Querying Knowledge and Data Bases by a Universal Description Logic with Recursion

Dissertation
zur Erlangung des Grades
Doktor der Naturwissenschaften
der Technischen Fakultät
der Universität des Saarlandes

von

Klaus Schild

Saarbrücken
1996

Tag es Kolloquiums:	31.05.1996
Dekan:	Prof. Dr. Helmut Bley
Vorsitzender:	Prof. Dr. Joachim Petersen
Berichterstatter:	Prof. Dr. Jörg Siekmann
	Prof. Dr. Bernhard Nebel, Ulm

Preface

This work came into being during my three years' stay at the German Research Center of Artificial Intelligence (DFKI) in Saarbrücken. The main motivation of my work was to bring together three fields of Computer Science which are only apparently unrelated: Knowledge Representation, propositional logics of programs, and relational databases. Knowledge Representation as an important subfield of artificial intelligence, however, was definitely the focus of my attention: I did not consider the remaining two fields in their own right, but rather studied them from a Knowledge Representation point of view. This biased perspective does make sense in that Knowledge Representation is quite a new field compared to the other two, so that one can hope research in Knowledge Representation may benefit from such a perspective. In fact, this hope came fully true. This is because the connection between Knowledge Representation and one of the two other fields is as close as it might possibly be: Standard Knowledge Representation languages, known as *description logics*, actually turned out to be just *notational variants* of well-known propositional logics of programs. This close connection made it possible to solve some important open problems in Knowledge Representation: One problem which could be clarified in this way concerns the relative expressive power of these Knowledge Representation languages. Although this issue should be placed at the very heart of research in Knowledge Representation, it was previously hardly investigated. A correspondence with specific propositional logics of programs yields nontrivial results in this respect. The same applies to sound and complete axiomatizations and inference algorithms, particularly for relatively powerful description logics. It is perhaps most interesting that in the same way it is possible to tackle a fundamental open problem in description logics: It is the problem of recursion, especially the problem of its meaning and expressive power and how it is to be dealt with algorithmically. Again, nontrivial results can be achieved by merely establishing an adequate correspondence.

These correspondences give us a host of new and interesting results. Among others, they tell us that inferences in many description logics are *decidable*, that is, they can be computed at least in principle. These results do not, however, tell us how to deal with such languages *efficiently*. On the contrary, a precise investigation of the exact computational complexity has shown that unrestricted reasoning in description logics is inherently intractable. So the question arises whether it is yet

possible to single out a specific subset of inferences that cannot only be computed efficiently, but can also be used in practice. It has turned out that one way to enable efficiency is to restrict the admissible structure of knowledge bases in a way that they correspond to what is usually called a *database*. It is exactly this restriction that guarantees tractability. It does so even for the most powerful description logic of the literature, whose inferences are known to be undecidable for the unrestricted case. This makes the relevant Knowledge Representation language an interesting alternative to traditional database query languages: Compared to traditional query languages, this description logic is restricted in its expressive power, but is also tractable. A more or less direct consequence of this tractability result concerns reasoning in exactly the same powerful description logic that we investigated as a database query language, but in a more traditional sense. The only restriction imposed here is the existence of at least one domain-closure axiom. The result then states that this restriction alone makes the originally undecidable inferences co-NP-complete (and thus decidable). Thus decidability can be gained just in return for a simple domain-closure axiom. Requiring the existence of a domain-closure axiom is to many applications no essential restriction at all.

Statements

The Introduction summarizes the main results of [Schild, 1991a]. Chapter 2 is based on [Schild, 1993b] and [Schild, 1994a]. Preliminary results of Chapter 3 and 4 were presented in [Schild, 1994b, 1994c,1995].

Acknowledgements

I am indebted to my thesis advisor, Jörg Siekmann, who not only provided me with a comfortable research environment, but also gave me all the necessary freedom for the kind of research I did.

I am particularly grateful to Bernhard Nebel, the second reader of my thesis, who accompanied my whole research career from its very beginning as a student at the Technical University in Berlin. Bernhard Nebel gave me valuable comments on earlier drafts of the thesis, just so as Jörg Siekmann did.

I also owe gratitude to Maurizio Lenzerni, who willingly became my third reader, although he was not given much time.

Furthermore, I would like to thank all my former colleagues of the TACOS project at the DFKI, in particular Franz Baader for giving me the opportunity to watch an brilliant researcher at work, and Martin Buchheit for showing me that a colleague

can also be a good friend.

Last not least, I am most thankful to the *Graduiertenkolleg "Effizienz und Komplexität von Algorithmen und Rechenanlagen"* of the Computer Science Department at the University of the Saarland, without whose financial support my stay in Saarbrücken certainly would not have been possible.

Berlin, July 1996 Klaus Schild

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich damit, die Ausdrucksfähigkeit von Beschreibungslogiken so zu erhöhen, daß diese für praktische Zwecke einsetzbar werden. Gleichzeitig soll jedoch die Berechnungskomplexität, die selbst ausdrucksschwachen Beschreibungslogiken inhärent ist, vermieden werden.

Beschreibungslogiken (auch terminologische Logiken genannt) haben sich in den letzten 20 Jahren innerhalb der Künstlichen Intelligenz als einer der wichtigsten Formalismen zur Repräsentation von Wissen durchgesetzt. Diese Art von Logiken wird zur formalen Spezifikation von Wissenrepräsentationssystemen benutzt, die in der Tradition des legendären Systems KL-ONE stehen [Brachman and Schmolze, 1985]. Die Spezifikation mittels einer Logik erlaubt es, sowohl die Leistungen eines Systems als auch die Bedeutung des abgespeicherten Wissens unabhängig von den in der konkreten Implementierung verwendeten Algorithmen zu charakterisieren. Erst dadurch werden Systeme vergleichbar und Wissensbasen wiederverwendbar.

Wissensrepräsentationssysteme in der Tradition von KL-ONE unterstützen die Definition von Begriffen und deren Beziehungen zueinander. Begriffe werden als intensional beschriebene Mengen von Objekten und Beziehungen als zweistellige Relationen zwischen Objekten aufgefaßt. Die definierten Begriffe werden in einer Subsumptionshierarchie organisiert. Beim Aufbau dieser Hierarchie werden nicht nur die explizit angegebenen Subsumptionsbeziehungen berücksichtigt, sondern auch jene, die in den gegebenen Definitionen nur implizit vorhanden sind und daher vom System erschlossen werden müssen. Dies ist insbesondere beim Aufbau größerer Begriffshierarchien von Bedeutung, da der Mensch oft nicht mehr in der Lage ist, das Zusammenspiel vieler einzelner Definitionen zu durchschauen. Mit Hilfe einer solchen Begriffshierarchie kann dann die eigentliche Anwendung sehr einfach repräsentiert werden.

Kapitel 1 der vorliegenden Arbeit gibt eine detaillierte Übersicht über die Entwicklung dieses Teilgebietes der Wissensrepräsentation. An dieser Stelle soll es genügen zu erwähnen, daß seit dem Abschluß des KL-ONE-Projektes eine Vielzahl weiterer Systeme entwickelt wurden, wobei sich zumindest eines dieser Systeme in der industriellen Großanwendung bewährt hat [Wright *et al.*, 1993].

Trotz dieses Erfolges gibt es noch einige ungelöste Probleme in diesem Bereich. Eines der wichtigsten betrifft die adäquate Behandlung von Rekursion. Rekursion ist ein

grundlegendes Ausdrucksmittel für Definitionen, nicht nur für Beschreibungslogiken. Dennoch ist es weder in KL-ONE noch in den Nachfolgersystemen möglich, Definitionen rekursiv zu formulieren. Hierfür gibt es im wesentlichen zweierlei Gründe. Erstens war sehr lange unklar, wie eine adäquate Semantik für Rekursion in diesem speziellen Fall auszusehen hat. Bei den üblicherweise für Rekursion verwendeten kleinsten und größten Fixpunktsemantiken stellt sich selbstverständlich die Frage, welche von diesen für Beschreibungslogiken adäquat ist [Nebel, 1990a, 1991]. Darüber hinaus war es lange Zeit unklar, wie diese semantischen Varianten der Rekursion in Beschreibungslogiken algorithmisch zu behandeln sind. Algorithmen, die sowohl korrekt als auch vollständig sind, wurden bisher lediglich für die ausdrucksschwächste aller Beschreibungslogiken gefunden [Baader, 1990a].

Kapitel 2 ist diesen offenen Fragen gewidmet. Rekursion wird dabei im Kontext einer Beschreibungslogik betrachtet, die heute als Standardsprache in ihrem Bereich gilt. Es handelt sich um die von Schmidt-Schauß und Smolka [1991] eingeführte Beschreibungslogik $\mathcal{ALC}$. Rekursion wird in dieser Sprache in ihrer Allgemeinheit betrachtet, lediglich die auch in anderen Bereichen übliche Beschränkung der formalen Monotonie wird vorausgesetzt. Diese Beschränkung gewährleistet die Sinnhaftigkeit rekursiver Definitionen.

Die Ergebnisse meiner in Kapitel 2 durchgeführten Untersuchungen können wie folgt zusammengefaßt werden: Zum einem wird die langandauerende Debatte unter Beschreibungslogikern, welche der verschiedenen Fixpunktsemantiken adäquat für Rekursion ist, geklärt. Diese Klärung wird durch eine ganaue Analyse dessen erreicht, was die verschiedenen Fixpunktsemantiken auszudrücken vermögen. Bei dieser Analyse sind reguläre Ausdrücke über Beziehungen grundlegend, also beispielsweise der transitive Abschluß einer Beziehung. Ein Ergebnis dieser Analyse ist, daß im Falle der Standardsprache $\mathcal{ALC}$ *beide* Fixpunktsemantiken – gegebenenfalls koexistierend – benötigt werden, um alle zulässigen regulären Ausdrücke definieren zu können. In diesem Sinne relativiert meine Analyse diejenige von Baader [1991], der eine ähnliche Analyse für die schwächste aller Beschreibungslogiken durchführte. Baader kam zu dem Ergebnis, daß in der von ihm betrachteten Sprache die kleinste Fixpunksemantik für sich allein genommen bereits für diese Zwecke ausreicht.

Andererseits zeigt meine Analyse auch, daß beide Fixpunktsemantiken Ausdrücke zu definieren in der Lage sind, die über reguläre hinausgehen. Damit konnte mit Hilfe von regulären Ausdrücken eine *echte* untere Schranke für die Ausdrucksmächtigkeit von Rekursion in $\mathcal{ALC}$ gefunden werden. Diese echte untere Schranke konnte auf alle zulässigen ω-regulären Ausdrücke erweitert werden. Diese unteren Schranken der Ausdrucksmächtigkeit von Rekursion in $\mathcal{ALC}$ stellen die ersten bewiesenen echten unteren Schranken im Bereich der Beschreibungslogiken dar.

Meine gesamte Analyse basiert auf der Beobachtung, daß zumindest für $\mathcal{ALC}$ die verschiedenen semantischen Varianten der Rekursion mit Hilfe einer speziellen Modallogik genau charakterisiert werden können. Es handelt sich um den sogennanten

propositionalen μ-Kalkül. Der propositionale μ-Kalkül wurde von Pratt [1981] und Kozen [1983] zum Schließen über Programmschemata nebenläufiger Programme vorgeschlagen und hat sich inzwischen zu einem Standard im Bereich der automatischen Programmverifikation entwickelt. Es besteht tatsächlich ein sehr enger Zusammenhang zwischen Rekursion in $\mathcal{ALC}$ und dem propositionalen μ-Kalkül, was sich dadurch ausdrückt, daß es eine Eins-zu-Eins Abbildung gibt. Diese Beobachtung ist eine Erweiterung der grundlegenden Korrespondenz zwischen Beschreibungslogiken und propositionalen Modallogiken, die in [Schild, 1991a] beschrieben wird.

