds semantics. Rantala (see [Ran03]) maintains, however, that with some restrictions on Kauppi's theory (e.g. excluding the "empty concept" G and contradictory concepts), the following would apply:

$$\mathbf{c} \geqslant \mathbf{d} \Leftrightarrow \text{for all worlds } w, c(w) \subseteq d(w)\ .$$

It is apparent that this definition would require us to identify intensional containment with the IS-A relationship. Moreover, this analysis casts aside some questions by omitting the empty concept and contradictory concepts.[1]

Situation semantics can be seen as an extension and criticism of possible worlds semantics. It is not explicit about the role of concepts, since it is concerned with meanings and information. In certain ways, Barwise and Perry are reluctant to admit the existence of concepts as such.[2] It seems, however, justifiable to identify meanings with concepts.

According to Perry in [Per99], "the basic idea of situation semantics is that in thought and action we use complexes of objects and properties to *directly* and *indirectly* classify parts and aspects of reality, or *situations*." Moreover, "we classify situations by what goes on in them; which properties objects have, and the relation they stand into one another in virtue of the events that comprise the situation".

The act of classifying situations according to some uniformities gives rise to meaningful relations. These uniformities are "invariants across real situations", like individuals, properties, relations, and locations [BP85].

A simple example of a situation is a *state of affairs*, expressed as a tuple of these invariants, a time indicator t, and a truth value (indicating whether the state of affairs holds at time t).

In [Per99] Perry states, too, that "various objects are built on the basic interplay of situations and states of affairs, permitting complex and abstract ways of classifying situations, including complex states of affairs, properties and relations. A key concept is a *type of situation* [..]. One type of situation may *involve* another: if there is a situation of the first type, there will also be one of the second type [..]. These sorts of states of affairs are called *constraints*".

Concerning meanings, Barwise and Perry maintain in [BP85] that a meaning is a relationship between utterances and described situations. In order to analyse so-called embedded sentences, as follows. Suppose "I am sitting" and "Carson City is west of L.A." both have the same truth value. In many traditional semantic accounts, attitude reports that involve embedded sentences cannot make a difference between two attitude reports if the embedded sentences have the same truth value (like "John thinks that I am sitting." and "John thinks Carson City is west of L.A."). However, in situation semantics, the reference/meaning of a sentence ("I am sitting" or "Carson City is west of L.A.") is not a truth value but a situation.

Though the theory can explain well how meanings come about, it probably fails to explain the special nature of intensional containment. For instance, the expression "kissing means touching" is explained "Kissings and touchings are uniformities across situations recognised by human beings in this culture - relational activities" [BP85].

Still, this does not completely undermine an interesting possibility of applying situation semantics in conceptual modelling. Situation semantics introduces a selection of constraints that can be applied to situations. We can easily identify some variants of intensional containment with necessary constraints and some with conditional constraints. Thus, a conceptual schema as a whole corresponds to a possible situation.

[1]For details, see Wood's critique in [Woo91] and Section 8.2.

[2]Barwise and Perry strongly put forward the idea of "Priority of External significance". This means that the information within things in the world comes first and gives rise to mental significances. There should not be other "privileged" realms like ideas or senses (concepts) (see [BP85], p.42). As opposed to "realism" (concepts exist independently) and "conceptualism" (concepts exist in minds), the position of Barwise and Perry could be described as "nominalism" (concepts do not exist as such).

7.3.2 HIT semantics

Since HIT semantics have been introduced in various sources ([Duz92], [Duz00c], [Duz00a], [Duz00b]), we only give a cursory account of the theory here. Moreover, we neglect the distinctions between TIL and HIT and regard them as a uniform theory. In practice, TIL is concerned with the meanings of expressions in general, whereas HIT is applied to meanings of data structures.

According to this approach, concepts are meanings of expressions (of a language) and constructions are methods of reaching them. A HIT conceptual schema consists of (HIT) attributes, that are constructions, and a set of consistency constraints connected to the attributes [Duz00c].

A HIT attribute is a function whose domain is a set of states-of-affairs (see below), and range is a set extensional functions (database tables). Thus, it associates with every state-of-affairs a database table. For instance, the attribute “Address of a person” selects a function which associates with each person his or her address.

In this short presentation, only the basic features of HIT-semantics can be explained. We shall first introduce constructions, then the base of sorts on which HIT-attributes operate, and finally HIT-attributes in general. Here, to make a distinction between the term “object” in HIT and, e.g. object oriented design, we use the term “T-object” to designate physical or conceptual objects. A type, like PERSON, is a set of T-objects. Moreover, constructions “build” T-objects, since they are ways by which T-objects are obtained; as such, they can be understood as functions that “return” simple or structured T-objects. A construction that builds a T-object is called a T-construction. In HIT-semantics, concepts and intensions are special kinds of T-objects, as will be explained later in this section.

A base is a collection of mutually disjoint non-empty sets. The *epistemic base* consists of (o) truth values, true and false; (ι) the Universe of Discourse, whose members are individuals; (τ) time points expressed by real numbers; and (ω) possible worlds. States of affairs are pairs $\langle w,t\rangle$, where $w \in \omega$ and $t \in \tau$. This is often expressed in the form of (wt).

A *type* is either

- an elementary type (a member of the base),
- if T_1, T_2 are types, then $(T_1 \to T_2)$, a set of partial functions from T_1 to T_2, is a type,
- if $T_1, .., T_n, (n > 1)$ are types, then $(T_1, .., T_n)$, a Cartesian product of T_1,..,T_n, is a type.

Construction can be defined as follows (see also [Duz00a]):

Let there be types for each of which there is an unlimited number of representatives, variables, and a total function (valuation) which assigns one object of the given type to each variable. A T-variable is a variable of type T, and a T-construction constructs a T-object..

- Atomic constructions are variables. A variable x v-constructs a T-object which the valuation v assigns to x.
- If X is a T-object, ${}^0X$ is a construction called trivialization. ${}^0X$ constructs simply X without any change.
- Let $A_1, .., A_n$ be $T_1, .., T_n$ constructions, respectively. $(A_1, .., A_n)$ is a $(T_1, .., T_n)$ construction called a *tuple*.
- Let $(A_1, .., A_n)$ be a $(T_1, .., T_n)$ construction. $A_{(1)}, .., A_{(n)}$ are $T_1, .., T_n$ constructions, respectively, called *projections*.

- Let F be a $((T_1,..,T_n) \to T)$ -construction and $A_1,..,A_n$ $T_1,..,T_n$ -constructions, respectively. $[F(A_1,..,A_n)]$ is a T-construction called the *application* of F to $(A_1,..,A_n)$.
- Let $x_1,..,x_n$ be pairwise different $T_1,..,T_n$ -variables and A a T-construction. Then $\lambda x_1,..,x_nA$ is a $((T_1,..,T_n) \to T)$ -construction called the λ *abstraction* of A on $x_1,..,x_n$.

For the purposes of discussing intensions of concepts, we define here the notion of *subconstruction*, as follows [Duz00b]:

Let C be a construction

- C is a subconstruction of C.
- Let C be ${}^0X$. If X is a construction, then X is a subconstruction of C.
- Let C be $[F(X_1,..,X_n)]$. Now, $F, X_1,..,X_n$ are subconstructions of C.
- Let C be $\lambda x_1,..,x_nX$. Now, X is a subconstruction of C.
- If A is a subconstruction of B and B is a subconstruction of C, then A is a subconstruction of C.

As we have seen, the definition of a type is rather general. In modelling, we are interested in specific cases, types that manifest themselves in the domain of application. These are called sorts and they are members of the base of sorts.

The *base of sorts* in HIT semantics is the set $\{E, D, \tau, \omega\}$, where E is a set of entity sorts (given solely by a property, like “being a person”), D is a set of descriptive sorts (corresponds to printable), τ is the set of time points, and ω the set of possible worlds as above.

It is not a logical necessity that a person, say N.N., has a certain address. Therefore, the (actual) world and the time point must be considered when expressing an attribute “Address of a person”, leading to the form of construction

$$(wt) \to ((T_1) \to (T_2)),$$

where T_1 and T_2 are sorts, in this case PERSON and ADDRESS, respectively.

The logical distinction between intensional and extensional T-objects is based on the form of the construction by which a T-object is achieved:

- A T-object that is *not* of the form $w \to T$ is an extension (also called intension of the 0th degree). Examples of this are, naturally, time points and individuals, but also analytical functions; that is, functions that are not dependent on possible worlds or points of time.
- Let T-object T be of intension of the nth degree. $(w \to T'')$ -object, where T'' is either T or $(t \to T), t \in \tau$ is an intension of $(n+1)$th degree.

Intensions of the 1st degree or higher are called, briefly, intensions. Specifically, $((wt) \to (T_1 \to T_2))$ is called an empirical function.

Because of its form, the function ”Address of a person” is an empirical function and thus an intension.

HIT-attributes are, indeed, empirical functions and of the form:

- $(wt) \to (T_1 \to T_2)$ (singular attributes) or

- $(wt) \to (T_1 \to (T_2 \to o))$ (multivalued attributes)

As stated earlier, a HIT-database schema consists of HIT-attributes and consistency constraints. Consistency constraints rule out impossible or undesired states of affairs in the world (see [Duz00b]). They enforce rules such as "for each material there is always a supplier". Consistency constraints are expressed using constructions, but we suppress the details here.

The conceptual schema in HIT-semantics is, however, not equal to a CONCEPT D diagram. In CONCEPT D diagrams, some of the relationships between concepts cannot be seen as empirical functions – especially some forms of intensional containment and functions. Thus, we consider concepts and intensional containment on the basis of constructions, following [Duz00b] and [Duz00a]:

- A concept is a closed construction, i.e. a construction without free variables.
- A simple concept is a construction ${}^0X$, where X is a variable (of any type) or an object that is not a construction.
- The content of a concept C is the set of subconstructions of C that are themselves concepts.

