MAX-PLANCK-INSTITUT
FÜR
INFORMATIK

Algebraic Terminological Representation

Renate A. Schmidt

MPI–I–91–216 November 1991

INFORMATIK

Im Stadtwald
D 66123 Saarbrücken
Germany

Author's Address

Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany.

Email: schmidt@mpi-sb.mpg.de

Publication Notes

This report is a reprint of a thesis prepared under the supervision of Professor Chris Brink for the degree of Master of Science in Mathematics at the University of Cape Town. The report is also available as Thesis Reprint TR 011 (July 1991) from the Department of Mathematics, University of Cape Town, 7700 Rondebosch, South Africa.

Abstract

This thesis investigates terminological representation languages, as used in KL-ONE-type knowledge representation systems, from an algebraic point of view. Terminological representation languages are based on two primitive syntactic types, called concepts and roles, which are usually interpreted model-theoretically as sets and relations, respectively. I propose an algebraic rather than a model-theoretic approach. I show that terminological representations can be naturally accommodated in equational algebras of sets interacting with relations, and I use equational logic as a vehicle for reasoning about concepts interacting with roles.

Acknowledgements

First and foremost I wish to thank my supervisor Prof. Chris Brink, who has been a constant source of inspiration, guidance and encouragement. For this I remain in his debt.

I am grateful to the following people with whom we corresponded and who sent us important material otherwise hard to obtain: Michael Böttner, Peter Jipsen, Karin Klabunde, Fritz Lehmann, Roger Maddux, Bernhard Nebel, Werner Nutt, Hans Jürgen Ohlbach, Peter Patel-Schneider, Jim Schmolze and Gert Smolka.

I am indebted to Prof. Ronald I. Becker and the Department of Mathematics for generous financial assistance and for the use of all the facilities. Collective thanks go to members of staff, especially Lyn Waldron, and fellow students, especially Ingrid Rewitzky.

Thanks are also due to the Foundation for Research Development for various post-graduate scholarships.

A significant part of my research was carried out while I was visiting the Australian National University on a vacation scholarship in the Summer of 1989/90. I wish to thank the ANU and the Research School of Social Sciences for this opportunity. I am especially grateful to Prof. Michael McRobbie and the Automated Reasoning Project for the pleasant working environment I was privileged to experience.

Finally, I wish to thank my parents and my sisters, Irmela and Karin.

Contents

Introduction

The aim of this thesis is to present an *algebraic* perspective of that branch of Knowledge Representation concerned with *terminological representation languages*. These languages originate from a system called KL-ONE, which arose in the late seventies from the debate on the role of logic in Artificial Intelligence, and in particular the clash between semantic networks and frames. Terminological representation languages have two primitive syntactic types, called *concepts* and *roles*. In the prevalent model-theoretic semantics concepts are interpreted as *sets* and roles as *binary relations*. The approach I propose is based on the fact that sets and relations have simple calculi which can be presented algebraically. The calculus of sets can be presented in the context of *Boolean algebras* and the calculus of relations in the context of *relation algebras*. Concepts and roles also interact in certain ways, and these can be modelled as interactions between sets and relations. For such interactions there also exist algebraic presentations, called *Boolean modules* and *Peirce algebras*. The representation of knowledge then becomes the formulation of certain equations in an algebraic context. But knowledge representation deals not only with *representing* given knowledge, it also deals with *inferring* knowledge which is implicit in the representation. In terminological representation inference amounts to calculation concerning interactions between sets and relations. This is formalised in the algebraic framework by the arithmetic of the respective algebras.

In summary, my work is motivated by the following observations: (i) Terminological representation and reasoning is modelled in calculi of sets and relations. (ii) For these calculi algebraic formalisations exist. In this thesis I combine these two facts and show that the algebraic framework provides a natural setting for both terminological representation and reasoning.

The thesis contains four chapters. The following summarises their contents:

Chapter 1: By way of a core example which I refer back to throughout the thesis I give a preview of terminological representation and of my proposal.

Chapter 2: This chapter is devoted to terminological representation. I present an extensive overview of its evolution from KL-ONE-based formalisms up to current developments. I formally define the syntax and model-theoretic semantics of two terminological languages, chosen to illustrate the expressiveness attained by such formalisms.

Chapter 3: This chapter is devoted to algebra. To start with I briefly outline the general algebraic notions and results relevant in subsequent sections and Chapter 4. There are five sections, presenting in turn Boolean algebras, relation algebras, Boolean modules, Peirce algebras and, finally, some other applications of the calculus of relations. In each of the first four sections I discuss the algebra in relation to the appropriate calculus and concentrate on the arithmetic required in Chapter 4. For background each section also contains a brief overview of the algebraic theory.

Chapter 4: In this chapter I motivate the proposed algebraic approach. I show how the semantics of a terminological language such as those of Chapter 2 can be accommodated in the algebraic framework presented in Chapter 3. I illustrate by means of the core example of Chapter 1 an algebraic method for generating terminological inferences. To substantiate my claims I present a number of case studies.

The thesis concludes with a List of Figures, an Index of Notation and a Bibliography.

Throughout it is assumed that the reader is familiar with the standard terminology and notation of set theory and first-order logic.

The numbering in each chapter is consecutive. For example, (2.12) refers to the 12th entity in Chapter 2, whether this be a definition, a theorem, a lemma or an example. Figures are numbered separately. Also, the axioms and arithmetical properties of the algebras in Chapter 3 are numbered separately within each section. For example, B7, R7, M7 and P7 refer to the seventh axiom or property of Boolean algebras, relation algebras, Boolean modules and Peirce algebras, respectively.

Citation of references is generally by author and year of publication, as for example in 'Tarski [1941]'. If an author has published more than one work in the same year I annotate the year with a letter (in no particular order), e.g., 'Patel-Schneider [1989a]' and 'Patel-Schneider [1989b]'. Secondary references (i.e. those I did not consult myself but concerning which I found some information in a primary reference) are marked with an asterisk.

A shorter and earlier version of this thesis is due to be published (Brink and Schmidt [1991]).

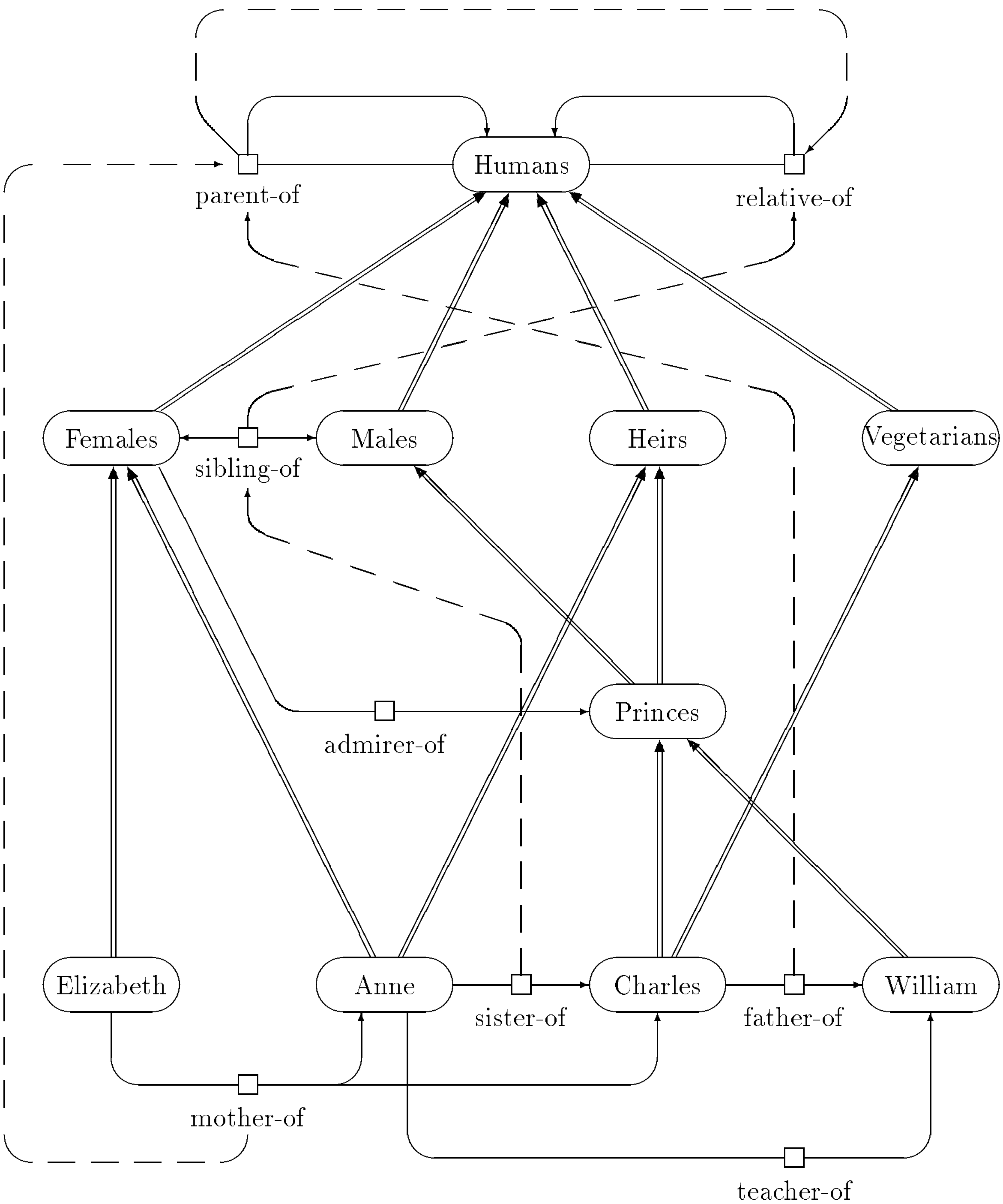
Chapter 1

Preview

In this chapter I introduce that field of knowledge representation which deals with *terminological representation languages* (also called *term subsumption languages* or *terminological logic*). I begin with a sample representation of a small knowledge domain, represented as a semantic network. My later exposition of terminological representation refers back where necessary to this standard example. I give a preview of the standard model-theoretic semantics for typical terminological languages, and I suggest that an algebraic approach may for certain purposes be more appropriate.

Consider the diagram of Figure 1.1, which I will call a *semantic network*. It represents some knowledge about a universe of people. The nodes represent *concepts* like 'Females', 'Princes' or 'Heirs' (= 'Heirs to the throne'). The directed edges marked with squares represent *roles*, like 'mother-of' or 'sister-of'. I will interpret concepts as sets and roles as binary relations. The double-line arrows between concepts indicate a *subsumption* relation, which is a partial order and forms a *concept taxonomy*. Analogously, the broken-line arrows between roles indicate a subsumption ordering under which roles form a poset, called a *role taxonomy*. For example, the diagram shows that 'Princes' is *subsumed by* 'Males', which is read as saying 'All princes are male' and interpreted to mean that the set of all princes is contained in the set of all males. Similarly the role 'mother-of' is subsumed by the role 'parent-of', in other words, 'mother-of' is a subrole of 'parent-of'. Concepts indentified with proper names, like 'Charles', are atoms in the concept taxonomy and are interpreted as singleton sets. The appearance

Figure 1.1: Sample semantic network



of a labelled arrow representing a role indicates a non-empty relation between concepts with the arrow head determining the direction of the relation. For example, the arrow labelled 'admirer-of' and directed from 'Females' to 'Princes' indicates that 'some females are admirers of some princes'. The double-headed arrow of roles like 'sibling-of' and 'relative-of' indicates a symmetric relation.

The semantic network thus represents some explicit facts which can be read off directly. For example:

(1.1) Elizabeth is female.

(1.2) All females are human.

(1.3) All mothers of someone are parents of that person.

(1.4) Charles is a father of William.

(1.5) Some females are siblings of some males.

In addition to such surface knowledge, the semantic network also contains some *implicit* facts, such as:

(1.6) Elizabeth is human.

(1.7) All sisters of someone are relatives of that person.

(1.8) Charles is a father of some prince.

(1.9) William is a child of Charles.

(1.10) Anne is an aunt of some prince.

(1.11) Some vegetarian is a parent of William.

Extracting this implicit information from the representation may seem straightforward to humans. But can this process be formalised? And if so, can it be mechanised? Research in knowledge representation is concerned with such questions. The aim in knowledge representation is to develop 'intelligent' systems for representing knowledge and reasoning about it. Included in such a system, called a *knowledge representation system*, is a *knowledge base*, that stores the explicit knowledge suitably expressed with a *representation scheme* or *language*. To extract the implicit information from the knowledge base, a knowledge representation system also has an *inference mechanism*.

Terminological representation systems are descendants of a knowledge representation system called KL-ONE. In terminological representation systems only *definitional* (or *terminological*) domain knowledge is represented. Definitional knowledge is knowledge that *defines* general notions and relationships, i.e. general interrelationships between concepts and roles. This excludes knowledge that, for example, contains assertions about the existence of individuals, as in

(1.12) There is someone called Charles who is a father of some prince.

This is said to be an *assertional* claim. The semantic network above contains only definitional information. Because the emphasis in many systems has been more on concepts than on roles, and because concepts are also referred to as 'terms', definitional information has become known as 'terminological' information.

The representation scheme of a terminological representation system is called a *terminological (representation) language* and uses a lexical notation rather than the graphic notation of semantic networks. Like any other formal language, a terminological language is defined in terms of its syntactic primitives and the operators on them. The syntactic primitives are concepts and roles; and the operators can be concept-forming or role-forming. Common concept-forming operators on concepts are 'and' (*conjunction*), 'or' (*disjunction*) and 'not' (*negation*). For example, the set of male heirs to the throne could be represented as (and Males Heirs), the set of males and females (that is, the union!) as (or Males Females), and the set of individuals who are not vegetarian as (not Vegetarians). These operators can also be applied to roles. In addition, there are role-forming operators like *inversion* and *composition*, which would respectively represent 'child-of' as (inverse parent-of) and 'aunt-of' as (compose sister-of parent-of). Most terminological languages also have operators which take both concepts and roles as arguments. For example, 'some' is a concept-forming operator on roles as in (some father-of Princes), which is interpreted as the set of fathers of (some) princes. The application to 'father-of' and an atomic concept like 'William' in (some father-of William) then represents the set of fathers of William. (I call the atoms in the concept taxonomy *atomic* concepts.) Subsumption relationships between concepts and also between roles are denoted by ' $\sqsubseteq$ '. Mutual subsumption is called *equivalence*, for which the symbol ' $\doteq$ ' is used.

Consider the explicit facts (1.1)–(1.4) of the sample representation in Figure 1.1. In a terminological language this information can be respectively represented by:

(1.13) $\mathsf{Elizabeth} \sqsubseteq \mathsf{Females}$

(1.14) $\mathsf{Females} \sqsubseteq \mathsf{Humans}$

(1.15) $\mathsf{mother\text{-}of} \sqsubseteq \mathsf{parent\text{-}of}$

(1.16) $\mathsf{Charles} \sqsubseteq (\mathsf{some\ father\text{-}of\ William})$.

Now consider the sentence (1.5). It means, that the intersection of the set of females with a set of siblings of some males is non-empty and can hence be represented by

(1.17) $(\mathsf{and\ Females\ (some\ sibling\text{-}of\ Males)}) \neq \bot$,

where '$\bot$' denotes the empty concept, that is, the *bottom concept* in the concept taxonomy.

Representing explicit knowledge thus essentially amounts to writing down subsumptions, and inference amounts to computing further subsumptions from the explicitly given ones. Hence we would like to be able to compute the formal representations of (1.6)–(1.11), namely:

(1.18) $\mathsf{Elizabeth} \sqsubseteq \mathsf{Humans}$

(1.19) $\mathsf{sister\text{-}of} \sqsubseteq \mathsf{relative\text{-}of}$

(1.20) $\mathsf{Charles} \sqsubseteq (\mathsf{some\ father\text{-}of\ Princes})$

(1.21) $\mathsf{William} \sqsubseteq (\mathsf{some\ (inverse\ parent\text{-}of)\ Charles})$

(1.22) $\mathsf{Anne} \sqsubseteq (\mathsf{some\ (compose\ sister\text{-}of\ parent\text{-}of)\ Princes})$

(1.23) $(\mathsf{and\ Vegetarians\ (some\ parent\text{-}of\ William)}) \neq \bot$.

A common inference mechanism in terminological systems is the *classifier* which is based on an algorithm, called the *subsumption algorithm*, that computes subsumption relationships. The classifier's task is to order (or 'classify') concepts and roles with respect to subsumption, the intention being to insert new concepts or roles in the correct position inside the appropriate taxonomy.

The common approach to specifying the semantics of terminological and KL-ONE-based representation languages is to do so *model-theoretically*, by formally associating concepts and roles respectively with sets and binary relations. In the semantics reasoning with concepts and roles then amounts to reasoning with sets and relations. In the model-theoretic paradigm such reasoning takes place in first-order logic, using the full resources of a first-order language for set theory.

In this thesis I propose another approach: the *algebraic* approach. I will interpret terminological representations of knowledge concerning the concept taxonomy by using equations from *Boolean algebra.* I will interpret terminological representations concerning the role taxonomy by using equations from *relation algebra.* And I will interpret terminological representations of the interactions between roles and concepts by using equations from a suitable algebra such as *Boolean Modules.* In this interpretation of terminological representation drawing inferences amounts to computing further equations from given ones. A natural vehicle of inference, therefore, is *equational logic.*

Chapter 2

Terminological Representation

In this chapter I give an overview of the development of knowledge representation in the KL-ONE system and its descendants, including terminological representation systems. I formally define two representative terminological representation languages. As a final section I append a Note on Sources.

2.1 Background

The main progenitors of terminological representation systems are *semantic networks* and *frames*. There is no universally accepted definition of a semantic network, and various styles exist. (The semantic network presented in the Preview does not adhere to any particular one of these styles, but it has some features common to all semantic network formalisms.) A semantic network is a *graphic* representation of some domain of knowledge. It can be viewed as a directed graph consisting of a collection of *nodes* connected by *links*. These nodes and links form a *taxonomy*, also called a *hierarchy*. Many semantic network formalisms contain only one taxonomy, usually the concept taxonomy. The semantic network of Figure 1.1 contains a concept taxonomy and a role taxonomy. The information implicitly contained in a semantic network representation can be viewed as the information which nodes *inherit* from other nodes (higher-up or lower-down) in the hierarchy.

Semantic networks were first introduced and named by Quillian [1968] as part of his attempt to model what he termed the human 'semantic memory'. His idea was to devise a formal representation scheme encoding the meaning of English words. Other early semantic network formalisms include the 'conceptual dependency' representations of Schank (Schank and Rieger III [1974]) and the 'structural descriptions' of Winston [1975].

In an influential paper Woods [1975] critically scrutinised the shortcomings of these early semantic networks. He pointed out that semantic networks are not well-defined. The descriptions are informal and fail to specify precisely the types of nodes and links that can be used, how the nodes and links can be combined to form a representation and the intended meanings of the nodes and links. In short, the descriptions fail to define unambiguously the *syntax* and *semantics* of a semantic network. Woods discussed the resulting confusions that arise when it is not specified whether the represented information has 'structural' (p. 58) (that is, definitional or terminological) or assertional import.

The failure to define carefully the syntax and semantics of semantic networks and other representation formalisms was also critised by Hayes [1974, 1977] and McDermott [1978]. One formalism that has a well-defined syntax and semantics is first-order logic. Since it has a precise mathematical language and inference structure with a well-defined semantic theory, Hayes [1977] proposed that logic be utilised in representation formalisms. He emphasised however that what is important is not so much the logical syntax but the notion of *meaning* (i.e. semantics) associated with first-order logic.

As a result of these studies, more attention was subsequently given to the 'semantics of semantic networks', and a number of formalisms evolved closely linked to logic. Patel-Schneider [1987a, p. 64] appropriately refers to these systems as 'logic-based semantic networks', which include among others the partitioned networks of Hendrix [1979], the propositional system of Schubert *et al* [1979*] and the SNePS semantic network of Shapiro [1979*] (see also Kumar [1990*]).

Research tended to concentrate on the adequacy of representation schemes, and gave little attention to formalising *inference* in semantic networks. In an effort to formalise inference (which amounts to computing the inherited information) Deliyanni and Kowalski [1979] proposed to extend semantic networks with explicitly specified deduction rules. These 'extended semantic networks' would do inference by resolution, in the logic programming paradigm. Such networks could then be regarded as syntactic variants of the language of first-order logic (p. 184).

For a more comprehensive account of semantic networks the interested reader could refer to Patel-Schneider [1987a, Section 4.1], Rich [1983, Chapter 7] and Nilsson [1980, Chapter 9]. In addition Findler [1979], Sowa [1990*] and *Computers and Mathematics with Applications* [1991*] contain collections of papers on semantic networks.

In another development, opposed to both semantic networks and logic, Minsky [1975] proposed a *frame*-based approach, which would support default specification and exception handling. With this approach knowledge is stored in data structures called *frames*, attached to which are some data manipulation routines, referred to as 'attached procedures'. Frames are intended to describe a prototypical object or situation of the domain of knowledge. Properties (or attributes) of such a prototypical entity are represented in components called *slots*. These slots may contain data values (e.g., numbers, Boolean values or strings), different kinds of pointers to other frames (e.g., 'is a' or 'a kind of' pointers) or attached procedures. The attached procedures enable the user to override the built-in inheritance procedures (i.e. inference procedures) and alter the default values, thus facilitating exception handling. Examples of frame-based representation systems are KRL (Winograd [1975], Bobrow and Winograd [1977] and FRL (Roberts and Goldstein [1977*]).

Less than fifteen years ago, R.J. Brachman and his co-workers at Bolt Beranek and Newman Inc. in Cambridge, Massachusetts started developing the KL-ONE knowledge representation system. KL-ONE is an attempt to combine the useful features of (logic-based) semantic networks and frames. The standard reference to KL-ONE is Brachman and Schmolze [1985]. In a forthcoming paper [1991] Woods and Schmolze give a broader account of knowledge representation in KL-ONE and its descendants including terminological systems.

In his PhD thesis and a series of papers [1977, 1979] Brachman had already addressed the foundational issues raised by Woods [1975], elaborated on the underlying confusions and inadequacies of contemporary semantic network formalisms and formulated his own theory of semantic networks, called 'structured inheritance networks' [1979, p. 34]. A node in a structured inheritance network was referred to as a *concept*. Brachman described a concept as a 'structured' representation of the 'abstraction of the commonalities' [1977, p. 130] from some set of objects in terms of their attributes (or relations) with respect to other concepts. A concept was also described as an 'intensional' [1977, p. 139] entity which is determined by its relationship to other concepts. The idea is that the concept 'Princes' in Figure 1.1, for example, represents more than just a set of objects. It represents a set of humans who are also related (by subsumption) to the concepts 'Males', 'Heirs to the throne', 'Charles' and 'William'. Consider also the concept 'Charles' as defined in the Figure 1.1. It represents the singleton set that is not only related by subsumption to other concepts like 'Princes', but that is also related by roles such as 'sister-of' and 'father-of' to concepts 'Anne' and 'William', respectively. In structured inheritance networks the relationships between concepts are represented by a variety of 'structural' links, also referred to as the 'epistemological primitives' [1977, p. 132]. They determine the *roles* and so-called *structural descriptions* of concepts. The structural description of a concept is intended to define the concepts in terms of roles and other concepts.

KL-ONE is essentially a frame-based system that incorporates the ideas of structured inheritance networks. In KL-ONE concepts and roles are analogous to frames and slots, respectively. Since its basic 'structural' units are concepts, Brachman and Schmolze [1985] describe KL-ONE as a concept-oriented formalism. Unlike frames, KL-ONE concepts allow neither defaults nor user intervention through attached procedures. Brachman [1985] shows why defaults and attached procedures are not compatible with KL-ONE-based knowledge representation. (Note that the earliest versions of KL-ONE did in fact allow attached procedures—see Woods and Schmolze [1991].) Brachman and Schmolze [1985, p. 179] described concepts as (frame-like) structures with three kinds of components that interrelate concepts. These specify

(i) the superconcepts (i.e. the subsuming concepts),
(ii) the roles and
(iii) the structural description of a concept.

In terminological representation schemes this view of concepts has changed. Concepts are not regarded as structures with components but as a syntactic types to which operations (like conjunction or disjunction) can be applied to form composite terms. In this view the three kinds of components of a concept can be represented by three kinds of relations between concepts. These are

(i) the subsumption relations that relate a concept to its superconcept,

(ii) the roles interpreted as binary relations and

(iii) an association with a composite concept description in terms of other concepts and roles,

respectively. In terminological languages the third component becomes an *operation*, called *structural description*, on concepts. I formally define this operation in Section 2.2 as part of the definition of the terminological language $\mathcal{U}$. Woods and Schmolze [1991] also adopt this refined view of concepts.

KL-ONE already has many of the operations for representing complex concept and role descriptions that are also available in terminological languages (e.g., conjunction, disjunction, negation, universal restriction (= 'value restriction') and role restriction). There are then two kinds of concepts in KL-ONE: *primitive* and *defined* (Brachman and Schmolze [1985, Section 2.2], Woods and Schmolze [1991, p. 11–12]). In the semantic diagram of the Preview all the concepts are primitive. In terminological languages primitive concepts are undefined concepts and the defined concepts are the compound ones like 'male heirs to the throne' and 'fathers of princes' which are represented in terms of other (primitive or defined) concepts and roles as (and Males Heirs) and (some father-of Princes), respectively. In other words, defined concepts are constructed with the operators of the representation language using other concepts and roles. In Vilain's [1985, p. 549] words, 'assigning a name to a complex term is tantamount to giving a definition to that name'. (Similarly one can distinguish between primitive and defined roles.)

Although Brachman and Schmolze [1985] has become the standard reference with regards to KL-ONE, in my experience this paper is difficult to read. The exposition does not provide a formal approach, even though such an approach is precisely what Hayes [1974, 1977], McDermott [1976, 1978] and also Israel [1983a] had been advocating. Despite the many elaborate examples presented in Brachman and Schmolze's graphic notation of semantic inheritance networks, many aspects of KL-ONE appear vague and confusing. For example, the meaning of 'the structural description of a concept' is inadequately specified.

According to Woods and Schmolze [1991], experience with KL-ONE revealed some shortcomings. For example, 'although the goals of KL-ONE included a well-defined semantics, a sufficient formal semantics was not provided'. Also 'some of the classifier's operations were not semantically justified' (p. 22). The *1981* KL-ONE *Workshop* (Schmolze and Brachman [1982*]) focussed on ways of improving KL-ONE. One prominent idea was to have separate representation schemes for *terminological* and *assertional* information, thus avoiding the confusion that Woods [1975] already addressed for semantic networks. Terminological information is also referred to as definitional, descriptional, structural or analytic. Assertional information is also referred to as synthetic or factual. The problem with distinguishing between these two kinds of information is that their exact meaning is not immediate. For one, 'there does not appear to be a commonly accepted meaning for "assertion"' (Woods and Schmolze [1991, p. 31]). In addition, 'it is problematic to establish a clear demarkation between analytic [i.e. terminological] statements and factual [i.e. assertional] statements' (Spinelli *et al* [1988, p. 33]). The difference between terminological and assertional information, as I understand it, is best explained with examples. Terminological statements can be viewed as expressing simple interrelationships, such as subsumption relationships, between concept and role descriptions. An example of a terminological statement is

(2.1) Charles is a father of some prince.

It can be expressed (in a suitable terminological language) in terms of subsumption as follows:

(2.2) $\mathsf{Charles} \sqsubseteq (\mathsf{some\ father\text{-}of\ Prince})$.

Assertional statements contain more information. As mentioned in the Preview they make assertions about the world and need to be expressed in a language with quantification. An example of an assertional statement that cannot be expressed in a terminological language is the statement (1.12) (on page 4 of the Preview). I essentially regard terminological information as information that can be expressed in a terminological language and assertional information as information that can only be expressed in a more powerful language such as that of first-order logic.

Brachman and Levesque [1982] argue that a knowledge representation system should be adequate in two respects: terminological and assertional. According to them (p. 190), 'terminological adequacy involves the ability to form the appropriate kind of technical vocabulary and understand the dependencies among the terms' and 'assertional adequacy involves the ability to form the kind of theory appropriate to the world knowledge of a system and understand the implications of the theory'. The formal distinction between terminological knowledge and assertional knowledge is fundamental to work on the systems NIKL (Schmolze [1989b]) and KRYPTON (Brachman *et al* [1983, 1985]. This work started in 1982. The main goal of the NIKL project was to develop an enhanced terminological representation formalism, while the main goal of the KRYPTON project was to incorporate a terminological representation formalism and a separate assertional representation formalism in one unifying system.

NIKL is the new implementation of KL-ONE that was developed in an effort to improve the representation scheme as well as the performance of the classifier of KL-ONE. In [1989b] Schmolze gives a comprehensive formal description of the language and the model-theoretic semantics of this new implementation. The language of NIKL is a terminological language, in which roles are treated on a par with concepts, and the operators are concept- *and* role-forming. Concepts and roles are perceived as being ordered with respect to subsumption in two separate taxonomies. This new 'enlightened' view of concepts and roles (discussed in Kaczmarek *et al* [1986, p. 979]) refines the perspective of KL-ONE where concepts are the principal syntactic types. Since roles are interpreted as binary relations, the 'RoleSet differentiation' operation of KL-ONE (Brachman and Schmolze [1985, p. 185], also known as role differentiation operation), which is used to represent the interrelationship between roles and their subroles, is in NIKL (and subsequent terminological formalisms) viewed as a relation, namely the subsumption relation.