Für den propositionalen μ-Kalkül wurden korrekte und vollständige Algorithmen erfolgreich entwickelt, beispielsweise von Vardi und Wolper [1984]. Aufgrund der erwähnten Eins-zu-eins-Korrespondenz können diese Algorithmen selbstverständlich auch für die Behandlung der verschiedenen semantischen Varianten rekursiver Definitionen in $\mathcal{ALC}$ verwendet werden.

Die in Kapitel 2 präsentierten Ergebnisse wurden bereits in [Schild, 1994a] veröffentlicht.

Kapitel 3 behandelt die Rekursion in einer Beschreibungslogik, die als *universell* bezeichnet werden kann, da sie alle Sprachmittel traditioneller Beschreibungslogiken vereint. Es handelt sich um die von Patel-Schneider [1987] eingeführte Referenzsprache $\mathcal{U}$. Zum ersten Mal in der Literatur terminologischer Logiken wird hier nicht nur die rekursive Definition von Begriffen, sondern auch von Beziehungen betrachtet. Die erforderliche Verallgemeinerung der in Kapitel 2 eingeführten semantischen Grundlagen ist mit keinerlei Schwierigkeiten verbunden. Die Ausdrucksfähigkeit rekursiver Definitionen in $\mathcal{U}$ wird exemplarisch verdeutlicht, indem gezeigt wird, daß Begriffe wie azyklische Graphen, Bäume, balancierte Bäume, sowie erfüllbare Und-Oder-Graphen einfach und elegant definiert werden können. Allerdings muß diese zusätzliche Ausdrucksfähigkeit damit erkauft werden, daß eine algorithmische Behandlung nicht mehr gleichzeitig korrekt und vollständig sein kann. Die zusätzliche Ausdrucksfähigkeit resultiert mit anderen Worten in der *Unentscheidbarkeit* der Systeminferenzen.

Kapitel 4 behandelt die Frage, unter welchen Bedingungen Systeminferenzen innerhalb dieses sehr ausdrucksstarken Rahmens dennoch effizient behandelt werden können. Auf den ersten Blick scheint dieses Unterfangen im Widerspruch zu der generellen Unentscheidbarkeit der Systeminferenzen zu stehen. Eine Möglichkeit, diesen Wiederspruch aufzulösen, ist, die zulässige *Struktur* von Wissensbasen einzuschränken. Eine naheliegende Einschränkung ist es beispielsweise, keinerlei unvollständiges Wissen zu erlauben, so daß die zulässigen Wissensbasen sich auf das beschränken, was man traditionell als *Datenbank* bezeichnet.

Eines der wichtigsten Ergebnisse der vorliegenden Arbeit ist, daß diese Einschränkung es tatsächlich ermöglicht, alle Systeminferenzen in polynomialer Zeit zu ziehen, wobei sich das entsprechende Problem auch als vollständig für die Klasse P herausstellt.

Aus beschreibungslogischer Sicht ist dieses Ergebnis von großer Bedeutung, da es sich um das *erste* wirkliche Polynomialzeitergebnis handelt. Alle anderen Ergebnisse dieser Art setzen eine künstliche Vorverarbeitung der gegebenen Begriffshierarchie voraus, die diese möglicherweise exponentiell vergrößert [Levesque and Brachman, 1987, Donini *et al.*, 1991]. Zudem gelten diese Resultate lediglich für Sprachen, deren Ausdrucksfähigkeit im Vergleich zu der von mir betrachteten Sprache minimal ist.

Aus der Sicht traditioneller Beschreibungslogiken ist eine relativ einfach zu beweisende Konsequenz dieses Polynomialzeitergebnisses besonders interessant. Diese besagt, daß die üblichen Inferenzen in der universellen Beschreibungslogik $\mathcal{U}$ mit Rekursion lediglich durch das Hinzufügen eines speziellen Axiomes co-NP-vollständig und somit entscheidbar werden. Dieses Axiom muß lediglich die in Betracht zu ziehenden Objekte auf eine vorgegebene *endliche* Menge beschränken. Hierbei ist zu beachten, daß in vielen Anwendungen eine solche Menge ohnehin vorgegeben ist.

Aus Sicht der Datenbanktheorie führt das hier erzielte Polynomialzeitresultat eine neue Anfrage- und Schemasprache für Datenbanken ein, die gleichzeitig komfortabel und in polynomialer Zeit auswertbar ist. "Polynomial auswertbar" bezieht sich in diesem Zusammenhang auf die *kombinierte Komplexität* und nicht etwa auf den wesentlich schwächeren Begriff der *Datenkomplexität* [Vardi, 1982]. Diese neue Anfragesprache deckt dabei (wie fast alle traditionellen Anfragesprachen) nicht alle theoretisch möglichen polynomialen Anfragen ab, ist jedoch strikt mächtiger als der binäre Relationenkalkül.

Aus Datenbanksicht stellt sich natürlich die Frage, ob das vorgestellte Resultat in Hinsicht auf die für Datenbanken unerläßlichen unbekannten Werte erweiterbar ist. Leider stellt sich heraus, daß dies nicht der Fall ist, da das mit diesen unbekannten Werten darstellbare unvollständige Wissen bereits ausreicht, um die co-NP-Vollständigkeit der Systeminferenzen zu verursachen. Allerdings ist es mir gelungen, für diesen Fall einen polynomialen, approximativen Algorithmus zu entwickeln, der zwar immer korrekt, aber nur dann vollständig ist, wenn kein Wert unbekannt ist.

Kapitel 5 schließlich faßt die gesamte Arbeit ausführlich zusammen und diskutiert ihre wichtigsten Resultate.

Contents

List of Figures

Chapter 1

Introduction

This introduction is not only to trace out the line of research that led to the present thesis, but also to make explicit those hypotheses which are presupposed without further analysis.

1.1 The Knowledge Representation Hypothesis

One of the key assumptions of research in artificial intelligence is that the simulation of intelligent behavior is not possible without vast amounts of knowledge. It was Smith [1982] who probably delineated this hypothesis best:

> Any mechanically embodied intelligent process will be comprised of structural ingredients that a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and b) independent of such external semantical attribution, play a formal but causal and essential role in engendering the behavior that manifests that knowledge.

This hypothesis explains the outstanding role of Knowledge Representation as a subfield of Artificial Intelligence. What Smith outlined above actually came to be called the *Knowledge Representation Hypothesis*. This hypothesis, however, has its own right as a general software-engineering paradigm, independent of whether or not the *mechanically embodied process* that we have in mind can be called *intelligent*. From the software-engineering point of view, the Knowledge Representation Hypothesis just describes a specific way of designing systems. There are two major characteristics that any system must posses if designed this way. First, it must be possible to view the basic data structures of the system as *propositions* representing the overall knowledge exhibited by the system. This presupposes that these data structures are

expressions drawn from some language having something like a truth theory. That is to say, for any such expression it must be clear how the world would have to look like in order to make it true. This is not to say, however, that the syntax of these expressions must resemble in any sense that of classical predicate logic. The second criterion such knowledge-based systems must posses is that the represented knowledge should play a causal role in the behavior of the system itself. In other words, the system's behavior should be determined exclusively by the knowledge that it explicitly represents. In combination with the first characteristic of knowledge-based systems, this implies that any implementation detail should be totally irrelevant for the system's behavior other than perhaps its efficiency.

1.2 The Semantics of Semantic Networks

Of course, once a system is accepted to be knowledge based, some method of representation has to be fixed. One popular method originating in natural-language processing are networks of interrelated semantic terms. The most influential early effort on this kind of *semantic networks* is certainly Quillian's [1966, 1967, 1969]. He aimed at a "theory of the structure of human long-term memory" [1967] which was supposed not only to bear psychological plausibility, but also to facilitate natural-language processing. Quillian's representation is somehow patterned in the organization of a dictionary. It is centered around what he called *type nodes*, where each type node represents a single English word concept. A definition of such a type node consists of a group of other type nodes (actually being pointers to them) with associative links of various kinds between them. These links include modification links (attaching modifying properties to a supertype), conjunction and disjunction links, as well as links pointing to the subject and the object of a verb. An outstanding role can be attributed to a special kind of link, the so-called "is a" or supertype link. This type of link makes it possible to organize word concepts in a generalization hierarchy. The hierarchical organization is supposed to make possible not only a psychologically plausible memory model, but also a computationally efficient one.

Quillian also gave a simple processing scheme to uncover information residing implicitly in semantic networks. In particular, given two word concepts, potential relationships between them might be inferred by means of a method which Quillian called *spreading activation*. This method propagates in a breadth-first manner some kind of activation signal through all links emanating from the corresponding type nodes until an intersection point has been reached. The paths to such intersection points are then postulated to indicate potential relationships between the two word concepts.

There were many efforts on evolving semantic networks following Quillian's initial work. These include the works of Schank [1973], Hendrix [1979], and Shapiro [1979],

just to mention a few of them. For a good overview of the evolvement of semantic networks the reader is referred to [Brachman, 1979] and [Sowa, 1991].

The idea of organizing knowledge in associative networks turns out to be fundamental, especially when knowledge about the generalization hierarchy is incorporated. However, as Woods [1975] put forward, typical semantic networks suffer from a lack of precise meaning. What Woods was concerned with was the semantics of the representation itself. Without such a semantics it is left open what a representation is supposed to represent. Consequently, different representations cannot be compared to one other. This very issue had been "previously brushed aside under the auspices of 'intuition'," as Brachman [1979] put it. The apparent intuitiveness of semantic networks, however, can be very misleading. One example of Woods' is that "is a"-links can have in principle two meanings. They can be read either as subclass or as membership relations. In semantic networks the decision which of these different readings is to be put into effect is left to the user's personal taste, so that representations might be ambiguous. But even if this kind of ambiguity was resolved, the precise meaning of a "is a"-link in the sense of, say, a subset relation would remain mysterious. In particular, no formal criterion is available in which cases such a link is to be placed and what the consequences of doing so are. On the other hand, links of this type play a crucial role in the system's behavior, especially in inheriting properties of a supertype to a subtype.

However, there were also early efforts on giving semantic networks a precise semantics. The first attempt leading in this direction is due to Hayes [1980], who translated parts of semantic networks into first-order logic.

1.3 The KL-ONE Perspective

In his Ph.D. thesis, supervised by Woods himself, Brachman [1977] followed the line of research traced out by the work on semantic networks. However, in acknowledging his supervisor's criticism of some of the most serious weaknesses of semantic networks, he actually went far beyond that line of research. Brachman's thesis led directly to the development of a knowledge representation system, called KL-ONE [Brachman and Schmolze, 1985]. KL-ONE's origin in semantic networks comes into light in at least two aspects. First, its graphical syntax much resembles that of semantic networks. In particular, its graphical notation is comprised of labeled nodes interrelated by associative links, including arcs corresponding to "is a"-links. Second, its original intent was to represent 'semantic' knowledge about English word concepts. In fact, "initially, the KL-ONE-project set out to develop a set of representational conventions that would be sufficient to express any concept expressible in natural language" say Woods and Schmolze [1992]. This statement must be judged in view of the fact that Woods was the principal investigator of the project and Schmolze was the key

developer of the system.