Some forms of intensional containment in CONCEPT D correspond very apparently to the last item: in HIT-semantics, concept C intensionally contains concept C', if and only if C' is a member of the content of C. If we accept this notion and if we equate CONCEPT D's intensional containment relation with it, we can use HIT semantics as the backgound theory of CONCEPT D.

It should be emphasised, like Palomäki in [Pal97] and [Pal02], that the contaiment relation established by subconstructions does *not* correspond to Kauppi's intensional containment relation. Therefore, we can not argue that we could provide the basis to Kauppi's concept theory using HIT semantics. However, we see HIT semantics as a viable option to establish the semantics of CONCEPT D diagrams. This will be considered in Section 7.4.1.

7.3.3 Theories of predication

Theories of predication are based on the idea of equating propositional functions, functions whose value range is a proposition, with concepts. Propositional functions can be logically analysed using second order predicate logic (2OPL). As Palomäki has demonstrated, different ontological views concerning concepts (nominalism, conceptualism, realism) can be studied this way as well.

These different ontological views give different interpretations to the following formula, known as the *comprehension principle*:

$$(\exists F^n)(\forall x_1)..(\forall x_n)(F^n(x_1,..,x_n) \leftrightarrow \phi),$$

where F^n is an n-place predicate variable, $n > 0$, F^n does not occur free in ϕ and ϕ is a well-formed formula (wff) of 2OPL with distinct individual variables $x_1,..,x_n$.

The main benefit of this approach is that it is a thorough formalisation of different views, and the formalisation contains semantics as well. As an example, we shall briefly examine logical realistic semantics that can be seen as a fruitful background for conceptual modelling. This study has been adopted from Palomäki's presentation in [Pal94] with only minor editing.

First, we need to introduce the syntax of 2OPL as follows:

- The terminal alphabet of language L_{2OPL} consists of logical constants $\neg, \rightarrow, \forall, \exists, =$; a countably infinite number of individual variables $x_i, i \in \mathbb{N}$; a countably infinite number of n-place predicate variables F^n for each $n \in \mathbb{N}$; a countably infinite number of m-place predicate constants P^m, for each $m \in \mathbb{N}$; the parentheses and the comma.

- The grammar can be expressed as follows: Wffs are expressions. Atomic wffs are (i) of the form $(x = y)$, where x and y are individual variables or (ii) of the form $F^n(x_1,...,x_n)$, where F^n is an n-place predicate variable or constant and $x_1,...,x_n$ are individual or predicate variables. Other wffs are of the form $\neg\phi$, $(\phi \rightarrow \theta)$, $(\forall x)\phi$, $(\exists x)\phi$, $(\forall F^n)\phi$, $(\exists F^n)\phi$, where ϕ and θ are wffs, x is an individual variable and F^n is an n-place predicate variable.

To define the semantics, two auxiliary tools will be needed: an interpretation function and a completely referential assignment.

Let L_{2OPL} be a language (i.e. the set of all wffs) and D a non-empty set. Let f be a function whose domain is L_{2OPL}. Thus, f is an interpretation function that maps (some) expressions of the language into individuals and sets of D, as follows. Let D^n be the set of all n-tuples whose constituents are in D. For all $n > 0, n \in \mathbb{N}, P^n \in L_{2OPL} : f(P^n) \subseteq D^n$ is a model for L_{2OPL}.

A completely referential assignment in D is a function A whose domain is the union of sets of individual and predicate variables such that $A(x) \in D$ for each individual variable x and $A(F^n) \subseteq D^n$ for each n-place predicate variable F^n $(n < 0, n \in N)$. When $d \in D$, let $A(d/x) = (A - \{\langle x, A(x)\rangle\}) \cup \{\langle x, d\rangle\}$ and when $X \subseteq D^n$ let $A(X/F^n) = (A - \{\langle F^n, A(F^n)\rangle\}) \cup \{\langle F^n, X\rangle\}$.

The semantics can be defined as follows, when L_{2OPL} is a language, $U = \langle D, f\rangle$ is its model, and A is a completely referential assignment in D:

- A satisfies $(x = y)$ in U if and only if $A(x) = A(y)$,
- A satisfies $P^n(x_1,..,x_n)$ in U if and only if $\langle A(x_1),..,A(x_n)\rangle \in f(P^n)$,
- A satisfies $\neg\phi$ in U if and only if A does not satisfy ϕ in U,
- A satisfies $(\phi \rightarrow \theta)$ in U if and only if A does not satisfy ϕ in U or A satisfies θ in U,
- A satisfies $(\forall x)\phi$ in U if and only if for all $d \in D$, $A(d/x)$ satisfies ϕ in U,
- A satisfies $(\exists x)\phi$ in U if and only if for some $d \in D$, $A(d/x)$ satisfies ϕ in U,
- A satisfies $F^n(x_1,..,x_n)$ in U if and only if $\langle A(x_1),..,A(x_n)\rangle \in A(F^n)$,
- A satisfies $(\forall F^n)\phi$ in U if and only if for all $X \subseteq D^n, A(X/F^n)$ satisfies ϕ in U.
- A satisfies $(\exists F^n)\phi$ in U if and only if for some $X \subseteq D^n, A(X/F^n)$ satisfies ϕ in U.

A theory is defined in a usual manner, i.e. a theory is $\langle S, L\rangle$ where S is a set of wffs of language L. In a theory of predication, L equals L_{2OPL} and S consists of a "normal" set of 2OPL axioms as well as the axiom schema of conceptualisation principle, cited above.

The semantics of the conceptualisation principle can now be equated with different ontological views of concepts. In the case of logical realism this task is simple, since according to logical realism every fully applied wff $\phi(x_1,..,x_n)$ is assumed to represent a real universal. Thus, if A is a completely referential assignment in D, as above, and $\phi(x_1,..,x_n)$ is a standard wff of 2OPL in which F^n does not occur free, then A satisfies $(\exists F^n)(\forall x_1)..(\forall x_n)(F^n(x_1,..,x_n) \leftrightarrow \phi)$ in U.

To sum up, according to theories of predication, concepts are propositional functions. Being based on second order predicate logic, the theory "automatically" contains a distinction between language, extensional and concept levels. On the other hand, the theory does not really explain the features needed in conceptual modelling (like intensional containment).

7.4 HIT semantics as a background theory of the semantics of CONCEPT D

7.4.1 Modelling constructs and CONCEPT D

According to a rather subjective survey of modelling situations in Chapter 2, the following features/constructs often occur in conceptual modelling:

- Representing some objects as entities;
- Representing an aggregation to connect some features of objects to the objects themselves. The features will be called attributes. Attributes are not considered to exist independently, i.e. without entities;
- Expressing that some attributes are necessary;
- Expressing that some attributes are identifying;
- Representing "member of" relationships by grouping;
- Representing arbitrary relationships between entities;
- Representing subclass-superclass relationships (IS-A relationships) between entities;
- Representing "part-of" relationships between entities and their attributes, or between entities;
- Representing functions by which derived information is calculated.

We indicated in Chapter 6 that all of these features are covered by CONCEPT D. In CONCEPT D, there is no difference between something being an attribute or entity, there are only nodes called concepts. The principal method of connecting nodes to each other is the relationship of (the direct) intensional containment, that covers aggregation, grouping, IS-A and part-of relationships (according to [Kan93]). Additional information is provided by cardinality constraints and identifiers (keys) that indicate identifying attributes. Conditions and functions are used, in addition to intensional containment, to carry information about how the nodes are related to each other. A more thorough account of the features of CONCEPT D can be found in Chapter 6. For the sake of brevity, in this chapter we only consider issues of entities, relationships – including IS-A relationships (generalisation, specialisation) – and aggregations. It should be noticed that in CONCEPT D, generalisation and specialisation are only variants of aggregation (see [KV90]) but here we discuss generalisation and aggregation as independent structures, as in Chapter 6. The graphical conventions of representing these structures were introduced in Chapter 6, too.

In our simplified CONCEPT D terminology:

- A concept is a node, identified by the name of the concept.

- The relation of direct intensional containment connects concepts to each other: concept **a** intensionally contains concept **b**, if (as in Section 1.3) concept **b** belongs to the knowledge contents of concept **a**. There can be cardinality constraints in the intensional containment between two concepts. By default, the relation is 1:1, but can be defined as being 1:n.

- There can be constraints between concepts that are contained by intensional containment either directly or indirectly.

- Aggregation, generalisation and specialisation are structures based on intensional containment. These structures are directed acyclic graphs with a root node.

- Aggregation can be simply a 1:n intensional containment between two concepts, like in the case of PERSONNEL and EMPLOYEE, where PERSONNEL contains N EMPLOYEEs (in this case, aggregation corresponds to grouping in the terminology of Chapter 2).

  In an aggregation structure, several concepts can be (directly) contained in a single one instead of one in the PERSONNEL - EMPLOYEE example. For instance, ZIP-CODE, CITY and STREET-ADDRESS can be contained in ADDRESS. This corresponds to aggregation in the terminology of Chapter 2.

- In a generalisation structure, a concept is expressed as a generalised version of concepts related to it. In [Kan93], several variants of generalisation (unconstrained, explicit and implicit) are introduced, but here we consider only unconstrained generalisation. Our form of generalisation is "automatically" considered to be a 1:1 relation (thus, no cardinality indicator is needed in our syntactic generalisation statements below).

- In a specialisation structure, there is a constraint between the specialised concept (e.g. SENIOR CITIZEN) and its criteria (AGE being greater than, say, 64 years). This is the only way we use constraints in this version of CONCEPT D.