In an early attempt to formalise the syntax and semantics of NIKL Schmolze and Israel [1983] also describe part of its classifier and in particular the subsumption algorithm. To my knowledge this is the earliest formal account of a KL-ONE-based formalism. Schmolze and Lipkis [1983] present a more informal account of NIKL. (It should be noted that the papers Schmolze and Israel [1983] and Schmolze and Lipkis [1983] actually deal with NIKL, the new version of KL-ONE—see Schmolze [1989b, p. 12].) A term frequently used in the literature but hardly ever explained is '(*role*) *fillers*'. A definition is given in Schmolze and Israel [1983, p. 34]: the filler of a role is interpreted as an element in the range of the relation associated with the role. For example in the semantic net of Chapter 1, the role fillers of 'admirer-of' are elements contained in the set of princes, and the role filler of 'father-of' is 'William'. (The term 'range' should not be confused with the 'range' operation available in some terminological languages, including NIKL as described in Schmolze [1989b]. In Section 2.2 I will define role fillers along with the common terminological operators, including 'range'.) Schmolze and Israel's formal treatment revealed that classification in NIKL is sound but not complete. This means that although every subsumption relation determined by the classifier is valid in the semantics, not every valid subsumption relation can be determined by the classifier.

In another attempt to clean up KL-ONE, the 'unifying approach' (Brachman and Levesque [1982]), that combines a terminological representation formalism with an assertional representation formalism, was adopted in a host of so-called *hybrid knowledge representation systems*. These include KRYPTON (Brachman *et al* [1983, 1985]), KL-TWO (Vilain [1985]), KANDOR (Patel-Schneider [1984]), MESON (Edelmann and Owsnicki [1986]) and BACK (Nebel and von Luck [1988]). Nebel and von Luck [1988] define a *hybrid knowledge representation formalism* to consist of two or more different subformalisms for representing different kinds of knowledge or knowledge in different kinds of representation formalisms.

The terminological component of KRYPTON, called the TBox, is similar to NIKL and includes a terminological language and a classifier. The assertional component, called the ABox, includes the language of standard first-order predicate logic and a suitable theorem prover (namely a connection-graph resolution theorem prover; see Stickel [1985*]). Naturally these two components need to interact in some way. In KRYPTON the interaction is accomplished with a mapping that translates TBox expressions into ABox expressions (Brachman *et al* [1983], Patel-Schneider [1987b]). In particular, concepts and roles are translated as unary and binary predicates, respectively. (This is analogous to the formalisation of frames and slots as unary and binary predicates in the semantics proposed by Hayes [1979].) Subsumption relations are translated as universally quantified closed implications. For example, the terminological relations

(2.3) $\mathsf{Princes} \sqsubseteq \mathsf{Males}$

(2.4) $\mathsf{father\text{-}of} \sqsubseteq \mathsf{parent\text{-}of}$

(2.5) $\mathsf{Charles} \sqsubseteq (\mathsf{some\ parent\text{-}of\ Princes})$

would respectively be mapped to the assertional expressions

(2.6) $(\forall x)[\mathsf{Princes}(x) \Rightarrow \mathsf{Males}(x)]$

(2.7) $(\forall x)(\forall y)[\mathsf{father\text{-}of}(x, y) \Rightarrow \mathsf{parent\text{-}of}(x, y)]$

(2.8) $(\forall x)[\mathsf{Charles}(x) \Rightarrow (\exists y)[\mathsf{parent\text{-}of}(x, y) \wedge \mathsf{Princes}(x)]]$.

Although every terminological statement is expressible in the more powerful assertional language, the advantage of having a terminological formalism is that its syntax is free of variables and quantifiers, therefore providing a more natural representation language. In addition it was hoped that since the language of the TBox is less expressive than the first-order language of the ABox, inference in the TBox would be computationally more efficient than first-order inference, which is undecidable. The terminological component can be thought of as a special-purpose formalism. Patel-Schneider [1987b, p. 66] views the terminological formalism as an 'auxiliary logic' of the assertional formalism which is the 'base logic' of the 'hybrid logic' (or hybrid formalism).

The approach to terminological representation in KRYPTON, KL-TWO, KANDOR, MESON and BACK is essentially the same. What varies is the expressiveness of the terminological languages. However, as Nebel and von Luck [1988, Section 3] note, the approaches to assertional representation differ significantly. Like KRYPTON, KL-TWO combines NIKL's language and classifier with a predicate logic theorem prover. The difference is that the terminological language of KL-TWO is more powerful and its assertional language is a propositional language which is a subset of the full first-order language of KRYPTON. The ABoxes of KANDOR and MESON in contrast are based on approaches used in databases. The ABox of BACK (the Berlin Advanced Computational Knowledge representation system) uses a combination of these two approaches (predicate logic and databases).

Besides the ones I mentioned, other hybrid formalisms exist. An example is OMEGA; see Saffiotti and Sebastiani [1989], Attardi and Simi [1986] and Attardi *et al* [1986]. I won't elaborate more on such systems and suggest that the interested reader consult also a recent publication by Nebel [1990b*].

In accordance with Woods [1975], Hayes [1977] and McDermott [1978], increasing attention has been given to *formalisation* in the description of representation formalisms. Terminological (and assertional) representation languages are being formally defined in terms of a fixed syntax with a well-defined model-theoretic semantics. KL-ONE is formally defined in Woods and Schmolze [1991], NIKL in Schmolze [1989b], KANDOR in Patel-Schneider [1984] and BACK in Nebel and von Luck [1988]. By precisely specifying the syntax and its intended meaning in a representation formalism, one is able to determine and analyse the expressive and deductive capabilities of the representation formalism. Is the formalism expressively adequate for a particular field of application? Is it deductively adequate? Is it efficient? These are questions that are important to users and that can be answered by formal investigations. Formal specification also allows one to compare different formalisms with respect to expressiveness and computational criteria. (Baader [1990] provides a definition of expressive power of KL-ONE-based knowledge representation languages that enables one to compare different representation languages formally.) By formally defining the language and classifier of NIKL, Schmolze and Israel [1983] made the unexpected discovery that inference in NIKL is incomplete. Naturally this discovery cast doubt on whether other terminological reasoners are in fact complete and efficient as had been believed.

There is a tradeoff between expressive power and computational tractability. The greater the expressive power of a language for representing knowledge, the harder it becomes to compute the needed inference in reasonable time (Brachman and Levesque [1984]). One of the advantages of using first-order logic as a representation formalism is that its language is very expressive. Unfortunately this expressiveness comes with a price: first-order reasoning is undecidable and hence intractable. Since terminological representation formalisms are expressively weaker than full first-order logic, it was hoped that terminological inference is tractable. This is unfortunately not so. Brachman and Levesque [1984] present a formal analysis of the computational cost in two simple terminological languages, called $\mathcal{FL}$ and $\mathcal{FL}^-$. $\mathcal{FL}$ is a *concept-description language* without negation and disjunction operators. (A concept-description language is a terminological language that can be used to construct complex concept descriptions which are ordered with respect to subsumption. No provision is made for role subsumption.) $\mathcal{FL}$ is a subset of the language in the terminological component of the early version (Brachman *et al* [1983]) of KRYPTON, as well as the more expressive terminological languages of NIKL and KL-ONE. Brachman and Levesque show that subsumption in $\mathcal{FL}$ is co-NP-complete, thus believed to be unsolvable in polynomial time. However, subsumption in $\mathcal{FL}^-$, a variant of $\mathcal{FL}$ that includes all the operators of $\mathcal{FL}$ but one (the *role restriction* operator), has quadratic time complexity and is tractable. (As a consequence the role restriction operator was omitted in the later version (Brachman *et al* [1985]) of KRYPTON.) A small increase in the expressiveness in $\mathcal{FL}^-$ to $\mathcal{FL}$, therefore, results in a dramatic increase of the computational complexity, from tractable to intractable. (The reader interested in the theory of computational complexity and undecidability is advised to refer to an excellent introduction by Harel [1987, Part 3]. The standard reference on intractable and NP-complete problems is the book by Garey and Johnson [1979].)

In an augmented version of [1984], Levesque and Brachman [1987] show that the tradeoff between expressive power and computational tractability is an underlying problem in a number of representation formalisms including first-order logic, databases, semantic networks and KL-ONE-based description formalisms. They argue that a knowledge representation system should be *dependable*, that is, inference should be sound and complete and should normally stop in a reasonable amount of time. Thus (p. 81):

> As responsible computer scientists, we should not be providing a general inferential service if all that we can say about it is that by and large it will probably work satisfactorily.

In view of the tradeoff there are (at least) two ways of developing dependable terminological representation formalisms:

(i) Limit the expressive power of the representation language, by omitting constructs that would lead to non-polynomial response time for correct inference.

(ii) Limit the inference capabilities of the formalism.

Levesque and Brachman are in favour of the former option, which Doyle and Patil [1989, p. 3] refer to as the 'restricted language thesis'. Since the computational complexity of the terminological languages such as $\mathcal{FL}$ and $\mathcal{FL}^-$ is very sensitive to the term-forming operators in their vocabulary, they suggest that further work should focus on establishing and analysing tractable and intractable languages. This view became very influential in shaping subsequent work. (Only recently has it been opposed, by Doyle and Patil [1989], who refer to it as one of two 'dogmas of knowledge representation'.)

As a consequence existing terminological formalisms were analysed for computational efficiency and a number were found to be not only intractable but undecidable. Schild [1988] showed that inference in $\mathcal{U}$, a very expressive terminological language introduced by Patel-Schneider [1987a] that includes most term-forming constructs from KL-ONE and NIKL, is undecidable. By analysing the computational tractability of a subformalism of NIKL and $\mathcal{U}$, Patel-Schneider [1989b] showed that subsumption in NIKL is undecidable. Schmidt-Schauß [1988] analysed an even smaller subset of the formalism investigated by Patel-Schneider [1989b], which turns out to be undecidable as well. Schmidt-Schauß thereby established that subsumption in KL-ONE is undecidable. In [1988] Nebel shows that subsumption in the terminological components of BACK and KANDOR is intractable. In order to guarantee timely responses knowledge representation systems such as KL-ONE, NIKL and BACK therefore have incomplete subsumption algorithms (Patel-Schneider [1989a], Schmidt-Schauß and Smolka [1988a], Nebel [1988]).

The paper by Levesque and Brachman [1987] also initiated the analysis of a family of *attributive concept-description languages* (so called because they aim to describe concepts by specifying restrictions on their attributes, i.e. roles), also known as $\mathcal{AL}$-languages. This analysis casts some light on the precise effect that including different syntactic operators in a language has on the computational cost of subsumption. The first $\mathcal{AL}$-language, called $\mathcal{ALC}$, was introduced and analysed by Schmidt-Schauß and Smolka [1988a, 1988b]. $\mathcal{ALC}$ extends the language $\mathcal{FL}$ with concept descriptions that are formed with the negation and disjunction operators. Subsumption in $\mathcal{ALC}$ is hence at most as efficient as subsumption in $\mathcal{FL}$, that is, it is at least co-NP-complete. Schmidt-Schauß and Smolka devise a 'constraint system' for deciding subsumption between concept descriptions that is based on an algorithm for checking satisfiability and show subsumption in $\mathcal{ALC}$ is in fact PSPACE-complete. Donini *et al* [1990] summarise the computational complexity of checking subsumption in the $\mathcal{AL}$-languages of which some were introduced and investigated earlier in the series of papers Hollunder [1989] and Hollunder *et al* [1990]. According to Donini *et al* [1990], except in two of the weakest languages (called $\mathcal{AL}$ and $\mathcal{ALN}$) checking satisfiability and subsumption in $\mathcal{AL}$-languages is of non-polynomial complexity. Nebel [1990a] shows that with respect to a given set (called a *terminology*) of subsumption and equivalence relations computing subsumption is co-NP-complete even for a minimal terminological representation language that is a subset of every other existing terminological language. Nebel characterises his work by the slogan: *Terminological reasoning is inherently intractable.* This suggests that limiting the expressiveness of terminological languages does not lead to useful and tractable (hence dependable) terminological formalisms.

Rather than choosing the first option for realising Levesque and Brachman's goal of developing a dependable formalism and avoiding the tradeoff between the expressive power and computational tractability, Patel-Schneider [1987a, 1987b, 1989a, 1989b, 1990] investigates the second option. So rather than limiting the number of syntactic constructs in a representation formalism, Patel-Schneider proposes to limit the inference capability. He uses a four-valued semantics instead of the standard two-valued semantics for specifying a terminological language, which he refers to as a *terminological logic.* The idea is that inference in a terminological logic with such a weakened semantics will support fewer subsumption relationships and promises to be more efficient. Patel-Schneider's aim is to find a suitably weak semantics such that inference is complete and tractable. His semantics is based on the relevance logic of Belnap [1975, 1977] and Anderson and Belnap [1975] which has four truth assignments: *true*, *false*, *neither* true nor false and *both* true and false. In [1987a, 1987b, 1990] Patel-Schneider also presents a hybrid formalism, called a 'hybrid logic', that incorporates a four-valued terminological logic and an assertional logic. Unfortunately Patel-Schneider's solution to devising a dependable inference scheme also has its problems. As he observes in [1989a, p. 333], the four-valued semantics is not as intuitive as the two-valued semantics and subsumption in the alternative semantics is limited to finding only the very easy inferences.

So it seems that neither limiting the expressive power nor limiting the inference capabilities of a terminological language are satisfactory options for finding sound, complete and tractable terminological reasoners. Nebel [1990a], Nebel and Smolka [1989], Schmidt-Schauß and Smolka[1988b] and Doyle and Patil [1989] thus support the argument for relaxing the requirements for dependable terminological reasoning by putting some more emphasis on a useful product, and less on being a 'responsible computer scientist'. Nebel and Smolka [1989] observe that despite the worst-case inherent intractability of all terminological reasoning, in practice 'it may well be the case that it is possible to find algorithms that are well-behaved in all "normal cases"'(p. 16). Especially Doyle and Patil [1989] oppose the viewpoint of Levesque and Brachman [1987] for computational tractability and instead argue in favour of expressiveness of language. Experience with representing medical knowledge in NIKL has led them to criticise the 'restricted language thesis' supported by Levesque and Brachman to ensure tractability. Thus (p. 5):

> The terminological facilities of such [restricted] systems are so expressively impoverished that the very purpose set out for general purpose representational utilities is defeated.

Doyle and Patil argue that rather than providing tractable inference knowledge representation systems should provide fully expressive languages, tolerate incomplete inference and provide a 'useful inference service through rational management of inference tools' (p. 7, 44).

This overview of terminological representation is by no means complete. A recent development worth mentioning is the study of the overlap between terminological representation and feature-based unification grammars (see Nebel and Smolka [1989], Schmidt-Schauß and Smolka [1988a], Smolka [1989]). Also notable is Schmolze's [1989a] proposal for generalising terminological representation by allowing n-ary role descriptions. In other developments data base models and techniques are being used in KL-ONE-type knowledge representation (see, e.g., Devanbu *et al* [1989], Borgida *et al* [1989] and Etherington *et al* [1989]).

2.2 Terminological Languages

In this section I define the syntax and the model-theoretic semantics of the languages $\mathcal{ALC}$ of Schmidt-Schauß and Smolka [1988a, 1988b] and $\mathcal{U}$ of Patel-Schneider [1987a]. For continuity I adapt the original definitions of $\mathcal{ALC}$ and $\mathcal{U}$. I base my definitions on the expositions of Schmidt-Schauß and Smolka [1988a, 1988b] and of Nebel and Smolka [1989] in particular. The syntax used for the languages is largely that used by Patel-Schneider [1987a] for $\mathcal{U}$. In order for the notation of $\mathcal{ALC}$ to resemble that of $\mathcal{U}$, I changed much of its original notation. I adopt the convention of starting concept names with a capital letter and role names with a small letter. Like Schild [1988] I prefer to distinguish notationally between a role and its converse. For example, I will denote the role representing the relation 'has as child' by 'has-child' and the role representing the relation 'is a child of' by 'child-of'. This contrasts with the prevelant tendency to use just 'child' for either. (It seems that 'child' is most often assumed to mean 'has as child', see, e.g., Patel-Schneider [1987a, 1989a], Nebel and Smolka [1989], Schmolze [1989b] and Levesque and Brachman [1987]).

As a consequence of the 'restricted language thesis' of Levesque and Brachman [1987], Schmidt-Schauß and Smolka [1988a, 1988b] introduced the language $\mathcal{ALC}$, which is less expressive than languages such as KL-ONE, NIKL and $\mathcal{U}$. $\mathcal{ALC}$ belongs to the class of *attributive concept description languages* or $\mathcal{AL}$*-languages* (Donini *et al* [1990]). It is 'fairly expressive and enjoys pleasant mathematical properties' (Schmidt-Schauß and Smolka [1988b, p. 4]) and also fits in well with the algebraic semantic specification of terminological languages which I will propose in Chapter 4.

The vocabulary of $\mathcal{ALC}$ consists of three disjoint sets of symbols: the alphabet of *primitive concepts*, the alphabet of *primitive roles* and the set of *structural* symbols. (In the literature primitive concepts and roles are also said to be 'atomic' or 'generic'.) There are two designated primitive concept symbols, called *top concept* '$\top$' and *bottom concept* '$\perp$'. The set of structural symbols includes 'and' (*conjunction*), 'or' (*disjunction*), 'not' (*negation*), 'some' (*existential restriction*) and 'all' (*universal restriction*). All these are referred to as *operators*.

In the Preview (on page 4) I gave examples of how these operators (with the exception of all) can be used to describe compound concept expressions such as ‘male heirs to the throne’, ‘males and females’, ‘not vegetarian’ and ‘fathers of some princes’. The operator all represents, for example, the set of ‘individuals who are admirers only of princes’ as the expression (all admirer-of Princes) (provided we assume everyone admires someone). These expressions are examples of *concept descriptions*, which are composed from roles and other concepts. Since the standard example in Chapter 1 has a domain consisting entirely of human beings the designated top and bottom concepts can be used respectively to represent ‘Humans’ (the set of all humans) as $\top$ and the set of no humans, i.e. the empty set, as $\bot$.

More formally, if ‘A’ is any primitive concept symbol then the concept descriptions, denoted by ‘C’ and ‘D’, can be constructed in terms of other concepts according to the following rule (in Backus Naur form):

(2.9) $\mathsf{C}, \mathsf{D} \longrightarrow \mathsf{A} \mid (\mathsf{and\ C\ D}) \mid (\mathsf{or\ C\ D}) \mid (\mathsf{not\ C})$.

Let ‘Q’ be a primitive role symbol. Extending the rule (2.9) the existential and universal restriction constructs are specified by:

(2.10) $\mathsf{C}, \mathsf{D} \longrightarrow \ \ldots \mid (\mathsf{some\ Q\ C}) \mid (\mathsf{all\ Q\ C})$.

Rules (2.9) and (2.10) recursively define every concept description C or D. (In the notation of Schmidt-Schauß and Smolka [1988a, 1988b], Nebel and Smolka [1989], Nebel [1990a], Hollunder [1989], Hollunder *et al* [1990] and Donini *et al* [1990] the concept descriptions in (2.9) and (2.10) are respectively denoted by A, $C \sqcap D$, $C \sqcup D$, $\neg C$, $\exists R \colon C$ (or $\exists R \,.\, C$) and $\forall R \colon C$ (or $\forall R \,.\, C$).)

The model-theoretic semantics of concept descriptions in $\mathcal{ALC}$ is given by an *interpretation* $\mathcal{I}$ which is defined as a pair $(\mathcal{D}^{\mathcal{I}}, \cdot^{\mathcal{I}})$. $\mathcal{D}^{\mathcal{I}}$ is a set thought of as the *domain* (or *universe*) *of interpretation* and $\cdot^{\mathcal{I}}$ is an *interpretation function* which assigns to every concept description C some subset $\mathsf{C}^{\mathcal{I}}$ of $\mathcal{D}^{\mathcal{I}}$ and to every role Q some binary relation $\mathsf{Q}^{\mathcal{I}}$ over the set $\mathcal{D}^{\mathcal{I}}$ (i.e. $\mathsf{Q}^{\mathcal{I}} \subseteq \mathcal{D}^{\mathcal{I}} \times \mathcal{D}^{\mathcal{I}}$). The interpretation function assigns meaning to the designated and complex concepts as specified by the following constraints:

$$
\begin{aligned}
(2.11) \qquad \top^{\mathcal{I}} &= \mathcal{D}^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\mathsf{and\ C\ D})^{\mathcal{I}} &= \mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}} \\
(\mathsf{or\ C\ D})^{\mathcal{I}} &= \mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}} \\
(\mathsf{not\ C})^{\mathcal{I}} &= (\mathsf{C}^{\mathcal{I}})' \qquad (= \mathcal{D}^{\mathcal{I}} - \mathsf{C}^{\mathcal{I}}) \\
(\mathsf{some\ Q\ C})^{\mathcal{I}} &= \left\{ x \mid (\exists y)[(x,y) \in \mathsf{Q}^{\mathcal{I}} \ \&\ y \in \mathsf{C}^{\mathcal{I}}] \right\} \\
(\mathsf{all\ Q\ C})^{\mathcal{I}} &= \left\{ x \mid (\forall y)[(x,y) \in \mathsf{Q}^{\mathcal{I}} \Rightarrow y \in \mathsf{C}^{\mathcal{I}}] \right\}.
\end{aligned}
$$

Besides containing the operator symbols the set of structural symbols contains two more: the *specialisation* '$\sqsubseteq$' and *equivalence* '$\doteq$' symbols. These are used to represent relationships between concept descriptions. In (1.13)–(1.16) of Chapter 1 I gave examples of specialisation relationships expressing the explicit information in the semantic network of Figure 1.1 given in (1.1)–(1.4). (Note: in Chapter 1 I referred to these as subsumption relationships, but strictly speaking subsumption (which I define in (2.14) below) is a semantic notion). With equivalence we can for example define 'females' as 'humans who are not male' by specifying that $\mathsf{Females} \doteq (\mathsf{and\ Humans\ (not\ Males)})$. Specialisation and equivalence relations which represent the explicitly given information in a knowledge base are called *terminological axioms*, because they are used as axioms when computing the implicitly represented information. Let 'σ' and 'τ' denote terminological axioms. Their syntax is formally defined by:

(2.12) $\sigma, \tau \longrightarrow \mathsf{C} \sqsubseteq \mathsf{D} \mid \mathsf{C} \doteq \mathsf{D}$.

A set of terminological axioms is referred to as a *terminology* and is denoted by T. A terminology can be viewed as a representation of a knowledge base.

An interpretation $\mathcal{I}$ of $\mathcal{ALC}$ is said to *satisfy* (or *model*) a terminological axiom σ, written $\models_{\mathcal{I}} \sigma$, iff the interpretations of the concepts are related to each other in certain ways. Namely:

$$
\begin{aligned}
(2.13) \quad &\models_{\mathcal{I}} \mathsf{C} \sqsubseteq \mathsf{D} \ \text{ iff } \ \mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}} \\
&\models_{\mathcal{I}} \mathsf{C} \doteq \mathsf{D} \ \text{ iff } \ \mathsf{C}^{\mathcal{I}} = \mathsf{D}^{\mathcal{I}}.
\end{aligned}
$$

More generally, an interpretation $\mathcal{I}$ is a model for the terminology T, written $\models_{\mathcal{I}} T$, iff *every* terminological axiom in T is satisfied by $\mathcal{I}$. A terminological axiom σ is *entailed by* (or the *consequence of*) a terminology T, written $T \models \sigma$, iff σ is satisfied by *every* model of T. In the case where T is empty, we write $\models \sigma$ and the axiom σ is said to be *valid.*

Now, define *subsumption* and *equivalence with respect to a terminology* T as follows:

(2.14) $\mathsf{C} \preceq_T \mathsf{D}$ iff $T \models \mathsf{C} \sqsubseteq \mathsf{D}$

$\mathsf{C} \approx_T \mathsf{D}$ iff $T \models \mathsf{C} \doteq \mathsf{D}$.

If $\mathsf{C} \preceq_T \mathsf{D}$ then the concept C is said to be *subsumed by* the concept D *in the terminology* T, or equivalently, D is said to *subsume* C *in* T. The descriptions C and D are said to be *semantically equivalent in the terminology* T if $\mathsf{C} \approx_T \mathsf{D}$. In case the terminology is empty, we may write $\mathsf{C} \preceq \mathsf{D}$ and $\mathsf{C} \approx \mathsf{D}$ instead of $\mathsf{C} \preceq_\emptyset \mathsf{D}$ and $\mathsf{C} \approx_\emptyset \mathsf{D}$, respectively. Note the following:

(2.15) $\mathsf{C} \preceq \mathsf{D}$ iff $\mathsf{C} \preceq_T \mathsf{D}$ for every terminology T

$\mathsf{C} \approx \mathsf{D}$ iff $\mathsf{C} \approx_T \mathsf{D}$ for every terminology T.

A concept description C is called *inconsistent* (or *incoherent*) in a terminology T iff $\mathsf{C} \approx_T \bot$, and *consistent* (or *coherent*) otherwise.

The subsumption relation $\preceq_T$ is a preorder, that is, it is a reflexive and transitive relation.

(2.16) **Lemma** For any terminology T, $\preceq_T$ is a preorder.

Proof. To show that $\preceq_T$ is reflexive, note that by (2.14) $\mathsf{C} \preceq_T \mathsf{C}$ is equivalent to $T \models \mathsf{C} \sqsubseteq \mathsf{C}$, which in turn is equivalent to saying that $\models_{\mathcal{I}} \mathsf{C} \sqsubseteq \mathsf{C}$ for every model $\mathcal{I}$ of T. By (2.13) this is equivalent to $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{C}^{\mathcal{I}}$ for every model $\mathcal{I}$ of T, which is true, since $\subseteq$ is reflexive.

Next, suppose $\mathsf{C} \preceq_T \mathsf{B}$ and $\mathsf{B} \preceq_T \mathsf{D}$. Then $T \models \mathsf{C} \sqsubseteq \mathsf{B}$ and $T \models \mathsf{B} \sqsubseteq \mathsf{D}$. Let $\mathcal{I}$ be any interpretation of T. Hence, $\models_{\mathcal{I}} \mathsf{C} \sqsubseteq \mathsf{B}$ and $\models_{\mathcal{I}} \mathsf{B} \sqsubseteq \mathsf{D}$, and using (2.13) this becomes $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{B}^{\mathcal{I}}$ and $\mathsf{B}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$. Since $\subseteq$ is transitive it follows that $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$, hence $\models_{\mathcal{I}} \mathsf{C} \sqsubseteq \mathsf{D}$. Therefore, since $\mathcal{I}$ was arbitrary, $T \models \mathsf{C} \sqsubseteq \mathsf{D}$ or equivalently $\mathsf{C} \preceq_T \mathsf{D}$. □

When two expressions subsume each other ($\mathsf{C} \preceq_T \mathsf{D}$ and $\mathsf{D} \preceq_T \mathsf{C}$), they can be shown to be equivalent ($\mathsf{C} \approx_T \mathsf{D}$). (The proof is similar to the one above.) Since the terms C

and D need not be identical, $\preceq_T$ is not an antisymmetric relation. Hence $\preceq_T$ is not a partial order. However, quotienting $\preceq_T$ with respect to the equivalence relation $\approx_T$ yields the partial order $(\preceq_T / \approx_T)$, associated with which is the poset of equivalence classes of the concepts, namely $(\mathbf{C}/\approx_T, \preceq_T/\approx_T)$, where $\mathbf{C}$ denotes the set of concept descriptions. In this poset the equivalent concepts are not distinguishable since they are associated to one particular equivalence class. Nebel and Smolka [1989] refer to this poset as the *concept taxonomy* in the terminology T. This completes the definition of the terminological language $\mathcal{ALC}$.

Since $\mathcal{ALC}$ is a concept-description language its treatment of concepts and roles is rather uneven. Concepts and roles can be combined to form new concepts, but not to form new roles. While the concepts are related to each other by the subsumption ordering, roles are not related in any way. Hollunder [1989], Hollunder *et al* [1990] and Donini *et al* [1990] introduce role conjunction and role subsumption to $\mathcal{ALC}$, obtaining the more expressive $\mathcal{AL}$-languages called $\mathcal{ALCR}$ and $\mathcal{ALCNR}$. Various other terminological languages including KL-ONE (Woods and Schmolze [1991]), NIKL (Schmolze [1989b]), and the terminological components of KRYPTON (Brachman *et al* [1983, 1985]), KANDOR (Patel-Schneider [1984]) and BACK (Nebel and von Luck [1988]) are equipped with both concept- and role-forming operators, as well as concept and role subsumption. Combining the different syntactic operators of these languages, Patel-Schneider [1987a] introduced a very expressive terminological language, called $\mathcal{U}$, which has the terminological languages of many systems as sublanguages. Since I aim to analyse the semantics of terminological languages in an algebraic framework, I am particularly interested in the different kinds of operators used in various languages and find $\mathcal{U}$ suitable for analysis.

As for $\mathcal{ALC}$, the vocabulary of $\mathcal{U}$ consists of the alphabet of primitive concepts and the alphabet of primitive roles as well as the set of structural symbols, which contains the operators of $\mathcal{U}$ and the symbols '$\sqsubseteq$' and '$\doteq$'. Like $\mathcal{ALC}$, $\mathcal{U}$ has two designated primitive concepts: the *top concept* '$\top$' and the *bottom concept* '$\bot$'. It has one designated primitive role 'self', called the *identity role*. As before 'A' denotes any primitive concept, 'C', 'D' any concept descriptions and 'Q' any primitive role. In addition, 'R' and 'S' denote any role descriptions. Concept description can be formed according to the following rule which specifies *conjunctions*, *disjunctions* and *negations*:

(2.17) $\mathsf{C}, \mathsf{D} \longrightarrow \mathsf{A} \mid (\mathsf{and\ C\ D}) \mid (\mathsf{or\ C\ D}) \mid (\mathsf{not\ C})$.

Other concept-forming operators are used to express *existential* and *universal restrictions* as specified by the extension:

(2.18) $\mathsf{C}, \mathsf{D} \longrightarrow \ldots \mid (\mathsf{some\ R\ C}) \mid (\mathsf{all\ R\ C})$.