It would be entirely inappropriate, however, to emphasize solely KL-ONE's similarities with semantic networks. For KL-ONE is commonly considered as a landmark not because it stays within the line of ideas set out by its predecessors, but because it extends beyond this line of ideas. In fact, its very intent was to surmount those deficiencies of semantic networks which have been uncovered inter alia by Woods [1975] and Brachman [1983]. In particular, KL-ONE radically deviates from its origins in at least three aspects:

Over and above all, representational notations are given without exception precise meanings by translating them into first-order logic [Schmolze and Israel, 1983]. This gives rise to representations the meaning of which is determined independent of any of the user's imaginations and independent of domain-specific terms other than those explicitly represented.

Second, a clear distinction is made between nodes representing classes of objects and those nodes which represent a single object only. A node of the former type is called in KL-ONE's terminology *concept*, whereas a node of the latter type is referred to as *nexus* or, as is nowadays more common, *individual* [Brachman and Schmolze, 1985, Section 10]. This makes it in turn possible to differentiate between links stating a subset relation and those which state a membership relation, a fundamental distinction blurred in semantic networks under the heading of "is a"-links. Links of the former type are called *subsumption* links, while links of the latter type are referred to as *assertional* links.

Last but not least, based on the precise semantics of KL-ONE's notational language, it is possible (and even straightforward) to define the exact meaning of subsumption and assertional links. This in turn renders possible an automatic classification of new concepts and individuals into a generalization hierarchy of previously defined concepts. *Classification* refers to the ability to insert a new concept definition into the subsumption hierarchy such that the new concept is directly linked to the most specific concepts it is subsumed by and to the most general concepts that it in turn subsumes. The term *recognition* is normally used instead of *classification* to refer to the process of computing the most specific concepts a given individual is an instance of. Without formal semantics, there would be no general criterion whatsoever, in accordance with which an algorithm can automatically infer implicit subsumption or assertional relations. It was Lipkis [1982] who devised the first classifier for KL-ONE, see also [Schmolze and Lipkis, 1983]. Recognition, however, was hardly tackled during the KL-ONE project. A notable exception was the work of Mark [1982], tackling a related problem. This problem concerns the computation of all individuals being instances of a given concept, the *realization* problem.

All in all, the KL-ONE project can be called successful in the long run: It not only led to a substantial amount of still ongoing research, including the development of

various systems, but at least one of its descendants eventually reached the stage of large-scale industrial applications. It was the CLASSIC system developed at AT&T which gained this honor [Wright *et al.*, 1993]. A thorough overview of the KL-ONE-family is given in [Woods and Schmolze, 1992].

1.4 Description Logics

From the very beginning of KL-ONE's history, special logics have been devised to underpin the representational foundations of KL-ONE and similar systems. These logics are to describe in a formally precise manner what can be represented by the relevant system and what computational service is provided. The first rigorously defined logic of this kind is due to Brachman and Levesque [1984]. Logics in the spirit of Brachman and Levesque are nowadays usually called *description logics*, but are also known under terms such as *concept language* or *terminological logic*. Of course, various logics of this kind are needed in order to capture all existing systems. What all these description logics have in common is the fact that they are comprised of two different syntactic categories, known as *concepts* and *roles*. While concepts are to be thought of as representing particular sets of objects, roles denote binary relations among objects. Roles are often treated as atomic, in which case they are restricted to simple *role names*. On the other hand, a description logic would not deserve its name if concepts could not be composed out of other concepts and roles. In fact, concepts can always be built up from simple concept and role names by applying a predefined set of so-called *concept-structuring primitives*.[1] The minimal repository of concept-structuring primitives shared by all description logics includes *concept conjunction* and *universal role quantification*. It is therefore convenient to refer to a description logic comprising no concept and role-structuring primitives other than this minimal repository as the *minimal* or *very weakest description logic*. Concept conjunction is an operator, usually designated by $\sqcap$, which takes two arbitrary concepts as its arguments and forms a new concept. The resulting concept represents the intersection of the two sets of objects represented by its arguments. Universal role quantification is an operator, usually designated by $\forall$:, which takes a role and a concept as its arguments and yields a new concept. In this case the resulting concept represents the set of all those objects for which each object related to them by the given role is an instance of the given concept. To be more precise, by *concept* and *role* we refer here to the set of objects and the binary relation among objects represented by the concept or role rather than the concept and role themselves. If, for instance, the concept names Student and GradCourse as well as the role name enrolled_in are given, then a compound concept such as Student $\sqcap$ $\forall$enrolled_in:GradCourse can be

[1] Brachman [1979] uses the term *epistemological primitive* rather than *concept-structuring primitive*. We shall, however, not adopt Brachman's terminology because the primitives to be referred to are dealing neither with sources of knowledge nor with justifications of beliefs.

formed out of them. Intuitively, this concept captures all those students who are enrolled in no courses other than graduate courses. But in contrast to what one might expect, this even includes all students enrolled in no course at all. This is due to the nature of universal quantification having no existential impact whatsoever. Of course, there are also concept-structuring primitives capable of excluding such idle fellows. In particular, existential role quantification is able to do so. Existential role quantification which is entirely analogous to universal role quantification is not provided by all systems though. What is almost always provided is a somewhat restricted version taking a role as its sole argument. In this case, existential role quantification is of the form $\exists R$ rather than $\exists R{:}C$. Concepts of the form $\exists R$, however, do suffice to preclude 'idle' students from being included in the concept above. Simply conjoining $\exists$enrolled_in impose the appropriate additional restriction on the concept.

In a formally precise fashion, meanings can be given to concepts and roles with the help of so-called *interpretations*. Such an interpretation, $\mathcal{I}$, consists of three components. The first component, $\Delta^{\mathcal{I}}$, is said to be the *domain* of the interpretation and is just an arbitrary nonempty set. Another component then fixes the interpretation of all concept and role names and is referred to as a *valuation over* $\Delta^{\mathcal{I}}$. Such a valuation is a function mapping concept names to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations over $\Delta^{\mathcal{I}}$. The remaining third component consists of what is called an *interpretation function*. Such an interpretation function, $.^{\mathcal{I}}$, extends $\mathcal{V}$ in a way that allows it to deal with arbitrary concepts and roles. In particular, it specifies how to interpret the concept-structuring primitives of interest by mapping them to particular set operations on $\Delta^{\mathcal{I}}$ or $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Any interpretation function maps $C \sqcap D$, for instance, to $C^{\mathcal{I}} \cap D^{\mathcal{I}}$, while $\forall R{:}C$ and $\exists R$ are mapped to $\{d \in \Delta^{\mathcal{I}} : R^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\}$ and $\{d \in \Delta^{\mathcal{I}} : R^{\mathcal{I}}(d) \neq \emptyset\}$ respectively. Here, $R^{\mathcal{I}}(d)$ is assumed to denote what might be called the application of the binary relation $R^{\mathcal{I}}$ to d. That is to say, it denotes the set of all those elements, e, of the domain for which $\langle d, e\rangle$ is a member of $R^{\mathcal{I}}$.

What all description logics also have in common is a simple facility to define concepts by abbreviating possibly compound concepts to simple concept names. If, for instance, GradStudent is a concept name, then a definition of this kind might look as follows:

$$\mathsf{GradStudent} \doteq \mathsf{Student} \sqcap \forall \mathsf{enrolled\_in{:}GradCourse} \sqcap \exists \mathsf{enrolled\_in}.$$

Expressions of this type are called *concept introductions*. Meaning is given to them by imposing additional restrictions on interpretations. In particular, every interpretation function has to map the concept name appearing at their left-hand side to exactly the same set as the concept at their right-hand side. Interpretations which meet this restriction are referred to as *models*.

A finite set of concept introductions then forms a *terminology*, at least if the condition is met that there is no concept name appearing more than once at a left-hand side of

a concept introduction. Often terminologies are additionally required to be *acyclic*, which means that the concept introductions can be ordered in a certain way. It must be possible to order them in such a way that the right-hand side of each concept introduction uses no concept name appearing at the left-hand side of some preceding concept introduction, including the concept introduction itself. This is to avoid cyclic dependencies among concept introductions. Of course, natural recursion is thereby excluded too. It is entirely straightforward to give meanings to terminologies, at least when they are acyclic. As with concept introductions, this is done by specifying what their models are. The models of a terminology are all those interpretations which are models of every concept introduction of the terminology.

The most important query posed to terminologies is whether one concept subsumes another, that is, whether the first one is more general than the second. The syntactic means by which such queries can be expressed is called an *inclusion axiom*. If both C and D are arbitrary concepts, then $C \sqsubseteq D$ forms a legal inclusion axiom. Of course, a model of such an inclusion axiom is an interpretation whose interpretation function reflects the intended subset relationship, i.e., the interpretation function has to map C to a subset of the set to which D is mapped. Given a terminology, $\mathcal{T}$, it then can be defined that a concept, C, *subsumes* another concept, D, just in case every model of the terminology is a model of $D \sqsubseteq C$ as well. In analogy to the corresponding notation for logical consequence, we write in this case $\mathcal{T} \models D \sqsubseteq C$. An inference of this kind—though a simple one—is that Student subsumes GradStudent with respect to the following terminology:

$$\begin{aligned} \mathsf{Student} &\doteq \mathsf{Person} \sqcap \forall \mathsf{enrolled\_in{:}Course} \sqcap \exists \mathsf{enrolled\_in}, \\ \mathsf{GradStudent} &\doteq \mathsf{Student} \sqcap \forall \mathsf{enrolled\_in{:}GradCourse} \sqcap \exists \mathsf{enrolled\_in}. \end{aligned}$$

If two concepts subsume each other, we say that they are *equivalent*. Moreover, a concept is said to be *coherent* if and only if it is not subsumed by the concept $\bot$. The symbol $\bot$ denotes a special concept being mapped to the empty set by every interpretation function.

While terminologies are to encapsulate the 'terminology' inherent in the application domain of interest, there clearly has to exist also a means of describing the application domain itself. Whenever it is possible this description of the application domain should make use of the extracted terminology. Such a description of the application domain is usually given by providing concepts and roles with particular instances. Of course, instances of concepts are objects of the domain, while instances of roles are always ordered pairs of such objects. As a matter of fact, instead of referring to the objects of the domain themselves, specific place holders are used. These place holders are referred to as *individual names*; and it is an *assertion* which provides concepts and roles with instances of the kind just described. A finite collection of assertions finally forms what is usually called a *knowledge base*. A typical knowledge

base of this kind looks as follows:

$$\mathsf{annie{:}GradStudent},$$
$$\langle \mathsf{annie}, \mathsf{DB\_IV} \rangle \mathsf{{:}enrolled\_in}.$$

It should be clear how to give assertions as well as knowledge bases a precise formal meaning in terms of models, just as it is already done for concept introductions and terminologies. Assertions can be used as queries, too, in which case they usually are evaluated with respect to a knowledge base along with a terminology. The answer depends in this case on whether every interpretation which is a model of both the knowledge base and the terminology is also a model of the query. Inferences of this kind include answering in the positive such queries as DB_IV:GradCourse when posed to the sample terminology and knowledge base above. This completes our brief outline of the kernel which KL-ONE and any of its descendants have in common.

At this stage, it is important to note that when presenting this minimal kernel, we did not intend to convince the reader of the representational or inferential merits of this kernel. We did intend, however, to demonstrate that systems such as KL-ONE can be specified at an abstract level independent of any specific detail of their concrete implementation. As a matter of fact, the actual expressive power of KL-ONE and any of its descendants extends far beyond what was introduced so far.