As an additional feature we present the type of concept to be either printable or abstract. While a distinguishing feature of CONCEPT D is that the user can build the externalised conceptual schema using almost entirely the terminology of the domain of application (i.e. no "attributes", "roles" or other technical terminology), we feel that the users can easily interpret whether a concept is printable or abstract. The benefit of this addition is that interpretation of the concept diagrams in terms of IS-A and aggregation becomes much easier, as we shall demonstrate in Section 7.4.3.

7.4.2 Syntax of limited CONCEPT D

We now define the syntax of this modified CONCEPT D language in a rather informal manner to allow for an explanation of the syntax.

concept(name,type)	Concepts are introduced using this notation. A name is a unique identifier of the concept; a type states if the concept is printable (p) or abstract (a). For instance: concept(student-card-number,p), concept(age,p), concept(student,a), concept(employee,a), concept(senior-citizen,a), concept(person,a).
intensionally-contains(c1,c2,n)	Intensional containment between two concepts, c1 and c2, is declared using this notation; concept c1 intensionally contains

	concept c2. The cardinality constraint (1:1 or 1:n) between the concepts is expressed by the third argument that is either '1' or 'n'. For example: intensionally-contains(student,student-card-number,1).
generalisation(c,c1,c2,..)	This notation declares generalisation of concept c using concept c1,c2,.. For example: generalisation(person,student,staff).
constraint(c1,c2,limit-of-c1)	This notation expresses a constraint between concepts c1 and c2. The only type of constraint considered here is an arithmetical one. For example: constraint(age,senior-citizen,$>$ 64).

Next, we define (informally) the constraints with which a valid schema of limited CONCEPT D must conform.

Ic statements and generalisation statements together define a graph in the following sense (for graph theoretical details, see, e.g. [Eve79]):

- In intensionally-contains statements, c1 and c2 form an ordered pair. Let the set of ordered pairs formed by intensionally contains statements be called V_1.
- In generalisation statements, c and c1, c and c2, .. form ordered pairs. Let the set of ordered pairs formed by generalisation statements be called V_2.
- In concept statements, let concept names be members of a set named N.
- $G = \langle N, V_1 \cup V_2 \rangle$ is a directed graph, where $V_1 \cup V_2$ are the edges of the graphs.
- A path is defined based on the transitive closure of the edges as usual.

Based on the graph and the statements, the following must hold:

- G is acyclic, connected and has a root vertex.
- $V_1 \cap V_2 = \emptyset$ (for practical reasons we want to make a difference between generalisation statements and intensionally-contains statements).
- Let a be any abstract concept and p any printable concept. $\langle p, a \rangle \notin V_1 \cup V_2$ (a printable concept may not directly intensionally contain an abstract concept).
- Let there be a constraint statement constraint(a, b, c). There must exist a path p where a precedes b (in a specialisation, the specialisation criteria must be intensionally contained in the concept that it specialises). Moreover, in intensionally-contains statements where a is the first argument, only '1' is acceptable as the last argument (all specialisations must be 1:1).

It is interesting to notice that the syntax defined above is much like a syntax of a Description Logic (see. e.g. [Bor95], [Lam96], [DLN$^+$92]). However, the actual logical (computational) components of CONCEPT D have not been defined.

As stated earlier, we concentrate on CONCEPT D structures whose semantic counterparts can be seen as entities, relationships – including IS-A relationships (generalisation, specialisation) and grouping – and aggregations (attributes). We can now identify the semantically different constructs in our limited version of CONCEPT D, and how they would correspond to HIT terminology namely:

- Nodes and edges: The nodes correspond to HIT entity sorts (abstract) or descriptive (printable). The edges have no semantics as such, but it depends on the construct in which they take part;
- Intensional containment: IS-A in the case of specialisation and generalisation, otherwise aggregation or grouping.

The semantics of these structures, based on HIT semantics, will be discussed in the next section.

7.4.3 Semantic counterparts of CONCEPT D in HIT

In a more formal discussion, we should give explicit rules for combinations of syntactic elements and thus explain what their semantical counterparts are. This kind of method would be very similar to the one used in DLs (e.g. [B$^+$89], [Lam96]) where the syntax is first introduced, and then, for instance, an IS-A statement is given its semantics. Here, we apply a more informal method in the discussion of CONCEPT D structures that can be identified as IS-A structures, grouping and aggregation.

Only specialisation and generalisation are recognised as IS-A structures. We have defined all specialisation and generalisation structures to be 1:1. We consider this a necessary relation. For specialisation and generalisation, intensional containment defined with contents and subconstruction as in Section 7.3.2 can be applied. However, we present another method as well, in 7.4.3.1. This corresponds to the "standard" semantics of IS-A -structures in HIT-semantics. In sections 7.4.3.2 and 7.4.3.3, we discuss aggregation and grouping. It should be emphasised that though these structures are called intensional containment in CONCEPT D, they do not correspond to intensional containment in HIT-semantics.

7.4.3.1: IS-A structures: Based on [Duz00b], we define:

An entity sort E_1 that is determined by property P_1 is a subtype of an entity sort E_2 if, in all the states of affairs, the population of E_1 is a subset of the population of E_2.

To analyse the IS-A structures more profoundly, we can use the notions of constructions and subconstructions in the sense of Materna [Mat00]. Here, we identify primitive concepts $C_1,..,C_m$. Concepts $C_{m+1}, C_{m+2},..$ are distinct from them and $C_1,..,C_m$ are their subconstructions.

A grouping structure between two concepts A and B is, for example, CLASS – STUDENT or CHAPTER – PARAGRAPH. In our version of CONCEPT D, it can be identified from a structure where there is only one concept (B) directly contained in another (A) and both A and B are of the same type (either printable or abstract). In Section 7.4.1 we indicated that in CONCEPT D, a grouping structure is a special case of an aggregation structure. Therefore, we discuss the semantics of other aggregation structures before grouping structures.

7.4.3.2: Aggregation structures: In HIT semantics, the counterpart of an aggregation structure is, simply, an attribute. Suppose a CONCEPT D structure where concept A directly contains concepts B$_1$, B$_2$, .. ,B$_n$. Earlier we stated that HIT attributes are empirical functions. Thus, in its simplest form, the corresponding attribute is $(wt) \rightarrow (A \rightarrow ((B_1, B_2, ..B_n) \rightarrow o))$.

7.4.3.3: Grouping structures: Respectively, a grouping structure is an attribute in HIT semantics as well. This attribute constructs multiple instances and, consequently, it is multi-valued. Given the example above, we express it as $(wt) \rightarrow (A \rightarrow (B \rightarrow o))$.

7.5 Summary and discussion: semantics and concept diagrams

Previously, in [Nii98] we tried to explicate the semantics of intensional containment by normal modal logic. According to this theory, concepts are predicates; there is a notion of logical necessity and intensional containment is based on that.

This account can be challenged in many ways; here we only state that it is profitable to seek alternatives that contribute to conceptual modelling from the point of view of semantics. We have identified the requirements of a suitable semantic theory to be as follows:

- A clear distinction of the language, occurrences and conceptual levels,
- A capability to express what is intensional and what is extensional,
- A capability to explain subconcept/superconcept relationships (IS-A, intensional containment). There should be sufficient means to map the different uses of intensional containment into different relations in the domain of application.
- A possibility to include concept-theoretical aspects into the theory.

We have considered the following alternatives: situation semantics in Section 7.3.1, HIT-semantics in Section 7.3.2 and theories of predication in Section 7.3.3. Among the alternatives, theories of predication focus on the philosophical background and situation semantics deals with the semantics of utterances in daily life situations. However, HIT-semantics has features by which it can, among the candidates, be best applied to conceptual modelling. This is because the theory clearly postulates concepts (as constructions), and defines a criteria for intensionality (intension as an empirical function, intensional containment as a containment between constructions). HIT-semantics is, furthermore, flexible enough to incorporate other concept-theoretical aspects. For instance, the structural limitations discussed in Chapter 4 can be expressed as consistency constraints. However, as indicated by Palomäki in [Pal97] and [Pal02] the intensional containment relation (based on subconstructions) of HIT does not equal to that in Kauppi's concept theory.

In Section 7.4 we discussed the other main theme of this chapter, i.e. the semantics of CONCEPT D diagrams. A limited version of CONCEPT D was introduced by means of a definition of its syntax in Section 7.4.2 and informal semantics in section 7.4.3. The semantics of the language was then discussed using the tools of HIT-semantics. As the discussion indicates, HIT-semantics is applicable for this purpose and could be applied to explain the semantic background of other formalisms as well.

Chapter 8

Summary and discussion: the Intensional Perspective in Conceptual Modelling

In this chapter, Section 8.1, we present the main points of the research. Moreover, in Section 8.2 we bring together issues related to the realm of "conceptual" in conceptual modelling. The issues discussed are: (i) the intensional perspective, (ii) how the intensional perspective can be seen from the point of view of semantics, and (iii) how the different modelling languages discussed here address the problem. This gives impetus to provide a view of (iv) how to combine the use of intensional and extensional modelling languages in a typical modelling situation. Finally, we discuss (v) other considerations related to modelling and address some items for future research.

8.1 Summary of the research

The principal hypothese and results of this research can be presented as follows:

- It is fruitful and possible to create a framework of conceptual modelling languages and compare them with each other. The categorisation of the languages can be based on how the languages express concepts and intensions. We have divided conceptual modelling languages in the categories of intensional, extensional, and hybrid languages based on that.

- It is meaningful to use the notions of first order predicate logic as a basis of discussion of semantics of any extensional language. By doing so, we have demonstrated that (a limited version) of the language of conceptual graph has expressive power equal to (slightly limited) first order predicate logic. We have demonstrated, too, that conceptual graphs are not always more intuitive than corresponding first order predicate logic formulas, contrary to the claim of John Sowa.

- It is possible to construct a modelling language based purely on Kauppi's concept theory. It is possible, too to create formal tools (an abstract implementation) to manage the externalised conceptual schemata of this kind of language, as has been demonstrated in this thesis.