Number restrictions are defined by:

(2.19) $\mathsf{C}, \mathsf{D} \longrightarrow \ldots \mid (\mathsf{atleast}\ n\ \mathsf{R}) \mid (\mathsf{atmost}\ n\ \mathsf{R})$,

where n is a non-negative integer. *Role value maps* and *structural descriptions* are respectively defined by the rule:

(2.20) $\mathsf{C}, \mathsf{D} \longrightarrow \ldots \mid (\mathsf{rvm\ R\ S}) \mid (\mathsf{sd\ C\ Rb}_1 \ldots \mathsf{Rb}_k)$,

where k is a positive integer. The 'Rb_i' denote the so-called *role bindings* and have one of two forms. Namely:

(2.21) $\mathsf{Rb}_i \longrightarrow (\subseteq \mathsf{R\ S}) \mid (\supseteq \mathsf{R\ S})$.

Rules (2.17)–(2.21) thus recursively define the set of concept descriptions.

The set of role descriptions is defined by rules (2.22)–(2.24) below. *Role conjunctions*, *disjunctions* and *negations* are defined by:

(2.22) $\mathsf{R}, \mathsf{S} \longrightarrow \mathsf{Q} \mid (\mathsf{and\ R\ S}) \mid (\mathsf{or\ R\ S}) \mid (\mathsf{not\ R})$.

Role inversions, *role compositions* and *transitive closures* are respectively defined by the extension:

(2.23) $\mathsf{R}, \mathsf{S} \longrightarrow \ldots \mid (\mathsf{inverse\ R}) \mid (\mathsf{compose\ R\ S}) \mid (\mathsf{trans\ R})$.

And *role restrictions* are defined according to:

(2.24) $\mathsf{R}, \mathsf{S} \longrightarrow \ldots \mid (\mathsf{restrict\ R\ C})$.

$\mathcal{U}$ can be viewed as an extension of $\mathcal{ALC}$. It has the same designated concepts $\top$ and $\bot$ as $\mathcal{ALC}$, and its operators and, or, not, some and all are defined as in $\mathcal{ALC}$, except that the some and all operators can be applied to any role description R, not just to primitive roles Q (compare (2.10) and (2.18)). As for $\mathcal{ALC}$ an interpretation $\mathcal{I}$ of $\mathcal{U}$ is defined in terms of a domain $\mathcal{D}^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$. Every concept description C is mapped by $\cdot^{\mathcal{I}}$ to a subset $\mathsf{C}^{\mathcal{I}}$ of $\mathcal{D}^{\mathcal{I}}$, and every role R is mapped to a binary relation $\mathsf{R}^{\mathcal{I}}$ over the set $\mathcal{D}^{\mathcal{I}}$. The semantics of the designated concepts and the concept-forming operators that $\mathcal{U}$ has in common with $\mathcal{ALC}$ is defined by (2.11) above, with the primitive role symbol 'Q' in the definition of some and all replaced by the more general role symbol 'R'. The semantics of the other concept-forming operators in $\mathcal{U}$ are given by the following conditions:

$$
\begin{array}{lrcl}
(2.25) & (\mathsf{atleast}\ n\ \mathsf{R})^{\mathcal{I}} & = & \left\{x \mid \mathrm{card}(\{y \mid (x,y) \in \mathsf{R}^{\mathcal{I}}\}) \geq n\right\} \\
& (\mathsf{atmost}\ n\ \mathsf{R})^{\mathcal{I}} & = & \left\{x \mid \mathrm{card}(\{y \mid (x,y) \in \mathsf{R}^{\mathcal{I}}\}) \leq n\right\} \\
& (\mathsf{rvm}\ \mathsf{R}\ \mathsf{S})^{\mathcal{I}} & = & \left\{x \mid (\forall y)[(x,y) \in \mathsf{R}^{\mathcal{I}} \Rightarrow (x,y) \in \mathsf{S}^{\mathcal{I}}]\right\} \\
& (\mathsf{sd}\ \mathsf{C}\ \mathsf{Rb}_1 \ldots \mathsf{Rb}_k)^{\mathcal{I}} & = & \left\{x \mid (\exists y)[(x,y) \in \bigcap_{i=1}^{k} \mathsf{Rb}_i{}^{\mathcal{I}} \ \&\ y \in \mathsf{C}^{\mathcal{I}}]\right\}.
\end{array}
$$

(For any set A, $\mathrm{card}(A)$ returns the cardinality of A.) Each role bindings construct Rb_i can have one of two forms and their respective interpretations are given by:

$$
\begin{array}{lrcl}
(2.26) & (\subseteq \mathsf{R}\ \mathsf{S})^{\mathcal{I}} & = & \left\{(x,y) \mid (\forall z)[(x,z) \in \mathsf{R}^{\mathcal{I}} \Rightarrow (y,z) \in \mathsf{S}^{\mathcal{I}}]\right\} \\
& (\supseteq \mathsf{R}\ \mathsf{S})^{\mathcal{I}} & = & \left\{(x,y) \mid (\forall z)[(y,z) \in \mathsf{S}^{\mathcal{I}} \Rightarrow (x,z) \in \mathsf{R}^{\mathcal{I}}]\right\}.
\end{array}
$$

In addition the semantics of the designated role and the role-forming operators are given by:

$$
\begin{array}{lrcl}
(2.27) & \mathsf{self}^{\mathcal{I}} & = & \left\{(x,x) \mid x \in \mathcal{D}^{\mathcal{I}}\right\} \\
& (\mathsf{and}\ \mathsf{R}\ \mathsf{S})^{\mathcal{I}} & = & \mathsf{R}^{\mathcal{I}} \cap \mathsf{S}^{\mathcal{I}} \\
& (\mathsf{or}\ \mathsf{R}\ \mathsf{S})^{\mathcal{I}} & = & \mathsf{R}^{\mathcal{I}} \cup \mathsf{S}^{\mathcal{I}} \\
& (\mathsf{not}\ \mathsf{R})^{\mathcal{I}} & = & (\mathsf{R}^{\mathcal{I}})' \qquad (= (\mathcal{D}^{\mathcal{I}} \times \mathcal{D}^{\mathcal{I}}) - \mathsf{R}^{\mathcal{I}}) \\
& (\mathsf{inverse}\ \mathsf{R})^{\mathcal{I}} & = & \left\{(x,y) \mid (y,x) \in \mathsf{R}^{\mathcal{I}}\right\} \\
& (\mathsf{compose}\ \mathsf{R}\ \mathsf{S})^{\mathcal{I}} & = & \left\{(x,y) \mid (\exists z)[(x,z) \in \mathsf{R}^{\mathcal{I}} \ \&\ (z,y) \in \mathsf{S}^{\mathcal{I}}]\right\} \\
& (\mathsf{trans}\ \mathsf{R})^{\mathcal{I}} & = & \mathsf{R}^{\mathcal{I}} \ \cup \ \bigcup\limits_{k \geq 1} \left\{(x,y) \mid (\exists z_1) \ldots (\exists z_k)[(x,z_1) \in \mathsf{R}^{\mathcal{I}} \right. \\
& & & \left. \ \&\ \forall (1 \leq i < k)[(z_i, z_{i+1}) \in \mathsf{R}^{\mathcal{I}}] \ \&\ (z_k, y) \in \mathsf{R}^{\mathcal{I}}]\right\} \\
& (\mathsf{restrict}\ \mathsf{R}\ \mathsf{C})^{\mathcal{I}} & = & \left\{(x,y) \mid [(x,y) \in \mathsf{R}^{\mathcal{I}} \ \&\ y \in \mathsf{C}^{\mathcal{I}}]\right\}.
\end{array}
$$

Thus the interpretations of the role forming operators are the usual set-theoretic operations: identity, intersection, union, complementation, converse, composition, transitive closure and range restriction. The set-theoretic definition of trans is rather unwieldy. However it can also be characterised by a recursive definition which I give in (4.6) of Section 4.1.

A *role filler* of some R is interpreted to be an element y in the domain $\mathcal{D}^{\mathcal{I}}$, such that there is an element $x \in \mathcal{D}^{\mathcal{I}}$ and $(x, y) \in \mathsf{R}^{\mathcal{I}}$. In other words, the set of role fillers is identical to the range of the relation $\mathsf{R}^{\mathcal{I}}$.

Examples of concept descriptions represented in terms of the and, or, not and some can be found in the Preview (on page 4). Above (on page 23) I gave an example of a concept description formed with the all operator. In the Preview I also represented the relation 'child-of' (or 'has-parent') as (inverse parent-of) and the relation 'aunt-of' in terms of the relations 'sister-of' and 'parent-of' as (compose sister-of parent-of). Here is a list of examples illustrating the other operators to concepts and roles from the standard example in Figure 1.1:

(2.28) (atleast 2 parent-of) represents the set of parents of at least two humans.

(2.29) (atmost 1 parent-of) represents the set of parents of at most one human.

(2.30) (rvm aunt-of relative-of) represents the set of humans who are relatives of all those of whom they are an aunt. (This should coincide with the set of all humans.)

(2.31) (and parent-of teacher-of) represents the relation of simultaneously being a parent and a teacher.

(2.32) (or sister-of admirer-of) represents the relation of being either a sister or being an admirer.

(2.33) (not father-of) represents the relation of not being a father.

(2.34) (trans parent-of) represents the relation 'is a parent or ancestor of' as the transitive closure of the 'parent-of' relation.

(2.35) (restrict sibling-of Males) represents the relation of being a sibling of males, in other words it represents the relation 'has as brother'.

The most complex and least obvious construct is the structural description, sd. The best descriptions I could find are those of Patel-Schneider [1987a] and Schild [1988]. Patel-Schneider [1987a, p. 88] describes structural description as 'a way of inter-relating role fillers by means of roles of some other object'. As an example he represents the concept 'project-broadcast message' or 'a message for which some project exists such that each sender of the message is a project-member of the project, and each project-member of the project is a recipient of the message' as

(and message (sd project ($\subseteq$ sender project-member) ($\supseteq$ recipient project-member))).

In his example Schild [1988, p. 3] defines the concept 'faithful husband' by

(and man (sd woman ($\subseteq$ has-child has-child))),

since 'a faithful husband should be a man for which there exists a woman who is the mother of all the man's children'. But even these examples seem unclear. So, I tried to come up with more intuitive examples that fit in with the sample knowledge of the semantic network in Figure 1.1. Consider the following role binding expression:

(2.36) ($\subseteq$ child-of child-of).

According to the definition in (2.26) it represents a relation by which an individual x is related to an individual y iff all individuals z who have x as a child also have y as a child. I.e., x is related to y iff all individuals z who are parents of x are also parents of y. In other words, the set of all parents of x coincides with the set of all parents of y. That is, x and y are siblings (half-brothers and -sisters excluded), provided we take anybody to be a sibling of him/herself. The expression

(2.37) (sd Males ($\subseteq$ child-of child-of))

thus represents (according to its definition in (2.25)) the set of individuals x who have a male sibling. That is, (2.37) represents the set of all those people who are either male (and hence qualify as their own male sibling) or have a brother. If one were opposed to this definition of 'sibling-of' and not want individuals to be their own siblings, one would have to provide for this and amend (2.37) as follows:

(2.38) (sd Males ($\subseteq$ self (not self)) ($\subseteq$ child-of child-of)).

In this representation no male is related to himself, since the role binding construct ($\subseteq$ self (not self)) is equivalent to (not self). (I will prove this in (4.27) of Section 4.3.) Observe that the syntactic definition of sd in (2.20) and (2.21) forces us to encode the role (not self) as a more complex and less intuitive expression. These examples illustrate that, first, it is not easy to find adequate English formulations for even small constructs such as (2.36) and (2.37). And second, as (2.38) illustrates it is also not easy to find adequate terminological representations for information formulated in English.

To complete the definition of $\mathcal{U}$ we need to extend the definition of subsumption and equivalence for concepts in $\mathcal{ALC}$ to subsumption and equivalence for roles. The rule

(2.39) $\sigma, \tau \longrightarrow \ \ldots \ | \ \mathsf{R} \sqsubseteq \mathsf{S} \ | \ \mathsf{R} \doteq \mathsf{S}$

extends the definition (2.12) of terminological axioms with *role specialisations* and *role equivalences*. Their semantics is as for the terminological axioms (2.13) that express relationships between concepts: specialisation is interpreted by the subset relation and equivalence by equality. I will not explicitly define the notions of *models* of terminological axioms, *entailment by a terminology*, *valid* terminological axioms, *role subsumption* and *role equivalence in a terminology*, *inconsistent* and *consistent* roles for $\mathcal{U}$. These are straightforward generalisations of the corresponding notions for $\mathcal{ALC}$. Just as the concept taxonomy is a poset of concept equivalence classes, the *role taxonomy* is defined to be the poset $(\mathbf{R}/\approx_{\mathcal{T}}, \preceq_{\mathcal{T}}/\approx_{\mathcal{T}})$ of role equivalence classes ordered with respect to the quotient of role subsumption, where $\mathbf{R}$ denotes the set of all role descriptions in the language.

I have slightly adapted the vocabulary of the language $\mathcal{U}$ as defined by Patel-Schneider [1987a] and Schild [1988]. They denote the 'inverse', 'compose' and 'restrict' operators by 'inv', 'comp' and 'vr', respectively. Rather than defining conjunction, disjunction and role composition as n-ary operators, without loss of generality they are here defined as binary operators. Instead of using Patel-Schneider's version of the existential rectriction operator some, I use the more general definition given by Schmidt-Schauß and Smolka [1988a, 1988b] and include the designated concepts $\top$ and $\bot$ in the vocabulary of $\mathcal{U}$. The some operator of Patel-Schneider is applied to a role R yielding the concept description (some R) which has the following interpretation:

(2.40) $(\mathsf{some}\ \mathsf{R})^{\mathcal{I}} \ = \ \left\{x \mid (\exists y)[(x,y) \in \mathsf{R}^{\mathcal{I}}]\right\}.$

It is easy to show that the two versions of some can be defined in terms of each other as follows:

(2.41) $(\mathsf{some}\ \mathsf{R}\ \mathsf{C}) \ \doteq \ (\mathsf{some}\ (\mathsf{restrict}\ \mathsf{R}\ \mathsf{C}))$

$(\mathsf{some}\ \mathsf{R}) \ \doteq \ (\mathsf{some}\ \mathsf{R}\ \top).$

Also, in $\mathcal{U}$ as defined by Patel-Schneider the concepts $\top$ and $\bot$ can be thought as abbreviating (some self) and (not $\top$), respectively. Hence my version of $\mathcal{U}$ is essentially equivalent to that of Patel-Schneider.

Suppose we use the symbol '∇' to abbreviate the expression (or self (not self)) and

'Λ' to abbreviate the expression ($\mathsf{not}\ \nabla$), then the following holds:

$$(2.42)\quad \begin{aligned} \nabla^{\mathcal{I}} &= \mathcal{D}^{\mathcal{I}} \times \mathcal{D}^{\mathcal{I}} \\ \Lambda^{\mathcal{I}} &= \emptyset. \end{aligned}$$

Hence ∇ and Λ can be regarded as the *top role* and the *bottom role*, respectively. The language of NIKL (Schmolze [1989b]) and the language $\mathcal{KL}$ (defined in Woods and Schmolze [1991]) have two interesting role-forming operators '**domain**' and '**range**'. The construct ($\mathsf{domain}\ \mathsf{C}$) represents the largest relation with the set represented by C as domain, while ($\mathsf{range}\ \mathsf{C}$) represents the largest relation with the set represented by C as range. Formally, their semantics is given by:

$$(2.43)\quad \begin{aligned} (\mathsf{domain}\ \mathsf{C})^{\mathcal{I}} &= \left\{(x,y) \mid x \in \mathsf{C}^{\mathcal{I}}\right\} \\ (\mathsf{range}\ \mathsf{C})^{\mathcal{I}} &= \left\{(x,y) \mid y \in \mathsf{C}^{\mathcal{I}}\right\}. \end{aligned}$$

These constructs can be defined in $\mathcal{U}$ by:

$$(2.44)\quad \begin{aligned} (\mathsf{domain}\ \mathsf{C}) &\doteq (\mathsf{inverse}\ (\mathsf{restrict}\ \nabla\ \mathsf{C})) \\ (\mathsf{range}\ \mathsf{C}) &\doteq (\mathsf{restrict}\ \nabla\ \mathsf{C}), \end{aligned}$$

which can be shown to be valid.

Although one can express most of the syntactic operators used in KL-ONE-type representation languages in $\mathcal{U}$, there are operators which, it seems, one cannot. One such operator is '**fillers**' (not to be confused with role fillers). It yields the concept description ($\mathsf{fillers}\ \mathsf{R}\ \mathsf{C}_1 \ldots \mathsf{C}_k$), for k any positive integer, which Patel-Schneider [1989a, p. 323] interprets to contain all those elements related by the role R to an element of each concept C_i, such that these elements in the C_i are all distinct. Formally this translates to:

$$(2.45)\quad (\mathsf{fillers}\ \mathsf{R}\ \mathsf{C}_1 \ldots \mathsf{C}_k)^{\mathcal{I}} = \{x \mid (\exists y_1) \ldots (\exists y_k)(\forall i)[(\forall j \neq i)[y_j \neq y_i]\ \&\ (x, y_i) \in \mathsf{R}^{\mathcal{I}}\ \&\ y_i \in \mathsf{C}_i^{\mathcal{I}}]\right\}.$$

This construct can be regarded as a generalised version of the operator **atleast**, since

$$(2.46)\quad (\mathsf{atleast}\ n\ \mathsf{R}) \approx (\mathsf{fillers}\ \mathsf{R}\ \underbrace{\top \ldots \top}_{n \text{ copies}}).$$

2.3 A Note on Sources

Standard textbooks on Artificial Intelligence like Charniak and McDermott [1985], Rich [1983] and Nilsson [1980] contain general introductions and overviews to knowledge representation. Knowledge representation is also surveyed in the papers of Delgrande and Mylopoulos [1986], Levesque [1986] and Mylopoulos and Levesque [1983]. A variety of research material deals with the 'philosophical' foundations of knowledge representation. For example, in [1983a] Israel discusses the semantics of semantic networks, and in [1983b] and [1985] he debates the role of (classical and nonmonotonic) logic in knowledge representation and the contrasting viewpoint of Minsky. Israel and Brachman [1981, 1984] discuss the merits of logic and some of the confusions in semantic networks. Brachman [1983] deals with the interpretations of links, specifically the 'is a' link in semantic networks and addresses the confusions that arise from the uniform treatment of links (e.g., the uniform treatment of subsumption and role links). In [1983, 1986] Woods identifies and discusses general issues that in his view are fundamental in knowledge representation.

Most of the work conducted in Artificial Intelligence and knowledge representation is published in journals, conference proceedings and technical reports. A rich source of references is the Springer-Verlag series *Lecture Notes in Computer Science* and its subseries *Lecture Notes in Artificial Intelligence.* Some of the most interesting papers in knowledge representation have appeared in special collections published as books. Brachman and Levesque [1985] is such a collection of the early important papers, and Findler [1979] contains a collection of the early papers on semantic networks. A collection on the 'principles of semantic networks' will appear in Sowa [1990*].

The most important source of material on knowledge representation are the major AI journals, which include *Artificial Intelligence* and *The AI Magazine.* In addition *Cognitive Science* and *Computational Intelligence* regularly publish contributions to research in knowledge representation. Papers also sporadically appear in the general Computer Science literature, e.g., *Proceedings of the IEEE* and the various ACM publications (e.g., the *SIGART Newsletter*). Special issues on knowledge representation and semantic networks include *IEEE Computer* [1983], *Proceedings of the IEEE* [1986] and

Computers and Mathematics with Applications [1991*].

Artificial Intelligence conferences are organised on a regular basis and usually have special sessions on knowledge representation. The major AI conferences, which are usually held annually or biannually, are the *International Joint Conference on Artificial Intelligence* (IJCAI), the *National Conference on Artificial Intelligence* (AAAI), the *European Conference on Artificial Intelligence* (ECAI) and the *German Workshop on Artificial Intelligence* (GWAI). Other conferences include the *Conference on Artificial Intelligence Applications* (organised by the IEEE Computer Society), the *International Symposium of Artificial Intelligence* and the *Conference of the Society for the Study of Artificial Intelligence and Simulation of Behaviour* (AISB). From time to time specialist conferences and workshops are held. Relevant ones include the *1981* KL-ONE *Workshop* (Schmolze and Brachman [1982*]), the *Knowledge Representation Workshop* [1983*], the *IEEE Workshop on Principles of Knowledge-Based Systems* [1984*] and the NIKL *Workshop* (Moore [1986*]). More recent conferences include the *Workshop on Inheritance Hierarchies in Knowledge Representation and Programming Languages* [1989*], the *First International Conference on Principles of Knowledge Representation and Reasoning* (Brachman *et al* [1989*]) and the *Workshop on Term Subsumption Languages in Knowledge Representation* (Patel-Schneider *et al* [1989]).

Almost all material initially appear as Technical Reports at the authors' home institution. These can be very difficult to get hold of, especially to workers in remote parts of the world. Fortunately (and on a personal note) AI workers are very approachable, and email works wonders.

Chapter 3

Algebras of Sets and Relations

The purpose of this chapter is to introduce algebraic structures which formalise reasoning with sets and relations. Since reasoning with concepts and roles is modelled as reasoning with sets and relations, these algebras form the basis for the *equational* approach to terminological inference which I propose in Chapter 4. The structures I consider are *Boolean algebras*, *relation algebras*, *Boolean modules* and a new class of algebras, called *Peirce algebras*. These algebras can be studied in the context of Universal Algebra. Standard expositions of one-sorted or homogeneous algebras can be found in, e.g., Burris and Sankappanavar [1981], Grätzer [1968, 1979] and Cohn [1981]. An exposition of many-sorted or heterogeneous algebras appears in Birkhoff and Lipson [1970].

For later reference I define some general algebraic notions.

(3.1) **Definition** A (*homogeneous*) *algebra* $\mathcal{A}$ is an ordered pair (A, F) with A any non-empty set and F a set of finitary operations on A. An *n-ary* (or *finitary*) operation on A is any function f from A^n to A. (In this chapter no operation has arity greater than two.) When F is finite, say $F = \{f_0, \ldots, f_{n-1}\}$, the algebra (A, F) is also denoted by $(A, f_0, \ldots, f_{n-1})$. A is called the *base set* of the algebra and is assumed to be closed under each operation in F. (In general, F is not necessarily finite and may be empty.) The operations in F are called the *fundamental operations* of the algebra. The set F of operations and their arities determine the *type* (or *similarity*) of an algebra.

I focus on algebras which are *equationally definable*, that is, those which are completely defined by equations (or *identities*). A class K of (similar) algebras is called an *equationally definable class* (or *equational class*), if there is a set of equations Σ such that K is the class of all algebras in which each equation in Σ is satisfied. Let Σ denote a set of equations and let e denote an equation. We write $\Sigma \models e$ if, given any algebra $\mathcal{A}$ in which each equation of Σ is satisfied, e is also satisfied in $\mathcal{A}$. The set of equations Σ satisfied in every algebra of a class K is called the *equational theory* of K.

Fundamental to the study of the equational theory of algebras are two important results, both due to Birkhoff [1935*]. The first provides a purely algebraic characterisation of equationally definable algebras.

(3.2) **Theorem** A non-empty class K of algebras is an equationally definable class iff it is closed under subalgebras, homomorphic images and direct products (i.e., it is a *variety*).

An equational class is thus synonymous with a variety. The second result (given in Theorem (3.3) below) establishes a correspondence between the equational theory of a class of algebras and equational logic.

Equational logic is a restricted form of first-order logic. Sentences in the language of equational logic come only in one form, namely as equations. (That is, the symbol '$=$' is the only predicate symbol in the language.) An *equation* is a pair of terms p and q, written '$p = q$'. The *terms* are recursively constructed as combinations of variables and operations on variables. The standard set of inference rules are the following:

Reflexivity: Given any term p, infer $p = p$.

Symmetry: Given any equation $p = q$, infer $q = p$.

Transitivity: Given any equations $p = q$ and $q = r$, infer $p = r$.

Replacement: Given any equation $p = q$ and any term r with p as subterm, infer $r = s$, where s is the term r in which p is replaced by q.

Substitution: Given any equation $p = q$ and any term r, for any given variable x occurring in $p = q$, infer $p[x/r] = q[x/r]$ (that is, the equation $p = q$ with every occurrence of x replaced by r).

The rule of reflexivity generates equations referred to as the *tautologies*. An equation e is *derivable* from the set of equational axioms Σ, written $\Sigma \vdash e$, if there is a finite proof, that is, a finite sequence of equations $(e_1, \ldots, e_n = e)$ such that each equation e_i is either an axiom in Σ or is inferred from earlier equations using the above rules. Birkhoff [1935*] gave the first proof that (as in first-order logic) derivability in equational logic is both sound and complete.

(3.3) **Theorem (Completeness Theorem for Equational Logic)** $\Sigma \models e$ iff $\Sigma \vdash e$.

This result gives us (as Burris and Sankappanavar [1981, p. 96] appropriately say) a 'two-edged sword' for studying the consequences from a set of equations. On the one hand, we can study *derivability* in equational logic (i.e. $\vdash$), and on the other hand, we can study *satisfaction* of equations in a class of algebras (i.e. $\models$). It is therefore not surprising that equational logic is treated in many standard references on Universal algebra, including Burris and Sankappanavar [1981, Chapter II §14]. Refer also to Henkin [1977]. For a survey of equational logic the reader is advised to refer to Tarski [1968*] and Taylor (in Appendix 4 of Grätzer [1979]).

One of the useful properties of varieties is that they possess *free algebras* (a result also due to Birkhoff [1935*], see also Burris and Sankappanavar [1981, §10]). In Section 4.2 I will exploit this fact to construct algebras from given sets of elements.

(3.4) **Definition** (Grätzer [1968]) Let K be a class of algebras and let $\mathcal{A} \in K$. Let $\mathcal{A}$ be *generated by* a set X, i.e. $\mathcal{A}$ is the smallest algebra in K containing X. $\mathcal{A}$ is said to be a *free algebra* over K if, for any algebra $\mathcal{A}' \in K$, and for any mapping $f : X \longrightarrow A'$, there is a homomorphism g of $\mathcal{A}$ into $\mathcal{A}'$ such that $f(x) = g(x)$ for all $x \in X$.

Free algebras are commonly constructed in two stages. First, a *term algebra* over a class K from the set X is constructed. This involves combining the elements in X through the fundamental operations in K in all possible ways, yielding 'absolutely freely generated' terms. An example of a term algebra (from the theory of formal languages) is the *word algebra* in which elements are combined by concatenation. In the second stage term algebras are transformed into K-algebras, by forming equivalence classes of equivalent terms. This involves quotienting the term algebra with respect to the congruence relation (operation preserving equivalence relation) determined by the axioms of K. The resulting algebra is a free K-algebra freely generated by X. The reader interested in the technical details could refer to Grätzer [1968, Chapter 4] and Burris and Sankappanavar [1981, Chapter II §10 & §11].

3.1 Boolean Algebras

There is more than one way of reasoning with sets and relations. One approach is to do so within the *elementary theory* of sets and relations. In this context reasoning takes place in first-order logic. The other approach is to do so in the *calculus* of sets and relations. In this framework, which I adopt here, one uses *equational* reasoning rather than first-order reasoning.

In this section I present the standard algebraic formalisation of the calculus of sets. I adopt the usual set-theoretic terminology and notation. Sets are denoted by A, B, C ... and the operations of union, intersection and complement by $\cup$, $\cap$ and $'$, respectively. These operations have some fundamental properties which can be formulated as equations. For example:

(3.5) $A \cup B = B \cup A$

(3.6) $A \cap B = B \cap A$

(3.7) $A \cup (A \cap B) = A$

(3.8) $A \subseteq B$ iff $A \cup B = B$.

To illustrate the difference between reasoning in the elementary theory of sets and the calculus of sets I give a small example. Suppose we want to prove the following theorem:

(3.9) $A \cap B \subseteq B$.

Proving this fact in the *elementary theory of sets* involves 'element-wise' reasoning. A proof would look something like this:

> If either A or B is empty then $A \cap B$ is empty and hence (3.9) is trivially satisfied. In case neither A nor B is empty, consider any element $x \in A \cap B$. By definition of intersection $x \in A \cap B$ iff $x \in A$ and $x \in B$. Hence $x \in A \cap B$ implies $x \in A$ and in particular $x \in B$. This completes the proof.

On the other hand, in the context of the *calculus of sets* the proposition (3.9) follows as an *equational* consequence of the properties (3.5)–(3.8) as in the following proof:

> By (3.8) it suffices to prove that $(A \cap B) \cup B = B$. By commutativity, (3.5)

and (3.6), and by absorption (3.7), we have:

$(A \cap B) \cup B \;=\; B \cup (A \cap B) \;=\; B \cup (B \cap A) \;=\; B.$

While the first approach involves 'local' reasoning (since we use the set-theoretic definitions of intersection and inclusion) this approach involves 'global' reasoning (since we use some fundamental properties). The point of the second approach is its conceptual simplicity: we do not need to talk about sets *and* elements, we only need to talk about sets.

The classic presentation of equational reasoning about sets involves the notion of a *Boolean algebra*. Boolean algebra, named after the G. Boole [1847*, 1854*], is presented in many textbooks such as Burris and Sankappanavar [1981], Grätzer [1968], Birkhoff [1973], Bell and Slomson [1971], Sikorski [1964], Halmos [1974] and Mendelson [1970]. Equational reasoning in Boolean algebra is also known as the *arithmetic* of Boolean algebra, to distinguish it from the *universal-algebraic* study of Boolean algebras.