1.5 A Fundamental Tradeoff

Formal foundations would be meaningless if a given system's behavior was not required to match its formal specification. The minimal condition to be imposed on the system's behavior is that it is *sound*. Soundness means that whenever the system draws some particular inference, this inference is actually valid with respect to the formal semantics. This criterion alone, however, is not very meaningful either. This is because systems drawing only trivial inferences or even no inference at all must inevitably be called sound. In particular, a system is always sound if it merely retrieves all those facts which are explicitly stored without drawing any additional inference. The same applies to systems drawing no inference at all. This is why one should impose on the system's behavior not only the condition of soundness, but also that of *completeness*. Completeness refers to the property that whenever a particular inference is valid according to the formal semantics, the system is actually capable of drawing this inference within a finite period of time. One major research topic is therefore to devise inference algorithms being sound as well as complete. This is a key issue not only for description logics, but for Knowledge Representation as a whole.

But once there is the commitment to sound and complete inference algorithms, the computational behavior of the system is tightly coupled with the general computa-

tional complexity of the inference problem itself. This explains the prominent role of complexity theory in the field. Complexity theory is concerned with the analysis of mathematically defined problems rather than the analysis of a given algorithm. This abstraction enables us to predict within certain bounds the computational behavior of *any* algorithm being sound and complete for the relevant problem. Suppose, for instance, a lower computational complexity bound for computing the subsumption relation has been established. Then we know for sure that *every* sound and complete algorithm will terminate on an infinite number of inputs not without consuming the amount of computational costs specified by the given lower bound. On the other hand, suppose some upper computational bound has been proven, but the algorithm at hand terminates on at least one input consuming computational costs which strictly exceed those specified by the upper bound. In this case, we know in turn for sure that the given algorithm is not *optimal*.

Of course, the more expressive a representation language, the harder is the computation of the corresponding inference. As Levesque and Brachman [1987] put forward, this raises the issue of finding an appealing tradeoff between the expressive power and the computational complexity. To single out such a tradeoff turned out to be one of the grand challenges, especially for description logics. Actually, to find such a tradeoff is not as easy as was originally thought. Although description logics were originally claimed to be computationally feasible, in 1988 it was possible to show that there are description logics satisfying this criterion by no means, even with the most liberal interpretation of 'feasible' one can think of. In particular, in [Schild, 1988] a description logic was given for which there does not exist any sound and complete algorithm at all. In such a case the corresponding inference problem is said to be *undecidable*. Since then a great number of similar undecidability results were established, including those given in [Patel-Schneider, 1989b] and [Schmidt-Schauß, 1989], just to mention the most important ones. The major challenge is therefore to single out concept languages the expressive power of which is sufficient to embrace a given range of application scenarios while inducing computational costs tolerable for this range of application scenarios. This certainly includes the exploration of the lower and upper computational complexity bounds. Another important issue is, of course, to compare different description logics in their expressive power. This issue, however, was hardly tackled at all, with the exceptions of [Baader, 1990a, 1991] as well as [Schild, 1994a]. In any case, as Doyle and Patil [1991] pointed out, we should always keep in mind that limited expressive power may threaten the goal of a general purpose representation language.

1.6 The Modal Logic's Perspective on Description Logics

In their seminal paper, Schmidt-Schauß and Smolka [1991] enhanced the expressive power of the minimal description logic by a concept-structuring primitive which was not investigated previously. They considered *concept negation* in its full generality. The reasons for Schmidt-Schauß and Smolka in doing so were twofold. First of all, and certainly above all, negation is an inherent part of any natural language and so it should also be included in formal languages supposed to capture fragments of it. If, for instance, our language includes concepts such as Student and GradStudent, from this point of view, there is no reason other than arbitrariness to declare expressions such as $\mathsf{Student} \sqcap \neg\mathsf{GradStudent}$ prohibited. On the other hand, concept negation provides for *concept disjunction* as well as for *existential role quantification* in its full generality. As is well-known, $C \sqcup D$ and $\exists R{:}C$ is equivalent to $\neg(\neg C \sqcap \neg D)$ and $\neg\forall R{:}\neg C$ respectively. Schmidt-Schauß and Smolka baptized the resulting description logic $\mathcal{ALC}$, an acronym for $\mathcal{A}$ttributive concept $\mathcal{L}$anguage with $\mathcal{C}$omplements. Nowadays $\mathcal{ALC}$ is something like a *standard* description logic, placed at the very center of theoretical research.

What makes their paper one of the landmarks in the history of description logics is that for the first time a sound and complete algorithm is given for a nontrivial description logic. Such an algorithm was previously known solely for a very small description logic. It was Levesque and Brachman [1987], who established both soundness and completeness in the latter case.

Rather than devising an algorithm for deciding subsumption between two concepts, Schmidt-Schauß and Smolka were concerned with deciding the coherence of single concepts. One of their motivation in doing so was that the two problems are tightly related. In particular, once the problem of deciding coherence is solved, subsumption between two concepts can be decided, too, at least if concept negation is available. This is because a concept, C, subsumes another concept, D, just in case the concept $D \sqcap \neg C$ is not coherent. Vice versa, coherence of a concept can, of course, be decided on the basis of whether or not the concept is subsumed by any other concept known to be incoherent, say, $CN \sqcap \neg CN$. Schmidt-Schauß and Smolka's algorithm is able to decide correctly the coherence of any concept of $\mathcal{ALC}$. The algorithm determines the coherence of the concept itself, but does not take into account any terminology.

Apart from the soundness and completeness of their algorithm, Schmidt-Schauß and Smolka proved that their algorithm is also near optimal. The worst case complexity of their algorithm actually matches the general lower complexity bound of coherence in $\mathcal{ALC}$ up to a polynomial. The lower bound that Schmidt-Schauß and Smolka were able to establish is as follows. They proved that every sound and complete algorithm for this problem terminates on an infinite number of inputs not before consuming

working space at best bounded above by some fixed polynomial in the size of the input. In technical terms, this means that the corresponding problem is PSPACE-hard. On the other hand, the working space consumed by the specific algorithm that they developed is in the worst case quadratically bounded in the size of the input.[2] The algorithm is therefore near optimal in this sense.

The results of [Schmidt-Schauß and Smolka, 1991] are well-known by now and form a cornerstone of the field. It was not discovered until recently, however, that equivalent results had been known for nearly one and a half decade before the publication of Schmidt-Schauß and Smolka's paper: Equivalent results were in fact already known in a field which appears to be unrelated at first sight. It is the field of *modal logic* in which equivalent results had previously been established.

A reason for the comparatively late discovery of this relationship may be the fact that from the very beginning, description logics have always been viewed as sublanguages of first-order predicate logic. When adopting this point of view, concept and role names correspond to unary and binary predicates respectively. The translation of all other well-formed expressions of, say, $\mathcal{ALC}$, are then determined by the following translation scheme:

$$\begin{aligned}
\forall R{:}C &\rightsquigarrow \forall Y.\tilde{R}(X,Y) \Rightarrow \tilde{C}(Y),\\
C \sqcap D &\rightsquigarrow \tilde{C}(X) \wedge \tilde{D}(X),\\
\neg C &\rightsquigarrow \neg\, \tilde{C}(X),\\
C \doteq D &\rightsquigarrow \forall X.\tilde{C}(X) \Leftrightarrow \tilde{D}(X),\\
C \sqsubseteq D &\rightsquigarrow \forall X.\tilde{C}(X) \Rightarrow \tilde{D}(X),\\
a{:}C &\rightsquigarrow \tilde{C}(a),\\
\langle a,b\rangle{:}R &\rightsquigarrow \tilde{R}(a,b).
\end{aligned}$$

Of course, one has to take special care of variables, particularly in the presence of nested role quantifications. Suppose, for instance, in the course of translating $\forall R{:}\forall S{:}C$, we are about to translate the subconcept $\forall S{:}C$. If the concept itself is translated into $\forall Y.\tilde{R}(X,Y) \Rightarrow (\widetilde{\forall S{:}C})(Y)$, then the subconcept $\forall S{:}C$ has to be mapped to $\forall Z.\tilde{S}(Y,Z) \Rightarrow \tilde{C}(Z)$ rather than to $\forall Y.\tilde{S}(X,Y) \Rightarrow \tilde{C}(Y)$.

Until recently this had been the predominant view on description logics. Careful inspection of this translation, however, reveals that it is correct though in some sense not the best one. This is because the translation into first-order logic lacks the property of a one-to-one mapping. If this translation were a one-to-one mapping, it would translate not only every syntactically well-formed expression of $\mathcal{ALC}$ into first-order logic, but it would, vice versa, also translate every well-formed first-order formulae into $\mathcal{ALC}$. This is, however, not possible in the case of $\mathcal{ALC}$, which does not

[2] In the original paper, the algorithm was analyzed by mistake as consuming in the worst case only linear space.

cover first-order logic to its full extent. As a matter of fact, no existing description logic covers all first-order predicate logic formulae. To discover suitable *fragments* of first-order logic is, of course, the whole business of the field. But only a one-to-one mapping would allow us to transfer results such as complete axiomatizations established for first-order logic to the description logic's case.

The syntax of $\mathcal{ALC}$, however, suggests a rather different translation. At least all those concepts of $\mathcal{ALC}$ which do not involve any occurrence of a role quantification can be translated to formulae of *propositional logic* in the most natural way. These concepts clearly correspond to propositional formulae in a one-to-one fashion. From a syntactic point of view, the translation into propositional logic is rather natural. From a semantic point of view, however, it gives rise to some confusion. How could one translate concepts into propositional formulae and vice versa, even though the former are supposed to denote sets of objects, while the latter denote just truth values? In other words, it is not immediately evident how this syntactically rather natural translation is reflected semantically.

This confusion can be readily solved though. The only thing one has to do is to adopt Kripke's [1963] well-known *possible worlds* approach. In order to give modalities such as 'necessarily' and 'possibly' precise meanings, Kripke suggested to view propositions as denoting sets of possible worlds rather than single truth values. Such a set of possible worlds is to be thought of as representing all those situations in which the relevant proposition holds. He thereby replaced a homogeneous view on the state of affairs by a pluralistic one. A formula is then considered to be necessarily true just in case it holds in all the worlds considered to be possible. On the other hand, a formula is possibly true just in case it holds in at least one of the worlds considered to be possible. What is or is not considered possible is reflected by some fixed binary relation among possible worlds, usually referred to as *accessibility relation*. Hintikka [1969] generalized Kripke's possible world setting in view of multiple accessibility relations. In Hintikka's setting each of these different accessibility relations is attributed to what might be called *agent*. In this generalized framework, the modality of necessity thus captures *personal* belief or knowledge. In any case, it is convenient to impose certain restrictions on accessibility relations if the variety of different modalities is to be taken into account. However, as far as the *normal propositional modal logic* is concerned, no such restriction at all is invoked. In this sense, normal modal logic can be considered the standard modal logic. In the multiple agent case, normal propositional modal logic is simply referred to as $\mathbf{K}_{(m)}$, the index m standing for m different agents.