- It is possible and meaningful to map a CONCEPT D schema to an IFO schema, under some constraints. This is beneficial, since it could combine the intuitiveness of CONCEPT D with precise semantics of IFO.

- It is possible to consider the semantics of concept diagrams (CONCEPT D schemata) from the point of view of semantical theories. It is possible, too, under some constraints, to employ HIT semantics to explicate the "meanings" of concept diagrams.

8.2 Discussion

(i) The intensional perspective in conceptual modelling.

In previous chapters we observed the different ways in which the term "intension" is understood. As a summary, the philosophical base is the distinction of the intension and the extension of a concept. The intension of a concept is seen as the internal contents ([eb-94b]), or the knowledge contents of the concept ([Kan93]). In Kauppi's theory in [Kau67] the intension is based on the (undefined) intensional containment relation. On the other hand, Bunge discusses the intension as a set of properties and relations "subsumed under the concept" (see [Bun67]), and Kangassalo's formalism in [Kan93] indicates a connection between "**x** is (partially) defined by **y**" and "**x** intensionally contains **y**". With possible worlds semantics, intensions are seen as combinations (sets) of extensions in all (accessible) possible worlds, and intensional containment can be based on subset relationships of these sets. With HIT-semantics, a conceptual containment relation can be based on "subconstructions", i.e. concepts are seen as constructions and the elements of construction A can be elements of construction B, too.

Still, the term "intensional" needs clarification. In database literature, almost anything related to the structure of the data (e.g. the design of the relations in a relational database) is called intensional, as opposed to the "extensional" real data. However, in the scope of this thesis, we see intensionality basically as being based on intensional containment. The view is expanded in Section (ii) below.

In Chapter 2, we tentatively divided modelling approaches into three categories: intensional, extensional and hybrid. The question of something being intensional is not very important in the extensional approach. In the intensional approach, something being intensional is nearly synonymous to something belonging to the contents of a concept. In hybrid approaches, both intensional and extensional features can be modelled and utilised, and the intensional features normally include IS-A relationships.

A purely intensional approach is discussed in Chapters 4 and 5. This kind of approach is clear and theoretically well-established, since it relies heavily on concept theory developed by Kauppi [Kau67]. However, this approach limits the description of the domain of application completely to the level of concepts and intensional containment. While this is most useful as a background, in many applications we are interested in making a difference between several kinds of language items (entities, relationships, attributes, IS-A, part-of, etc.).

COMIC methodology [Kan93] and CONCEPT D modelling language [Kan83] have a broader view of the intensional approach than that of the "pure core" described in Chapters 4 and 5.

We can call CONCEPT D a concept definition oriented modelling language; the user of the language expresses his or her knowledge of the domain of application in the form of concept structures that resemble definitions. Together, these concept structures form a CONCEPT D (externalised conceptual) schema. The structures consist of concepts, intensional containment relations among them, and knowledge primitives other than concepts, such as constraints, identifying keys, etc. The relation of intensional containment is used to express both general IS-A -type relationships (that probably prevail independently of the domain of application), and highly domain dependent relationships, too. For instance, using the relation the user can express that the concept of **PERSON** contains the concepts **LIVING BEING** and **ADDRESS**. There, the containment of **LIVING BEING** should be considered a traditional IS-A relationship, while the containment of **ADDRESS** is more like an contingent attribute (not every person in every situation has an address).

In order to make a distinction between different kinds of intensional containment structures, two different approaches are proposed in this thesis. In both approaches we have only considered a subset of CONCEPT D in order to make the comparisons between CONCEPT D and other modelling languages more feasible. The first one, in Chapter 6, suggests translating CONCEPT D schemata into IFO schemata, IFO being a well known modelling language. However, given the different natures of CONCEPT D, the translation appears forced. Another approach, in Chapter 7, is based on the investigation of different semantical theories. We primarily consider HIT semantics of Materna, Duzi and others [Duz01] and proceed to show that using rather simple rules with a limited CONCEPT D language a mapping can be established between structures in a CONCEPT D schema and HIT data model.

(ii) Semantics

In his influential paper [Woo91], Woods discusses the role of concepts in knowledge presentation languages, especially Description Logics, like KL-ONE and its followers. Woods does not like to identify the notion of concept with the notion of predicate in FOPL, and emphasises that concepts are structural and "intensional in the sense in which [morning star] and [evening star] are intensionally distinct concepts". Furthermore, Woods criticises possible worlds semantics, where (in its some forms) concepts are identified with their extensions in all the accessible worlds: **prime number less than one** and **round square** have empty extensions in all possible worlds, but they are still distinct concepts.

Wood suggests "conceptual descriptions" that would preserve the structural features needed, and thus make the difference between **morning star** and **evening star**, on one hand, and **prime number less than one** and **round square**, on the other. However, Woods does not give an exact definition of a concept in his paper. Therefore we discuss another possibility of analysis, which preserves the intensional emphasis of Woods but establishes a more integral theory as well. This theory can better explain how to address the question of **prime number less than one** versus **round square**.

Woods seems to emphasise that concepts are something structural and abstract, distinct from expressions. We shall see how these notions can be explained in the context of Materna's analysis in [Mat00]. There, concepts have extensions and intensions; the contents (Kauppi's intension) of a concept is a set "but the concept itself can be construed as some procedure that 'organizes' the elements of the contents" [Mat00]. Thus, a concept can be seen as this procedure *and* the contents (the details of this view, especially constructions, are explained in Section 7.3.2).

Let us now define some clarifying notions, based on [Mat00]:

- We say that (linguistic) expressions denote objects. As a special case, they can *refer* to things in the world (or in an imaginary world). In such a case, the expression is empirical. Non-empirical expressions can be found in formal sciences.

- Expressions E1 and E2 are synonymous if and only if they express one and the same concept.

- Expressions E1 and E2 are equivalent if and only if they denote one and the same object.

- E1 and E2 are coincident if and only if they are empirical and they share the same intension in the actual world and time.[1]

It is rather easy to find coincident expressions ("human", "featherless biped") as well as equivalent ones ("a polygon with three angles", "a polygon with three sides", cf. [Woo91]). But we notice an "anomaly" in the definition of synonymity: concepts are expressed, not denoted. This is due to the theory that concepts are constructions, as discussed in 7.3.2. When it comes to concepts, both their contents and the way of putting them together matter, so the concepts of **prime number less than one** and **round square** are, indeed, distinct concepts. Naturally, the expressions "morning star" and "evening star" are coincident.

We have now seen that Woods was right in his emphasis of the "intensional" or, at least, concept-oriented approach in modelling. But using Materna's theory, we can express this concept oriented emphasis in a more concise manner. In what follows, we shall examine how the idea that both the contents and the procedure of organization (putting things together) matter when it comes to concepts.

(iii) Language support

What follows is obvious with respect to CONCEPT D: **Evening star** intensionally contains **celestial body** and **shines in the evening**, **morning star** includes **celestial body** and **shines in the morning**; thus, the concepts are different. This naturally applies to concepts like **prime number less than one** and **round square**.

This, however, is just the contents side of concepts. The procedure of organization, "putting together", can be illustrated by the example in Figures 8.1 and 8.2. Here, we follow Bernard Bolzano's notion (in his book *Wissenshaftslehre*) that the concepts **an uneducated son of an educated father** and **educated son of an uneducated father** differ. Following CONCEPT D's background theory (see [Kan93]), we assume here the following view of intensional containment: "concept **a** intensionally contains concept **b** if the knowledge content of concept **b** is a part of the knowledge contents of concept **a**." This enables us to present intensional containment relationships, such as **an uneducated son of an educated father** intensionally contains **uneducated son**".[2]

We see in Figures 8.1 and 8.2 that the "putting together" is crucial to make a difference between the concepts **an uneducated son of an educated father** and **an educated son of an uneducated father**, both of which eventually contain the same basic concepts (**father, son, educated, uneducated**). Naturally, Figure 8.1 explicates the concepts **an uneducated son of an educated father** and **an educated son of an uneducated father** better, since it illustrates the concepts of **educated son**, **uneducated father**, **uneducated son** and **educated father**. The explication or "discovery" of these concepts can be supported by a methodology like the one described in Section 4.6.

The same example could, of course, be introduced using a DL, like in the simplistic presentation of figure 8.3. It would be, however, rather artificial to try to present it using a language like ER.

[1]According to Materna's definition "the value of the intension denoted by them is the same in the actual world and time".

[2]If we want to avoid the potentially ambiguous term "knowledge contents of a concept", we can equate "concept **x** is (partially) defined by concept **y**" relation with the intensional containment relation, too, as in Chapter 6.

Bolzano's example was analyzed by Kauppi in [Kau67], but in her analysis Kauppi used "relation concepts". This extension to her concept theory is not presented in this thesis.

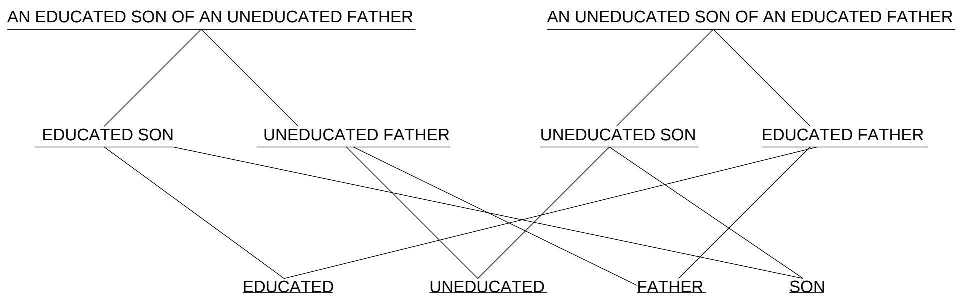

Figure 8.1: A "proper" way of "putting together" **an uneducated son of an educated father** and **an educated son of an uneducated father**.