(3.10) **Definition** A *Boolean algebra* is an algebra $\mathcal{B} = (B, +, \cdot, ', 0, 1)$ such that for each $a, b, c \in B$ the following hold:

B1 $\quad a + a = a, \qquad a \cdot a = a$

B2 $\quad a + b = b + a, \qquad a \cdot b = b \cdot a$

B3 $\quad a + (b + c) = (a + b) + c, \qquad a \cdot (b \cdot c) = (a \cdot b) \cdot c$

B4 $\quad a \cdot (a + b) = a, \qquad a + a \cdot b = a$

B5 $\quad a + b \cdot c = (a + b) \cdot (a + c), \qquad a \cdot (b + c) = a \cdot b + a \cdot c$

B6 $\quad a + 0 = a, \qquad a \cdot 1 = a$

B7 $\quad a + a' = 1, \qquad a \cdot a' = 0.$

The two binary operations $+$ and $\cdot$ are respectively referred to as *join* (or *sum*) and *meet* (or *product*), the unary operation $'$ as *complement* and the constants (i.e. the nullary operations) 0 and 1 as *zero* and *unit*, respectively. Unless guided by the parentheses the association for the operations is left to right with $'$ binding tightest, then $\cdot$ and finally $+$. (Note that here 'B' denotes the base set of a Boolean algebra $\mathcal{B}$.)

The above axiomatisation of Boolean algebra is neither independent nor minimal, and many other axiomatisations exist (as shown by Huntington [1904*, 1933*]). A Boolean algebra can alternatively be defined as a bounded, complemented and distributive lattice $(B, \leq)$, where $\leq$ is a partial order defined on B such that for each $a, b \in B$:

B8 $\quad a \leq b \;$ iff $\; a + b = b \;$ (or $\; a \cdot b = a$).

A *lattice* is a partially ordered set in which each pair of elements has a least upper bound (a join) and an greatest lower bound (a meet). Lattices are characterised by axioms B1 to B4.

The next result lists some elementary arithmetical properties of Boolean algebra which follow from the axioms.

(3.11) **Theorem** In any Boolean algebra the following properties hold:

B9 $\quad a + 1 = 1, \qquad a \cdot 0 = 0$

B10 $\quad a'' = a$

B11 $\quad (a + b)' = a' \cdot b', \qquad (a \cdot b)' = a' + b'$

B12 $\quad a \leq b \;$ iff $\; b' \leq a' \;$ iff $\; a' + b = 1 \;$ iff $\; a \cdot b' = 0$

B13 $\quad 0' = 1, \qquad 1' = 0.$

The paradigm example of a Boolean algebra is a *full Boolean algebra* (also called a *power set algebra*) $\mathcal{B}(U) = (\mathbf{2}^U, \cup, \cap, ', \emptyset, U)$ over some non-empty set U, called the *universe*. $\mathbf{2}^U$ denotes the set of all subsets of U and is partially ordered by $\subseteq$. We also write $\mathcal{B}(U) = (\mathbf{2}^U, \subseteq)$. Let F be a set of subsets of some set U, i.e. $F \subseteq \mathbf{2}^U$, such that F is closed under $\cup$, $\cap$ and $'$. $(F, \subseteq)$ is called a *field of sets on* U (also referred to as a *proper Boolean algebra*). Any field of sets is a subalgebra of a full Boolean algebra.

A Boolean algebra $\mathcal{B}$ is said to be *atomic* iff for each non-zero element $b \in B$ there is some *atom* $a \in B$ such that $a \leq b$, where a is a minimal non-zero element in B with respect to the ordering $\leq$. For example, every full Boolean algebra $\mathcal{B}(U)$ is atomic, its atoms being the singleton sets. Also, every finite Boolean algebra is atomic. In fact, any finite Boolean algebra must have 2^n elements (for some non-negative integer n). These elements correspond to the 2^n subsets of some set of n atoms. Any finite Boolean algebra is thus isomorphic to some full Boolean algebra. More generally, the following very important theorem due to Stone [1936*] asserts that any Boolean algebra is isomorphic to a *subalgebra* of a full Boolean algebra.

(3.12) **Theorem (Representation Theorem for Boolean Algebras)** Every Boolean algebra is *representable*, i.e. isomorphic to some field of sets.

This theorem implicitly states that reasoning about sets is successfully captured by the arithmetic of Boolean algebra. This means that for the calculus of sets 'element-wise' reasoning is unnecessary.

A Boolean algebra $\mathcal{B}$ may be closed under arbitrary joins and meets. That is, it may happen that for any subset A of B, the least upper bound (written $\sum_{a \in A} a$) and the greatest lower bound (written $\prod_{a \in A} a$) exist. Then $\mathcal{B}$ is called *complete*. Tarski [1935*] shows that any complete and atomic Boolean algebra is isomorphic to a full Boolean algebra. This generalises Theorem (3.12).

The class of Boolean algebras is a standard example of an equationally definable class (i.e. a variety). *Free Boolean algebras* exist and have been extensively studied. See, for example, Sikorski [1964] and Halmos [1974].

3.2 Relation Algebras

In this section I consider the *algebra of binary relations*. A *binary relation* over some non-empty set is a subset of the Cartesian product U^2 ($= U \times U = \{(x,y) \mid x \in U \;\&\; y \in U\}$). The set U is called the *universe*. All relations considered in this thesis are binary and are denoted by R, S, T .... New relations can be formed using the set-theoretic (or *Boolean*) operations union, intersection and complement. In addition, relations are endowed with *relational operations* and *constants*, such as:

(3.13) *Composition*: $R\,;S = \{(x,y) \mid (\exists z)[(x,z) \in R \;\&\; (z,y) \in S]\}$

(3.14) *Converse*: $R^{\smile} = \{(x,y) \mid (y,x) \in R\}$

(3.15) *Identity*: $Id = \{(x,x) \mid x \in U\}$.

(Composition is also referred to as *relational* (or *relative*) *product* and the identity relation as the *diagonal relation*. Many authors denote the identity relation by Id_U or $Id \lceil U$ to indicate the universe explicitly.) For example, if R is the relation 'is a brother of' and S is the relation 'is a parent of' then $R\,;S$ is the relation 'is an uncle of' and $R^{\smile}$ is the relation 'has as brother'. Just as the Boolean operations are governed by equational laws so are the relational operations and constants. For example:

(3.16) Composition is associative: $(R\,;S)\,;T = R\,;(S\,;T)$

(3.17) Converse is an involution: $R^{\smile\smile} = R$

(3.18) Converse distributes over ; and reverses the order: $(R\,;S)^{\smile} = S^{\smile}\,;R^{\smile}$

(3.19) Id is an identity of composition: $R\,;Id = R = Id\,;R$.

As with sets there is more than one way of reasoning with relations: First, within the *elementary theory of relations*, and second, within the *calculus of relations*. Our interest lies with the equational approach, which avoids 'element-wise' reasoning. The calculus of relations originated in the nineteenth century with A. De Morgan [1847*], C.S. Peirce [1870*, 1931–1935*] and E. Schröder [1890–1895*]. Peirce and Schröder recognised that many familiar properties of relations can be formulated equationally. For example:

(3.20) **Theorem** Let R be any binary relation over some non-empty universe U. Then:

(i) R is reflexive iff $Id \subseteq R$

(ii) R is symmetric iff $R = R^{\smile}$

(iii) R is anti-symmetric iff $R \cap R^{\smile} \subseteq Id$

(iv) R is transitive iff $R\,;R \subseteq R$

(v) R is single-valued iff $R^{\smile}\,;R \subseteq Id$.

Proof. I only prove (iv). The other proofs are similar. $R\,;R \subseteq R$ iff $(\forall x)(\forall y)[(x,y) \in R\,;R \Rightarrow (x,y) \in R]$ iff $(\forall x)(\forall y)[(\exists z)[(x,z) \in R \;\&\; (z,y) \in R] \Rightarrow (x,y) \in R]$ iff $(\forall x)(\forall y)[(\forall z)\neg[(x,z) \in R \;\&\; (z,y) \in R]$ or $(x,y) \in R]$ iff $(\forall x)(\forall y)(\forall z)[\neg[(x,z) \in R \;\&\; (z,y) \in R]$ or $(x,y) \in R]$ iff $(\forall x)(\forall y)(\forall z)[[(x,z) \in R \;\&\; (z,y) \in R] \Rightarrow (x,y) \in R]$ iff R is transitive. $\square$

(Note that by (3.8), for example, the inclusions can be rewritten as equations.) A reflexive, symmetric and transitive relation is also known as an *equivalence relation* and a single-valued relation as a *partial function.* The identity relation Id and the *universal relation* U^2 are (extreme) examples of equivalence relations.

In a seminal paper [1941] Tarski distinguished the calculus of relations from the 'elementary theory' of relations, and proposed a set of equations (including those of (3.16)–(3.19)) as axioms for the formalisation of the calculus of relations. (These appear in Definition (3.21) below.) His work gave rise to *relation algebras*, the first definition of which appeared in Jónsson and Tarski [1948*]. I use the definition from Chin and Tarski [1951], as adapted in Tarski [1955*]. (The reader interested in a historic account of relation algebras could refer to, e.g., Maddux [1990a] and Brink [1988].)

(3.21) **Definition** A *relation algebra* is an algebra $\mathcal{R} = (R, +, \cdot, ', 0, 1, ;, ^{\smile}, e)$ satisfying the following axioms for each $r, s, t \in R$:

R1 $(R, +, \cdot, ', 0, 1)$ is a Boolean algebra

R2 $r\,;(s\,;t) = (r\,;s)\,;t$

R3 $r\,;e = r = e\,;r$

R4 $r^{\smile\smile} = r$

R5 $(r+s)\,;t = r\,;t + s\,;t$

R6 $(r+s)^{\smile} = r^{\smile} + s^{\smile}$

R7 $(r\,;s)^\smile = s^\smile ; r^\smile$

R8 $r^\smile ; (r\,;s)' \leq s'$.

For convenience parentheses are omitted according to the convention that $^\smile$ and $'$ bind tightest, then ; and $\cdot$ and subsequently $+$. As in Boolean algebra $+$, $\cdot$, $'$, 0, 1 are known as join, meet, complement, zero and unit respectively. The operations ; and $^\smile$ are known respectively as *relative product* (or *composition*) and *conversion*. The designated element e in R is called the *identity* element. (Although my use of the symbol 'R' is overloaded, it will be clear from the context whether it denotes a binary relation or the base set of a relation algebra $\mathcal{R}$.)

Observe that most of the axioms specify familiar properties. Axioms R2, R3 and R4 define an *involutive monoid* $(R, ;, ^\smile, e)$. Axioms R5, R6 and R7 define ; and $^\smile$ (essentially) as distributive operations. R8 implicitly deals with *residuation* (to which I will return). Observe also that relation algebras are defined purely in terms of equational axioms (since R8 can be rewritten as an equation, using B8). In fact, the class of relation algebras forms a variety (that is, the class is equationally definable).

The standard example of relation algebras are proper relation algebras. A *proper relation algebra over some non-empty universe* U is defined by $(F, \cup, \cap, ', \emptyset, U^2, ;, ^\smile, Id)$ where F is a non-empty family of binary relations between elements in U (that is, $F \subseteq \mathbf{2}^{U^2}$). The operations ; and $^\smile$ correspond to the relation-theoretic operations of composition and converse defined by (3.13) and (3.14). In the case where F is the set of *all* subsets of U^2, i.e. $F = \mathbf{2}^{U^2}$, the algebra is called the *full relation algebra over the set* U which I denote by $\mathcal{R}(U)$. In general, an algebra of binary relations need not have the universal relation U^2 as its unit. Such an algebra is referred to merely as a *proper relation algebra* (with the definition of complementation appropriately adapted). A definition of proper relation algebras can be found in Jónsson and Tarski [1952] and Tarski and Givant [1987, p. 239].

Since proper relation algebras are relation algebras every property derivable from the axioms of relation algebras is satisfied in the elementary theory of relations. The question is: does the arithmetic of relation algebras capture the elementary theory of relations, in the same sense as Boolean algebras capture the elementary theory of sets?

More technically, is there a representation theorem of the same strength as Stone's theorem (Theorem (3.12))—i.e. to the effect that every relation algebra is isomorphic to some proper relation algebra? Lyndon [1950*] showed that the answer is negative: there are non-representable relation algebras. This means there are properties in the elementary theory of binary relations which cannot be proved in the framework of relation algebra. The class of relation algebras which are representable has been extensively studied. Early references are Jónsson and Tarski [1951, 1952], Tarski [1955*] and Monk [1964*]; later work include McKenzie [1970*], Maddux [1978a] and Maddux [1989] (in which further references can be found). We know from Tarski [1955*] that the class of representable relation algebras forms a variety (an equational class of algebras). But from Monk [1964*] we know the class of representable relation algebras is not finitely axiomatisable (it is not a finitely based variety), which means that neither is the set of true equations in the calculus of relations.

Another problem with a negative solution is the *decidabilty problem* of relation algebra. The elementary theory of relations, being a form of first-order logic in which binary predicates appear, is certainly undecidable. As Tarski [1941] noted, there is also no decision method for deciding whether a statement (free of variables) is true in the calculus of relations. Tarski also proved that the equational theory of relation algebra is undecidable. (This result was already announced in Chin and Tarski [1951] but his proof appears only in Tarski and Givant [1987, §8.5]. Maddux [1978b*] presents a different proof.) According to Maddux [1990b] some restricted classes of equations yield more tractable problems; see, e.g., Schönfeld [1982*]. For further references see also Tarski and Givant [1987, §8.7].

Another natural question is whether there is a systematic procedure for transforming every property of binary relations formulated as a first order sentence into an equivalent sentence formulated in relation algebras. Again, the answer is negative. This result is attributed by Tarski [1941] to Korselt (published in Löwenheim [1915*]).

Despite all these negative results relation algebra is still surprisingly powerful. Its arithmetic is very rich, and although it may not be possible to express and derive every property of relations a lot can be achieved. Namely, Tarski (see Chin and Tarski [1951, p. 341]) made the astounding claim that every problem concerning the derivability of a mathematical statement from a set of axioms reduces to the problem of whether an equation is derivable from a set of equations in the calculus of relations. Therefore in principle the whole of mathematical research can be carried out within the framework of this calculus. This claim is fully explained and motivated in a major publication by Tarski and Givant [1987].

Chin and Tarski [1951] give the most extensive treatment of the arithmetic of relation algebras. Additional properties are proved in (among others) Jónsson and Tarski [1952], Henkin *et al* [1985, p. 212–214] and Jónsson [1988]. In [1982] Jónsson gives a concise survey of relation algebra and lists (without proof) some of the more important properties in the arithmetic of relation algebra. The following theorem lists some properties I will need for later work.

(3.22) **Theorem** (Chin and Tarski [1951]) In any relation algebra $\mathcal{R}$ the following properties are satisfied for each $r, s, t, u \in R$:

R9 $\quad e^{\smile} = e, \qquad 0^{\smile} = 0, \qquad 1^{\smile} = 1$

R10 $\quad r \leq s$ iff $r^{\smile} \leq s^{\smile}$

R11 $\quad (r \cdot s)^{\smile} = r^{\smile} \cdot s^{\smile}, \qquad r'^{\smile} = r^{\smile\prime}$

R12 $\quad r\,;0 = 0 = 0\,;r, \qquad 1\,;1 = 1$

R13 $\quad r\,;(s+t) = r\,;s + r\,;t$

R14 $\quad$ if $r \leq s$ then $t\,;r \leq t\,;s$ and $r\,;t \leq s\,;t$

R15 $\quad (r\,;s) \cdot t = 0$ iff $(r^{\smile}\,;t) \cdot s = 0$ iff $(t\,;s^{\smile}) \cdot r = 0$

R16 $\quad (r\,;s) \cdot (t\,;u) \leq r\,;[(r^{\smile}\,;t) \cdot (s\,;u^{\smile})]\,;u.$

Relation algebras contain some special elements, which I define for later reference.

(3.23) **Definition** Let r be an element in a relation algebra.

(i) r is a *reflexive* element iff $e \leq r$

(ii) r is a *symmetric* element iff $r = r^{\smile}$

(iii) r is a *transitive* element iff $r\,;r \leq r$

(iv) r is a *equivalence* element iff $r = r^{\smile}$ and $r\,;r \leq r$

(v) r is a *functional* element iff $r^{\smile}\,;r \leq e$

It follows directly from Theorem (3.20) that reflexive, symmetric and transitive elements in a proper relation algebra are relations with the corresponding properties. Also, every functional element is single-valued and every reflexive equivalence element is an equivalence relation. In Section 3.4 I need the following property, proved in Chin and Tarski [1951].

R17 If $r \leq e$ then r is an equivalence element.

Special elements which I will use in Chapter 4 to interpret the two role binding constructs $(\subseteq \mathsf{R}\ \mathsf{S})$ and $(\supseteq \mathsf{R}\ \mathsf{S})$ in terminological languages are the *residual* elements. Residuation has been extensively studied and plays an important role in lattice theory and algebra. Residual elements in algebraic systems are defined in, e.g., Birkhoff [1973] and Blyth and Janowitz [1972].

(3.24) **Definition** (Jónsson [1982]) The *right* (respectively *left*) *residual* of an element r over an element s in a relation algebra is the largest element u (respectively v) such that

$$s\,;u \leq r \quad \text{(respectively} \quad v\,;s \leq r).$$

The right residual u of r over s is denoted by $s\backslash r$ and the left residual v by r/s.

(3.25) **Theorem** Let $\mathcal{R}$ be any relation algebra with $r, s, t \in R$. Then:

R18 $s\,;t \leq r$ iff $t \leq s\backslash r$

R19 $t\,;s \leq r$ iff $t \leq r/s$

R20 $s\backslash r = (s^{\smile}\,;r')'$ and $r/s = (r'\,;s^{\smile})'$

R21 $r^{\smile} = r'\backslash e' = e'/r'$

R22 $r = r^{\smile\prime}\backslash e' = e'/r^{\smile\prime}$

R23 $r\backslash r$ and r/r are both reflexive and transitive.

Proof. R18 and R19 are standard definitions for right and left residuation. See Blyth and Janowitz [1972].

R20: I only show the first equality, but the second equality can be proved similarly. For any $t \in R$, $s\,;t \leq r$ iff $(s\,;t)\cdot r' = 0$ iff $(s^{\smile}\,;r')\cdot t = 0$ iff $t \leq (s^{\smile}\,;r')'$ by B12 and R15. Hence by definition $s\backslash r = (s^{\smile}\,;r')'$.

To prove R21 use R20, B10, R3 and R11: $r'\backslash e' = (r'^{\smile}\,;e'')' = (r'^{\smile}\,;e)' = r'^{\smile\prime} =$

$r''^{\smile} = r^{\smile}$. Similarly $e'/r' = r^{\smile}$.

R22 is immediate by R21 and R4.

For a proof of R23, see Pratt [1990a, Proposition 3]. (This result is originally due to Peirce (1893); see Maddux [1990a, p. 13, (A34)].) □

Observe that the residuals (as given by R20) are characterised by axiom R8. R20 implies that both right and left residuals exist in every relation algebra. By R22 every element r in a relation algebra is in fact a residual. (R20 was used by Brink [1979] to give alternative axiomatisations for relation algebras.) R21 implies that the residuation operations could be taken as fundamental operations for relation algebras in place of conversion.

(3.26) **Theorem** In any proper relation algebra the right and left residuals of a relation R over a relation S are respectively given by:

(i) $(x, y) \in S\backslash R$ iff $(\forall z)[(z, x) \in S \Rightarrow (z, y) \in R]$

(ii) $(x, y) \in R/S$ iff $(\forall z)[(y, z) \in S \Rightarrow (x, z) \in R]$.

Proof. (i) By R20, (3.13), (3.14) and using first-order logic $(x, y) \in S\backslash R$ iff $(x, y) \in (S^{\smile} ; R')'$ iff $\neg(\exists z)[(x, z) \in S^{\smile}$ & $(z, y) \in R']$ iff $(\forall z)[(x, z) \notin S^{\smile}$ or $(z, y) \in R]$ iff $(\forall z)[(z, x) \in S \Rightarrow (z, y) \in R]$.

(ii) Analogously $(x, y) \in R/S$ iff $(x, y) \in (R' ; S^{\smile})'$ iff $\neg(\exists z)[(x, z) \in R'$ & $(z, y) \in S^{\smile}]$ iff $(\forall z)[(x, z) \in R$ or $(z, y) \notin S^{\smile}]$ iff $(\forall z)[(y, z) \in S \Rightarrow (x, z) \in R]$. □

To model the transitive closure operator in terminological languages I require relation algebras to have arbitrary joins. This is given in *complete* relation algebras in which the underlying Boolean algebra is complete. If the underlying Boolean algebra is atomic the relation algebra is said to be *atomic*, its *atoms* being the atoms in the underlying Boolean algebra.

(3.27) **Definition** Let r be an element in a relation algebra.

(i) r is a *right-ideal* element iff $r ; 1 = r$

(ii) r is a *left-ideal* element iff $1 ; r = r$

(iii) r is an *ideal* element iff $1 ; r ; 1 = r$.

In a proper relation algebra over a set U with unit U^2, a right-ideal element is a relation $R = \mathrm{dom}(R) \times U$, a left-ideal element a relation $R = U \times \mathrm{ran}(R)$ and the only ideal elements are $\emptyset$ and U^2. (Note: $\mathrm{dom}(R)$ and $\mathrm{ran}(R)$ denote respectively the domain and range of the relation R and are formally defined in (3.29) and (3.30) of Section 3.3.) Observe that the relations (defined in (2.43)) presenting domain and range expressions are examples of right- and left-ideal elements, respectively. Although there is no direct way of expressing the notions of domain and range in the calculus of relations (because they are sets, not relations), their properties can often be expressed indirectly with right- and left-ideals (see Chin and Tarski [1951, pp. 360]). For example, the two relations R and S have the same domain iff $R ; U^2 = S ; U^2$.

To establish a result in Peirce algebras (which I discuss in Section 3.4) I need the following property:

R24 If s is a right-ideal element then $r \cdot s = (s \cdot e) ; r$.

Proof. Applying R14 to $s \cdot e \le e$ and using R3 we obtain $(s \cdot e) ; r \le e ; r = r$. Similarly $(s \cdot e) ; r \le s ; r \le s ; 1 = s$ since s is a right-ideal element. Hence $(s \cdot e) ; r \le r \cdot s$. It remains to be shown that $r \cdot s \le (s \cdot e) ; r$. For this I use R16. $r \cdot s = s \cdot r = (e ; s) \cdot (e ; r) \le e ; [(e^{\smile} ; e) \cdot (s ; r^{\smile})] ; r = [e \cdot (s ; r^{\smile})] ; r$ by B2, R3, and R9. Hence since $r^{\smile} \le 1$ we get $r \cdot s \le [e \cdot (s ; 1)] ; r = (e \cdot s) ; r = (s \cdot e) ; r$ using R14, $s ; 1 = s$ and B2. $\square$

Ideal elements have interesting algebraic properties. (For example, the set of ideal elements forms a Boolean algebra. Likewise do the sets of right- and left-ideal elements. See Chin and Tarski [1951]). Ideal elements have the important property that they can be used to characterise *simple relation algebras*. In general, an algebra $\mathcal{A}$ is *simple* iff the only congruence relations over A are the identity relation I_A and the universal relation A^2. The next theorem establishes a more natural *arithmetical* characterisation of simple relation algebras.

(3.28) **Theorem** (Jónsson and Tarski [1952, Theorem 4.10]) For every non-trivial relation algebra $\mathcal{R}$ the following are equivalent.

(i) $\mathcal{R}$ is simple.

(ii) $\mathcal{R}$ has exactly two distinct ideal elements, namely 0 and 1.

(iii) For every $r \in R$, $\quad r \neq 0$ iff $1\,;r\,;1 = 1$.

Note that in (iii) Jónsson and Tarski only use the implication $(\forall r \in R)\ [r \neq 0 \Rightarrow 1\,;r\,;1 = 1]$. But the converse is easily shown. For suppose that $1\,;r\,;1 = 1$ but that $r = 0$, then $1\,;0\,;1 = 1$, hence $0 = 1$ by R12, contradicting the non-triviality of $\mathcal{R}$. For my purposes the theorem is important since it allows me to formulate any inequality of the form $r \neq 0$ in simple relation algebras as an equation.

It is worth mentioning that even stronger results are available. Namely, every Boolean combination of equations in a simple relation algebra can be equivalently formulated as an equation of the form $r = s$ and even as an equation of the form $r = 1$. (This fact was established already by Schröder [1890–1895*] for the calculus of relations; see Maddux [1990a, p. 13].) There is even an effective transformation procedure, for which see, e.g., Tarski [1941] and Jónsson [1982].

It is easy to verify that every proper relation algebra over a non-empty set U with unit U^2 is simple (by verifying that condition (iii) of Theorem (3.28) holds). Hence every full relation algebra $\mathcal{R}(U)$ is also simple. In fact Jónsson and Tarski [1952, Theorem 4.30] characterised a full relation algebra as a complete and atomic relation algebra $\mathcal{R}$ which is simple and $r\,;1\,;r^{\smile} \leq e$ holds, for every atom $r \in R$. An earlier characterisation of full relation algebras was given by McKinsey [1940].

In Section 4.2 I construct *free relation algebras*. These exist and have been treated by various authors including Tarski and Givant [1987, Chapter 8], Maddux [1978b*] and recently by Andréka *et al* [1990, 1991].

3.3 Boolean Modules

To model reasoning with concepts and roles *interacting* with each other I now consider an algebraic formalisation of sets *interacting* with relations. Besides the set-forming operations on sets, that is, intersection, union and complement (which are accommodated in a Boolean algebra as discussed in Section 3.1) there are also *set-forming operations on relations*. The following are examples of such operations:

(3.29) *Domain*: $\mathrm{dom}(R) = \{x \mid (\exists y)[(x, y) \in R]\}$

(3.30) *Range*: $\mathrm{ran}(R) = \{y \mid (\exists x)[(x, y) \in R]\}$.

Other operations combine a relation and a set to yield a set. For example:

(3.31) *Peirce product*: $R : A = \{x \mid (\exists y)[(x, y) \in R \;\&\; y \in A]\}$

(3.32) *Image*: $R\,\text{"}A = \{y \mid (\exists x)[(x, y) \in R \;\&\; x \in A]\}$.

For our application Peirce product is the most convenient operation. (It was named by Brink [1978, 1981] in honour of the nineteenth century American logician C.S. Peirce [1870*] who first used it.) The Peirce product $R : A$ is the set of all those elements related by R to *some* element in A. For example, if R is the relation 'is an admirer of' and A is the set of princes, then $R : A$ is the set of 'admirers of (some) princes'. The other operations are variants of Peirce product, since:

(3.33) $\mathrm{dom}(R) = R : U$

(3.34) $\mathrm{ran}(R) = R^{\smile} : U$

(3.35) $R\,\text{"}A = R^{\smile} : A$.

In this section we are interested in the equational laws satisfied by Peirce product. For example:

(3.36) Id is an identity of Peirce product: $Id : A = A$

(3.37) Peirce product distributes over union: $R : (A \cup B) = R : A \,\cup\, R : B$

(3.38) Peirce product is weakly associative: $R : (S : A) = (R ; S) : A$.

Arithmetic with such identities constitutes the *calculus of sets interacting with relations*. Just as Boolean algebras were introduced in an attempt to formalise the calculus of sets and relation algebras in an attempt to formalise the calculus of relations, Brink [1978] introduced *Boolean modules* in an attempt to formalise the calculus of sets interacting with relations through Peirce product. Accordingly, a Boolean module is defined as a two-sorted algebra, which can be regarded as a Boolean algebra with a multiplication (the algebraic counterpart of Peirce product) from a relation algebra. (A *two-sorted algebra* has two base sets with the fundamental operations defined on either base and additional fundamental operations defined on elements in both base sets. For a formal definition of many-sorted or heterogeneous algebras see, e.g., Ehrig and Mahr [1985, p. 16], Manes and Arbib [1986, p. 322] and Birkhoff and Lipson [1970].)

(3.39) **Definition** (Brink [1988]) A *Boolean module* is a two-sorted algebra $\mathcal{M} = (\mathcal{B}, \mathcal{R}, :)$, where $\mathcal{B} = (B, +, \cdot, ', 0, 1)$ is a Boolean algebra, $\mathcal{R} = (R, +, \cdot, ', 0, 1, ;, \smile, e)$ is a relation algebra and $:$ is a mapping $\mathcal{R} \times \mathcal{B} \longrightarrow \mathcal{B}$ (called *Peirce product* and written $r : a$ instead of $:(r, a)$ for any $r \in R$, $a \in B$) such that for any $r, s \in R$ and $a, b \in B$:

M1 $\quad r : (a + b) = r : a + r : b$

M2 $\quad (r + s) : a = r : a + s : a$

M3 $\quad r : (s : a) = (r ; s) : a$

M4 $\quad e : a = a$

M5 $\quad 0 : a = 0$

M6 $\quad r^{\smile} : (r : a)' \leq a'$.

The order of precedence among the operations is $'$ and $\smile$, $:$, $;$, $\cdot$ and $+$ (in decreasing order). Note that the operations ($+$, $\cdot$ and $'$) and the constants (0 and 1) in the Boolean algebra $\mathcal{B}$ are not notationally distinguished from those in the underlying Boolean algebra of the relation algebra $\mathcal{R}$. Nevertheless the association of the symbols should be clear from the context.

If $\mathcal{B}(U)$ is the full Boolean algebra and $\mathcal{R}(U)$ is the full relation algebra over some non-empty set U then $\mathcal{M}(U) = (\mathcal{B}(U), \mathcal{R}(U), :)$, with $:$ the Peirce product defined by (3.31), is an example of a Boolean module. I will refer to $\mathcal{M}(U)$ as the *full Boolean module* over U. The standard models of the axiomatisation of Boolean modules are the *proper Boolean modules* which are defined more generally than full Boolean modules. For the purposes of my exposition it suffices to say that a proper Boolean module is essentially a two-sorted algebra of a proper Boolean algebra and a proper relation algebra together with Peirce product on relations. (See Brink [1981] for a formal definition of proper Boolean modules and further examples of Boolean modules.)