From a formal point of view, $\mathbf{K}_{(m)}$ extends propositional logic just by one additional formation rule. This rule states that if α is an arbitrary formula, then so is $K_i\alpha$. Here, K_i must be one of the $K_1, ..., K_m$ different agents. Modal formulae of the form $K_i\alpha$ are to be read as "agent i believes in the truth of α." Meanings are given to any formula of $\mathbf{K}_{(m)}$ with the help of what came to be called a *Kripke structure*. Such

a Kripke structure is comprised of three components. The first component is an arbitrary set, containing all relevant possible worlds. The second component is a so-called *assignment function*. Such an assignment function, π, assigns a particular set of possible worlds to each propositional variable. This set contains all those possible worlds in which the corresponding propositional variable is supposed to hold. The remaining third component contains for each of the m different agents, K_i, a binary relation, $\mathcal{P}_i$, among possible worlds. If $w\mathcal{P}_i w'$, that is, if $\mathcal{P}_i$ contains as its member the ordered pair $\langle w, w' \rangle$, then agent K_i is assumed to consider w' as possible whenever K_i is in the state of belief represented by w.

The truth value of an arbitrary formula, α, of $\mathbf{K}_{(m)}$ in an arbitrary possible world, w, is determined with respect to such a Kripke structure. Technically speaking, if $\mathcal{M}$ is such a Kripke structure, then an expression of the form $\mathcal{M} \models_w \alpha$ is evaluated. This expression is intended to mean that the possible world w is among those in which α holds. Expressions of this kind are evaluated according to the following semantic rules. If P is a propositional variable, then $\mathcal{M} \models_w P$ holds if and only if w is a member of $\pi(P)$. Of course, π is the assignment function of the given Kripke structure. The second semantic rules states that $\mathcal{M} \models_w \alpha \wedge \beta$ holds if and only if both $\mathcal{M} \models_w \alpha$ as well as $\mathcal{M} \models_w \beta$ holds. Another semantic rule deals with negation. It states that $\mathcal{M} \models_w \neg\alpha$ holds just in case $\mathcal{M} \models_w \alpha$ does not. Finally, $\mathcal{M} \models_w K_i\alpha$ holds if and only if $\mathcal{M} \models_{w'} \alpha$ holds whenever w' is a possible world such that $w\mathcal{P}_i w'$. Satisfiability is then to be understood in the framework of $\mathbf{K}_{(m)}$ as follows: A formula, α, is *satisfiable* if and only if there is at least one Kripke structure, $\mathcal{M}$, and at least one possible world, w, such that $\mathcal{M} \models_w \alpha$.

An alternative but ultimatively equivalent style of semantics is often stated instead of the one just given. Clearly the mapping π can be extended to deal with arbitrary formulae of $\mathbf{K}_{(m)}$ in a way suggested by the given semantic rules. In particular, $\wedge$ and $\neg$ have then to be mapped to set intersection and complementation respectively. The mapping of $K_i\alpha$ is somewhat more involved. $K_i\alpha$ is mapped to the set of all those possible worlds, w, such that every possible world related to w by the accessibility relation $\mathcal{P}_i$ is a member of the set to which the extended version of π maps α. In this way, the traditional semantic rules given above have directly been codified into the assignment function π. But then it suffices to make a simple check whether or not w is a member of the set to which the so extended π maps the given formula.

At this stage, the semantic parallel with $\mathcal{ALC}$ should be evident. The only thing one has to realize is the fact that nothing prevents us from reading objects as possible worlds and vice versa. Then it should be obvious that those operators which correspond to each other by their syntax are mapped to exactly the same set-theoretical operations. The modal logic $\mathbf{K}_{(m)}$ is actually nothing but a notational variant of $\mathcal{ALC}$. The following table should suffice to explain this close relationship:

$\mathcal{ALC}$	$\mathbf{K}_{(m)}$
CN	P_{CN}
$C \sqcap D$	$\alpha_C \wedge \alpha_D$
$\forall R{:}C$	$K_R \alpha_C$
C is coherent	α_C is satisfiable

This proves that the close relationship with propositional modal logic becomes evident as soon as description logics are viewed as propositional logics rather than first-order logics. The first-order logic point of view can therefore be considered as the main obstacle for uncovering the quite obvious one-to-one correspondence with propositional modal logic.

The correspondence with $\mathbf{K}_{(m)}$ was first observed in [Schild, 1991a].[3] Once this observation is made, the host of results established for $\mathbf{K}_{(m)}$ can immediately be carried over to the description logic's case. These include lower and upper computational complexity bounds for deciding satisfiability in $\mathbf{K}_{(m)}$ [Ladner, 1977]. Notably, these bounds correspond exactly to those established by Schmidt-Schauß and Smolka [1991].[4]

The one-to-one correspondence with propositional modal logic yields not only results which are anyway known for the corresponding description logics, but there are also many interesting consequences previously unknown. This particularly applies to results on the axiomatization, on the model theory as well as on the expressive power of $\mathcal{ALC}$. For instance, Lemmon [1966] gave a complete equational axiomatization of the equivalence relation in propositional modal logic. His axiomatization is based on that for Boolean algebras. Notably, compared to the axiomatization of Boolean algebra, Lemmon needed not more than two additional equations so as to deal with modal operators. The additional equations $K_i true = true$ and $(K_i\alpha) \wedge (K_i\beta) = K_i(\alpha \wedge \beta)$ do suffice. When translated into $\mathcal{ALC}$ these two equations correspond to $\forall R{:}\top = \top$ and $(\forall R{:}C) \sqcap (\forall R{:}D) = \forall R{:}(C \sqcap D)$. We thereby gain an elegant axiomatization of the equivalence relation in $\mathcal{ALC}$ too. Such an axiomatization of $\mathcal{ALC}$ does not only give rise to an alternative, purely syntactic view on logics, but can also serve as a basis for prototypical implementations [Quantz *et al.*, 1994]. For details the reader is referred to [Schild, 1991a].

Another interesting result which was established for $\mathbf{K}_{(m)}$, but was at best only implicitly present in the work of Schmidt-Schauß and Smolka [1991] is a property called *Finite Model Property*. This property is to be understood as follows. If an

[3] As a matter of fact, Schmidt [1991] independently observed a semantic parallel with propositional modal logics. However, she did not explicitly state the correspondence with $\mathbf{K}_{(m)}$, nor did she substantially exploit this semantic parallel.

[4] Ladner [1977] proved his upper bound for the single agent case only. His result is therefore restricted to $\mathcal{ALC}$ with only one single role name. However, Ladner's proof can straightforwardly be extended to deal with the multi-agent case too, as Halpern and Moses [1985] noted.

arbitrary formula, say, α, is satisfiable, then there always exists at least one Kripke structure with only a *finite* number of possible worlds and there exists at least one possible world, w, such that $\mathcal{M} \models_w \alpha$. The Finite Model Property is established with the help of so-called *p-morphisms*, describing what kind of relationship must exist between two Kripke structures so as to be indistinguishable by any formula of $\mathbf{K}_{(m)}$. See e.g. Chapter 5 of Hughes and Cresswell's [1984] companion of modal logic. In this sense p-morphism can be considered as a way of describing the expressive power of $\mathbf{K}_{(m)}$ in terms of those semantic structures it is able to distinguish. The significance of this result for a knowledge representation language such as $\mathcal{ALC}$ should be evident.

1.7 The Dynamic Logic's Perspective on Description Logics

The applicability of Kripke-style semantics turned out to extend far beyond its original intent. In particular, it proved to be a proper semantic framework for logics of programs. These logics are designed in the spirit of Hoare [1969] for the purpose of reasoning about the correctness and termination of programs. It was Pratt [1976] who first argued that possible worlds could be seen as program *states*. In doing so we can interpret accessibility relations as possibly nondeterministic programs. Of course, this is only possible when we adopt the common view of programs as state transitions. But then propositional modal logic as its stands provides a basic framework for reasoning about programs. From this point of view, the modality $K_a\alpha$ can be read as "postcondition α holds after any successful termination of the program a." Whenever such a reading is advocated the notation $[a]\alpha$ is usually preferred to $K_a\alpha$. The well-known Hoare assertion $\alpha\{a\}\beta$ can therefore be captured by a modal formula of the form $\alpha \Rightarrow [a]\beta$. If $\langle a\rangle\alpha$ is treated as an abbreviation of $\neg[a]\neg\alpha$, then termination of the program a can directly be expressed by $\langle a\rangle true$.

Following Harel *et al.* [1977] modal logics are referred to as *dynamic logics* whenever this alternative interpretation is put into effect. The term *dynamic* reflects the fact that the underlying intuition is of states and state transitions, both dynamic in nature. In acknowledgement of Hennessy and Milner's [1985] investigations on this dynamic reading of $\mathbf{K}_{(m)}$, $\mathbf{K}_{(m)}$ is usually referred to as *Hennessy-Milner Logic* whenever this reading is preferred. Although the Hennessy-Milner Logic is able to reason about Hoare assertions and termination, it can do so only if the programs upon which the reasoning takes place are atomic rather than compound. This is of course by no means sufficient. The minimal programming repository should include a possibility of executing two programs consecutively as well as while-loops along with if-then-else guards. Fischer and Ladner [1979] proposed to resort to the following fundamental inventory:

$a;b$ meaning "run a and b consecutively in order;"
a^* meaning "repeat a a nondeterministically chosen number of times ≥ 0;"
$a \cup b$ meaning "nondeterministically execute either a or b;"
$\alpha?$ meaning "test α and continue if the result is true."

Except for the last one, all these constructs are regular in nature. This inventory of programming facilities forms a good basis for more complex programs such as while-loops and if-then-else guards. It is instructive to verify that the following abbreviations do match the intuition behind these complex programs:

$$\begin{aligned} \textbf{if } \alpha \textbf{ then } a \textbf{ else } b &\stackrel{\text{def}}{=} (\alpha?;a) \cup (\neg\alpha?;a), \\ \textbf{while } \alpha \textbf{ do } a &\stackrel{\text{def}}{=} (\alpha?;a)^*;\neg\alpha. \end{aligned}$$

But then we are in a position to phrase such inferences as the following. Suppose the condition α constitutes an invariant of a program, say, a. This is to say that α is satisfied in any possible termination state of a if it is also satisfied before the execution. This invariance property can easily be captured by the dynamic formula $[a^*](\alpha \Rightarrow [a]\alpha)$. If the invariant property α is also satisfied by the current state, then it can be inferred that any termination of the program **while** β **do** a meets the postcondition $\alpha \wedge \neg\beta$. This inference is reflected by the validity of the following dynamic formula:

$$\Big(\alpha \wedge [a^*](\alpha \Rightarrow [a]\alpha)\Big) \;\Rightarrow\; [\textbf{while } \beta \textbf{ do } a](\alpha \wedge \neg\beta).$$

Fischer and Ladner [1979] chose the name *propositional dynamic logic* (or *PDL* for short) to refer to the extension of Hennessy-Milner Logic by the basic programming facilities introduced above. PDL had been placed at the very heart of the research on propositional logics of programs till the mid 1980's. This role is, however, nowadays taken over by the more expressive propositional μ-calculus.

Unaware of PDL's potential influence on his work, Baader [1990a] proposed to augment $\mathcal{ALC}$ by regular role expressions. His motivation in doing so was the observation that at least in some special cases regular role expressions turn out to be an alternative to recursively defined concepts. A recursively defined concept that can be handled in this way is, for instance, $CN \doteq C \sqcap \forall RN{:}CN$. Rather than developing an algorithm capable of dealing with recursion, he devised one for what he called *regular extension of* $\mathcal{ALC}$. This extension modifies $\mathcal{ALC}$ in such a way that roles are not restricted to simple role names, but compound roles defined by the following formation rule are admissible as well. If R and S are both roles, then so are $R \circ S$, R^*, as well as $R \sqcup S$. The meaning of these new role-structuring primitives should be evident. The operator * corresponds to the reflexive-transitive closure, while $\circ$ and $\sqcup$ denote the composition and the union of two binary relations. Baader was able to establish both soundness and completeness of the algorithm he devised, without, however, analyzing the worst-case complexity of the algorithm. He does not

analyze the general computational complexity of the corresponding subsumption or equivalence problem either.