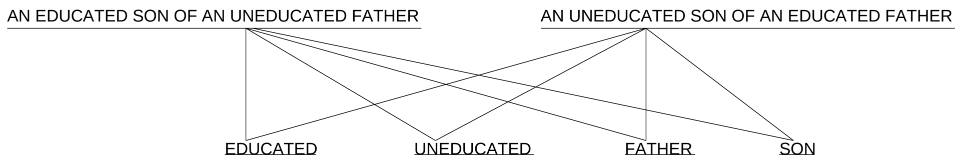
AN EDUCATED SON OF AN UNEDUCATED FATHER
AN UNEDUCATED SON OF AN EDUCATED FATHER
EDUCATED
UNEDUCATED
FATHER
SON

Figure 8.2: The same example showing only the contents, but less "putting together".

uneducated-son-of-educated-father $\doteq$
(and uneducated-person (all father-slot educated-person)
(atleast 1 father-slot))

educated-son-of-uneducated-father $\doteq$
(and educated-person (all father-slot uneducated-person)
(atleast 1 father-slot))

educated-person $\dot{\leq}$ person

uneducated-person $\dot{\leq}$ person

person $\dot{\leq} \top$

Figure 8.3: A DL presentation of the previous example.

(iv) Combining intensional and extensional approaches

As mentioned in Chapter 1, a modeller may design something that does not yet exist in the domain of application. The modeller probably visualises and evaluates the prospective features of his design by using concepts. These concepts do not necessarily currently have a non-empty extension, but (unless they are logically impossible) they have an extension in some possible "world", at least the one envisioned by the modeller. If the modeller is successful, the concepts will have an extension, partially because of his modelling activity. We can assume as well that the intellectual activity of the modeller is (at least on the conceptual level) more or less the same whether he uses an intensional or extensional modelling language when reporting the results of his modelling.

In Chapter 6 we stated that it would be profitable to combine the benefits of the intensional approach (easiness, semantic relativism and support for the process of conceptual modelling) with those of the extensional approach (popularity, most often simple and unambiguous semantics). To achieve that goal, we discussed the possibility of designing the externalised conceptual model by using a language that resembles CONCEPT D, and then translating the model into IFO notation. Here, we propose a slightly alternative method.

In Section 2.4 we presented a simple intensional modelling language and in Figure 2.5, a modelling example. As we can see in the figure, the externalised conceptual schema is quite readable and can probably be used as a medium of communication between a modeller and domain experts. This could be utilised as follows:

- The modeller collects information about the domain of application and reports the results using an intensional modelling language like CONCEPT D or the simple language of Section 2.4.
- As long as the modelling is based solely on concepts and the intensional containment relation, the modeller can utilise the tools (legality checking, finding association relations) developed in Chapters 4 and 5. This will possibly guide the modeller to discover "new" concepts that he will find useful to add in the schema.
- When the modeller thinks to have found all the concepts needed to express the information contents, the perspective can be changed from a purely conceptual one. That is, in order to make the semantics of the schema more explicit and to facilitate a simple mapping of this schema and a database, the modeller adds information about how the intensional containment relation should be interpreted in each of its occurrences in the schema.

Figure 2.5 with the information of the interpretation of each intensional containment edge added can be seen in Figure 8.4. There, A stands for aggregation, IS-A stands for an IS-A relationships and GR stands for a grouping. Niemi, Nummenmaa and Thanish have presented a language that resembles our notation in [NNT00].

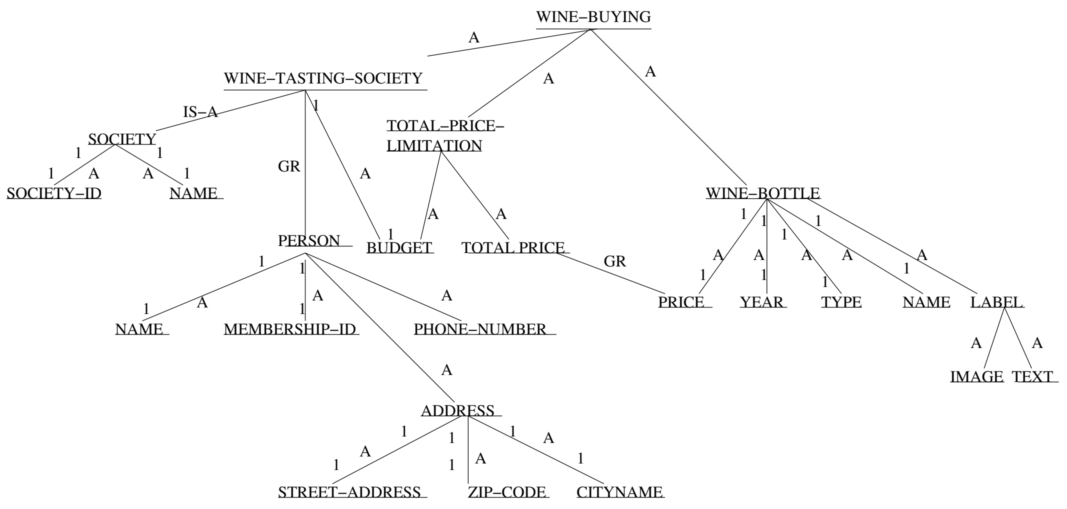

Figure 8.4: Figure 2.5 with interpretation symbols of intensional containment edges.

(v) Other considerations and future research

We can state the main points of intensions and semantics in conceptual modelling as follows:

- It appears to be useful to discuss intensional and extensional modelling languages or, to see the question from a different angle, to discuss intensional and extensional features in modelling languages. The criterion for a language to be intensional is that it states some sort of intensional containment relation, i.e., "intensional = based on intensional containment"[3];
- With the intensional approach (and intensional modelling language), we can use concept operations and possibly utilise them in concept analysis and discovery;
- Constructing semantics suitable for the intensional approach is possible (e.g. with HIT semantics), but complicated. However, if we combine intensional and extensional perspectives, we can facilitate the explication of the semantics of externalised conceptual schemata.

We can ask whether in conceptual modelling the choice of the modelling language really matters. In engineering (including software engineering) people have been using fundamentally extensional modelling languages for over a decade, though from the theoretical point of view we see their shortcomings. However, if we want our (externalised conceptual) schemata to be well-understood and to avoid paradoxes, we should rely on semantically well-established theories. In engineering, we normally consider extensional, though complex, objects. But the role of a semantically well-established theory is more important in the fields where the subject matter is not necessarily tangible; natural language processing and codes of law are good examples of that.

As explained in [Pal94], it is not too uncommon to see theory construction as a modelling task. In what follows, we shall shortly discuss the modelling methodology and its theory connection.

In Chapters 2 and 6, we discussed the possible gain for conceptual modelling if a COMIC-like methodology were combined with a modelling language that is the most beneficial in each of the domains of applications. Among the benefits of COMIC we find simplicity and concentration on the conceptual level. Moreover, Vaden [Vad94] has discussed how COMIC methodology corresponds to structural views of scientific theories.[4]

CONCEPT D is a natural choice for a modelling language when using COMIC methodology, but for many specific purposes CONCEPT D is not the most applicable language. In order to be able to use COMIC in connection with a modelling language other than CONCEPT D, in Chapters 6 and 7 we presented how to map CONCEPT D externalised conceptual schemata with externalised conceptual schemata that are expressed using a more conventional language.

Naturally, we have the possibility of abandoning COMIC in favour of other methodologies, which are numerous. In Object Role Modelling (ORM, [BBMP95]), one of the recent achievements has been to define an axiomatised kernel that relates modelling constructs (in this case object types, generalisation, specialisation, grouping, etc.) to each other and presents constraints to them. ORM can be criticised on the basis that it does not support a clear theoretical distinction of what is intensional and what is not – this task is left to the modeller. Other possible starting points for a methodology could be the following:

- A methodology based on the background theory of Conceptual Graphs (see Chapter 3.3 and [Sow84]). The theory is many-faceted, consisting of at least the conceptual graph notation itself; type hierarchy; proof theory; theory of heuristics in conceptual graph based artificial intelligence systems; and ways to represent data flow within the graphs.

[3]This applies, naturally, to languages defined in the context of this thesis. For languages of intensional logic, see [vB88].

[4]It must be mentioned, however, that the structural view probably cannot be defined as precisely as the more traditional statement view, and that Tichy [Tic71] has demonstrated how to manipulate statement view based theories using second order predicate logic, that is the basis of HIT, too.

- A methodology based on description logics and formal ontologies. This kind of methodology would possibly be closely related to existing COMIC, but allow one to use a description logic as the language of representing the externalised conceptual schema. This sort of approach could easily incorporate the ontological analysis of different IS-A relationships as in [GW00] and the logical rigour of Catarzi's and Lenzerini's "conceptual data base modelling" [CL92].

- A methodology based on HIT -semantics. As we have seen in Chapter 7, HIT-semantics has its background in thorough logical theory, but the basic modelling construct (attribute) is flexible and relatively easy to understand.

All of these possibilities also serve as items for further studies. The goal is, naturally, to develop intension-supporting modelling languages and methodologies. This kind of a language could be in some senses analogous to the popular XML (Extensible Markup Language) formalism in software development (see, e.g. [B$^+$01]). This would guarantee the portability of externalised conceptual schemata; there would be a common understanding of the guidelines of their usage and a rich variety of tools supporting their design. A simple implementation of an XML based modelling tools is discussed in [NS03].

Another direction of further studies is that of mereology (see e.g. [Sim87]). The mereological relation is apparently similar to the intensional containment, and mereology could be used as a background of the semantics of intensional containment.

Bibliography

[AFGP96] A. Artale, E. Franconi, N. Guarino, and L. Pazzi. Part-whole relations in object-centered systems: an overview. *Data and Knowledge Engineering*, 20(3), 1996.