As for relation algebras the question arises whether we can algebraically formulate and derive *every* property satisfied in the calculus of sets interacting with relations. More formally the question is whether every Boolean module is representable, that is, isomorphic to a subalgebra of a full Boolean module. The fact that relation algebras are not representable seems to preclude a positive answer. (If $\mathcal{R}$ is a non-representable relation algebra then the Boolean module $\mathcal{M} = (\mathcal{B}, \mathcal{R}, :)$ is not representable either.) Based on the notion of *weak representabilty* for relation algebras (considered by Jónsson and Tarski [1952]), Brink [1978, 1981] established weak representability for a certain class of Boolean modules (those satisfying *bijectivity*, that is, for each $r, s \in R$ if for each $a \in B$, $r : a = s : a$ then $r = s$). Pretorius [1990] obtained the same result for a wider class of algebras (namely, Boolean algebras with normal additive unary operators).

Nevertheless the arithmetic of Boolean modules is sufficiently powerful, for us to derive more than we need in Section 4.3. In the following theorem I list some essential arithmetical properties of Boolean modules which are proved in Brink [1981, p. 296–297].

(3.40) **Theorem** In any Boolean module $\mathcal{M}$ the following hold for each $a, b \in B$ and $r, s \in R$:

M7 $\quad a \leq b \quad \Rightarrow \quad r : a \leq r : b$

M8 $\quad r \leq s \quad \Rightarrow \quad r : a \leq s : a$

M9 $\quad r : (a \cdot b) \leq (r : a) \cdot (r : b)$

M10 $\quad (r \cdot s) : a \leq (r : a) \cdot (s : a)$

M11 $\quad r : 0 = 0$

M12 $\quad 1 : 1 = 1$

M13 $\quad (r' : 1)' \leq r : 1$

M14 $\quad (r : a) \cdot b \leq r : ((r^{\smile} : b) \cdot a)$

M15 $\quad a \leq 1 : a$.

As can already be seen from (3.31), and as will be discussed further in Section 4.1, Peirce product is a natural algebraic version of the terminological operator some. In an attempt to find also a natural algebraic version of the the terminological operator all I have been investigating two variants of Peirce product: $(r : a')'$ and $(r' : a)'$. In a full Boolean module $\mathcal{M}(U)$ these variants are interpreted as follows:

(3.41) **Theorem** Given any binary relation R over some non-empty set U and any $A \subseteq U$, we have:

(i) $(R : A')' = \{x \mid (\forall y)[(x, y) \in R \Rightarrow y \in A]\}$

(ii) $(R' : A)' = \{x \mid (\forall y)[y \in A \Rightarrow (x, y) \in R]\}$.

Proof. By the definition of Peirce product (3.31) and the standard laws of first-order logic, $x \in (R : A')'$ iff $\neg(\exists y)[(x, y) \in R \,\&\, y \notin A]$ iff $(\forall y)[(x, y) \notin R$ or $y \in A]$ iff $(\forall y)[(x, y) \in R \Rightarrow y \in A]$.

Also, $x \in (R' : A)'$ iff $\neg(\exists y)[(x, y) \notin R \,\&\, y \in A]$ iff $(\forall y)[(x, y) \in R$ or $y \notin A]$ iff $(\forall y)[y \in A \Rightarrow (x, y) \in R]$. □

It is difficult to give an adequate English formulation for the variant $(R : A')'$. Provided the domain of R is the entire universe U, $x \in (R : A')'$ iff every element y to which x is related by R is in A, that is, x is related by R *only* to elements in A. The other variant has a more natural translation, namely $x \in (R' : A)'$ iff x is related by R to *every* element in A. Suppose R is the relation 'is an admirer of' and A is the set of princes, then $(R : A')'$ is the set of 'admirers only of princes' provided every human admires someone, while $(R' : A)'$ is the set of 'admirers of all princes'.

To reason about the all construct we only need to investigate the properties of the variant in the form $(r : a')'$.

(3.42) **Theorem** In any Boolean module $\mathcal{M}$ the following conditions hold for each $a, b \in B$ and $r, s \in R$.

M16 $\quad a \leq b \quad \Rightarrow \quad (r : a')' \leq (r : b')'$

M17 $\quad s \leq r \quad \Rightarrow \quad (r : a')' \leq (s : a')'$

M18 $\quad (r : (a \cdot b)')' = (r : a')' \cdot (r : b')'$

M19 $\quad ((r + s) : a')' = (r : a')' \cdot (s : a')'$

M20 $\quad (r : a')' + (r : b')' \leq (r : (a + b)')'$

M21 $(r:a')'+(s:a')' \leq ((r\cdot s):a')'$

M22 $(r:1)' \leq (r:a')'$

M23 $(r:a')' \leq r:a$ iff $r:1=1$

M24 $(r:a)\cdot(r:a')' = (r:1)\cdot(r:a')'$

M25 $(r:a')'\cdot(r:(a\cdot b'))' = (r:(a\cdot b)')'$.

Proof. M16: By B12 and M7 $a \leq b$ iff $b' \leq a'$ $\Rightarrow$ $r:b' \leq r:a'$ iff $(r:a')' \leq (r:b')'$.
M17 follows analogously by B12 and M8.
M18: By B11 and M1 $(r:(a\cdot b)')' = (r:(a'+b'))' = (r:a'+r:b')' = (r:a')'\cdot(r:b')'$.
M19 follows analogously by B11 and M2.
M20 follows by B11, B12 and M9: $(r:a')'+(r:b')' = ((r:a')\cdot(r:b'))' \leq (r:(a'\cdot b'))' = (r:(a+b)')'$.
M21 follows analogously by B11, B12 and M10.
M22: Since $(r:1)'+(r:a')' = ((r:1)\cdot(r:a'))' \leq (r:(1\cdot a'))' = (r:a')'$ using B11, B12, M9 and B6, the result follows by B12.
M23: Assume $(r:a')' \leq r:a$. Then since $1 \geq a$ and $a \geq 0$ we have $r:1 \geq r:a \geq (r:a')' \geq (r:0')' = (r:1)'$ by M7 and M16. Using B8 and B7, $r:1 = r:1+(r:1)' = 1$. Conversely, assume $r:1=1$. Then $(r:a')'\cdot(r:a)' = (r:a'+r:a)' = (r:(a'+a))' = (r:1)' = 1' = 0$ by B11, M1, B7 and B13. Thus by B12 we get $(r:a')' \leq r:a$.
M24: By M7 $r:a \leq r:1$. Hence $(r:a)\cdot(r:a')' \leq (r:1)\cdot(r:a')'$. To establish equality we need to show that $(r:1)\cdot(r:a')' \leq (r:a)\cdot(r:a')'$. Since $(r:1)\cdot(r:a')' \leq (r:a')'$ it suffices to prove $(r:1)\cdot(r:a')' \leq r:a$. Consider $[(r:1)\cdot(r:a')']'+r:a = (r:1)'+r:a'+r:a = (r:1)'+r:(a'+a) = (r:1)'+r:1 = 1$ by B11, B10, M1 and B7. Hence by B12, $(r:1)\cdot(r:a')' \leq r:a$.
M25 follows by B11, M1, B5, B7 and B12: $(r:a')'\cdot(r:(a\cdot b'))' = (r:a'+r:(a\cdot b'))' = (r:(a'+a\cdot b'))' = (r:((a'+a)\cdot(a'+b')))' = (r:(1\cdot(a\cdot b)'))' = (r:(a\cdot b)')'$. □

Like full relation algebras, full Boolean modules satisfy an arithmetical condition that charaterises *simple Boolean modules*. This condition will allow us to express as equations certain inequalities involving Boolean elements. (If necessary we could use this condition also to derive some more properties of sets interacting with relations.)

To define simple Boolean modules as simple algebras (defined on page 50), it must be shown that Boolean modules can be regarded as algebras in the sense of Definition (3.1).

In Brink [1978, 1981] (where Boolean modules were first defined) a Boolean module is defined as a module over a given relation algebra $\mathcal{R}$, called the (left) Boolean $\mathcal{R}$-module. (The reader familiar with ring theory will note the similarity of the following definition with that of a *module over a ring*, see Burris and Sankappanavar [1981, p. 25].)

(3.43) **Definition** Let $\mathcal{R} = (R, +, \cdot, ', 0, 1, ;, \breve{}, e)$ be a relation algebra. A *(left) Boolean $\mathcal{R}$-module* is an algebra $(B, +, \cdot, ', 0, 1, \{f_r\}_{r \in R})$ with $f_r(a)$ written as $r : a$ for any $r \in R$ and $a \in B$ such that $\mathcal{B} = (B, +, \cdot, ', 0, 1)$ is a Boolean algebra, and for any $r, s \in R$ and $a, b \in B$ the axioms M1–M6 are satisfied.

This implies a Boolean $\mathcal{R}$-module (or just Boolean module for short) can be regarded as a Boolean algebra $\mathcal{B}$ with additional unary fundamental operations indexed by the elements in the relation algebra $\mathcal{R}$. By M1 each of these operations f_r distributes over Boolean addition. (We say each f_r is *additive*.) A Boolean module is therefore a *Boolean algebra with operators* (for a definition of which see Jónsson and Tarski [1951]). Viewing Boolean modules as homogeneous algebras has the distinct advantage that one can study their algebraic theory (including representability) in the general context of Universal Algebra. In particular, we can use the definition of a simple algebra to define *simple Boolean modules*. Like simple relation algebras, these are determined by their ideal elements.

(3.44) **Definition** An *ideal* element in a Boolean module is an element a in the underlying Boolean algebra such that $1 : a = a$.

(As for relation algebras the set of ideal elements can be shown to form a Boolean algebra.) In a full Boolean module $\mathcal{M}(U)$ the only ideal elements are the empty set and the universe U. The parallel result of Theorem (3.28) is the next result.

(3.45) **Theorem** (Brink [1981, Theorem 4.1]) For every non-trivial Boolean module $\mathcal{M}$ the following are equivalent.

(i) $\mathcal{M}$ is simple.

(ii) $\mathcal{M}$ has exactly two distinct ideal elements, namely 0 and 1.

(iii) For every $a \in B$, $\quad a \neq 0 \;$ iff $\; 1 : a = 1$.

(In fact in (iii) Brink only uses the implication $(\forall a \in B)\ [a \neq 0 \ \Rightarrow\ 1 : a \ = 1]$. But as for simple relation algebras the converse holds trivially, since $0 \neq 1$.) Since $U^2 : A \ = \ U$ for $A \neq \emptyset$, any proper Boolean module over a non-empty set U with unit U^2 is simple, and in particular any full Boolean module is simple. Theorem (3.45) is important since it allows for an *arithmetical* characterisation of simple Boolean modules. This result will allow me to reformulate in a simple Boolean module any inequality of the form $a \neq 0$ as an equation.

Although *free Boolean modules* have not been studied we know they exist since the class of Boolean modules forms a variety (in fact a discriminator variety as Pretorius [1990] showed).

3.4 Peirce Algebras

In the previous section I presented Boolean modules as algebras representing a calculus in which sets interact with relations to form new *sets*. Since concepts also interact with roles to form new roles we are interested in an algebra in which sets and relations also interact to form new *relations*. Such an algebra should formalise *relation-forming operations on sets*, like:

(3.46) *Domain restriction*: $R \lceil A = \{(x,y) \mid (x,y) \in R \;\&\; x \in A\}$

(3.47) *Range restriction*: $R \rfloor A = \{(x,y) \mid (x,y) \in R \;\&\; y \in A\}$

(3.48) *Cartesian product*: $A \times B = \{(x,y) \mid x \in A \;\&\; y \in B\}$

(3.49) *Right cylindrification*: ${}^{c}A = \{(x,y) \mid y \in A\}$

(3.50) *Left cylindrification*: $A^{c} = \{(x,y) \mid x \in A\}$.

These operations are interdefinable. One can for example define the operations in (3.46)–(3.49) with left cylindrification as follows:

(3.51) $R \lceil A = R \cap A^{c}, \qquad A^{c} = U^{2} \lceil A$

(3.52) $R \rfloor A = R \cap A^{c\smile}, \qquad A^{c} = (U^{2} \rfloor A)^{\smile}$

(3.53) $A \times B = A^{c} \cap B^{c\smile}, \qquad A^{c} = A \times U$

(3.54) ${}^{c}A = A^{c\smile}, \qquad A^{c} = ({}^{c}A)^{\smile}$.

In this section I extend Boolean modules to accommodate also relation-forming operations on sets. The resulting algebras are called *Peirce algebras*. Since the operations of (3.46)–(3.50) are interdefinable it suffices to formalise one of these in Peirce algebras. Peirce algebras were introduced by Britz [1988] in an attempt to accommodate the *extended relation algebras* of Suppes [1976]. (Suppes uses extended relation algebras in the context of computational linguistics to analyse the semantics of fragments of the English language. I return to extended relation algebras in Section 3.5.) A Peirce algebra is essentially a Boolean module $(\mathcal{B}, \mathcal{R}, :)$, endowed with an extra operation from the underlying Boolean algebra $\mathcal{B}$ to the underlying relation algebra $\mathcal{R}$. This operation is the algebraic counterpart to left cylindrification.

(3.55) **Definition** Let $\mathcal{B}$ be a Boolean algebra and $\mathcal{R}$ be a relation algebra. A *Peirce algebra* is a two-sorted algebra $\mathcal{P} = (\mathcal{B}, \mathcal{R}, :, {}^{c})$ with $(\mathcal{B}, \mathcal{R}, :)$ a Boolean module and

${}^c : B \longrightarrow R$ a mapping such that for every $a \in B$ and $r \in R$:

P1 $\quad a^c : 1 = a$

P2 $\quad (r : 1)^c = r \,;1.$

I refer to ^{c} simply as the *cylindrification* operation. The assumed order of precedence (in descending order) is ^{c}, $'$ and ${}^\smile$, then $:$, $;$, $\cdot$ and finally $+$.

The motivating example of a Peirce algebra is what I call the *full Peirce algebra* $\mathcal{P}(U) = (\mathcal{B}(U), \mathcal{R}(U), :, {}^c)$ over a non-empty set U with $(\mathcal{B}(U), \mathcal{R}(U), :)$ the full Boolean module over U and ^{c} the left cylindrification operation on sets defined by (3.50). To verify that full Peirce algebras are indeed Peirce algebras we need to establish that P1 and P2 are true in $\mathcal{P}(U)$. Recall that $\mathrm{dom}(R) = R : U$. The first axiom states that the domain of $A^c = A \times U$ is A, which is true. As for the second axiom, R composed with the universal relation U^2 is the set of (x, y) such that $x \in \mathrm{dom}(R)$ and $y \in U$, that is, $R \,; U^2 = \mathrm{dom}(R) \times U$. Hence by (3.53) $R \,; U^2 = (\mathrm{dom}(R))^c = (R : U)^c$.

Consequently any property true in a Peirce algebra is also true in the calculus of sets and relations interacting with each other. It is not known whether the converse holds. In fact very little is known about universal-algebraic aspects of Peirce algebras. However, (in Section 4.2) I will assume that it is possible to construct *free Peirce algebras*. My assumption is based on results obtained by Birkhoff and Lipson [1970]. They show that many fundamental theorems for homogeneous (or one-sorted) algebras carry over to heterogeneous (or many-sorted) algebras. In particular, Birkhoff and Lipson define and construct *free heterogeneous algebras*. (Unlike Boolean modules Peirce algebras cannot be viewed as homogeneous algebras.)

My main concern is whether the axiomatisation of Peirce algebras is adequate for deriving the basic properties of those relation-forming operations on sets which model terminological operators. In preparation for Chapter 4 I introduce a *restriction* operation $\rfloor$ and a multiplication $\times$ defined by:

P3 $\quad r \rfloor a = r \cdot a^{c\smile}$

P4 $\quad a \times b = a^c \cdot b^{c\smile}.$

By (3.52) and (3.53) the operations $\rfloor$ and $\times$ are the respective algebraic counterparts to the range restriction operation and the Cartesian product.

The next theorem lists a number of arithmetical properties of Peirce algebras. I prove only those not already contained in Britz [1988].

(3.56) **Theorem** In any Peirce algebra $(\mathcal{B}, \mathcal{R}, :, {}^c)$ the following hold for each $a, b \in B$ and $r, s \in R$.

P5 $0^c = 0, \qquad 1^c = 1$

P6 a^c is a right-ideal element, i.e. $a^c ; 1 = a^c$

P7 $(a + b)^c = a^c + b^c$

P8 $a = b$ iff $a^c = b^c$

P9 $a \leq b$ iff $a^c \leq b^c$

P10 $(a \cdot b)^c = a^c \cdot b^c$

P11 $a'^c = a^{c\prime}$

P12 $a^c \cdot e$ is an equivalence element, i.e. $(a^c \cdot e)^\smile = a^c \cdot e$ and $(a^c \cdot e) ; (a^c \cdot e) \leq a^c \cdot e$

P13 $r \cdot a^c = (a^c \cdot e) ; r, \qquad a^c = (a^c \cdot e) ; 1$

P14 $r \cdot a^{c\smile} = r ; (a^c \cdot e), \qquad a^{c\smile} = 1 ; (a^c \cdot e)$

P15 $(a^c \cdot e) : 1 = a$

P16 $(r \cdot a^{c\smile}) : 1 = r : a$

P17 $(r \cdot a^{c\smile}) : b = r : (a \cdot b)$.

Proof. P12 follows by R17 since $a^c \cdot e \ \leq \ e$.

P13 follows immediately by R24 since by P6 a^c is a right ideal element. To establish $a^c = (a^c \cdot e) ; 1$ let $r = 1$.

P14: Using R4, R11, P13, R7 and P12 we get $r \cdot a^{c\smile} = (r \cdot a^{c\smile})^{\smile\smile} = (r^\smile \cdot a^{c\smile\smile})^\smile = (r^\smile \cdot a^c)^\smile = ((a^c \cdot e) ; r^\smile)^\smile = r^{\smile\smile} ; (a^c \cdot e)^\smile = r ; (a^c \cdot e)$. Let $r = 1$ then $a^{c\smile} = 1 ; (a^c \cdot e)$.

P15: By P2 and P13, $((a^c \cdot e) : 1)^c = (a^c \cdot e) ; 1 = a^c$. By applying P8 we then get the required result.

P16 follows by P15, M3 and P14: $r : a = r : ((a^c \cdot e) : 1) = (r ; (a^c \cdot e)) : 1 = (r \cdot a^{c\smile}) : 1$.

P17: Using P16, P10 and R11 we get $r : (a \cdot b) = (r \cdot (a \cdot b)^{c\smile}) : 1 = (r \cdot a^{c\smile} \cdot b^{c\smile}) : 1$. Therefore by P16, $r : (a \cdot b) = (r \cdot a^{c\smile}) : b$. □

Properties P16 and P17 are important properties of the restriction operation (and as we will see in Chapter 4 also of the terminological operator restrict). P16 and P17 can be reformulated as follows:

P16′ $(r \rfloor a) : 1 = r : a$

P17′ $(r \rfloor a) : b = r : (a \cdot b)$.

Using P17′ we can also reformulate M25 in terms of restriction:

M25′ $(r : a')' \cdot ((r \rfloor a) : b')' = (r : (a \cdot b)')'$.

More properties for restriction can be routinely derived using the axioms, Theorem (3.56) and elementary properties in Boolean algebra. I list some without proof.

P18 $r \rfloor (a + b) = r \rfloor a + r \rfloor b$

P19 $(r + s) \rfloor a = r \rfloor a + s \rfloor a$

P20 $r \rfloor 1 = r, \qquad r \rfloor 0 = 0, \qquad 0 \rfloor a = 0$

P21 $1 \rfloor a = a^{c\smile}$

P22 $a \leq b \quad \Rightarrow \quad r \rfloor a \leq r \rfloor b$

P23 $r \leq s \quad \Rightarrow \quad r \rfloor a \leq s \rfloor a$

P24 $(r \cdot s) \rfloor (a \cdot b) = ((r \cdot s) \rfloor a) \rfloor b = (r \rfloor a) \cdot (r \rfloor b)$

P25 $r ; (s \rfloor a) = (r ; s) \rfloor a$.

Just as routinely we can derive properties of $\times$. In Section 4.3 I refer to the following property:

P26 $(a \times b)^{\smile} = b \times a$.

It follows from the definition of $\times$ (in P4) and the fact that $^{\smile}$ is an involution and distributes over $\cdot$ (see R4 and R11).

3.5 Other Applications

The main thrust of this thesis is to show the application of relation-algebraic notions to knowledge representation. However there are also a number of other application areas in Computer Science, and in this section I discuss some of them. In particular, I discuss the work of Suppes [1976] in computational linguistics and the work of Kozen [1980] and Pratt [1979] in the area of logics of programs. Aspects of these are relevant to terminological representation.

In [1976] and other papers [1973, 1979, 1981] Suppes aims at a systematic analysis of the model-theoretic semantics of fragments of natural language. In Suppes [1979, p. 49] he says:

> The central idea is that the syntax of first-order logic is too far removed from that of any natural language, to use it in a sensitive analysis of the meaning of ordinary utterances.

Instead he proposes an algebraic approach, using so-called *extended relation algebras*.

(3.57) **Definition** An *extended relation algebra* $\mathcal{E}(U)$ over a domain U (a non-empty set), is a subset of $\mathbf{2}^U \cup \mathbf{2}^{U^2}$ closed under the operations of union, complementation, converse, composition and image.

Complementation of sets is taken with respect to U and complementation of relations with respect to U^2. In Böttner [1986] an extended relation algebra is also assumed closed under domain restriction.

Note that extended relation algebras are of model-theoretic nature and are not abstract algebras as defined in Definition (3.1). Instead of calling them algebras they are more appropriately thought of as *calculi*, in the same sense as in the preceding sections. As mentioned in Section 3.4, Britz [1988] suggests that extended relation algebras provide standard models for Peirce algebras. This remains open.

With extended relation algebras Suppes characterises the semantics of English language phrases and sentences. The syntax is specified (Chomsky-style) by a grammar G, called a *phrase structure grammar*. The semantics is defined in two steps. First, the grammar G is extended to a so-called (*potentially*) *denoting grammar* by associat-

Figure 3.1: Semantic association in a denoting grammar

	Lexical Production Rule	Semantic Function
(i)	S $\longrightarrow$ NP + VP	$[\mathrm{NP}] \cap [\mathrm{VP}] \neq \emptyset$
(ii)	NP $\longrightarrow$ N	$[\mathrm{NP}] = [\mathrm{N}]$
(iii)	NP $\longrightarrow$ Adj + N	$[\mathrm{NP}] = [\mathrm{Adj}] \cap [\mathrm{N}]$
(iv)	VP $\longrightarrow$ TV + NP	$[\mathrm{VP}] = [\mathrm{TV}] : [\mathrm{NP}]$

ing each production rule of G with a semantic function. This denoting grammar then determines the meaning of phrases and sentences. For example, the semantics of the phrase 'male vegetarian' and the sentence 'Anne admires Charles' are determined by the semantic associations summarised in Figure 3.1. Symbols S, NP, VP, N, Adj and TV denote 'sentence' (or 'start symbol'), 'noun phrase', 'verb phrase', 'noun', 'adjective' and 'transitive verb', respectively. The square brackets indicate the interpretation function. If the adjective 'male' is interpreted as the set of male people and the noun 'vegetarian' as the set of vegetarians, the intersection [male] $\cap$ [vegetarian] defines the meaning of 'male vegetarian'. According to Figure 3.1 (iv) the verb phrase 'admires Charles' is interpreted as the set of admirers of Charles, given by the Peirce product [admire] : [Charles]. (In (iv) Suppes uses the image operation (defined in (3.32)). But recall that image is a variant of Peirce product.) The semantics of the sentence 'Anne admires Charles' is therefore given by

$$[\text{Anne}] \ \cap [\text{admire}] : [\text{Charles}] \ \neq \emptyset.$$

This illustrates how meaning is assigned to a phrase or sentence by converting its grammatic definition (which Suppes views as a grammatic derivation tree) to a semantic definition (which he views as a semantic tree) via the denotational assignments to the production rules which determine the syntax of the phrase or sentence.

In the second step a *model structure* (U, v) is defined for the phrase structure grammar G. U is any non-empty set regarded as the domain or universe and v, called a *valuation*, is a (partial) function from the vocabulary of terminal symbols in G to the extended relation algebra $\mathcal{E}(U)$. That is, v maps terminal symbols to either sets in $\mathbf{2}^U$ or binary relations in $\mathbf{2}^{U^2}$.

Figure 3.2: Interpretation of verb phrases containing quantifier words

	Verb phrase	Interpretation
(i)	eat all fruit	$([\text{eat}]' : [\text{fruit}])'$
(ii)	eat some fruit	$[\text{eat}] : [\text{fruit}]$
(iii)	eat no fruit	$([\text{eat}] : [\text{fruit}])'$
(iv)	do not eat some fruit	$[\text{eat}]' : [\text{fruit}]$

This algebraic approach has the advantage that it is free of variables and quantifiers over variables. Consequently, according to Suppes [1981, p. 405] the analysis of the semantics of natural language fragments can be carried out directly in English, avoiding the translation into another language (e.g., into the first-order language). Furthermore, it allows the development of a syntactic derivation system for direct inference in the English language. (I won't elaborate on this system, but see Suppes [1981].)

Since Suppes translates English language phrases and sentences as algebraic expressions, which as we will see in Chapter 4 can be associated with terminological expressions, his work is relevant to the problem of finding adequate terminological representations for information formulated in English and vice versa. In [1981] Suppes demonstrates how phrases and sentences with quantifier words (such as 'all', 'some' and 'no') in object and subject position are interpreted in the framework of extended relation algebras. For example, the verb phrases listed in Figure 3.2 are interpreted by variants of the image operation, here appropriately translated as variants of Peirce product. When each of these verb phrases is combined with quantified subjects the semantics of the resulting sentences is of the form similar to that of the sentences given in Figure 3.3.

Also relevant to terminological representation (in particular to the interpretation of the all construct) is the semantics of phrases of the form:

(3.58) 'eat only fruit'.

Böttner [1985] interprets this phrase by $[\text{eat}] : [\text{fruit}] - [\text{eat}] : [\text{fruit}]'$, or equivalently:

(3.59) $[\text{eat}] : [\text{fruit}] \ \cap \ ([\text{eat}] : [\text{fruit}]')'$.

Figure 3.3: Interpretation of sentences containing the word 'all'

	Sentence	Interpretation
(i)	Some persons eat all fruit	$[\text{persons}] \cap ([\text{eat}]' : [\text{fruit}])' \neq \emptyset$
(ii)	All persons eat all fruit	$[\text{persons}] \subseteq ([\text{eat}]' : [\text{fruit}])'$
(iii)	No person eats all fruit	$[\text{persons}] \cap ([\text{eat}]' : [\text{fruit}])' = \emptyset$

As Böttner pointed out in [1990], $([\text{eat}] : [\text{fruit}]')'$ alone inadequately interprets (3.58). If 'eat only fruit' were to be interpreted as $([\text{eat}] : [\text{fruit}]')'$ one would not be able to deduce that persons who eat only fruit are also persons who eat (some) fruit, since in general

(3.60) $([\text{eat}] : [\text{fruit}]')' \not\subseteq [\text{eat}] : [\text{fruit}]$.

(For suppose [fruit] is empty. Then $[\text{eat}] : [\text{fruit}]$ is empty (by M11), but $([\text{eat}] : [\text{fruit}]')'$ is not necessarily empty, since $([\text{eat}] : \emptyset')' = [\text{eat}] : U = (\text{dom}([\text{eat}]))'$ by (3.33).) By M23 and (3.33) we have

(3.61) $([\text{eat}] : [\text{fruit}]')' \subseteq [\text{eat}] : [\text{fruit}]$ iff $\text{dom}([\text{eat}]) = \text{U}$.

But to decree that the domain of each relation must be the entire universe of discourse does not seem feasible. (For example, we would not want to include the instances of [fruit] in the domain of [eat].) However the interpretation (3.59) suggested by Böttner is contained in $[\text{eat}] : [\text{fruit}]$, ensuring that persons eating only fruit also eat some fruit.

In the paper [1985] Böttner not only analyses the semantics of sentences like 'John loves only Mary' with 'only' in object position, but also of sentences like 'Only John loves Mary' and also like 'All boys except John love Mary'. In other papers [1989, 1986] he investigates the algebraic interpretation of anaphoric expressions and English imperatives. (An expression is *anaphoric* if it refers back to earlier contexts as in 'John loves himself', 'John and Mary like each other' and 'John likes his toys'.)

Besides being relevant to this thesis as regards representing knowledge in a calculus of sets and relations, the work of Suppes and Böttner (in particular Böttner's analysis of English imperatives in [1986]) also relate to the algebraic side of a certain logic of programs, called *dynamic logic*, introduced by Pratt [1976*]. (For a survey of dynamic logic refer to Harel [1984] and Parikh [1981].) Dynamic logic is a vehicle for reasoning about program characteristics such as correctness, termination and equivalence. It provides a formalism for studying assertions about the state of programs before and after execution. (A *state* of a program is an assignment of values to program variables.) Dynamic logic can thus be viewed as a logic of propositions acted upon by programs. The algebraic versions of propositional dynamic logic are *dynamic algebras*, introduced by Kozen [1980] and Pratt [1979]. In a dynamic algebra the propositions are presented in a Boolean algebra and the programs in a *Kleene algebra*.