In Baader's [1990a] paper, a semantic parallel between the regular extension of $\mathcal{ALC}$ and the propositional dynamic logic was not realized because at that time the close relationship between description logics and propositional modal logic was not known either. However, once the basic correspondence between $\mathcal{ALC}$ and $\mathbf{K}_{(m)}$ is established, it becomes quite obvious that Baader's regular extension of $\mathcal{ALC}$ is nothing but a notational variant of PDL. To be accurate, it is a notational variant of a particular fragment of PDL. In contrast to full PDL, this fragment does not involve any test, that is, programs of the shape $\alpha?$. This correspondence is a straightforward extension of the basic correspondence between $\mathcal{ALC}$ and $\mathbf{K}_{(m)}$ and was also first observed in [Schild, 1991a]. If the one-to-one correspondence between test-free PDL and Baader's regular extension of $\mathcal{ALC}$ is to cover tests, too, then the latter has to be slightly extended. In particular, Baader's version of the regular extension of $\mathcal{ALC}$ has to be enriched by the identity role, ϵ, as well as by so-called *role restrictions*. Role restrictions are frequently emerging in terminological knowledge representation systems and are special roles of the form $R|C$ projecting the range of the role R to the concept C. The role restriction $\mathsf{has\_child}|\mathsf{Woman}$, for instance, can be thought of as representing the *has daughter* relationship. The so extended regular extension of $\mathcal{ALC}$ is easily recognized as a notational variant of full PDL. In particular, the test $\alpha?$ corresponds to a role of the form $\epsilon|C_\alpha$ and, vice versa, the roles ϵ and $R|C$ correspond to the programs $true?$ and $a_R;\alpha_C?$ respectively.

The PDL literature contains a number of interesting results which turned out to be new for the corresponding description logics. The most important result is certainly that Fischer and Ladner [1979] were able to fix the exact lower and upper computational complexity bounds for deciding satisfiability of arbitrary PDL formulae. Both bounds match and are deterministic exponential-time in nature. Notably, none of these bounds was previously known for the corresponding regular extension of $\mathcal{ALC}$.

Fischer and Ladner [1979] also established the Finite Model Property for PDL. They do so with the help of what came to be called *Fischer-Ladner closure*. It is exactly this technique which in the meantime has gained broad acceptance as a powerful proof technique in the field of description logics too. Complete axiomatizations are also known for PDL, see [Pratt, 1979] and [Kozen and Parikh, 1981]. These and other results are summarized in [Schild, 1991a]

Not only PDL itself has been thoroughly investigated, but also several extensions and variants of it. The most important effort to be mentioned in this connection is Danecki's work [1984] on program intersection, the work of Ben-Ari *et al.* [1982] and Halpern and Reif [1983] on deterministic programs, Vardi's [1985] investigations of the converse of programs, as well as the work of Passy and Tinchev [1985] on data constants. In each case, it was possible to establish the decidability of the corresponding extension of PDL. Incidentally, all the additional features just mentioned

are significant not only within the framework of program logics, but also within description logics. All these additional programming features actually correspond to the following common concept and role-structuring primitives in the order in which they are mentioned above: role conjunction, functional roles (also known as *features*), the inverse of a role, as well as so-called *individual concepts*, denoting singleton sets rather than arbitrary sets of objects.

It was criticized, though, that all these correspondences lack at least two features that ought to be present in any description logic. On the one hand, the correspondences presented so far do not incorporate a concept-structuring primitive which is simply indispensable to most modeling tasks, namely *number restrictions*. These are special concepts of the form $\exists^{\geq n} R$ and $\exists^{\leq m} R$ imposing certain restrictions on the possible number of fillers of the role R. In particular, they restrict the possible number of the objects related by the role R in a way suggested by the notation $\exists^{\geq n}$ and $\exists^{\leq m}$. It was also criticized that it is not obvious how assertions can be integrated into these correspondences as well. This is in contrast to the fact that axioms can quite easily be integrated into these correspondences. In fact, Baader [1991] and Schild [1991a] noted independently that it is possible to simulate arbitrary axioms *within* the regular extension of $\mathcal{ALC}$ itself. De Giacomo and Lenzerini [1994a, 1994c] recently put forward that both assertions and at least a special kind of number restrictions can also be encoded within the regular extension of $\mathcal{ALC}$ if features are available. However, the employed simulations are rather involved.

Unnoticed till now remained the fact that there are extensions of PDL that allow the expression not only of arbitrary axioms, but also of any kind of assertion. This can be done in a rather straightforward way in *combinatory PDL*. Combinatory PDL has been launched by a group at the Sofia University in Bulgaria in order to remove some of PDL's major deficiencies [Passy and Tinchev, 1991]. This extension of PDL additionally comprises what is called the *universe program* as well as a special kind of propositional variables, referred to as *data constants* or *names*. The universe program, ν, denotes the universal relation among states, that is, a binary relation comprising any ordered pair of two states. The universe program enables us to capture unqualified quantification over states with the help of formulae of the form $[\nu]\alpha$. Data constants, usually denoted by *const* with appropriate indices, are to be interpreted as singleton sets containing exactly one state rather than an arbitrary number of states. Combinatory PDL thus removes a fundamental paradox traditional PDLs suffer from: Although the intuition underlying PDL is based on states and state transitions, the syntax of PDL itself is not capable of referring to a *single* state.

We already mentioned that data constants are known in the field of description logics as individual concepts. But then, having additionally the universe program at our disposal, axioms as well as assertions can straightforwardly be translated into the so extended PDL. The following translation scheme indicates how the corresponding

translations into combinatory PDL look like:

$$\begin{aligned} C \doteq D &\rightsquigarrow [\nu](\alpha_C \Leftrightarrow \alpha_D), \\ C \sqsubseteq D &\rightsquigarrow [\nu](\alpha_C \Rightarrow \alpha_D), \\ a{:}C &\rightsquigarrow \langle\nu\rangle(const_a \wedge \alpha_C), \\ \langle a, b\rangle{:}R &\rightsquigarrow \langle\nu\rangle(const_a \wedge \langle a_R\rangle const_b). \end{aligned}$$

Combinatory PDL has been the subject of thorough investigations, a comprehensive overview of which is given in [Passy and Tinchev, 1991]. In particular, Passy and Tinchev [1991] showed that the upper and lower computational complexity bounds of deciding satisfiability in combinatory PDL match the corresponding bound for PDL itself, even with deterministic programs added to combinatory PDL. As far as model theory is concerned, the Finite Model Property was established by Gargov [1985]. On top of this, Passy and Tinchev [1985] gave deep insights into the consequences of adopting different kinds of semantics for data constants. The semantic accounts for data constants differ in whether or not each semantic state is required to have at least one syntactic counterpart (i.e., a particular data constant denoting this very state). Last but not least, complete axiomatics were given for combinatory PDL too, including extensions of combinatory PDL which encompass additional formulae and programs corresponding to features, role conjunctions, as well as to number restrictions. In addition, formulae were axiomatized which correspond to what was introduced in the field of description logics under the term *role-value map*. For details the reader is referred to [Passy and Tinchev, 1991].

What the Thesis Is and Is Not About

We shall not enter into the details of the correspondences mentioned so far. Instead, in Chapter 2 we shall give the details of another correspondence not mentioned in the introduction. This correspondence will involve a special logic of programs, known as the *propositional μ-calculus*. The reasons in favor of this correspondence are many-fold: First, the μ-calculus is strictly stronger in expressive power than PDL. This means that the correspondence with the propositional μ-calculus subsumes the correspondences with PDL and $\mathbf{K}_{(m)}$.

Second, this correspondence with the propositional μ-calculus will solve an important open problem of description logics, which has to do with recursion. There has been an intensive debate in description logics on which kind of semantics is appropriate for recursion and how the different kinds of semantics can be dealt with algorithmically. Preliminary results in this direction are only known for the very weakest description logic [Baader, 1990b], the correspondence with the propositional μ-calculus will clarify this debate to a large extent.

Last but not least, for PDL as well as for $\mathbf{K}_{(m)}$ there already exist several excellent textbooks. $\mathbf{K}_{(m)}$ is thoroughly introduced in [Hughes and Cresswell, 1968, 1984], [Chellas, 1980], and [Bull and Segerberg, 1984]. Comprehensive introductions to PDL are given, for instance, in [Parikh, 1979], [Harel, 1984], [Goldblatt, 1987], and [Kozen and Tiuryn, 1990]. With the close connection of description logics with PDL and $\mathbf{K}_{(m)}$ in mind, all these text books can be read as if written to the corresponding description logic.

The results of Chapter 2 on recursion in the standard description logic $\mathcal{ALC}$ seem to be promising. However, $\mathcal{ALC}$ was advocated by its inventors because of its "pleasant mathematical properties" [Schmidt-Schauß and Smolka, 1991] rather than because of its practical significance for knowledge representation purposes. This raises the question whether the syntactic and semantic foundations of recursion laid in Chapter 2 can be generalized so as to deal with more expressive languages. Chapter 3 will demonstrate that this is in fact possible in a quite straightforward manner. In particular, it will be shown that these foundations of recursion can be extended to a description logic which can be called universal: it is universal in the sense that it encompasses all traditional concept and role-structuring primitives. Such high expressive power, however, is gained at the expense of decidability.

But then there is the question whether any mechanical reasoning can take place in this general setting and Chapter 4 is devoted to exactly this issue. The main result will be that even *tractable* reasoning can take place in this setting. However, the only way to achieve tractability is to abandon incomplete knowledge from knowledge bases. The resulting knowledge bases are then much more a traditional relational databases than a knowledge bases. This will immediately lead to a database-theoretical interpretation of the tractability result. Independent of this interpretation, it is important to note that the tractability result to be presented in Chapter 4 is, in effect, the very first tractability result for terminological reasoning, all other results depend on terminologies being somehow artificially preprocessed.

An overview of the thesis together with a summary discussion of its main results is compiled in the final Chapter 5.

Chapter 2

The μ-Calculus' Perspective on Recursion in Description Logics

2.1 Recursion in Description Logics

Recursion is a fundamental means of definition in all areas of computer science. However, it is still open how to tackle recursion in description logics, notwithstanding that nonrecursive definitions can be dealt with easily. So let us begin the discussion with nonrecursive definitions. A typical example of a nonrecursive definition in description logics is the following, which defines leaves as nodes that do not have any branch:

$$\mathsf{Leaf} \doteq \mathsf{Node} \sqcap \neg\exists \mathsf{branch}{:}\top.$$

It is perfectly straightforward to state the meaning of such a *concept introduction* in set-theoretical terms. As usual, the meaning of concepts and concept introductions is given in terms of *interpretations* and *models*. An *interpretation* $\mathcal{I}$ over a *domain* $\Delta^{\mathcal{I}}$ maps the universal concept $\top$ to $\Delta^{\mathcal{I}}$, each concept name CN to an arbitrary subset $CN^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and each role name RN to a binary relation $RN^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$. Moreover, the logical connectives $\sqcap$, $\sqcup$ and $\neg$ are interpreted as the corresponding set operations on $\Delta^{\mathcal{I}}$, whereas $\exists RN{:}$ and $\forall RN{:}$ represent existential and universal quantification over the relation $RN^{\mathcal{I}}$. The meaning of a concept introduction is then given by requiring that an interpretation is a *model* of $C \doteq D$ if and only if the interpretation maps C and D to exactly the same subset of the domain. In the case of the concept introduction given above, this means that each model has to satisfy the following equation:

$$\mathsf{Leaf}^{\mathcal{I}} = \mathsf{Node}^{\mathcal{I}} \cap \{d \in \Delta^{\mathcal{I}} : \textit{there is no } e \textit{ such that } \langle d, e\rangle \in \mathsf{branch}^{\mathcal{I}}\}.$$

There are also algorithms to compute both the subsumption and the equivalence relation between concepts, even with respect to finite sets of concept introductions similar to the one just considered [Schmidt-Schauß and Smolka, 1991].