[AH84] S. Abiteboul and R. Hull. IFO: A formal semantic database model. In *Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, Waterloo, Ontario, Canada, April 2nd-4th, 1984.

[AH87] S. Abiteboul. and R. Hull. IFO: A formal semantic database model. *ACM Transactions on Database Systems*, 12(4), 1987.

[ASCD99] A. Analyti, N. Spyratos, P. Constantopoulos, and M. Doerr. Inheritance under participation constraints and disjointness. In H. Jaakkola, H. Kangassalo, and E. Kawaguchi, editors, *Information Modelling and Knowledge Bases X*. IOS Press, 1999.

[Ata99] M. J. Atallah, editor. *Algorithms and Theory of Computation Handbook*. CRC Press, 1999.

[B$^+$89] A. Borgida et al. CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*. Portland, Oregon, May 31st - June 2nd, 1989.

[B$^+$01] M. Birbeck et al. *Professional XML*. Wrox, 2nd edition, 2001.

[Bar89] J. Barwise. *The Situation in Logic*. Number 17 in CSLI Lecture Notes. CSLI, 1989.

[BBMP95] G.H.W.M. Bronts, S.J. Brouwer, C.L.J. Martens, and H.A. Proper. A unifying object role modelling theory. *Information Systems*, 20(3), 1995.

[BCN92] C. Batini, S. Ceri, and S. Navathe. *Conceptual Data Base Design: An Entity-Relationship Approach*. Benjamin/Cummings, 1992.

[Ber99] A. Berztiss. Concepts, objects and domains. In H. Jaakkola, H. Kangassalo, and E. Kawaguchi, editors, *Information Modelling and Knowledge Bases X*. IOS Press, 1999.

[Bla96] S. Blackburn. *The Oxford Dictionary of Philosophy*. Oxford University Press, 1996.

[Bor91] A. Borgida. Knowledge representation, semantic modeling: Similarities and differences. In H. Kangassalo, editor, *Entity-Relation Approach: The Core of Conceptual Modelling*. Elsevier Science Publishers, 1991.

[Bor95] A. Borgida. Description logics in data management. *IEEE transactions on knowledge and data engineering*, 7(1), 1995.

[BP85] J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, 1985.

[Bra83] R.J. Brachman. What IS-A is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer*, 16(10), 1983.

[BS79] R. Bradley and N. Swartz. *Possible Worlds*. Blackwell, 1979.

[BS85] R.J. Brachman and J. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2), 1985.

[BS92] S. Bergamaschi and C. Sartori. On taxonomic reasoning in conceptual design. *ACM Transactions on Database Systems*, 17(3), 1992.

[BS98] F. Baader and U. Sattler. Description logics with aggregates and concrete domains, part ii (extended). Technical Report LTCS-Report 98-02, Aachen University of Technology, Research Group for Theoretical Computer Science, 1998.

[Bun67] M. Bunge. *Scientific Research I: The Search for System*. Springer-Verlag, 1967.

[BW77] D. Bobrow and T. Winograd. An overview of KRL, a knowledge representation language. *Cognitive Science*, 1(1), 1977.

[Che76] P. Chen. The entity-relationship model - towards a unified view of data. *ACM Transactions on Database Systems*, 1(1), 1976.

[CL92] T. Catarci and M. Lenzerini. Conceptual database modeling through concept modeling. In S. Oshuga, H. Kangassalo, H. Jaakkola, K. Hori, and N. Yonezaki, editors, *Information Modelling and Knowledge Bases III*. IOS Press, 1992.

[CLN98] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*. Kluwer Academic Publisher, 1998.

[DLN$^+$92] F.M. Donini, M. Lenzerini, D. Nardi, B. Hollunder, W. Nutt, and A. M. Spaccamela. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 53(2/3), 1992.

[DLNN97] F.M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1), 1997.

[Duz92] M. Duzi. *Logic and Data Semantics*. PhD thesis, Dept. of Logic, Institute of Philosophy, Czechoslovak Academy of Sciences, 1992.

[Duz00a] M. Duzi. A contribution to the discussion on concept theory. Discussion paper in the 10th European-Japanese Conference on Information Modelling and Knowledge Bases, May 2000.

[Duz00b] M. Duzi. Logical foundations of conceptual modelling. A manuscript, 2000.

[Duz00c] M. Duzi. Two approaches to conceptual data modelling. In O. Majer, editor, *Topics in Conceptual Analysis and Modelling*. Academy of Sciences of the Czech Republic, 2000.

[Duz01] M. Duzi. Logical foundations of conceptual modelling using HIT data model. In H. Jaakkola, H. Kangassalo, and E. Kawaguchi, editors, *Information Modelling and Knowledge Bases XII*. IOS Press, 2001.

[eb-94a] The history and kinds of logic. In *The New Encyclopaedia Britannica, Macropaedia*, volume 23. Encyclopaedia Britannica, 15th edition, 1994.

[eb-94b] Intension and extension. In *The New Encyclopaedia Britannica, Macropaedia*, volume 6. Encyclopaedia Britannica, 15th edition, 1994.

[eb-94c] Ontology. In *The New Encyclopaedia Britannica, Macropaedia*, volume 8. Encyclopaedia Britannica, 15th edition, 1994.

[EN94] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Benjamin/Cummings, 2nd edition, 1994.

[Eve79] S. Even. *Graph Algorithms*. Computer Science Press, 1979.

[FLS94] G. Falquet, M. Leonard, and J. Sindayamaze. F2Concept: a database system for managing classes' extensions and intensions. In H. Jaakkola, H. Kangassalo, T. Kitahashi, and A. Markus, editors, *Information Modelling and Knowledge Bases V*. IOS Press, 1994.

[GF92] M. Genesereth and R. Fikes. *Knowledge Interchange Format – Reference Manual*. Computer Science Department, Stanford University, Stanford, California, 3rd edition, 1992. Logic Group Technical Report Logic-92-1.

[GG95] N. Guarino and P. Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In N.J.I Mars, editor, *Towards Very Large Knowledge Bases*. IOS Press, 1995.

[Gre91] R. Mac Gregor. The evolving technology of classification-based knowledge representation systems. In J.F. Sowa, editor, *Principles of Semantic Networks*. Morgan Kaufman, 1991.

[Gua97] N. Guarino. Understanding, building and using ontologies. *International Journal of Human Computer Studies*, 46(2/3), 1997.

[GW00] N. Guarino and C. Welty. Ontological analysis of taxonomic relationships. In A. Laender, S. Liddle., and W. Storey, editors, *Conceptual Modeling - ER 2000, 19th International conference of Conceptual Modeling*, Lecture Notes in Computer Science 1920, Salt Lake City, Utah USA, October, 2000.

[Hau86] A. Hautamäki. *Points of View and their Logical Analysis*, volume 41 of *Acta Philosophica Fennica*. Philosophical Society of Finland, 1986.

[Hau01a] R. Hausser. *Foundations of Computational Linguistics, Human-Computer Communication in Natural Language*. Springer, 2nd. edition, 2001.

[Hau01b] R. Hausser. The four basic ontologies of semantic interpretation. In H. Jaakkola, H. Kangassalo, and E. Kawaguchi, editors, *Information Modelling and Knowledge Bases XII*. IOS Press, 2001.

[HK87] R. Hull and R. King. Semantic data modelling: Survey, applications and research issues. *ACM Computing Surveys*, 19(3), 1987.

[HLvR96] T.W.C. Huibers, M. Lalmas, and C.J. van Rijsbergen. Information retrieval and situation theory. *SIGIR Forum*, 30(1), 1996.

[HM81] M. Hammer and D. McLeod. Database description with SDM: A semantic database model. *ACM Transactions on Database Systems*, 6(3), 1981.

[HT00] T. Horrocks and S. Tobies. Reasoning with axioms: Theory and practice. In A. Cohn, F. Giunchilia, and B. Selman, editors, *KR 2000, Principles of Knowledge Representation and Reasoning, Proceedings of the Seventh International Conference*, Breckenridge, Colorado, USA, April 11-15, 2000, 2000.

[JBR99] I. Jacobson, G. Booch, and J. Rumbauch. *The Unified Software Development Process*. Addison-Wesley, 1999.

[Jec78] T. Jech. *Set Theory*. Academic Press, 1978.

[JN93] K. Järvelin and T. Niemi. Deductive information retrieval based on classifications. *Journal of the American Society for Information Science*, 44(10), 1993.

[Joh90] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. Elsevier Science Publishers, 1990.

[JP89] M. Jackman and C. Pavelin. Conceptual graphs. In G.A. Ringland and D.A. Duce, editors, *Approaches to Knowledge Representation*. Wiley and Sons, 1989.

[Jun98] M. Junkkari. The modelling primitives for component relationships and a 'design by example' method. Technical Report A-1998-13, University of Tampere, Dept. of Comp. Sci., 1998.

[Jun01] M. Junkkari. The systematic object-oriented representation for managing intensional and extensional aspects in modeling of part-of relationships. Technical Report A-2001-5, University of Tampere, Dept. of Comp. Sci., 2001.

[Kan82] H. Kangassalo. On the concept of concept in a conceptual schema. In H. Kangassalo, editor, *First Scandinavian Research Seminar on Information Modelling and Database Management*, volume 17 of *series B*. University of Tampere, 1982.

[Kan83] H. Kangassalo. Concept D - a graphical formalism for representing concept structures. In H. Kangassalo, editor, *Second Scandinavian Research Seminar on Information Modelling and Database Management*, volume 19 of *series B*. University of Tampere, 1983.

[Kan92] H. Kangassalo. On the concept of concept for conceptual modelling and concept detection. In S. Oshuga, H. Kangassalo, H. Jaakkola, K. Hori, and N. Yonezaki, editors, *Information Modelling and Knowledge Bases III*. IOS Press, 1992.

[Kan93] H. Kangassalo. Comic: A system and methodology for conceptual modelling and information construction. *Data and Knowledge Engineering*, 9, 1993.