Kleene algebras are meant to formalise a calculus of (non-deterministic) programs. Programs can be interpreted as binary relations over the sets of possible input and output states. In this interpretation the program denotations form a calculus of relations with basic operations union $\cup$, composition $;$ and a reflexive transitive closure operation *, called the *Kleene closure*. Let α and β denote programs which are interpreted as relations R and S respectively. Then

(i) 'do α or β' (non-deterministic choice) is interpreted by $R \cup S$,

(ii) 'do α then do β' (sequence) by $R\,;S$, and

(iii) 'do α zero or more times' (iteration) by R^*,

where R^* is defined by:

(3.62) $R^* = Id \ \cup \ R \ \cup \ R\,;R \ \cup \ R\,;R\,;R \ \cup \ \ldots$.

Accordingly a *Kleene algebra* is defined as an algebra $\mathcal{K} = (K, +, 0, \,;, ^*, e)$ with a join $+$, a relative composition operation $;$ and a *star* operation * (these essentially being the respective algebraic counterparts to (i)–(iii) above) satisfying a set of equational axioms. For a definition of Kleene algebras refer to Kozen [1980]. In the standard models of Kleene algebras the star operation is a reflexive transitive closure operation (see Kozen [1980, p. 356]) that has the following property: for any $r \in K$

(3.63) $r^* = \sum_{n=0}^{\infty} r^n$

where $\sum$ is taken with respect to $+$ and r^n is defined by:

(3.64) $r^0 = e \quad \text{and} \quad r^{n+1} = r\,;r^n \quad (\text{for } n \geq 0)$.

The difference between Kleene algebras and relation algebras is that Kleene algebras need not form Boolean algebras (note the absence of a meet and a complementation operation), and they do not include a converse operation (although some definitions do, e.g., the one in Pratt [1990a]). Also, Kleene algebras have a star operation which the relation algebras defined in Section 3.2 do not. However *relation algebras with transitive closure* have been defined by Ng in [1984*] and together with Tarski in [1977].

A *dynamic algebra* is then a two-sorted algebra $(\mathcal{B}, \mathcal{K}, \diamond)$, with $\mathcal{B}$ a Boolean algebra, $\mathcal{K}$ a Kleene algebra and $\diamond$ a multiplication over the Boolean algebra from the Kleene algebra. This multiplication satisfies certain equational axioms which characterise the interaction between propositions and programs. (The axioms of $\diamond$ are similar to those satisfied by Peirce product in Boolean modules, but also accommodate the star operation). A definition of dynamic algebras can be found in Kozen [1981].

Work in this field continues. In a recent paper Pratt [1990a] discusses dynamic algebras in relation to relation algebras. In [1990b] he introduces more powerful structures than Kleene algebras but which are weaker than the Ng-Tarski relation algebras with transitive closure. These new algebras are called *action algebras*. (Interestingly, residuation plays an important role in these.) Recent work by Kozen on Kleene algebras appears in [1990a*] and [1990b*]. In a 'very preliminary draft' Jónsson [Draft] tentatively outlines another algebraic treatment of programs and program specification which is more extensively based on universal-algebraic notions.

In conclusion I list some references to other applications of the algebra of relations. Schmidt and Ströhlein [1985] consider some requirements for relation algebra applied in the (relational) theory of graphs and programs. Maddux [1983] presents a sequent calculus for the calculus of relations, and Wadge [1975] and Hennessy [1980] develop natural deduction systems for the calculus of relations. Further references can be found in Brink [1988], which discusses the history of relations as well as their applications in Boolean modules, extended relation algebras and dynamic algebras.

Chapter 4

Algebraic Terminological Representation

In Chapter 2 I discussed terminological representation formalisms and defined two typical terminological languages, and in Chapter 3 I gave an overview of algebras of sets and relations. I this chapter I relate these two topics. First, I accommodate the model-theoretic semantics of terminological languages in the algebraic framework. With the exception of the number restriction operators each terminological operator can be expressed algebraically. This enables me to use the algebraic apparatus of Boolean algebras, relation algebras, Boolean modules and Peirce algebras as an inference mechanism for reasoning about concepts and roles. I then demonstrate the algebraic approach with a number of case studies.This motivates my claim that terminological reasoning can be handled equationally.

4.1 Algebraic Semantics

In this section I show how the semantics of a terminological language can be presented algebraically. The language I treat is a sublanguage of $\mathcal{U}$ which I will call $\mathcal{U}^-$, obtained by dropping the number restriction constructs atleast and atmost. Its model-theoretic semantics was discussed at some length in Section 2.2. The algebraic apparatus I use is that of Chapter 3. There I presented Boolean algebras, relation algebras, Boolean modules and Peirce algebras as formalisations of different operations on sets and relations.

As before an interpretation $\mathcal{I}$ of $\mathcal{U}^-$ is a pair $(U, \cdot^{\mathcal{I}})$ with U ($= \mathcal{D}^{\mathcal{I}}$) the universe of interpretation and $\cdot^{\mathcal{I}}$ the interpretation function. A concept C is interpreted as a set $\mathsf{C}^{\mathcal{I}} \subseteq U$ and a role R as a binary relation $\mathsf{R}^{\mathcal{I}}$ over the set U. Instead of defining the constraints on $\cdot^{\mathcal{I}}$ model-theoretically I will here define the constraints in the algebraic context. To emphasise this context, I use the notation of Chapter 3 and abbreviate $\mathsf{C}^{\mathcal{I}}$, $\mathsf{D}^{\mathcal{I}}, \ldots$ and $\mathsf{R}^{\mathcal{I}}$, $\mathsf{S}^{\mathcal{I}}, \ldots$ by C, D, $\ldots$ and R, S, $\ldots$, respectively. I follow quite closely the order of exposition of Section 2.2, to which the reader should refer.

The interpretation of the concept descriptions defined in (2.11) (with 'Q' replaced by 'R') can be rewritten as follows:

(4.1)
$$\begin{aligned}
\top^{\mathcal{I}} &= U \\
\bot^{\mathcal{I}} &= \emptyset \\
(\mathsf{and\ C\ D})^{\mathcal{I}} &= C \cap D \\
(\mathsf{or\ C\ D})^{\mathcal{I}} &= C \cup D \\
(\mathsf{not\ C})^{\mathcal{I}} &= C' \\
(\mathsf{some\ R\ C})^{\mathcal{I}} &= R\,{:}\,C \\
(\mathsf{all\ R\ C})^{\mathcal{I}} &= (R\,{:}\,C')'.
\end{aligned}$$

The designated top and bottom concepts ($\top$ and $\bot$) and the Boolean operators (**and**, **or** and **not**) are defined as before. The **some** operator is assigned to the Peirce product (defined in (3.31)) and the **all** operator is assigned to that variant of Peirce product which I considered in Theorem (3.41) (i).

The interpretation of the role value map as defined in (2.25) can be reformulated in terms of Peirce product (or the domain operation defined in (3.29)) as follows:

(4.2) $$(\mathsf{rvm\ R\ S})^{\mathcal{I}} = ((R \cap S')\,{:}\,U)' \qquad (= (\mathrm{dom}(R \cap S'))').$$

Proof. $(x, y) \in (\mathsf{rvm\ R\ S})^{\mathcal{I}}$ iff $(\forall y)[(x, y) \in R \Rightarrow (x, y) \in S]$ (by (2.25)) iff $(\forall y)[(x, y) \notin R$ or $(x, y) \in S]$ iff $\neg(\exists y)[(x, y) \in R \;\&\; (x, y) \notin S]$ iff $\neg(\exists y)[(x, y) \in R \cap S']$ iff $x \notin \mathrm{dom}(R \cap S')$ (by (3.29)) iff $x \in (\mathrm{dom}(R \cap S'))'$ iff $x \in ((R \cap S')\,{:}\,U)'$ (by (3.33)). $\square$

As is apparent from the definition of Peirce product in (3.31) and the definition of structural description in (2.25) its new formulation is given by:

$$(4.3) \qquad (\textsf{sd C Rb}_1 \ldots \textsf{Rb}_k)^{\mathcal{I}} \;=\; (\bigcap_{i=1}^{k} Rb_i):C$$

(where Rb_i abbreviates $\textsf{Rb}_i{}^{\mathcal{I}}$). Recall that the role bindings $\textsf{Rb}_i$ have one of two forms: $(\subseteq \textsf{R S})$ or $(\supseteq \textsf{R S})$. It is immediate by (2.26) and Theorem (3.26) (ii) that the semantics of role bindings in the form $(\supseteq \textsf{R S})$ coincides with the definition of a left residual in the calculus of relations. Below I prove that role bindings in the form $(\subseteq \textsf{R S})$ can be expressed as right residuals. Because the residuals can be defined (by R20) in terms of relational composition and conversion the role bindings can now be formulated as follows:

$$\begin{array}{llll} (4.4) & (\subseteq \textsf{R S})^{\mathcal{I}} = (R\,;S^{\smile\prime})' & & (= R^{\smile}\backslash S^{\smile}) \\ & (\supseteq \textsf{R S})^{\mathcal{I}} = (R'\,;S^{\smile})' & & (= R/S). \end{array}$$

Proof. (Of the formulation of $(\subseteq \textsf{R S})^{\mathcal{I}}$.) $(x,y) \in (\subseteq \textsf{R S})^{\mathcal{I}}$ iff $(\forall z)[(x,z) \in R \Rightarrow (y,z) \in S]$ (by (2.26)) iff $(\forall z)[(z,x) \in R^{\smile} \Rightarrow (z,y) \in S^{\smile}]$ iff $(x,y) \in R^{\smile}\backslash S^{\smile}$ (by Theorem (3.26) (i)) iff $(x,y) \in (R^{\smile\smile}\,;S^{\smile\prime})'$ (by R20) iff $(x,y) \in (R\,;S^{\smile\prime})'$ (by R4). □

Next I reformulate the interpretation of the role descriptions as defined in (2.27). The model-theoretic semantics of the identity role **self** coincides with the definition (given in (3.15)) of the identity relation Id. The Boolean operators **and**, **or** and **not** (applied this time to roles) are defined as before. As is apparent from (3.14) and (3.13) the **inverse** and **compose** operator can be equivalently defined with conversion $^{\smile}$ and relational composition ;, respectively. Thus

$$\begin{array}{lrcl} (4.5) & \textsf{self}^{\mathcal{I}} & = & Id \\ & (\textsf{and R S})^{\mathcal{I}} & = & R \cap S \\ & (\textsf{or R S})^{\mathcal{I}} & = & R \cup S \\ & (\textsf{not R})^{\mathcal{I}} & = & R' \\ & (\textsf{inverse R})^{\mathcal{I}} & = & R^{\smile} \\ & (\textsf{compose R S})^{\mathcal{I}} & = & R\,;S. \end{array}$$

The interpretation of (**trans** R) (given in (2.27)) can be characterised by a recursive definition:

$$(4.6) \qquad (\textsf{trans R})^{\mathcal{I}} \;=\; R \;\cup\; R\,;(\textsf{trans R})^{\mathcal{I}}.$$

This unfolds to

$$(4.7) \qquad (\mathsf{trans\ R})^{\mathcal{I}} = R \cup R\,;R \cup R\,;R\,;R \cup \ldots = \bigcup_{n=1}^{\infty} R^n$$

where $R^1 = R$ and $R^{n+1} = R\,;R^n$. (Note that $\bigcup_{n=1}^{\infty} R^n = R\,;R^*$ where R^* is the Kleene closure, that is, the reflexive transitive closure, defined by (3.62).) The restrict operator can be formulated (by (3.47)) as range restriction or (by (3.52)) it can be formulated using (left) cylindrification. Namely:

$$(4.8) \qquad (\mathsf{restrict\ R\ C})^{\mathcal{I}} = R \cap C^{c\smile} \qquad (= R \rfloor C).$$

It is feasible to define further designated roles and operators in $\mathcal{U}^-$. These are the top and bottom roles ($\nabla$and Λ) defined as on page 31 and the domain and range operators defined as in (2.44). Their semantics is given by

$$(4.9) \qquad \nabla^{\mathcal{I}} = U^2, \qquad \Lambda^{\mathcal{I}} = \emptyset$$

and

$$(4.10) \qquad (\mathsf{domain\ C})^{\mathcal{I}} = C^c \qquad (= C \times U)$$
$$(\mathsf{range\ C})^{\mathcal{I}} = C^{c\smile} \qquad (= U \times C).$$

Proof. (Of (4.10).) Using (2.44), (4.5), (4.8), (4.9), R4 and (3.53) we get: $(\mathsf{domain\ C})^{\mathcal{I}} = (\mathsf{inverse\ (restrict\ \nabla\ C)})^{\mathcal{I}} = (U^2 \cap C^{c\smile})^{\smile} = C^{c\smile\smile} = C^c = C \times U$.
Analogously $(\mathsf{range\ C})^{\mathcal{I}} = (\mathsf{restrict\ \nabla\ C})^{\mathcal{I}} = U^2 \cap C^{c\smile} = C^{c\smile} = U \times C$. □

By P6 (domain C) is therefore interpreted as a right-ideal element, and since the converse of a right-ideal element is a left-ideal element (for a proof see Chin and Tarski [1951]), (range C) is interpreted as a left-ideal element. This verifies an earlier remark in Section 3.2 (page 50).

The interpretation of the terminological axioms used to specify specialisation and equivalence relations between concepts and roles is defined as in Section 2.2. Namely:

$$(4.11) \qquad \models_{\mathcal{I}} \mathsf{C} \sqsubseteq \mathsf{D} \ \text{ iff } \ C \subseteq D$$
$$\models_{\mathcal{I}} \mathsf{C} \doteq \mathsf{D} \ \text{ iff } \ C = D,$$

and likewise for specialisations and equivalences between roles.

I have thus shown that the model-theoretic semantics of terminological expressions in $\mathcal{U}^-$ can be formulated in terms of constants and operations in the calculus of sets and relations interacting with each other. These constants and operations have algebraic counterparts in the algebras presented in Chapter 3 which capture (or attempt to capture) their corresponding calculi. Each terminological expression can therefore be directly associated with algebraic terms as summarised in Figure 4.1. The figure lists the different terminological expressions, their respective interpretations (derived above) as well as their associated algebraic formulations. (The term $\prod_{i=1}^{k} r_i$ used in (iii) denotes the product $r_1 \cdot r_2 \cdot \ldots \cdot r_k$.)

I now discuss in more detail the associations of the different kinds of terminological operators with algebraic operations. From primitive ones, new concepts and roles arise by using one of four kinds of operators:

(i) **Concept-forming operators on concepts:** These are the Boolean operators and, or and not. Each operator is assigned to a set-forming operation on sets (see Figure 4.1 (i)) and is thus catered for in the calculus of sets. Since Boolean algebra captures the calculus of sets these concept-forming operators can be captured in the context of Boolean algebra.

(ii) **Role-forming operators on roles:** These are the operators listed (with the designated roles) in category (ii) of Figure 4.1. Their interpretations are defined with the relation-forming operations in the calculus of relations. Relation algebras thus cater for these role-forming operators in the same way as they cater for the calculus of relations. (As pointed out in Section 2.2, relation algebras do not fully capture the calculus of relations.)

(iii) **Concept-forming operators on concepts and roles:** Their semantics is accommodated in the calculus of sets which interact with relations through Peirce product. Hence in the same way as Boolean modules cater for this calculus, Boolean modules also cater for the operators some, all, rvm and sd.

(iv) **Role-forming operators on concepts and roles:** These operators are interpreted in the calculus of relations interacting with sets as formalised by Peirce algebra. That is, the properties of domain, range and restrict are formalised in Peirce algebra.

Note that the terminological constants have algebraic counterparts as well. The des-

Figure 4.1: Algebraic Semantics of $\mathcal{U}^-$

	Terminological expression	Interpretation	Algebraic term
(i)	$\top$	U	1
	$\bot$	$\emptyset$	0
	(and C D)	$C \cap D$	$a \cdot b$
	(or C D)	$C \cup D$	$a + b$
	(not C)	C'	a'
(ii)	∇	U^2	1
	Λ	$\emptyset$	0
	self	Id	e
	(and R S)	$R \cap S$	$r \cdot s$
	(or R S)	$R \cup S$	$r + s$
	(not R)	R'	r'
	(inverse R)	$R^{\smile}$	$r^{\smile}$
	(compose R S)	$R\,;S$	$r\,;s$
	(trans R)	$\bigcup_{n=1}^{\infty} R^n$	$\sum_{n=1}^{\infty} r^n$
	($\subseteq$ R S)	$(R\,;S^{\smile\prime})' = R^{\smile}\backslash S^{\smile}$	$(r\,;s^{\smile\prime})' = r^{\smile}\backslash s^{\smile}$
	($\supseteq$ R S)	$(R'\,;S^{\smile})' = R/S$	$(r'\,;s^{\smile})' = r/s$
(iii)	(some R C)	$R:C$	$r:a$
	(all R C)	$(R:C')'$	$(r:a')'$
	(rvm R S)	$((R \cap S'):U)'$	$((r \cdot s'):1)'$
	(sd C $\mathsf{Rb}_1 \ldots \mathsf{Rb}_k$)	$(\bigcap_{i=1}^{k} Rb_i):C$	$(\prod_{i=1}^{k} r_i):a$
(iv)	(domain C)	C^c	a^c
	(range C)	$C^{c\smile}$	$a^{c\smile}$
	(restrict R C)	$R \cap C^{c\smile} = R\rfloor C$	$r \cdot a^{c\smile} = r\rfloor a$

ignated top and bottom concepts $\top$ and $\bot$ are associated with the top and bottom elements 1 and 0, of the partial order in a Boolean algebra. The designated top and bottom roles ∇ and Λ are similary associated with 1 and 0, this time in a relation algebra. The constant corresponding to the third designated role self is the identity element e in a relation algebra. The corresponding relationships to $\sqsubseteq$ and $\doteq$ expressions are inclusions and equations in the relevant algebras.

With one exception every algebraic operation introduced in Chapter 3 can be associated with a terminological operator. The exception is the $\times$ operation defined in Section 3.4 for Peirce algebras. I will use this operation in Section 4.3.

To conclude this section I illustrate how the algebraic properties relate to the semantic theory of $\mathcal{U}^-$. It is an axiom (R4) in relation algebra that conversion is an involution. Formally:

(4.12) $\quad r^{\smile\smile} = r \qquad$ for every element r in a relation algebra.

In the calculus of relations a corresponding identity is satisfied (see (3.17)). This implies that for any role description R the associated terminological statement

(4.13) (inverse (inverse R)) $\doteq$ R

is satisfied in any interpretation $\mathcal{I}$ of $\mathcal{U}^-$ (since $(\mathsf{inverse}\ (\mathsf{inverse}\ \mathsf{R}))^{\mathcal{I}} = (\mathsf{R}^{\mathcal{I}})^{\smile\smile} = \mathsf{R}^{\mathcal{I}}$). This in turn implies that (4.13) is valid (in symbols, $\models$ (4.13)). Hence

(4.14) $\quad$ (inverse (inverse R)) $\approx$ R $\qquad$ for any role description R.

This transformation from an arithmetical identity like (4.12) to a semantic equivalence relation like (4.14) works for each universal identity in relation algebra. (By a universal identity I mean an equational axiom or equational property of relation algebra.) Every such universal identity thus determines a semantic equivalence relation between roles in $\mathcal{U}^-$. Although an inclusion like $r \leq 1$ is an implicit equation and determines a semantic equivalence, it can be seen to determine the subsumption relationship R $\preceq$ ∇. Analogously we can show that any universal identity in Boolean algebras, Boolean modules and Peirce algebras appropriately determine $\approx$ expressions for $\mathcal{U}^-$.

4.2 Algebraic Reasoning

In this section I propose an *algebraic* approach to *reasoning about* terminological expressions formulated in the language $\mathcal{U}^-$. My proposal is based on the algebraic formulations presented in Section 4.1. There I showed that concept descriptions can be regarded as forming a *Boolean algebra* and role descriptions as forming a *relation algebra*. Furthermore, concepts interacting with roles form, depending on the operators, a *Boolean module* or a *Peirce algebra*. I propose to make use of the arithmetic of these algebras to calculate inferences phrased in $\mathcal{U}^-$.

To illustrate my approach I use in the first instance the core example presented in the Preview. The semantic network in Figure 1.1 contains some explicit information (such as (1.1)–(1.5)) but it also contains some implicit information (such as (1.6)–(1.11)). The semantic network is a (graphic) representation of explicit facts specified by the user, formulated in a terminological language such as $\mathcal{U}^-$. Recall that these explicit facts are formulated as *terminological axioms*, each of which is a $\sqsubseteq$ or $\doteq$ expression. The set of terminological axioms makes up the *terminology*, and this corresponds to the semantic diagram. The aim now is to extract by some inference mechanism also knowledge *implicit* in the diagram, or terminology.

I now describe the algebraic method I propose for computing such inferences. Recall (from Chapter 1 and Section 2.2) that the concepts in a terminology are ordered with respect to the subsumption relation and form a poset, called the *concept taxonomy*. The set of concepts in the semantic network of Figure 1.1 form the concept taxonomy depicted in Figure 4.2. Analogously the set of roles forms a poset called the *role taxonomy*, which for the sample network is depicted in Figure 4.3. Thus the first step is to separate the concept taxonomy from the role taxonomy.

The next step involves the generation of free algebras. The idea is that the given primitive concepts and roles are used as generators for constructing compound concepts and roles. Consider first the concept taxonomy. New concepts are generated by combining the given primitive ones using the Boolean operations. That is, every pair of concepts generates both a meet and a join, and every concept generates a concept as its complement. In this way we generate many new concepts. For example, from the

Figure 4.2: Concept poset

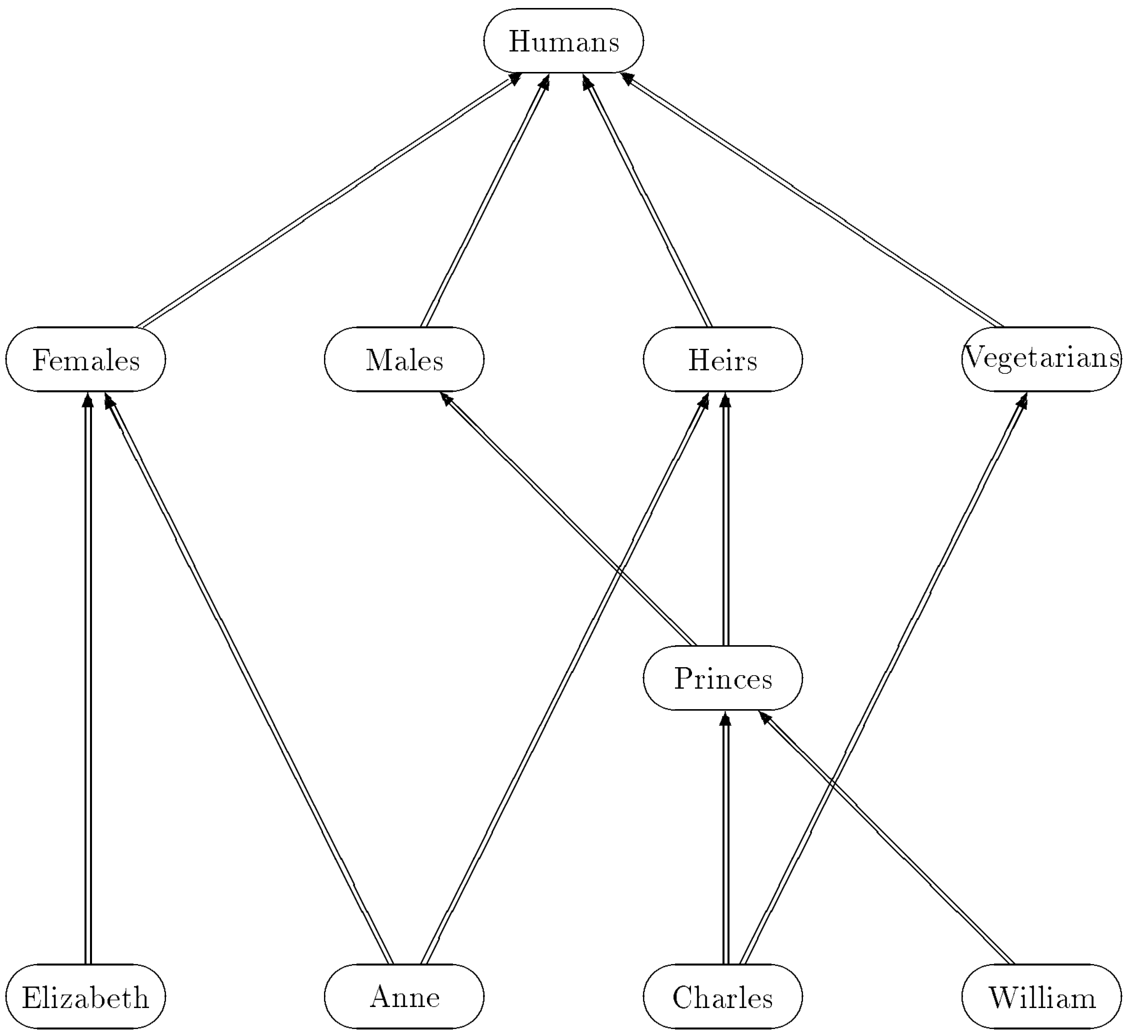
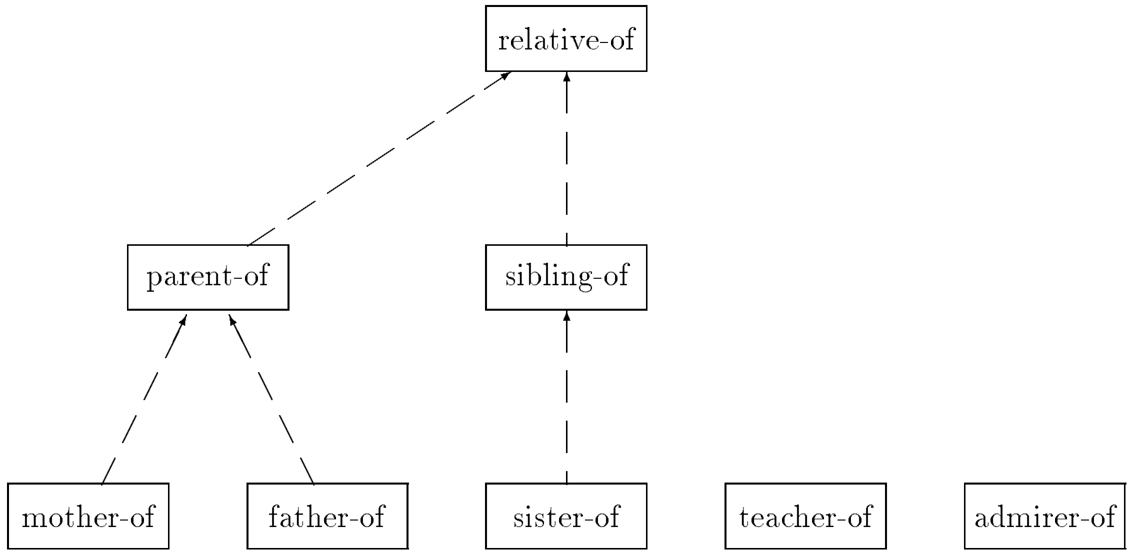


Figure 4.3: Role poset



concepts in Figure 4.2 we generate 'male heirs to the throne' as the meet of 'males' and 'heirs to the throne', 'females and vegetarians' as the join of 'females' and 'vegetarians', 'not princes' as the complement of 'princes' and so on. This generation is constrained by the axioms of Boolean algebra. For example, since join is commutative (by B2) the concept 'vegetarians and females' will not be generated in addition to the concept 'females and vegetarians'. Since 'humans' is the top concept in our example it generates the bottom concept as its complement, in accordance with the property B13 in Boolean algebras that the complement of the unit coincides with the zero. The user can assign certain names to compound concepts by specifying appropriate equivalences. For example he/she may want to call the bottom concept 'nobody'. This would be facilitated if the expression $\mathsf{Nobody} \doteq (\mathsf{not\ Humans})$ is added to the terminology. Like the axioms, user constraints also limit the number of concepts being generated. Since princes are male (formulated as $\mathsf{Princes} \sqsubseteq \mathsf{Males}$), for example, the concept 'male prince' (the meet of 'males' and 'princes') coincides with 'princes' and will therefore not be generated. In this way we generate a *free Boolean algebra* of concept descriptions. These are the concept descriptions formulated in $\mathcal{U}^-$ with the and, or and not operators.

As is the case in general, from n atomic concepts we would generate a Boolean algebra of 2^n concepts.

Consider next the role taxonomy. As for concepts we can freely generate a Boolean algebra of role descriptions from the given primitive roles (including the designated identity role self). Using the Boolean operations we generate new roles such as 'both sister of and admirer of', 'mother of or teacher of', 'not sibling of' etc. Again this generation is in accordance with the axioms of Boolean algebra and possible user constraints. The user may wish to specify that the roles 'mother of' and 'father of' are mutually exclusive and exhaust the role 'parent of'. A way of generating the top role is as the join of self and its complement. We obtain then the bottom role analogous to how we did the bottom concept. With roles we go further than with concepts. We use the *relational operations* of converse and composition to generate additional roles as constrained by the axioms of relation algebra and the relevant user constraints. In this way we generate a *free relation algebra* of role descriptions. Roles like the following will be generated: 'has as parent' as the converse of 'parent of', 'mother of parent of' as the composition of 'mother of' and 'parent of', 'has not as teacher' as the complement of the converse of 'teacher of' and so on. Again the user can assign certain names to roles. For example, he/she may want to define 'child of' as 'has as parent', 'grandmother of' as 'mother of parent of' and 'pupil of' as 'has as teacher'. The role descriptions thus generated are those defined in $\mathcal{U}^-$ with the Boolean operators and the relational operators inverse and compose. Observe that roles expressed with the trans operator and the role binding constructs are implicitly generated in terms of the relational operators. Unlike Boolean algebras freely generated from a finite set of elements, freely generated relation algebras are in general not finite. For example, from the standard example we obtain 'parent of', 'grandparent of', 'great-grandparent of', ...