Problems arise, however, when cyclic or recursive concept introductions enter the picture. For example, it is entirely natural to define a tree recursively as a node which has only trees as branches:[1]

$$\mathsf{Tree} \doteq \mathsf{Node} \sqcap \forall \mathsf{branch}{:}\mathsf{Tree}. \tag{2.1}$$

The models of this cyclic concept introduction are characterized by the following equation:

$$\mathsf{Tree}^{\mathcal{I}} = \mathsf{Node}^{\mathcal{I}} \cap \{d \in \Delta^{\mathcal{I}} : \textit{for all } e \textit{ such that } \langle d, e\rangle \in \mathsf{branch}^{\mathcal{I}},\ e \in \mathsf{Tree}^{\mathcal{I}}\}.$$

Unfortunately, such recursive equations do not always have unique solutions, even when the interpretation of all undefined concept and role names is fixed. Take, for instance, an interpretation $\mathcal{I}$ the domain of which is $\mathbb{N}$, the set of all natural numbers. Suppose, moreover, $\mathsf{Node}^{\mathcal{I}}$ is $\mathbb{N}$, while $\mathsf{branch}^{\mathcal{I}}$ is the successor relation on $\mathbb{N}$. Such an interpretation is a model of (2.1) just in case the following equation is satisfied:

$$\begin{aligned}\mathsf{Tree}^{\mathcal{I}} &= \mathbb{N} \cap \{n \in \mathbb{N} : \textit{for all } m \textit{ such that } m = n+1,\ m \in \mathsf{Tree}^{\mathcal{I}}\} \\ &= \{n \in \mathbb{N} : n+1 \in \mathsf{Tree}^{\mathcal{I}}\}.\end{aligned}$$

The question, then, is whether in all these models the nodes are actually trees or not. However, the recursive equation above does not tell us anything about that: It has two conflicting solutions, viz. one in which $\mathsf{Tree}^{\mathcal{I}}$ is $\mathbb{N}$, and one in which it is the empty set. This gives rise to the question which, if any, of these conflicting solutions is to be preferred. In fact, there is an ongoing discussion on which kind of solution generally accords best with our intuition.[2] In essence, there are three rivals which should be taken into consideration. First, simply allowing *all* solutions results in what Nebel [1990a, Chapter 5.2.3] called *descriptive semantics*. The remaining two alternatives allow only those solutions which are the least or greatest ones with respect to the interpretation of all *defined* concepts (i.e., those concept names which appear on the left-hand side of a concept introduction). The terms *least* and *greatest*, however, apply only to solutions which agree in the interpretation of all undefined concept and role names. Nebel [1990a, Chapter 5.2.2] called solutions of this kind *least* and *greatest fixed-point models*. The previously mentioned model which interprets Tree as the empty set is, therefore, a least fixed-point model of (2.1), whereas the other one interpreting Tree as $\mathbb{N}$ is a greatest fixed-point model. But even if we stick to one of these alternatives, it is not clear at all how to obtain the corresponding inference algorithms, except for very small languages [Baader, 1990b].

In some cases, the consequences of choosing one of these semantics can be clarified in terms of the reflexive-transitive closure R^* of a role. Baader [1990b, Theorem 4.3.1]

[1] If this definition were intended to represent trees *accurately*, the concept Node would have to be defined properly, i.e., it would have to be defined in such a way that each Node has at most one branch-predecessor.

[2] Among others, Nebel [1990a, 1991], Baader [1990b], as well as Dionne *et al.* [1992, 1993] have contributed to this discussion.

showed that the greatest fixed-point semantics forces the recursive definition of a tree (2.1) to be equivalent to $\mathsf{Tree} \doteq \forall \mathsf{branch}^*{:}\mathsf{Node}$. This is, however, neither the case for the descriptive nor for the least fixed-point semantics. Taking for granted that recursion is commonly used to express the reflexive-transitive closure of a role, Baader then came to the conclusion that the greatest fixed-point semantics comes off best [Baader, 1990b, page 626]. Baader's conclusion is somewhat misleading in that it applies only to the particular description logic he considered. As a matter of fact, he would have come to the opposite conclusion if, instead of concept conjunction and universal role quantification, he considered the corresponding dual concept-structuring primitives. For the dual concept-structuring primitives, that is, concept disjunction and existential quantification over roles, the situation is just the other way round. To see this, take the following definition of a non-tree as opposed to the above of a tree:

$$\mathsf{NonTree} \doteq \neg\mathsf{Node} \sqcup \exists \mathsf{branch}{:}\mathsf{NonTree}. \tag{2.2}$$

This is to say, a non-tree is something which is either no node or which has some branch being a non-tree. We shall see below that in this case only the *least* fixed-point semantics forces (2.2) to be equivalent to $\mathsf{NonTree} \doteq \exists \mathsf{branch}^*{:}\neg\mathsf{Node}$. As $\exists \mathsf{branch}^*{:}\neg\mathsf{Node}$ is equivalent to $\neg\forall \mathsf{branch}^*{:}\mathsf{Node}$, this means that the least fixed-point semantics of (2.2) expresses the very opposite of the greatest fixed-point semantics of (2.1). Anyway, insofar as solely finite trees are concerned, the least fixed-point semantics seems to be more adequate in that it excludes infinite chains of the role branch. In fact, we shall see that it forces (2.1) to be equivalent to $\mathsf{Tree} \doteq (\forall \mathsf{branch}^*{:}\mathsf{Node}) \sqcap \neg\exists \mathsf{branch}^{\omega}$, where the concept $\exists \mathsf{branch}^{\omega}$ stipulates the existence of some infinite chain of the role branch. We thus allow only acyclic structures of finite depth, clearly a necessary condition for being a finite tree.

This shows that in order to express the reflexive-transitive closure of a role in a recursive manner, in some cases it is necessary to resort to the greatest fixed-point semantics, while in others the least fixed-point semantics must be invoked. The reader might object that in the presence of negation, one could always employ dualities such as the one between $\exists \mathsf{branch}^*{:}\neg\mathsf{Node}$ and $\neg\forall \mathsf{branch}^*{:}\mathsf{Node}$. In particular, instead of expressing $\exists \mathsf{branch}^*{:}\neg\mathsf{Node}$, one could first capture $\forall \mathsf{branch}^*{:}\mathsf{Node}$ with the help of $\mathsf{Tree} \doteq \mathsf{Node} \sqcap \forall \mathsf{branch}{:}\mathsf{Tree}$. Of course, in order to achieve this, greatest fixed-point semantics must be invoked. If one would then simply add the concept introduction $\mathsf{NonTree} \doteq \neg\mathsf{Tree}$, then NonTree should finally capture $\exists \mathsf{branch}^*{:}\neg\mathsf{Node}$. Such a representation does not work though. This is because terminologies of the form $\{\mathsf{Tree} \doteq \mathsf{Node} \sqcap \forall \mathsf{branch}{:}\mathsf{Tree}, \mathsf{NonTree} \doteq \neg\mathsf{Tree}\}$ do not always have any greatest fixed-point model at all, nor have they always any least fixed-point model! To see this, observe that it is impossible to maximize (or to minimize) the interpretation of Tree and its complement, $\neg\mathsf{Tree}$, at the same time, unless there is only one possible interpretation. Take, for instance, the interpretation over $\mathbb{N}$ from above. This interpretation is a model of the terminology $\{\mathsf{Tree} \doteq \mathsf{Node} \sqcap \forall \mathsf{branch}{:}\mathsf{Tree}, \mathsf{NonTree} \doteq \neg\mathsf{Tree}\}$

just in case the following two equations are simultaneously satisfied:

$$\mathsf{Tree}^{\mathcal{I}} = \{n \in \mathbb{N} : n+1 \in \mathsf{Tree}^{\mathcal{I}}\}, \tag{2.3}$$

$$\mathsf{NonTree}^{\mathcal{I}} = \mathbb{N} \setminus \mathsf{Tree}^{\mathcal{I}}. \tag{2.4}$$

These equations have exactly two solutions in common, namely one in which $\mathsf{Tree}^{\mathcal{I}}$ is $\mathbb{N}$ and $\mathsf{NonTree}^{\mathcal{I}}$ is the empty set, while in the second it is the other way round. Of course, neither solution is a least or greatest one with respect to both $\mathsf{Tree}^{\mathcal{I}}$ and $\mathsf{NonTree}^{\mathcal{I}}$.

This proves not only that both kinds of fixed-point semantics are needed, but that they are even needed in *coexistence* if we want to express the reflexive-transitive closure of roles in the context of existential as well as universal role quantifications. The notion of least and greatest fixed-point semantics as considered in the terminological logics literature does not take into consideration this fact. To overcome this deficiency, we introduce prefixes μ and ν as explicit references to least and a greatest fixed-point semantics. In addition, explicit fixed-point prefixes allow us to limit the scope of fixed points. For instance, $\mu\{(2.1), \mathsf{NonTree} \doteq \neg\mathsf{Tree}\}$ yields the simultaneous least fixed point of the two concept introductions, while $\{\mu\{(2.1)\}, \mu\{\mathsf{NonTree} \doteq \neg\mathsf{Tree}\}\}$ refers to two least fixed points computed locally. A terminology with a sibngle fixed-point prefix in front of it is called *least* or *greatest fixed-point terminology*, depending on whether the prefix is μ or ν.. A finite collection of such fixed-point terminologies then forms a *complex fixed-point terminology.*

Having complex fixed-point terminologies at our disposal, we can even reason about the different kinds of semantics. For instance, we can conclude that the greatest fixed-point semantics of (2.1) in fact expresses the very opposite of the least fixed-point semantics of (2.2):

$$\nu\{(2.1)\},\ \mu\{(2.2)\} \models \mathsf{NonTree} \doteq \neg\mathsf{Tree}.$$

As it turns out, the overall situation can be analyzed in terms of the well-investigated *propositional μ-calculus* in a perfectly straightforward way. This was first observed in [Schild, 1993b, 1993c] and [Schild, 1994a]. The propositional μ-calculus is an extension of the propositional multi-modal logic $\mathbf{K}_{(m)}$, launched by Pratt [1981] and Kozen [1983] as a special logic to reason about (concurrent) programs. The propositional μ-calculus extends $\mathbf{K}_{(m)}$ by fixed-point operators of the form $\mu x.\alpha$ and $\nu x.\alpha$ where α can be an arbitrary formula of the propositional μ-calculus. However, a restriction, called *formal monotonicity*, is imposed on α which requires that the variable x may occur only positively in α. The formulae $\mu x.\alpha$ and $\nu x.\alpha$ explicitly represent the least and the greatest fixed-points of a function loosely associated with the lambda-expression $\lambda x.\alpha$. As x may occur only positively in α, this function is known to be monotone. According to the well-known Knaster-Tarski Theorem, this function therefore has both a unique least as well as a unique greatest fixed-point [Tarski, 1955].