[Kan96] H. Kangassalo. Conceptual description for information modelling based on intensional containment relation. In F. Baader, M. Buchheit, M.A. Jeusfeld, and W. Nutt, editors, *Knowledge Representation meet Databases, Proceedings of the 3rd Workshop KRDB'96*, Budapest, Hungary, August 13th, 1996.

[Kan00] H. Kangassalo. Frameworks of information modelling: Construction of concepts and knowledge by using the intensional approach. In S. Brinkkemper, E. Lindencrona, and A. Solvberg, editors, *Information Systems Engineering, State of the Art and Research Themes*. Springer, 2000.

[Kau67] R. Kauppi. *Einführung in die Theorie der Begriffssysteme*, volume 15 of *series A*. University of Tampere, 1967.

[KKJH00] E. Kawaguchi, H. Kangassalo, H. Jaakkola, and I.A. Hamid, editors. *Information Modelling and Knowledge Bases XI, Preface*. IOS Press, 2000.

[KKP90] S. Kari, H. Kangassalo, and J. Pösö. CQL – conceptual query language: A visual user interface to application data bases. In H. Kangassalo, S. Ohsuga, and H. Jaakkola, editors, *Information Modelling and Knowledge Bases*. IOS Press, 1990.

[KV90] H. Kangassalo and A. Viitanen. A concept data base for conceptual schemata. In H. Kangassalo, S. Ohsuga, and H. Jaakkola, editors, *Information Modelling and Knowledge Bases*. IOS Press, 1990.

[Lam96] P. Lambrix. *Part-Whole Reasoning in Description Logics*. PhD thesis, Linköping Studies in Science and Technology No. 448, 1996.

[Lan78] B. Langefors. *Theoretical analysis of information systems*. Studentlitteratur, 4th edition, 1978.

[LB85] H.J. Levesque and R.J. Brachman. A fundamental tradeoff in knowledge representation and reasoning. In R.J. Brachman and H.J. Levesque, editors, *Readings in Knowledge Representation*. Morgan Kaufman, 1985.

[Lip76] S. Lipschutz. *Discrete Mathematics*. Schaum's outline of Theory and Problems. McGraw-Hill, 1976.

[Mar97] E. Marjomaa. *Aspects of relevance in information modelling: methodological principles and conceptual problems*. PhD thesis, University Of Tampere, 1997.

[Mar02] E. Marjomaa. Peircean reorganization in conceptual modeling terminology. *Journal of Conceptual Modeling*, January 2002.

[Mat00] P. Materna. Two notions of concepts. In O. Majer, editor, *Topics in Conceptual Analysis and Modelling*. The Institute of Philosophy, Academy of Sciences of the Czech Republic, 2000.

[MM01] E. Marcos and A. Marcos. A philosophical approach to the concept of data model: Is a data model, in fact, a model? *Information Systems Frontiers*, 3(2), 2001.

[Mot94] A. Motro. Intensional answers to database queries. *IEEE Transactions on Knowledge and Data Engineering*, 6(3), 1994.

[Nau72] D. Nauta. *The meaning of information.* Mouton, 1972.

[NH89] G. Nijssen and T. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach.* Prentice-Hall, 1989.

[Nie99] T. Niemi. Transforming Concept D schema into relational database schema. In H. Jaakkola, H. Kangassalo, and E. Kawaguchi, editors, *Information Modelling and Knowledge Bases X.* IOS press, 1999.

[Nie00] T. Niemi. New approaches to intensional concept theory. In E. Kawaguchi, H. Kangassalo, H. Jaakkola, and I.A. Hamid, editors, *Information Modelling and Knowledge Bases XI.* IOS press, 2000.

[Nii98] M. Niinimäki. *Käsitteellinen mallintaminen, ekstensionaaliset ja intensionaaliset käsitekielet (Conceptual modelling, extensional and intensional concept languages).* Licenciate Thesis, University of Tampere, 1998.

[Nii99] I. Niiniluoto. *Critical Scientific Realism.* Oxford University Press, 1999.

[NN01] T. Niemi and J. Nummenmaa. A query method based on intensional concept definition. In H. Jaakkola, H. Kangassalo, and E. Kawaguchi, editors, *Information Modelling and Knowledge Bases XII.* IOS Press, 2001.

[NNT00] T. Niemi, J. Nummenmaa, and P. Thanisch. Applying dependency theory to conceptual modelling. In O. Majer, editor, *Topics in Conceptual Analysis and Modelling.* The Institute of Philosophy, Academy of Sciences of the Czech Republic, 2000.

[NP98] J.F. Nilsson and J. Palomäki. Towards computing with extensions and intensions of concepts. In P.J. Charrel, H. Jaakkola, H. Kangassalo, and E. Kawaguchi, editors, *Information Modelling and Knowledge Bases IX.* IOS Press, 1998.

[NS03] M. Niinimäki and V. Sivunen. Experiences in computer assisted xml-based modelling. In H. Kangassalo, E. Kawaguchi, Y. Kiyoki, and H. Jaakkola, editors, *Information Modelling and Knowledge Bases XV.* IOS Press, 2003.

[Pal94] J. Palomäki. *From Concepts to Concept Theory.* PhD thesis, University of Tampere, 1994.

[Pal97] J. Palomäki. Three kinds of containment relations of concepts. In H. Kangassalo, J.F. Nilsson, H. Jaakkola, and S. Ohsuga, editors, *Information Modelling and Knowledge Bases VIII.* IOS Press, 1997.

[Pal02] J. Palomäki. Intensional vs. conceptual content of concepts. In H. Kangassalo and E. Kawakuchi, editors, *Proceedings of the 12th European-Japanese Conference on Information Modelling and Knowledge Bases*, Krippen, Germany, May 27 - 30, 2002, 2002.

[Per99] J. Perry. Semantics, situation. In *Concise Routledge Encyclopedia of Philosophy.* Routledge, 1999.

[Poh96] K. Pohl. Requirements engineering: An overview. In A. Kent and J. Williams, editors, *Encyclopedia of Computer Science and Technology, Vol. 36. Supplement 21*. Marcel Deccer Inc, 1996.

[Qui68] M.R. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*. MIT Press, 1968.

[Ran88] D. Mac Randal. Semantic networks. In G.A. Ringland and D.A. Duce, editors, *Approaches to Knowledge Representation*. Research Studies Press, 1988.

[Ran03] V. Rantala. Possible worlds. In L. Haaparanta and I. Niiniluoto, editors, *Analytic Philosophy in Finland*, Poznan Studies in the Philosophy of the Sciences and the Humanities. Rodopi, 2003.

[Rob73] D. D. Roberts. *The Existential Graphs of Charles S. Peirce*. Mouton, 1973.

[Sat95] U. Sattler. A concept language for engineering applications with part-whole relations. In A. Borgida and D. Nardi, editors, *Proceedings of the International Workshop on Description Logics DL-95*, Roma, Italy, 1995.

[Sim87] P. Simons. *Parts: A Study in Ontology*. Clarendon, 1987.

[Sow84] J.F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison Wesley, 1984.

[Sow00] J. F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing, 2000.

[Sup57] P. Suppes. *Introduction to Logic*. D. van Nostrand Company, 1957.

[SV93] H. Salminen and J. Väänänen. *Johdatus logiikkaan (Introduction to logics)*. Gaudeamus, 1993.

[SW98] J. Simpson and E. Weiner, editors. *The Oxford English Dictionary*. Oxford University Press, 1998.

[Tau91] B. Tauzovich. Towards temporal extensions to the entity-relationship model. In T. J. Teorey, editor, *Proceedings of the 10th International Conference on Entity-Relationship Approach (ER'91)*, San Mateo, California, USA, 23rd-25th October, 1991.

[TCI87] Information processing systems Technical Committee ISO/TC 97. *Information processing systems – Concepts and Terminology for the Conceptual Schema and the Information Base*. ISO, TR 9007:1987 (E), 1987.

[Tic71] P. Tichy. Synthetic components of a finite class of postulates. *Archiv für Mathematische Logik und Grundlagenvorschung*, 2(17), 1971.

[Tic88] P. Tichy. *The Foundations of Frege's Logic*. De Gruyter, 1988.

[TJ92a] Niemi T and K. Järvelin. Operation-oriented query language approach for recursive queries – part 1: Functional definition. *Information systems*, 17(1), 1992.

[TJ92b] Niemi T and K. Järvelin. Operation-oriented query language approach for recursive queries – part 2: Prototype implementation and its integration with relational databases. *Information systems*, 17(1), 1992.

[Vad94] T. Vaden. Conceptual modelling as theory construction: Some implications for the metalevel treatment. In H. Jaakkola, H. Kangassalo, T. Kitahashi, and A. Markus, editors, *Information Modelling and Knowledge Bases V*. IOS Press, 1994.

[vB88] J. van Benthem. *A manual of intensional logic*. CSLI, 1988.

[vD97] D. van Dalen. *Logic and Structure*. Springer, 3rd edition, 1997.

[VvB82] G. Verheijn and J. van Bekkum. Niam: An information analysis method. In T.W. Olle, H.G. Sol, and A. Verrijn-Stuart, editors, *Information Systems Design Methodologies: A Comparative Review (CRIS '82 Proceedings)*. Elsevier Science Publishers, 1982.

[WCH87] M.E. Winston, R. Chaffin, and D. Herrman. A taxonomy of part-whole relations. *Cognitive Science*, 11, 1987.

[Win75] H.P. Winston. Learning structural descriptions from examples. In H. P. Winston, editor, *Phychology of Computer Vision*. McGraw-Hill, 1975.

[Woo91] W.A. Woods. Understanding subsumption and taxonomy: A framework for progress. In J.F. Sowa, editor, *Principles of Semantic Networks*. Morgan Kaufman Publishers, 1991.