As a third step we freely generate all the *interactions between concepts and roles that yield concepts*. That is, using Peirce product we freely generate a *Boolean module* over the Boolean algebra of concepts and a relation algebra of roles. In this way we generate additional concepts such as 'mother of Charles' or 'sister of (some) princes'. The generation will also yield variants of Peirce product such as 'fathers of no females', 'not pupils of Anne', 'admirers of all princes' and 'teachers only of heirs to the throne'.

Other variants generated include those concepts which are represented in $\mathcal{U}^-$ as role value maps and structural descriptions. Examples are 'relatives of all those of whom they are an aunt', 'admirers of all those of whom they are a child', 'humans with brothers' and 'princes with siblings'.

The fourth and final step consists of the free generation of all *interactions between concepts and roles yielding roles*. We use the cylindrification operation to generate a *Peirce algebra* over the Boolean module of concepts and roles. Recall that I used cylindrification to model the restrict, domain and role operators of $\mathcal{U}^-$. The free generation thus yields new roles such as 'being a mother of heirs to the throne' or 'being a sibling of males' which are represented in $\mathcal{U}^-$ as (restrict mother-of Heirs) and (restrict sibling-of Males), respectively. We could also generate roles expressed with domain and range which do not necessarily have a natural English translation.

I have thus shown how implicit relationships between concepts and roles can *in principle* be generated. An implementation cannot attempt to do free generation of entire algebras since the free algebras are in general infinite. I envisage that in practice only part of the relevant algebra is generated in response to a user query, namely that part which will be sufficient for answering the query.

4.3 Case Studies

This section is devoted to a number of case studies of deducing implicit knowledge from explicitly given facts within the equational framework of the algebras of Chapter 3. We have already seen in Section 4.1 how terminological claims can be formulated with algebraic operators. On the basis of Section 4.2 I assume that new concepts and roles are generated as required. Deriving further claims will thus amount to proof along the lines of those in Chapter 3. However, for ease of exposition I will present these in an informal mixture of English and algebraic operators.

Still keeping to the standard example of Figure 1.1 in Chapter 1, I will show how the implicit facts listed in (1.6)–(1.11) can be derived equationally. In (4.15)–(4.20) below I list these claims again. There are three parts to each claim:

(i) the English formulation (given in (1.6)–(1.11)),

(ii) the terminological formulations (given in (1.18)–(1.23)), and

(iii) the algebraic formulations (which I will prove).

(4.15) (i) Elizabeth is human

(ii) $\mathsf{Elizabeth} \sqsubseteq \mathsf{Humans}$

(iii) $\text{Elizabeth} \leq \text{Humans}$

(4.16) (i) All sisters of someone are relatives of that person

(ii) $\mathsf{sister\text{-}of} \sqsubseteq \mathsf{relative\text{-}of}$

(iii) $\text{sister-of} \leq \text{relative-of}$

(4.17) (i) Charles is a father of some prince

(ii) $\mathsf{Charles} \sqsubseteq (\mathsf{some\ father\text{-}of\ Princes})$

(iii) $\text{Charles} \leq \text{father-of}:\text{Princes}$

(4.18) (i) William is a child of Charles

(ii) $\mathsf{William} \sqsubseteq (\mathsf{some\ (inverse\ parent\text{-}of)\ Charles})$

(iii) $\text{William}\times\text{Charles} \leq \text{child-of}$ (where $\text{child-of} = \text{parent-of}^{\smile}$)

(4.19) (i) Anne is an aunt of some prince

(ii) $\mathsf{Anne} \sqsubseteq (\mathsf{some\ (compose\ sister\text{-}of\ parent\text{-}of)\ Princes})$

(iii) $\text{Anne} \leq \text{aunt-of}:\text{Princes}$ (where $\text{aunt-of} = \text{sister-of};\text{parent-of}$)

(4.20) (i) Some vegetarian is a parent of William

(ii) (and Vegetarians (some parent-of William)) $\neq \perp$

(iii) Vegetarians $\cdot$ (parent-of : William) $\neq 0$.

The reader is reminded that inclusions (that is, $\leq$ expressions) are in fact disguised equations. To handle *algebraic* inequalities like (iii) in (4.20) I make use of the properties of *simple Boolean modules*, discussed in Section 3.3. In particular, I use Theorem (3.45), by which in a simple Boolean module $a \neq 0$ iff $1 : a = 1$ (for any Boolean element a).

The *terminological* inequality (ii) in (4.20) is strictly speaking not well-formulated in $\mathcal{U}^-$ (as defined in Section 2.2). Neither is the earlier example given in (1.17):

(4.21) (and Females (some sibling-of Males)) $\doteq\!\!\!\!\!/ \ \perp$.

To cater for such inequalities I assume that any model of a terminology forms a full Peirce algebra (over some non-empty set U). Its underlying Boolean module is simple, because the underlying Boolean module of a full Peirce algebra is full and every full Boolean module is simple (refer to page 58). Then again according Theorem (3.45), we may regard a terminological expression of the form C $\doteq\!\!\!\!\!/ \ \perp$ as an abbreviation of the expression (some ∇ C) $\doteq \top$. In particular, then, (4.20) (ii) and (4.21) can be regarded as respectively abbreviating:

(4.22) (some ∇ (and Vegetarians (some parent-of William))) $\doteq \top$

(4.23) (some ∇ (and Females (some sibling-of Males))) $\doteq \top$.

The same method works also for role inequalities. By Theorem (3.28), in a simple relation algebra $r \neq 0$ iff $1 ; r ; 1 = 1$ (for every element r). Hence we may in the terminology regard claims of the form R $\doteq\!\!\!\!\!/ \ \Lambda$ as abbreviations of (compose (compose ∇ R) ∇) $\doteq$ ∇.

To infer the algebraic formulations of the claims in (4.15)–(4.20) I therefore use the arithmetic of *Peirce algebras with simple underlying Boolean modules.*

(4.15)′ By (1.1) and (1.2) we know that Elizabeth $\leq$ Females and Females $\leq$ Humans. In a Boolean algebra $\leq$ is a transitive relation. Hence Elizabeth $\leq$ Humans.

(4.16)′ Analogous to (4.15)′. Since sister-of $\leq$ sibling-of and sibling-of $\leq$ relative-of is given, we obtain sister-of $\leq$ relative-of by transitivity of $\leq$.

(4.17)′ William is a prince, i.e. William $\leq$ Princes, implies father-of : William $\leq$

father-of : Princes using M7. Therefore since Charles $\leq$ father-of : William by (1.4), Charles $\leq$ father-of : Princes.

(4.18)′ In (4.17)′ we expressed that Charles is a father of William in terms of Peirce product. It can also be expressed as Charles$\times$William $\leq$ father-of. Hence since father-of $\leq$ parent-of, Charles$\times$William $\leq$ parent-of. By R4, P26 and R10 it follows then that William$\times$Charles $=$ (William$\times$Charles)$^{\smile\smile}$ $=$ (Charles$\times$William)$^{\smile}$ $\leq$ parent-of$^{\smile}$ $=$ child-of. Observe that we could have expressed the claim which we set out to prove by William $\leq$ child-of : Charles. To derive the claim in this form would require some properties of Peirce product applied to atoms.

(4.19)′ There is more than one way of proving this. One possibility is:

Anne $\leq$ sister-of : Charles (since 'Anne is a sister of Charles' is given)
$\leq$ sister-of : (father-of : Princes) (by (1.8) and M7)
$\leq$ sister-of : (parent-of : Princes) (since father-of $\leq$ parent-of, M8 and by M7)
$=$ (sister-of ; parent-of) : Princes (by M3)
$=$ aunt-of : William.

(4.20)′ Charles $\leq$ Vegetarians is given. Hence Vegetarians $\cdot$ Charles $=$ Charles by B8. Charles is an atom in the semantic diagram. Therefore Charles $\neq 0$, which implies Vegetarians $\cdot$ Charles $\neq 0$. Since father-of $\leq$ parent-of and Charles $\leq$ father-of : William, by M8 Charles $\leq$ parent-of : William. Thus $0 \neq$ Vegetarians $\cdot$ Charles $\leq$ Vegetarians $\cdot$ (parent-of : William).

A strictly equational proof would rely on Theorem (3.45) and would go as follows: By (iii) of Theorem (3.45) Charles $\neq 0$ iff 1 : Charles $= 1$. By B8 Charles $\leq$ Vegetarians iff Charles $\cdot$ Vegetarians $=$ Charles. Therefore, since father-of $\leq$ parent-of and Charles $\leq$ father-of : William, $1 =$ 1 : Charles $=$ 1 : (Vegetarians $\cdot$ Charles) $\leq$ 1 : (Vegetarians $\cdot$ (father-of : William)) $\leq$ 1 : (Vegetarians $\cdot$ (parent-of : William)). Hence $1 =$ 1 : (Vegetarians $\cdot$ (parent-of : William)) which implies Vegetarians $\cdot$ (parent-of : William) $\neq 0$.

I have now discharged the challenge made in Chapter 1, of deducing some implicit knowledge for the standard example of Figure 1.1. However, as is clear from Section 2.2, there are many more terminological operators than those of the standard example. I therefore give also some further examples covering such other terminological operators. Namely, I will derive some implicit relationships expressed with the rvm, trans, restrict and sd operators. These relationships involve the sample constructs given in (2.30), (2.34), (2.35), (2.37) and (2.38) of Section 2.2.

(4.24) The expression (rvm aunt-of relative-of) from (2.30) represents the set of all humans who are relatives of all those of whom they are an aunt, which we expect coincides with the set of all humans. This intuitively obvious fact is also relatively easy to prove formally. Our aim is to show that Humans $\doteq$ (rvm aunt-of relative-of) is derivable from the terminology of the given facts in the semantic network. Recall the algebraic formulation of rvm constructs given in Figure 4.1 (iii) according to which the associated algebraic term of (rvm aunt-of relative-of) is $((\text{aunt-of}\cdot\text{relative-of}'):1)'$. So, algebraically we aim to show that $\text{Humans} = ((\text{aunt-of}\cdot\text{relative-of}'):1)'$ is true.

To prove this I assume that 'aunt-of' is defined as in (4.19) as $\text{sister-of}\,;\text{parent-of}$. By (4.16) $\text{sister-of} \leq \text{relative-of}$. It is given that $\text{parent-of} \leq \text{relative-of}$. Thus using R14 repeatedly we obtain $\text{aunt-of} = \text{sister-of}\,;\text{parent-of} \leq \text{relative-of}\,;\text{parent-of} \leq \text{relative-of}\,;\text{relative-of}$. I also assume it is given that 'being a relative of' is a transitive relation. According to Theorem (3.20) (iv) this is specified by $\text{relative-of}\,;\text{relative-of} \leq \text{relative-of}$. Hence $\text{aunt-of} \leq \text{relative-of}$, which is equivalent to $\text{aunt-of}\cdot\text{relative-of}' = 0$ by B12. Therefore $((\text{aunt-of}\cdot\text{relative-of}'):1)' = (0:1)' = 0' = 1 = \text{Humans}$ (by M5 and B13), as required.

(4.25) Using (2.34) the fact that 'Elizabeth is a parent or an ancestor of William' can be expressed in $\mathcal{U}^-$ as Elizabeth $\sqsubseteq$ (some (trans parent-of) William). Algebraically it becomes $\text{Elizabeth} \leq (\sum_{n=1}^{\infty}\text{parent-of}^{n}):\text{William}$. To cater for this example I use the arithmetic of *complete* relation algebras which provide for arbitrary joins (and meets). This is justified since we model terminologies in full Peirce algebras. These have full underlying relation algebras which are necessarily complete. Here is a proof then:

Since Elizabeth is a mother of Charles who is a father of William, Elizabeth is a parent of Charles who is a parent of William. In other words, Elizabeth is a grandparent of William. Formally, since $\mathrm{Elizabeth} \le \text{mother-of}:\mathrm{Charles}$, $\text{mother-of} \le \text{parent-of}$, $\mathrm{Charles} \le \text{father-of}:\mathrm{William}$, and $\text{father-of} \le \text{parent-of}$, by using M8, M7 and M3 we get $\mathrm{Elizabeth} \le \text{mother-of}:\mathrm{Charles} \le \text{parent-of}:\mathrm{Charles} \le \text{parent-of}:(\text{father-of}:\mathrm{William}) \le \text{parent-of}:(\text{parent-of}:\mathrm{William}) = (\text{parent-of};\text{parent-of}):\mathrm{William}$. Since $\sum_{n=1}^{\infty}\text{parent-of}^{n} = \text{parent-of} + \text{parent-of};\text{parent-of} + \ldots$ it follows that $\text{parent-of};\text{parent-of} \le \sum_{n=1}^{\infty}\text{parent-of}^{n}$. By M8 it follows then that $\mathrm{Elizabeth} \le (\sum_{n=1}^{\infty}\text{parent-of}^{n}):\mathrm{William}$.

(4.26) According to (2.35) the relation 'has as brother' is represented by the role (restrict sibling-of Males). The sentence 'Anne has brothers' can then be represented in $\mathcal{U}^-$ as Anne $\sqsubseteq$ (some (restrict sibling-of Males) Humans). Algebraically it is expressed as $\mathrm{Anne} \le \text{has-brother}:\mathrm{Humans}$ where $\text{has-brother} = \text{sibling-of}\rfloor\mathrm{Males}$. To prove this I use M7 and M8 as well as P16′.

Given that $\mathrm{Charles} \le \mathrm{Princes} \le \mathrm{Males}$ and $\text{sister-of} \le \text{sibling-of}$ and given that Anne is a sister of Charles, we get $\mathrm{Anne} \le \text{sister-of}:\mathrm{Charles} \le \text{sister-of}:\mathrm{Males} \le \text{sibling-of}:\mathrm{Males} = (\text{sibling-of}\rfloor\mathrm{Males}):1 = (\text{sibling-of}\rfloor\mathrm{Males}):\mathrm{Humans}$.

(4.27) Structural description constructs are paradigm examples of constructs that are hard to translate in English. In Section 2.2 I discussed the sample expressions

(sd Males ($\subseteq$ child-of child-of))

and

(sd Males ($\subseteq$ self (not self)) ($\subseteq$ child-of child-of))

and their associated problems. But from Figure 4.1 (ii) and (iii) it is apparent that these expressions do have algebraic formulations, namely $(\text{child-of}^{\smile}\backslash\text{child-of}^{\smile}):\mathrm{Males}$ and $((e^{\smile}\backslash e'^{\smile})\cdot(\text{child-of}^{\smile}\backslash\text{child-of}^{\smile})):\mathrm{Males}$, respectively.

I claimed that (sd Males ($\subseteq$ child-of child-of)) represents the set of all people who are either male or have brothers. To show, e.g., that Males is subsumed by this expression, i.e. Males $\sqsubseteq$ (sd Males ($\subseteq$ child-of child-of)), is now routine. To prove this I need to show that Males $\leq$ (child-of$^{\smile}\backslash$child-of$^{\smile}$) : Males. By R23 child-of$^{\smile}\backslash$child-of$^{\smile}$ (the algebraic formulation of the construct ($\subseteq$ child-of child-of)) is reflexive. By Definition (3.23) (i) this means e $\leq$ child-of$^{\smile}\backslash$child-of$^{\smile}$. Using M4 and M8 we get Males $=$ e : Males $\leq$ (child-of$^{\smile}\backslash$child-of$^{\smile}$) : Males. Just as easy to prove is my claim (on page 30 in Section 2.2) that the subexpression ($\subseteq$ self (not self)) of the second sd expression is equivalent to (not self). For this we need to establish that $e^{\smile}\backslash e'^{\smile} = e'$. It follows by R11, R9 and R21 that $e^{\smile}\backslash e'^{\smile} = e^{\smile}\backslash e^{\smile\prime} = e\backslash e' = e'^{\smile} = e^{\smile\prime} = e'$.

As a final example (which involves the all construct) consider the following expression from Patel-Schneider [1990, p. 14]:

(4.28) (and Persons (and (all has-child Lawyers) (all (restrict has-child Lawyers) Doctors)))
$\sqsubseteq$ (and Persons (all has-child Doctors)).

This is quite complicated. It is also not clear how to read this in English. Patel-Schneider himself translates the subsumed term by 'the class of people whose children are all lawyers and whose children who are lawyers are all doctors', and the subsuming term by 'the class of anyone whose children are all doctors'. Note that with this translation the same point would arise as in the discussion in Section 3.5 on the intended meaning of the phrase 'eat only fruit'. Namely, representing the set of individuals whose children are all doctors by (all has-child Doctors) or algebraically by (has-child : Doctors$'$)$'$ means it is impossible to deduce that such individuals have children who are doctors. However, the point of this example is, presumably, to illustrate the inference mechanism rather than the expressiveness of the formalism. In this respect the subsumption relation has a perfectly manageable algebraic formulation, which is persons $\cdot$ (has-child : Lawyers$'$)$'$ $\cdot$ ((has-child $\rfloor$ Lawyers) : Doctors$'$)$'$ $\leq$ persons $\cdot$ (has-child : Doctors$'$)$'$. It is 'perfectly manageable' in the sense that it can be proved algebraically in the same way as the other examples above. The proof goes as follows:

It suffices to prove (has-child : Lawyers$'$)$'$ $\cdot$ ((has-child $\rfloor$ Lawyers) : Doctors$'$)$'$ $\leq$ (has-child : Doctors$'$)$'$. To prove this I use the property M25$'$ (given on page 62) as reformulated from M25 proved in Section 3.3. So

$(\text{has-child} : \text{Lawyers}')' \cdot ((\text{has-child} \rfloor \text{Lawyers}) : \text{Doctors}')'$

$= (\text{has-child} : (\text{Lawyers} \cdot \text{Doctors})')'$ (by M25$'$)

$\leq (\text{has-child} : \text{Doctors}')'$ (using M16).

My principal objective with the case studies of this section has been to illustrate the power and elegance of the equational algebraic approach. Even quite complicated formulations from $\mathcal{U}^-$ have straightforward translations in the algebraic framework, and reasonably straightforward deductions from the terminological axioms—and of course the algebraic axioms.

Studying terminological representation languages from an algebraic point of view has several spinoffs. It makes for easier analysis of terminological expressions (especially those constructed with operators otherwise difficult to handle). Furthermore, it provides one with a useful link to other areas where relation-algebraic concepts have been applied.

Analysis of structural description sd has revealed that this operator is redundant, since it is interdefinable with the some operator. That the sd operator can be defined in terms of some and the other terminological operators is apparent form the algebraic presentation of sd (given in (ii) and (iii) of Figure 4.1). To show that some can be expressed with sd, consider the algebraic presentation $r^{\smile} \backslash s^{\smile}$ of the role binding construct ($\subseteq$ R S). R22 implies that for any element r in a relation algebra $r = r^{\smile\prime} \backslash e'$. Hence using R11 and R9 any r can be identically formulated as $r'^{\smile} \backslash e'^{\smile}$. Therefore (since any algebraic identity determines a semantic equivalence), for any role description R

(4.29) R $\approx$ ($\subseteq$ (not R) (not self)).

Also, it is immediate by R22 that

(4.30) R $\approx$ ($\supseteq$ (not self) (not (inverse R))).

Using these results it is easy to encode any role description as a role binding construct of either kind. More importantly, these results imply that anything expressible with some can be expressed with sd since, e.g.,

(4.31) (some R C) $\approx$ (sd C ($\subseteq$ (not R) (not self))).

So, sd and some are interdefinable. In the light of the problems associated with using structural description this raises doubts about the value of including sd and role binding constructs in terminological languages at all.

Also contributing to simplifying the analysis of terminological expressions is the link between their algebraic representations and the linguistic analysis of Suppes and Böttner (discussed in Section 3.5). This link can be utilised to provide valuable assistance when representing given information formulated in English as terminological expressions and likewise when translating represented information into ordinary English. For example, according to Figure 3.2 (iii) the set of 'admirers of no princes' is represented by $(\text{admirer-of}:\text{Princes})'$ which translates to the terminological expression (not (some admirer-of Princes)). Using (3.58) and (3.59) we represent the set of 'admirers only of princes' by $\text{admirer-of}:\text{Princes} \;\cap\; (\text{admirer-of}:\text{Princes}')'$ or in a terminological language by

(and (some admirer-of Princes) (all admirer-of Princes)).

Reversing this process, given for example the terminological expression

(and Humans (not (some (not admirer-of) Princes))) $\doteq \; \bot$

its algebraic formulation is $\text{Humans} \cdot (\text{admirer-of}'\colon \text{Princes})' = 0$ (or in the calculus $\text{Humans} \;\cap\; (\text{admirer-of}'\colon \text{Princes})' = \emptyset$) which according to Figure 3.3 (iii) translates to 'no human admires all princes'.

Terminological representation can also be linked to algebras that formalise a transitive closure operation. As mentioned earlier (in Section 3.2 on page 49 and in (4.25)) the trans construct, which is interpreted as a transitively closed relation (namely the arbitrary union $\bigcup_{n=1}^{\infty} R^n$) can be catered for in complete relation algebras (since these have arbitrary joins). For certain purposes it may however be more advantageous to have an algebraic axiomatisation of a transitive closure operation. Such axiomatisations exist for example in the algebras mentioned in Section 3.5. In particular the Ng-Tarski relation algebras have a transitive closure operation and the Kleene algebras, the dynamic algebras and also the action algebras have a star operation intended to formalise reflexive transitive closure. Some work still needs to be done to cater for transitively closed relations interacting with sets (by for example extending Boolean modules with a transitive closure operation or by strengthening the underlying Kleene algebra in a dynamic algebra). The reader interested in formalisations of transitive closures should consult the references given in Section 3.5.

In Section 4.1 I showed that $\mathcal{U}^-$, a sublanguage of $\mathcal{U}$, can be accommodated in the algebraic context. Unlike $\mathcal{U}$, $\mathcal{U}^-$ does not include the number restriction operators atleast and atmost. It remains to establish whether it is possible to accommodate these operators. Consider the expression (atleast 4 R), for example. To present this expression algebraically it must be possible to formulate equationally the statement that 'there exist four elements to which an element is related by R'. The algebras I have been focussing on are remarkably powerful, but are expressively weaker than full first-order logic. Thus, not every first-order statement can be represented equationally. More specifically, it is known (see Tarski and Givant [1987]) that first-order statements containing more than three variables cannot be represented equationally in relation algebras. For example, 'there exist four elements' is such a statement. (According to Tarski and Givant [1987, p. xi] this follows from a result by Korselt as documented in Löwenheim [1915*].) This suggests that atleast expressions have no algebraic representation in the context of this thesis. Since any atmost expression can be defined in terms of the atleast operator (see, e.g., Patel-Schneider [1987a, p. 91]) and since atleast is a special case of the fillers operator (see (2.46)), both the atmost and fillers operators are likewise not presentable in the present context. This does not exclude the possibility that other adequate formalisations can be found.

4.4 Concluding Remarks

As outlined in the Introduction the goal of this thesis has been to show that the algebras of sets and relations are natural vehicles for *representing* given terminological knowledge, and also for *inferring* implicit knowledge from what is given. In order to accomplish this goal, I showed that the standard model-theoretic semantics of certain terminological representation languages can be accommodated in the algebraic framework. I established natural associations between terminological and algebraic representations (summarised in Figure 4.1). Interactions between concepts and roles are then algebraically characterised, and the generation of new concepts and roles from old amounts to the generation of free algebras, as discussed in Section 4.2. Finally, in Section 4.3 I used equational reasoning to derive those inferences in the core example which were presented in Chapter 1 and also some fairly complex inferences with terminological constructs. I also linked terminological representation with other areas of application, notably computational linguistics.

In conclusion, I claim for the algebraic approach the following advantages.

An existing mathematical framework: In Section 2.1 I outlined the history of terminological representation. It is a relatively new field of research and by and large, the development has been implementation driven and rather *ad hoc*. It seems that only recently research has started to focus on formal aspects such as semantics and tractability. In contrast the algebras presented here are formal mathematical structures. Their origins lie in the calculi of sets and relations which go back more than 100 years to G. Boole, A. De Morgan, C.S. Peirce and E. Schröder. As is evident from the bulk of literature available the algebras have been extensively studied and continue to be of considerable interest. The algebraic approach thus has the advantage of embedding new work into old.

Expressiveness: In this respect the algebras are quite powerful. Many elementary statements concerning sets and relations can be formulated algebraically. For example, in Theorem (3.20) I listed some such formulations of familiar properties of relations. Schröder [1890–1895*], who systematically studied the calculus of relations, was even led to conjecture (wrongly as it turned out) that every elementary statement about relations can be formulated equationally in the calculus of relations. I mention again Tarski's claim (finally proved together with Givant in [1987]) that practically all of mathematical research can be carried out in a formalism based on relation algebra. This is truly remarkable, especially since one cannot express every first-order statement in the relation-algebraic context. With regards to terminological representation I established that the algebras are sufficiently expressive to cater for the language $\mathcal{U}^-$ which includes most existing terminological operators (the exceptions being the number restriction operators).

Ease of use: This is apparent from Chapter 3 and the case studies presented in Section 4.3. The main reasons why these algebras are so easy to work with are: first, the form of the algebraic language (resulting in simple and elegant formulations for first-order statements or terminological expressions), and second, the natural axiomatisations they provide for reasoning with sets and relations (and now also with concepts and roles).

Other Areas of Application: The algebras of sets and relations already have some firm links to other areas in Computer Science. I mentioned various applications in Section 3.5, of which at least two (namely computational linguistics and logics of programs) are useful to terminological representation.

Possible mechanisation: Since the algebras considered here are defined with equational axioms, I envisage an implementation based on equational logic. Equational logic is a well-established field of mathematical logic and has been implemented in various forms, for example as term rewriting systems (like that of Hsiang [1985]) or as equational logic programming systems (like that of O'Donnell [1985]). More on rewriting techniques can be found in Huet and Oppen [1980], Jouannaud [1985*] and Lescanne [1987*], and on equational logic programming in Goguen and Meseguer [1986*] and Hölldobler [1989]. For an introduction to term rewriting and logic programming see, e.g., Jorrand [1988]. Equational logic also forms the basis for unification theory (for a survey see Siekmann [1987]) and an extension of many-sorted logic (Cohn [1989]) called order sorted equational logic (Smolka *et al* [1989]). In both Siekmann [1987] and Smolka *et al* [1989] further references can be found to various other automated deduction approaches. One such approach which has already been oriented towards knowledge representation applications is that of Aït-Kaci and Nasr [1986] and Aït-Kaci *et al* [1989] implementing lattice and inheritance operations.

List of Figures

Index of Notation

Bibliography

(Note: Some of the references listed here as 'to appear' or 'forthcoming' may have been published already but have been inaccessible to me.)

Aït-Kaci, H., and Nasr, R. [1986]. LOGIN: A Logic Programming Language with Built-In Inheritance. *Journal of Logic Programming* **3**, 185–215.

Aït-Kaci, H., Boyer, R., Lincoln, P., and Nasr, R. [1989]. Efficient Implementation of Lattice Operations. *ACM Trans. Prog. Lang. Syst.* **11** (1), 115–146.

Anderson, A.R., and Belnap, N.D. [1975]. *Entailment: The Logic of Relevance and Necessity.* Vol. 1, Princeton University Press, Princeton, NJ.

Andréka, H., Jónsson, B., and Németi, I. [1990]. Relatively Free Relation Algebras. In Bergman, C.H., Maddux, R.D., and Pigozzi, D.L. (Eds.), *Algebraic Logic and Universal Algebra in Computer Science. Lecture Notes in Computer Science* **425**, 1–14.

Andréka, H., Jónsson, B., and Németi, I. [1991]. Free Algebras in Discriminator Varieties. To appear in *Algebra Universalis.*

Attardi, G., and Simi, M. [1986]. A Description-Oriented Logic for Building Knowledge Bases. *Proceedings of the IEEE* **74** (10), 1335–1344.

Attardi, G., Corradini, A., Diomedi, S., and Simi, M. [1986]. Taxonomic Reasoning. *European Conference on AI* **1**, Brighton, UK, 236–245.

Baader, F. [1990]. A Formal Definition for Expressive Power of Knowledge Representation Languages. Research Report RR-90-05, DFKI, Projektgruppe WINO, Kaiserslautern, Germany.

Bell, J.L., and Slomson, A.B. [1971]. *Models and Ultraproducts: An Introduction.* North-Holland, Amsterdam.

Belnap, N.D. [1975]. How a Computer Should Think. In Ryle, G. (Ed.), *Contemporary Aspects of Philosophy.* Proceedings of the Oxford International Symposium, Oxford, England, 30–56.

Belnap, N.D. [1977]. A Useful Four-Valued Logic. In Epstein, G., and Dunn, J.M. (Eds.), *Modern Uses of Multiple-Valued Logic.* Reidel Publ. Co., Dordrecht, Holland, 8–37.

Birkhoff, G. [1935*]. On the Structure of Abstract Algebras. *Proc. Camb. Philos. Soc.* **31**, 433–454.

Birkhoff, G., and Lipson, J.D. [1970]. Heterogeneous Algebras. *Journal of Combinatorial Theory* **8**, 115–133.

Birkhoff, G. [1973]. *Lattice Theory.* Third Edition, American Mathematical Society Colloquium Publications, Providence, Rhode Island.

Blyth, T.S., and Janowitz, M.F. [1972]. *Residuation Theory.* Pergamon Press, Oxford, England.

Bobrow, D.G., and Winograd, T. [1977]. An Overview of KRL, a Knowledge Representation Language. *Cognitive Science* **1** (1), 3–46. Reprinted in Brachman and Levesque [1985, 263–285].

Boole, G. [1847*]. *The Mathematical Analysis of Logic, Being an Essay Toward a Calculus of Deductive Reasoning.* London and Cambridge.

Boole, G. [1854*]. *An Investigation of the Laws of Thought.* London.

Böttner, M. [1985]. Variable-free Semantics for 'Only' and 'Except'. Manuscript, Institut für Kommunikationsforschung und Phonetik, Universität Bonn, Germany.

Böttner, M. [1986]. State Transition Semantics. Manuscript. To appear in *Theoretical Linguistics.*

Böttner, M. [1989]. Variable-free Semantics for Anaphora. Manuscript, Department of Mathematics, University of Cape Town, South Africa. To appear in *Journal of Philosophical Logic.*

Böttner, M. [1990]. Personal communication.

Borgida, A., Brachman, R.J., McGuinness, D.L., and Resnick, L.A. [1989]. CLASSIC: A Structural Data Model for Objects. *Proceedings of the ACM SIGMOD-89 International Conference on Management of Data*, Portland, Oreg., 58–67.

Brachman, R.J. [1977]. What's in a concept: Structural Foundations for Semantic Networks. *Int. J. Man-Machine Studies* **9**, 127–152.

Brachman, R.J. [1979]. On the Epistomological Status of Semantic Networks. In Findler [1979, 3–50].

Brachman, R.J. [1983]. What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *IEEE Computer* **16** (10), 30–36.

Brachman, R.J. [1985]. "I Lied about the Trees" or, Defaults and Definitions in Knowledge Representation. *AI Magazine* **6** (3), 80–93.

Brachman, R.J., and Levesque, H.J. [1982]. Competence in Knowledge Representation. *Proc. Nat. Conf. AI*, 189–192.

Brachman, R.J., and Levesque, H.J. [1984]. The Tractability of Subsumption in Frame-Based Description Languages. *Proc. Nat. Conf. AI*, 34–37.

Brachman, R.J., and Levesque, H.J. (Eds.) [1985]. *Readings in Knowledge Representation.* Morgan Kaufmann Publ. Inc., Los Altos.

Brachman, R.J., and Schmolze, J.G. [1985]. An Overview of the KLONE Knowledge Representation System. *Cognitive Science* **9** (2), 171–216. Reprinted in Mylopoulos and Brodie [1989*].

Brachman, R.J., Fikes, R.E., and Levesque, H.J. [1983]. KRYPTON: A Functional Approach to Knowledge Representation. *IEEE Computer* **16** (10), 67–73. A revised version appears in Brachman and Levesque [1985, 411–429].

Brachman, R.J., Gilbert, V.P., and Levesque, H.J. [1985]. An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON. *International Joint Conference on Artificial Intelligence*, Los Angeles, California, 532–539. Reprinted in Mylopoulos and Brodie [1989*].

Brachman, R.J., Levesque, H.J., and Reiter, R. [1989*]. *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann Publ. Inc., San Mateo, California.

Brink, C. [1978]. The Algebra of Relations. PhD Dissertation, University of Cambridge.

Brink, C. [1979]. Two Axiom Systems for Relation Algebras. *Notre Dame Journal of Formal Logic* **20** (4), 909–914.

Brink, C. [1981]. Boolean Modules. *Journal of Algebra* **71** (2), 291–313.

Brink, C. [1988]. On the Application of Relations. *S. Afr. J. Philos.* **7** (2), Special Interdisciplinary Edition Devoted to Logic, 105–112.

Brink, C., and Schmidt, R.A. [1991]. Subsumption Computed Algebraically. To appear in *Computers and Mathematics with Applications*, Special Issue on Semantic Networks in Artificial Intelligence.

Britz, K. [1988]. Relations and Programs. M.Sc. Thesis, Department of Computer Science, University of Stellenbosch, South Africa.

Burris, S., and Sankappanavar, H.P. [1981]. *A Course in Universal Algebra.* Springer-Verlag, Berlin.

Charniak, E., and McDermott, D. [1985]. *Introduction to Artificial Intelligence.* Addison-Wesley, Reading, Mass.

Chin, L.H., and Tarski, A. [1951]. Distributive and Modular Laws in the Arithmetic of Relation Algebras. *Univ. Calif. Publ. Math.* **1** (9), 341–384.

Cohn, A.G. [1989]. Taxonomic Reasoning with Many-Sorted Logics. *Artificial Intelligence Review* **3**, 89–128.

Cohn, P.M. [1981]. *Universal Algebra.* Reidel Publ. Co., Dordrecht, Holland.

Computers and Mathematics with Applications [1991*]. Special Issue on Semantic Networks in Artificial Intelligence. To appear.

Delgrande, J.P., and Mylopoulos, J. [1986]. Knowledge Representation: Features of Knowledge. In Bibel, W., and Jorrand, Ph. (Eds.), *Fundamentals in Artificial Intelligence: An Advanced Course. Lecture Notes in Computer Science* **232**, 3–36.

Deliyanni, A., and Kowalski, R.A. [1979]. Logic and Semantic Networks. *Comm. ACM* **22** (3), 184–192.

De Morgan, A. [1847*]. *Formal Logic: The Calculus of Inference, Necessary and Probable.* Taylor and Walton, London.

Devanbu, P., Selfridge, P.G., Ballard, B.W., and Brachman, R.J. [1989]. A Knowledge-Based Software Information System. *International Joint Conference on Artificial Intelligence*, 110–115.

Donini, F.M., Lenzerini, M., Nardi, D., Hollunder, B., and Nutt, W. [1990]. The Complexity of Concept Description Languages. Submitted to the *IEEE Symposium on Foundations of Computer Science* (FOCS 90).

Doyle, J., and Patil, R.S. [1989]. Two Dogmas of Knowledge Representation: Language Restrictions, Taxonomic Classification, and the Utility of Representation Services. MIT/LCS/Technical Memo 387.b.

Edelmann, J., and Owsnicki, B. [1986]. Data Models in Knowledge Representation Systems: A Case Study. *10th German Workshop on Artificial Intelligence*, 69–74.

Ehrig, H., and Mahr, B. [1985]. *Fundamentals of Algebraic Specification* 1: *Equations and Initial Semantics.* Springer-Verlag, Berlin.

Etherington, D.W., Borgida, A., Brachman, R.J., and Kautz, H. [1989]. Vivid Knowledge and Tractable Reasoning: Preliminary Report. *International Joint Conference on Artificial Intelligence*, 1146–1152.

Findler, N.V. (Ed.) [1979]. *Associative Networks: Representation and Use of Knowledge by Computers.* Academic Press, New York.

Garey, M.R., and Johnson, D.S. [1979]. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman & Co., San Fransisco, CA.

Goguen, J.A., and Meseguer, J. [1986*]. Eqlog: Equality, Types, and Generic Modules for Logic Programming. In DeGroot, D., and Lindstrom, G. (Eds.), *Logic Programming, Functions, Relations and Equations.* Prentice Hall, Englewood Cliffs, NJ.

Grätzer, G. [1968]. *Universal Algebra.* First Edition, D. van Nostrand Co. Inc., Princeton, NJ.

Grätzer, G. [1979]. *Universal Algebra.* Second Edition, Springer-Verlag, Berlin.

Halmos, P.R. [1974]. *Lectures on Boolean Algebras.* Springer-Verlag, Berlin.

Harel, D. [1984]. Dynamic Logic. In Gabbay, D., and Guenther, F. (Eds.), *Handbook of Philosophical Logic.* Vol. II, Reidel Publ. Co., Dordrecht, Holland, 497–604.

Harel, D. [1987]. *Algorithmics: The Spirit of Computing.* Addison-Wesley, Reading, Mass.

Hayes, P. [1974]. Some Problems and Non-Problems in Representation Theory. *Proc. AISB Summer Conference*, University of Sussex, 63–79. Reprinted in Brachman and Levesque [1985, 3–22].

Hayes, P. [1977]. In Defense of Logic. *International Joint Conference on Artificial Intelligence*, 559–565.

Hayes, P. [1979]. The Logic of Frames. In Metzing, D. (Ed.), *Frame Conceptions and Text Understanding.* W. De Gruyter and Co., Berlin, 46–61. Reprinted in Brachman and Levesque [1985, 288–295].

Hendrix, G.G. [1979]. Encoding Knowledge in Partitioned Networks. In Findler [1979, 51–92].

Henkin, L. [1977]. The Logic of Equality. *American Mathematical Monthly* **84** (8), 597–612.

Henkin, L., Monk, J.D., and Tarski, A. [1985]. *Cylindric Algebras: Part II.* Studies in Logic and the Foundations of Mathematics **115**, North-Holland, Amsterdam.

Hennessy, M.C.B. [1980]. A Proof-System for the First-Order Relational Calculus. *Journal of Computer and System Sciences* **20**, 96–110.

Hölldobler, S. [1989]. *Foundations of Equational Logic Programming. Lecture Notes in Artificial Intelligence* **353** (Subseries of LNCS).

Hollunder, B. [1989]. Subsumption Algorithms for Some Attributive Concept Description Languages. SEKI Report SR-89-16, Fachbereich Informatik, Universität Kaiserslautern, Germany.

Hollunder, B., Nutt, W., and Schmidt-Schauß, M. [1990]. Subsumption Algorithms for Concept Description Languages. To appear in *Proceedings of the 9th European Conference on AI* (ECAI 90). A shortened version of Hollunder, B., and Nutt, W. [1990]. Subsumption Algorithms for Concept Languages. Research Report RR-90-04, DFKI, Kaiserslautern, Germany.

Hsiang, J. [1985]. Refutational Theorem Proving using Term-Rewriting Systems. *Artificial Intelligence* **25**, 255–300.

Huet, G., and Oppen, D.C. [1980]. Equations and Rewrite Rules: A Survey. In Book, R.V. (Ed.), *Formal Language Theory: Perspectives and Open Problems.* Academic Press, New York, 349–405.

Huntington, E.V. [1904*]. Sets of Independent Postulates for the Algebra of Logic. *Trans. A.M.S.* **5**, 288–309.

Huntington, E.V. [1933*]. New Sets of Independent Postulates for the Algebra of Logic, with Special Reference to Whitehead and Russell's *Principia Mathematica. Trans. A.M.S.* **35**, 274–304.

IEEE Computer **16** (10) [1983]. Special Issue on Knowledge Representation.

IEEE Workshop on Principles of Knowledge-Based Systems [1984*]. Denver, Colorado.

Israel, D.J. [1983a]. Interpreting Network Formalisms. *Computers and Mathematics with Applications* **9** (1), Special Issue on Computational Linguistics, 1–13.

Israel, D.J. [1983b]. The Role of Logic in Knowledge Representation. *IEEE Computer* **16** (10), 37–41. Earlier version of Israel [1985].

Israel, D.J. [1985]. A Short Companion to the Naive Physics Manifesto. In Hobbs, J.R., and Moore, R.C. (Eds.), *Formal Theories of the Commonsense World.* Ablex Publ. Co., Norwood, New Jersey.

Israel, D.J., and Brachman, R.J. [1981]. Distinctions and Confusions: A Catalogue Raisonne. *International Joint Conference on Artificial Intelligence*, 452–459.

Israel, D.J., and Brachman, R.J. [1984]. Some Remarks on the Semantics of Representation Languages. In Brodie, M.L., Mylopoulos, J., and Schmidt, J.W. (Eds.), *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages.* Springer-Verlag, Berlin, 119–146. Based on Israel and Brachman [1981].

Jónsson, B. [1982]. Varieties of Relation Algebras. *Algebra Universalis* **15**, 273–298.

Jónsson, B. [1988]. Relation Algebras and Schröder Categories. *Discrete Mathematics* **70**, 27–45.

Jónsson, B. [Draft]. Program Specifications as Boolean Operators: A Very Preliminary Draft. Department of Mathematics, Vanderbilt University, Nashville, TN.

Jónsson, B., and Tarski, A. [1948*]. Representation Problems for Relation Algebras. Abstract 89, *Bulletin of the American Mathematical Society* **54**, 80 & 1192.

Jónsson, B., and Tarski, A. [1951]. Boolean Algebras with Operators, Part I. *American Journal of Mathematics* **73**, 891–939.

Jónsson, B., and Tarski, A. [1952]. Boolean Algebras with Operators, Part II. *American Journal of Mathematics* **74**, 127–162.

Jorrand, P. [1988]. Fundamental Mechanisms for Artificial Intelligence Programming Languages: An Introduction. In Nossum, R.T. (Ed.), *Advanced Topics in Artificial Intelligence. Lecture Notes in Artificial Intelligence* **345** (Subseries of LNCS), 1–40.

Jouannaud, J.-P. (Ed.) [1985*]. *Rewriting Techniques and Applications. Lecture Notes in Computer Science* **202**.

Kaczmarek, T.S., Bates, R., and Robbins, G. [1986]. Recent Developments in NIKL. *Proc. Nat. Conf. AI*, 978–985.

Knowledge Representation Workshop [1983*]. Santa Barbara, CA.

Kozen, D. [1980]. A Representation Theorem for Models of $*$-free PDL. In de Bakker, J., and van Leeuwen, J. (Eds.), *Automata, Languages and Programming. Lecture Notes in Computer Science* **85**, 351–362.

Kozen, D. [1981]. On the Duality of Dynamic Algebras and Kripke Models. In Engeler, E. (Ed.), *Logic of Programs. Lecture Notes in Computer Science* **125**, 1–11.

Kozen, D. [1990a*]. A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events. Technical Report 90-1123, Cornell University.

Kozen, D. [1990b*]. On Kleene Algebras and Closed Semirings. In Rovan, B. (Ed.), *Mathematical Foundations of Computer Science. Lecture Notes in Computer Science* **452**, 26–47.

Kumar, D. [1990*]. *Current Trends in SNePS-Semantic Network Processing System. Lecture Notes in Artificial Intelligence* **437** (Subseries of LNCS).

Lescanne, P. (Ed.) [1987*]. *Rewriting Techniques and Applications. Lecture Notes in Computer Science* **256**.

Levesque, H.J. [1986]. Knowledge Representation and Reasoning. *Annual Review of Computer Science* **1**, 255–287. Reprinted in Mylopoulos and Brodie [1989*].

Levesque, H.J., and Brachman, R.J. [1987]. Expressiveness and Tractability in Knowledge Representation and Reasoning. *Computational Intelligence* **3**, 78–93. An extended version of Brachman and Levesque [1984].

Löwenheim, L. [1915*]. Über Möglichkeiten im Relativkalkül. *Math. Ann.* **76**, 447–470.

Lyndon, R.C. [1950*]. The Representation of Relational Algebras. *Annals of Mathematics* **51**, 707–729.

Maddux, R.D. [1978a]. Some Sufficient Conditions for the Representability of Relation Algebras. *Algebra Universalis* **8**, 162–172.

Maddux, R.D. [1978b*]. Topics in Relation Algebras. PhD Dissertation, University of California, Berkeley.

Maddux, R.D. [1983]. A Sequent Calculus for Relation Algebras. *Annals of Pure and Applied Logic* **25**, 73–101.

Maddux, R.D. [1989]. Nonfinite Axiomatizability Results for Cylindric and Relation Algebras. *Journal of Symbolic Logic* **54** (3), 951–974.

Maddux, R.D. [1990a]. The Origin of Relation Algebras in the Development and Axiomatization of the Calculus of Relations. Manuscript, Department of Mathematics, Iowa State University, Ames, Iowa.

Maddux, R.D. [1990b]. Personal communication with C. Brink.

Manes, E.G., and Arbib, M.A. [1986]. *Algebraic Approaches to Program Semantics.* Springer-Verlag, Berlin.

McDermott, D. [1976]. Artificial Intelligence Meets Natural Stupidity. *ACM SIGART Newsletter* **57**. Reprinted in Haugeland, J. (Ed.) [1981]. *Mind Design.* Bradford Books, MIT Press, Cambridge, Mass., 141–160.

McDermott, D. [1978]. Tarskian Semantics, or No Notation Without Denotation! *Cognitive Science* **2** (3), 277–282.

McKenzie, R. [1970*]. Representations of Integral Relation Algebras. *Michigan Math. J.* **17**, 279–287.

McKinsey, J.C.C. [1940]. Postulates for the Calculus of Binary Relations. *Journal of Symbolic Logic* **5** (3), 85–97.

Mendelson, E. [1970]. *Theory and Problems of Boolean Algebra and Switching Circuits.* Schaum's Outline Series, McGraw-Hill, New York.

Minsky, M. [1975]. A Framework for Representing Knowledge. In Winston, P.H. (Ed.), *The Psychology of Computer Vision.* McGraw-Hill, New York, 211–277. Excerpts reprinted in Haugeland, J. (Ed.) [1981]. *Mind Design.* Bradford Books, MIT Press, Cambridge, Mass., 95–128. Also, reprinted in Brachman and Levesque [1985, 245–262].

Monk, J.D. [1964*]. On Representable Relation Algebras. *Michigan Math. J.* **11**, 207–210.

Moore, J.D. [1986*]. NIKL *Workshop* Summary. Information Sciences Institute, University of Southern California.

Mylopoulos, J., and Levesque, H.J. [1983]. An Overview of Knowledge Representation. *7th German Workshop on Artificial Intelligence*, 142–157.

Mylopoulos, J., and Brodie, M. (Eds.) [1989*]. *Readings in Artificial Intelligence and Databases.* Morgan Kaufmann Publ. Inc., San Mateo, California.

Nebel, B. [1988]. Computational Complexity of Terminological Reasoning in BACK. *Artificial Intelligence* **34**, 371–383.

Nebel, B. [1990a]. Terminological Reasoning is Inherently Intractable. *Artificial Intelligence* **43**, 235–249. Also available as Nebel, B. [1989]. IWBS Report 82, IWBS, IBM Deutschland, Stuttgart, Germany.

Nebel, B. [1990b*]. *Reasoning and Revision in Hybrid Representation Systems. Lecture Notes in Artificial Intelligence* **422** (Subseries of LNCS).

Nebel, B., and Smolka, G. [1989]. Representation and Reasoning with Attributive Descriptions. IWBS Report 81, IWBS, IBM Deutschland, Stuttgart, Germany. To appear in Bläsius, K.-H., Hedtstück, U., and Rollinger, C.-R. (Eds.), *Sorts and Types in Artificial Intelligence.* Springer-Verlag, Berlin.

Nebel, B., and von Luck, K. [1988]. Hybrid Reasoning in BACK. In Ras, Z.W., and Saitta, L. (Eds.), *Methodologies for Intelligent Systems* **3**, North-Holland, New York, 260–269.

Ng, K.C., and Tarski, A. [1977]. Relation Algebras with Transitive Closure. Abstract 742-02-09, *Notices of the American Mathematical Society*, A29–A30.

Ng, K.C. [1984*]. Relation Algebras with Transitive Closure. PhD Dissertation, University of California, Berkeley.

Nilsson, N. [1980]. *Principles of Artificial Intelligence.* Tiago, Palo Alto, California.

O'Donnell, M.J. [1985]. *Equational Logic as a Programming Language.* MIT Press, Cambridge, Mass.

Parikh, D. [1981]. Propositional Dynamic Logic of Programs: A Survey. In Engeler, E. (Ed.), *Logic of Programs. Lecture Notes in Computer Science* **125**, 102–144.

Patel-Schneider, P.F. [1984]. Small can be Beautiful in Knowledge Representation. AI Technical Report 37, Schlumberger Palo Alto Research.

Patel-Schneider, P.F. [1987a]. Decidable, Logic-Based Knowledge Representation. PhD Dissertation, University of Toronto.

Patel-Schneider, P.F. [1987b]. A Hybrid, Decidable, Logic-Based Knowledge Representation System. *Computational Intelligence* **3**, 64–77.

Patel-Schneider, P.F. [1989a]. A Four-Valued Semantics for Terminological Logics. *Artificial Intelligence* **38**, 319–351.

Patel-Schneider, P.F. [1989b]. Undecidability of Subsumption in NIKL. *Artificial Intelligence* **39**, 263–272.

Patel-Schneider, P.F. [1990]. Practical, Object-Based Knowledge Representation for Knowledge-Based Systems. *Information Systems* **15** (1), 9–19.

Patel-Schneider, P.F., *et al* [1989]. Report on the *Workshop on Term Subsumption Languages in Knowledge Representation*, Thorn Hill, New Hampshire, USA.

Peirce, C.S. [1870*]. Description of a Notation for the Logic of Relatives, Resulting from an Amplification of the Conceptions of Boole's Calculus of Logic. *Mem. Amer. Acad.* **9**, 317–378. Reprinted in Peirce [1931–1935*].

Peirce, C.S. [1931–1935*]. *The Collected Papers of Charles Sanders Peirce.* Vol. III, Hartshorne, C., and Weiss, P. (Eds.), Harvard University Press, Cambridge, Mass.

Pratt, V.R. [1976*]. Semantical Considerations on Floyd-Hoare Logic. 17*th IEEE Symposium on Foundations of Computer Science*, 109–121.

Pratt, V.R. [1979]. Dynamic Algebras: Examples, Constructions, Applications. Technical Report MIT/LCS/TM-138, MIT Laboratory for Computer Science.

Pratt, V.R. [1990a]. Dynamic Algebras as a Well-Behaved Fragment of Relation Algebras. In Bergman, C.H., Maddux, R.D., and Pigozzi, D.L. (Eds.), *Algebraic Logic and Universal Algebra in Computer Science. Lecture Notes in Computer Science* **425**, 77–110.

Pratt, V.R. [1990b]. Action logic and Pure Induction. Department of Computer Science, Stanford. To appear in JELIA-90.

Pretorius, J.P.G. [1990]. The Algebra and Topology of Boolean Modules. M.Sc. Thesis, Thesis Reprint TR 008, Department of Mathematics, University of Cape Town, South Africa.

Proceedings of the IEEE **74** (10) [1986].

Quillian, M.R. [1968]. Semantic Memory. In Minsky, M. (Ed.), *Semantic Information Processing.* MIT Press, Cambridge, Mass., 216–270.

Rich, E. [1983]. *Artificial Intelligence.* McGraw-Hill Book Co., New York.

Roberts, R.B., and Goldstein, I.P. [1977*]. The FRL Manual. MIT Lab AI Memo 409, Massachusetts Institute of Technology, Cambridge, Mass.

Saffiotti, A., and Sebastiani, F. [1989]. Towards a Hybrid Logic of Acquaintance and Awareness. In Cohn, A.G. (Ed.), *Proceedings of the 7th Conference of the Society for the Study of Artificial Intelligence and Simulation of Behaviour* (AISB 89). Morgan Kaufmann Publ. Inc., San Mateo, California, 95–103.

Schank, R.C., and Rieger III, C.J. [1974]. Inference and the Computer Understanding of Natural Language. *Artificial Intelligence* **5**, 373–412. Reprinted in Brachman and Levesque [1985, 119–139].

Schild, K. [1988]. Undecidability of Subsumption in $\mathcal{U}$. KIT-Report 67, Department of Computer Science, Technische Universität Berlin, Germany.

Schmidt, G., and Ströhlein, T. [1985]. Relation Algebras: Concept of Points and Representability. *Discrete Mathematics* **54**, 83–92.

Schmidt-Schauß, M. [1988]. Subsumption in KLONE is Undecidable. SEKI Report SR-88-18, Fachbereich Informatik, Universität Kaiserslautern, Germany. Also in Brachman *et al* [1989*, 421–431].

Schmidt-Schauß, M., and Smolka, G. [1988a]. Attributive Concept Description with Unions and Complements. SEKI Report SR-88-21, Fachbereich Informatik, Universität Kaiserslautern, Germany.

Schmidt-Schauß, M., and Smolka, G. [1988b]. Attributive Concept Description with Complements. To appear in *Artificial Intelligence*.

Schmolze, J.G. [1989a]. Terminological Knowledge Representation Systems Supporting n-ary Terms. In Brachman *et al* [1989*, 432–443].

Schmolze, J.G. [1989b]. The Language and Semantics of NIKL. Technical Report 89–4, Department of Computer Science, Tufts University, Medford, MA. To appear in *Computational Intelligence*.

Schmolze, J.G., and Brachman, R.J. [1982*]. Proceedings of the *1981* KLONE *Workshop*. BBN Report No. 4842, Bolt Beranek and Newman Inc.

Schmolze, J.G., and Israel, D. [1983]. KLONE: Semantics and Classification. BBN Technical Report No. 5421, Bolt, Beranek and Newman Inc., 27–39.

Schmolze, J.G., and Lipkis, T. [1983]. Classification in the KLONE Knowledge Representation System. *International Joint Conference on Artificial Intelligence*, 330–332.

Schönfeld, W. [1982*]. Upper Bounds for a Proof-Search in a Sequent Calculus for Relational Equations. *Z. Math. Logik Grundlagen Math.* **28**, 239–246.

Schröder, E. [1890–1895*]. *Vorlesungen über die Algebra der Logik III: Algebra und Logik der Relative.* B. G. Teubner, Leibzig (1890–1895).

Schubert, L.K., Goebel, R.G., and Cerone, N.J. [1979*]. The Structure and Organization of a Semantic Net for Comprehension and Inference. In Findler [1979, 121–175].

Shapiro, S.C. [1979*]. The SNePS Semantic Network Processing System. In Findler [1979, 179–203].

Siekmann, J. [1987]. Unification Theory. Technical Report TR-ARP-17/87, Automated Reasoning Project, Australian National University, Canberra.

Sikorski, R. [1964]. *Boolean Algebras.* Springer-Verlag, Berlin.

Smolka, G. [1989]. Feature Constraint Logics for Unification Grammars. IWBS Report 93, IWBS, IBM Deutschland, Stuttgart, Germany.

Smolka, G., Nutt, W., Goguen, J.A., and Meseguer, J. [1989]. Order-Sorted Equational Computation. *Resolution of Equations in Algebraic Structures* **2**, 297–367.

Sowa, J.F. (Ed.) [1990*]. *Principles of Semantic Networks: Explorations in the Representation of Knowledge.* Morgan Kaufmann Publ. Inc., San Mateo, California. Forthcoming.

Spinelli, G., Gaglio, S., Costa, M., Frixione, M., Traversa, M., and Zolezzi, M. [1988]. Was King Arthur a King by Definition: Methodological Reflections on Knowledge Representation in a Historic Domain. *AI Communications* **1** (1), 32–41.

Stickel, M.E. [1985*]. Automated Deduction by Theory Resolution. *Journal of Automated Reasoning* **1**, 333-356.

Stone, M.H. [1936*]. The Theory of Representation for Boolean Algebras. *Trans. A.M.S.* **40**, 37–111.

Suppes, P. [1973]. Semantics of Context-Free Fragments of Natural Languages. In Hintikka, K.J.J., Moravcsik, J.M.E., and Suppes, P. (Eds.), *Approaches to Natural Language.* Reidel Publ. Co., Dordrecht, Holland, 370–394.

Suppes, P. [1976]. Elimination of Quantifiers in the Semantics of Natural Language by Use of Extended Relation Algebras. *Rev. Int. de Philosophie* **30**, 243–259.

Suppes, P. [1979]. Variable-Free Semantics for Negations with Prosodic Variation. In Saarinen, E., Hilpinen, R., Niiniluoto, I., and Hintikka, M.P. (Eds.), *Essays in Honour of Jaakko Hintikka.* Reidel Publ. Co., Dordrecht, Holland, 49–59.

Suppes, P. [1981]. Direct Inference in English. *Teaching Philosophy* **4**, 405–418.

Tarski, A. [1935*]. Zur Grundlegung der Boole'schen Algebra. I, *Fundamenta Mathematicae* **24**, 177–198. A revised English translation is published in Tarski, A. [1956]. *Logic, Semantics, Metamathematics: Papers from* 1923 *to* 1938. Claredon Press, Oxford, 320–341.

Tarski, A. [1941]. On the Calculus of Relations. *Journal of Symbolic Logic* **6**, 73–89.

Tarski, A. [1955*]. Contributions to the Theory of Models. *Indag. Math.* **17**, 56–174.

Tarski, A. [1968*]. Equational Logic and Equational Theories of Algebras. In Schmidt, H.A., *et al* (Eds.), *Contributions to Mathematical Logic.* North-Holland, Amsterdam, 275–288.

Tarski, A., and Givant, S. [1987]. *A Formalization of Set Theory without Variables.* American Mathematical Society Colloquium Publications **41**, Providence, Rhode Island.

Vilain, M. [1985]. The Restricted Language Architecture of a Hybrid Representation System. *International Joint Conference on Artificial Intelligence*, 547–551.

Wadge, W.W. [1975]. A Complete Natural Deduction System for the Relational Calculus. Theory of Computation Report **5**, University of Warwick.

Winograd, T. [1975]. Frame Representation and the Declarative/Procedural Controversy. In Bobrow, D.G., and Collins, A.M. (Eds.), *Representation and Understanding: Studies in Cognitive Science.* Academic Press, New York, 185–210. Reprinted in Brachman and Levesque [1985, 357–370].

Winston, P.H. [1975]. Learning Structural Descriptions from Examples. In Winston, P.H. (Ed.), *The Psychology of Computer Vision.* Mc-Graw Hill, New York, 157–209. Reprinted in Brachman and Levesque [1985, 141–168].

Woods, W.A. [1975]. What's in a Link: Foundations for Semantic Networks. In Bobrow and Collins (Eds.), *Representation and Understanding: Studies in Cognitive Science.* Academic Press, New York, 35–82. Reprinted in Brachman and Levesque [1985, 217–241].

Woods, W.A. [1983]. What's Important about Knowledge Representation? *IEEE Computer* **16** (10), 22–27.

Woods, W.A. [1986]. Important Issues in Knowledge Representation. *Proceedings of the IEEE* **74** (10), 1322–1334. Adapted from Woods [1983].

Woods, W.A., and Schmolze, J.G. [1991]. The KLONE Family. To appear in *Computers and Mathematics with Applications*, Special Issue on Semantic Networks in Artificial Intelligence.

Workshop on Inheritance Hierarchies in Knowledge Representation and Programming Languages [1989*] Viareggio, Italy.