[Yov93] M. Yovits. Data and information. In A. Ralston and E. Reilly, editors, *Encyclopedia of Computer Science*. Chapman & Hall, 3rd edition, 1993.

[Zal00] E. Zalta. A (Leibnizian) theory of concepts. In U. Meixner and A. Newen, editors, *Philosophiegeschichte und logische Analyse, Band 3*. Mentis, 2000.

[Zho98] L. Zhongwan. *Mathematical Logic for Computer Science*. World Scientific, 2nd edition, 1998.

Appendix A

FOPL CG Translations

G1. a is a painting:

$$painting(a).$$

The formula is in the translation form. The counterpart for this formula is the conceptual graph

$$[PAINTING : a].$$

G2. There exists (at least one) painting:

$$\exists x(painting(x)).$$

The formula is in the translation form. The counterpart for this formula is the conceptual graph

$$[PAINTING : x].$$

G3. There exists something that is not a painting:

$$\exists x(\neg(painting(x))).$$

The formula is in the translation form. The counterpart for this formula is the conceptual graph

$$[\neg PAINTING : x].$$

G4. There is no (such thing as) kitsch.

$$\neg(\exists x(kitsch(x))).$$

The formula is in the translation form. The counterpart for this formula is the conceptual graph

$$\neg[[KITSCH : x]].$$

G5. There are artists and collectors:

$$(\exists x(artist(x)) \wedge \exists y(collector(y))).$$

In the translation form, the formula is $(\exists x(\exists y((artist(x)) \wedge (collector(y)))))$. The graph counterpart for this formula is

$$[[ARTIST : x] \wedge [COLLECTOR : y]].$$

G6. Someone is both an artist and a collector.

$$\exists x((artist(x) \wedge collector(x))).$$

The formula is in the translation form. The graph counterpart for this formula is

$$[[ARTIST : x] \wedge [COLLECTOR : x]].$$

G7. There is at least someone who is an artist and not a collector.

$$\exists x((artist(x) \wedge \neg(collector(x)))).$$

The formula is in the translation form. The counterpart for this formula is the conceptual graph

$$[[ARTIST : x] \wedge [\neg COLLECTOR : x]].$$

G8. Everything is art:

$$\forall x(art(x)).$$

The formula is in the translation form. The counterpart for this formula is the conceptual graph

$$[ART : \forall x].$$

G9. For everything, there is something that is more beautiful:

$$\forall x(\exists y(morebeautiful(y,x))).$$

Using the inverse relation we get:

$$\forall x((\exists y)(morebeautiful^{-1}(x,y))),$$

that is the translation form of this formula. The graph counterpart is thus:

$$[\top_c : \forall x] \leftarrow (MOREBEAUTIFUL) \leftarrow [\top_c : y].$$

G10. There is something that is more beautiful than everything else:

$$\exists x(\forall y(morebeautiful(x,y))).$$

The formula is in the translation form. Its graph counterpart is

$$[\top_c : x] \rightarrowtail (MOREBEAUTIFUL) \rightarrowtail [\top_c : \forall y].$$

G11. The collector b buys painting a:

$$((collector(b) \wedge painting(a)) \wedge buy(b,a)).$$

The counterpart of this formula is the conceptual graph

$$\begin{array}{c}[[[COLLECTOR : b] \rightarrowtail (BUY) \rightarrowtail [PAINTING : a] \\ \wedge \\ [[PAINTING : a] \wedge [COLLECTOR : b]]].\end{array}$$

According to conditions CGM[1], this means the same as

$$[COLLECTOR : b] \rightarrowtail (BUY) \rightarrowtail [PAINTING : a].$$

G12. Some collector buys the painting a:

$$\exists x(((collector(x) \wedge painting(a)) \wedge buy(x,a))).$$

The translation process is similar to that in G11, resulting in

$$[COLLECTOR : x] \rightarrowtail (BUY) \rightarrowtail [PAINTING : a].$$

G13. There exists a collector who does not buy painting a:

$$\exists x((collector(x) \wedge (\neg(buy(x,a)) \wedge painting(a)))).$$

The translation form of the formula is

$$\exists x((\neg(buy(x,a)) \wedge (collector(x) \wedge painting(a)))).$$

The counterpart for this formula is the conceptual graph

$$\begin{array}{c}[[COLLECTOR : b] \rightarrowtail (\sim BUY) \rightarrowtail [PAINTING : a] \\ \wedge \\ [[PAINTING : a] \wedge [COLLECTOR : b]]].\end{array}$$

According to conditions CGM, this means the same as

$$[COLLECTOR : b] \rightarrowtail (\sim BUY) \rightarrowtail [PAINTING : a].$$

[1]See page 44. Naturally, there should be a syntactic rule to justify this short cut, too.

G14. No collector buys painting a:

$$\neg(\exists x((collector(x) \wedge (painting(a) \wedge buy(x,a))))).$$

The translation form of the formula is

$$\neg(\exists x(buy(x,a) \wedge (collector(x) \wedge painting(a)))).$$

The counterpart for this formula is the conceptual graph

$$\begin{gathered}\neg[[COLLECTOR : x] \leftarrowtail (BUY) \leftarrowtail [PAINTING : a] \\ \wedge \\ [COLLECTOR : x] \wedge [PAINTING : a]].\end{gathered}$$

G15. No collector buys any painting:

$$\neg(\exists x(\exists y((collector(x) \wedge (painting(y) \wedge buy(x,y)))))).$$

The translation process, similar to the one in the previous example, leads to:

$$\begin{gathered}\neg[[[COLLECTOR : x] \leftarrowtail (BUY) \leftarrowtail [PAINTING : y] \\ \wedge \\ [[COLLECTOR : x] \wedge [PAINTING : y]]]].\end{gathered}$$

G16. Each painting has been painted by some artist:

$$\forall x(painting(x) \rightarrow \exists y((artist(y) \wedge paintedby(x,y)))).$$

The process of transforming the formula into the translation form is as follows:
Eliminate $\rightarrow$:

$$\forall x(\neg(painting(x) \wedge \neg(\exists y((artist(y) \wedge paintedby(x,y)))))).$$

Move quantifiers, move binary relations closest to quantifiers:

$$\forall x(\exists y((\neg(painting(x) \wedge \neg((paintedby(x,y) \wedge artist(y))))))).$$

The graph counterpart of this formula is:

$$\neg[[[PAINTING : \forall x] \wedge \neg[[PAINTING : x] \rightarrowtail (PAINTEDBY) \rightarrowtail [ARTIST : y]]]].$$

According to the semantics of the quantifier $\forall$ and the shortcut rule S6, this graph is equal to

$$[PAINTING : \forall x] \rightarrowtail (PAINTEDBY) \rightarrowtail [ARTIST : y].$$

G17. There exists a painting that all the collectors want:

$$\exists x((painting(x) \wedge \forall y((collector(y) \rightarrow want(x,y))))).$$

The translation process is similar to that in the previous example. Its result is the graph:

$$[[PAINTING : x] \wedge$$
$$[COLLECTOR : \forall y] \rightarrowtail (WANT) \rightarrowtail [PAINTING : x]].$$

Appendix B

The definitions of intensional relations and operations as presented by Kauppi

- Comparable: Df_H“$aHb'' =_{df}$ “$(\exists x)(a \geq x \& b \geq x)''$
- Incomparable: $Df_{\text{⌶}}$“$a\text{⌶}b'' =_{df}$ “ $\sim (\exists x)(a \geq x \& b \geq x)''$
- Compatible: $Df_{\overset{|}{\wedge}}$“$a \overset{|}{\wedge} b'' =_{df}$ “$(\exists x)(x \geq a \& x \geq a)''$
- Incompatible: $Df_{\underset{|}{\vee}}$“$a \underset{|}{\vee} b'' =_{df}$ “ $\sim (\exists x)(x \geq a \& x \geq a)''$
- Homogen-compatible: $Df_{\overset{|}{\triangle}}$ “$a\overset{|}{\triangle}b'' =_{df}$ “$aHb \& a \overset{|}{\wedge} b \& \sim a \geq b \& \sim b \geq a''$
- Heterogen-compatible: $Df_{\overset{|}{\text{⚠}}}$“$a\overset{|}{\text{⚠}}b'' =_{df}$ “$a\text{⌶}b \& a \overset{|}{\wedge} b''$
- Opposite: $Df_{\underset{|}{\nabla}}$“$a\underset{|}{\nabla}b'' =_{df}$ “$aHb \& a \underset{|}{\vee} b''$
- Isolated: $Df_{\underset{|}{\overline{\nabla}}}$“$a\underset{|}{\overline{\nabla}}b =_{df}$ “$a\text{⌶}b \& a \underset{|}{\vee} b''$
- Sum: $Df_{\oplus}$“$c = a \oplus b'' =_{df}$ “$(x)(x \geq c \leftrightarrow x \geq a \& x \geq b)''$
- Product: $Df_{\otimes}$“$c = a \otimes b'' =_{df}$ “$(x)(c \geq x \leftrightarrow a \geq x \& b \geq x)''$
- Negation: Df_{-}“$b = \overline{a}'' =_{df}$ “$(x)(x \geq b \leftrightarrow x \underset{|}{\vee} a)''$
- Difference: $Df_{\ominus}$“$c = a \ominus b'' =_{df}$ “$(x)(c \geq x \leftrightarrow a \geq x \& b\text{⌶}x)''$
- Quotient: $Df_{\oslash}$“$c = a \oslash b'' =_{df}$ “$(x)(x \geq c \leftrightarrow x \geq a \& x \underset{|}{\vee} b)$

Notes:

- There is a printing error in [Kau67], p. 44 in the definition of opposite.
- Logical symbols are as follows: negation ($\sim$), conjunction (&), universal quantifier (x).
- The symbol $\geq$ is used here instead of Kauppi's $>$.