It will turn out that least and greatest fixed-point terminologies can be expressed straightforwardly in terms of such explicit fixed-point operators. The only prerequisite for such a representation will be that recursively defined concepts may occur only positively in their definition. In what follows, we shall use this correspondence to provide a deeper insight into the computational complexity and the expressive power of the different kinds of fixed-point semantics.

2.2 The Standard Description Logic $\mathcal{ALC}$

Let us first fix the concept language which will form the basis for our investigations in this chapter. We consider the standard terminological logic $\mathcal{ALC}$, thoroughly investigated by Schmidt-Schauß and Smolka [1991] in their seminal paper. $\mathcal{ALC}$ is a well-known description logic, which covers the basic expressive power of a description logic. It constitutes, for instance, the core language of the system $\mathcal{KRIS}$ [Baader and Hollunder, 1991].

Definition 1. Assume $\mathcal{N}$ is the union of two disjoint, infinite sets, called $\mathcal{N}_\mathcal{C}$ and $\mathcal{N}_\mathcal{R}$, which contain neither $\top$ nor $\bot$.[3] The elements of these sets are called **concept names** and **role names** respectively. The concepts and roles of $\mathcal{ALC}$ are inductively defined as follows.

1. Every role name is a role of $\mathcal{ALC}$.
2. Every concept name, $\top$, and $\bot$ are concepts of $\mathcal{ALC}$.
3. If C and D are concepts of $\mathcal{ALC}$ and RN is a role name, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall RN{:}C$ and $\exists RN{:}C$ are all concepts of $\mathcal{ALC}$.
4. These are the concepts and roles of $\mathcal{ALC}$.

Of course, we may use parentheses to resolve ambiguities.

As already mentioned in the introduction, concept names are interpreted as arbitrary subsets of some domain, while role names are interpreted as binary relations over the domain. For this very purpose, so-called *$\mathcal{L}$-valuations* are introduced, which fix the interpretation of all elements of a set $\mathcal{L}$ of concept and role names.

Definition 2. Assume $\mathcal{L}$ is a set of concept and role names and assume Δ is an arbitrary set. An **$\mathcal{L}$-valuation** $\mathcal{V}$ over Δ is a function which maps each concept name of $\mathcal{L}$ to a subset of Δ and each role name of $\mathcal{L}$ to a binary relation over Δ.

[3] Of course, both sets should be acceptable in deterministic polynomial time, for instance, by means of some finite state automaton.

We shall frequently make use of the fact that each $\mathcal{L}$-valuation $\mathcal{V}$ over Δ can be viewed as a subset of $\mathcal{L} \times (2^{\Delta} \cup 2^{\Delta \times \Delta})$, i.e., it can be viewed as the set $\{\langle TN, \mathcal{V}(TN)\rangle : TN \in \mathcal{L}\}$.

Before specifying of how arbitrary concepts of $\mathcal{ALC}$ are interpreted, we introduce a useful projection operation on binary relations.

Notation 1. Assume $r \subseteq \Delta \times \Delta$ is an arbitrary binary relation over Δ and $d \in \Delta$. Then $\boldsymbol{r(d)}$ is defined to be $\{e \in \Delta : \langle d, e\rangle \in r\}$.

Definition 3. An **interpretation** $\mathcal{I}$ is a triple $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$, where $\Delta^{\mathcal{I}}$ is a set, called the **domain** of $\mathcal{I}$, and $\mathcal{V}$ is a $\mathcal{N}$-valuation over $\Delta^{\mathcal{I}}$. Moreover, $.^{\mathcal{I}}$ is a function, called the **interpretation function** of $\mathcal{I}$, which maps concepts of $\mathcal{ALC}$ to subsets of $\Delta^{\mathcal{I}}$ and role names to binary relations over $\Delta^{\mathcal{I}}$. It extends $\mathcal{V}$ to deal with arbitrary concepts of $\mathcal{ALC}$ and is inductively defined as follows: $\top^{\mathcal{I}}$ is $\Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}}$ is $\emptyset$, and $TN^{\mathcal{I}}$ is $\mathcal{V}(TN)$ whenever $TN \in \mathcal{N}$. Now, suppose $C^{\mathcal{I}}$ as well as $D^{\mathcal{I}}$ have already been defined, where C and D are concepts of $\mathcal{ALC}$. Then $.^{\mathcal{I}}$ is defined as follows:

$$\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}},\\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}},\\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},\\
(\forall RN{:}C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : RN^{\mathcal{I}}(d) \subseteq C^{\mathcal{I}}\},\\
(\exists RN{:}C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : RN^{\mathcal{I}}(d) \cap C^{\mathcal{I}} \neq \emptyset\}.
\end{aligned}$$

It can easily be verified that the interpretation function $.^{\mathcal{I}}$ of every interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$ is uniquely determined by $\Delta^{\mathcal{I}}$ together with the $\mathcal{N}$-valuation $\mathcal{V}$.

Having specified the syntax and the meaning of the basic expressions, it remains to define what exactly constitutes a terminology and what its meaning is.

Definition 4. Assume $\mathcal{L}$ is a set of concepts and roles. If C and D are two concepts of $\mathcal{L}$ and R and S are two roles of $\mathcal{L}$, then both $\boldsymbol{C \doteq D}$ and $\boldsymbol{R \doteq S}$ are called **axioms** of $\mathcal{L}$. Moreover, such axioms are called **concept** or **role introductions** of $\mathcal{L}$ whenever C is a concept name and R is a role name. Axioms of the form $T \doteq T \sqcap T'$ are called **inclusion axioms** and can be abbreviated by $\boldsymbol{T \sqsubseteq T'}$. If T is a concept or role name, then such an inclusion axiom is also called **primitive concept introduction** and **primitive role introduction** respectively. A **terminology** of $\mathcal{L}$ is a finite set $\mathcal{T}$ of concept or role introductions of $\mathcal{L}$ such that for every concept or role name TN there is at most one concept or role T such that $TN \doteq T$ is an element of $\mathcal{T}$.

In case of $\mathcal{ALC}$, however, we do not consider role introductions because no complex roles are available in this description logic.

Definition 5. A concept or role name $TN \in \mathcal{N}$ is called **defined** in the terminology $\mathcal{T}$ if and only if there is a concept or role T such that $TN \doteq T$ is an element of $\mathcal{T}$. Moreover, TN is **primitive** in $\mathcal{T}$ if and only if it occurs in $\mathcal{T}$, but is not defined in $\mathcal{T}$. We denote with $\mathbf{def}(\mathcal{T})$ the set of all concept and role names which are defined in $\mathcal{T}$, while $\mathbf{prim}(\mathcal{T})$ is the set of all those concept and role names which are primitive in $\mathcal{T}$. Finally, $\mathbf{undef}(\mathcal{T})$ is defined to be $\mathcal{N} \setminus def(\mathcal{T})$.

Note that $undef(\mathcal{T})$ comprises *all* concept and role names which are not defined in $\mathcal{T}$, no matter whether they occur in $\mathcal{T}$ or not.

In order to state the meaning of terminologies we have to specify their models. As usual, models are interpretations forcing something to hold. In case of terminologies, a model is simply an interpretation respecting every concept and role introduction of the terminology in the sense that the left-hand side of the concept or role introduction must denote the same set as the right-hand side. As terminologies such as $\{CN \doteq \neg CN\}$ should not have any model, the domain of a model is required to be nonempty.

Definition 6. An interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ with a nonempty domain $\Delta^{\mathcal{I}}$ is a **model** of an axiom $T \doteq T'$ if and only if $T^{\mathcal{I}} = T'^{\mathcal{I}}$, and it is a **model** of a set of axioms if and only if it is a model of each axiom of the set.

Recall that we treated an inclusion axiom of the form $T \sqsubseteq T'$ as an abbreviation of $T \doteq T \sqcap T'$. It should be stressed that the models of $T \sqsubseteq T'$ are exactly those interpretations which interpret T and T' according to the intended subset relation. That is, an interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ is a model of $T \sqsubseteq T'$ if and only if $T^{\mathcal{I}} \subseteq T'^{\mathcal{I}}$.

Having the notion of a model at hand, we can now define semantic relations such as subsumption and equivalence.

Definition 7. Suppose $\mathcal{A} \cup \{T \doteq T'\}$ is an arbitrary set of axioms. Then $\mathcal{A}$ is said to **entail** $T \doteq T'$ if and only if every model of $\mathcal{A}$ is also a model of $T \doteq T'$. Whenever this is the case, we write $\boldsymbol{\mathcal{A} \models T \doteq T'}$, possibly omitting the curly brackets of the set $\mathcal{A}$, and $\mathcal{A}$ altogether if it is empty. We say that T' **subsumes** T with respect to $\mathcal{A}$ if and only if $\mathcal{A} \models T \sqsubseteq T'$. Moreover, C and D are **equivalent** with respect to $\mathcal{A}$ if and only if $\models T \doteq T'$. Finally, a concept is **coherent** if and only if it is not equivalent to $\bot$ with respect to $\emptyset$.

For the sake of convenience, we may omit the phrase "with respect to $\mathcal{A}$" whenever $\mathcal{A}$ is the empty set. We close this section with a formal definition of cyclic and acyclic terminologies.

Definition 8. Assume $\mathcal{T}$ is some terminology and assume $TN \doteq T$ and $TN' \doteq T'$ are concept or role introductions of $\mathcal{T}$. We say that $TN \doteq T$ **directly uses** $TN' \doteq T'$ if

and only if T involves an occurrence of TN'. If $\mathcal{T}$-*uses* denotes the transitive closure of *directly uses* over $\mathcal{T}$, then two concept or role introductions are defined to be **mutually dependent** within $\mathcal{T}$ if and only if they $\mathcal{T}$-use each other.

The reader may check that the relation *mutually dependent within* $\mathcal{T}$ is always transitive as well as symmetric, but it is not necessarily reflexive.

Definition 9. A terminology $\mathcal{T}$ is **cyclic** if and only if it contains concept or role introductions which are mutually dependent within $\mathcal{T}$; otherwise it is **acyclic**.

It can easily be seen that as far as acyclic terminologies $\mathcal{T}$ of $\mathcal{ALC}$ are concerned, every $undef(\mathcal{T})$-valuation $\mathcal{V}_u$ can be uniquely extended to a model of $\mathcal{T}$. This is to say, there is exactly one model of $\mathcal{T}$ which *extends* $\mathcal{V}_u$ in the sense that an interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ is defined to **extend** a $\mathcal{L}$-valuation $\mathcal{V}_u$ over Δ if and only if $\Delta^{\mathcal{I}} = \Delta$ and $\mathcal{V}_u \subseteq \mathcal{V}$. However, we have already seen in the introduction that this does not apply to *cyclic* terminologies.

2.3 Formal Monotonicity in $\mathcal{ALC}$

So far we took all models of terminologies as admissible. We now introduce prefixes μ and ν to invoke least or greatest fixed-point semantics.

Definition 10. Assume $\mathcal{L}$ is a set of concepts and $\mathcal{T}$ is an arbitrary terminology of $\mathcal{L}$. Then $\mu\mathcal{T}$ is called a **least fixed-point terminology** of $\mathcal{L}$, whereas $\nu\mathcal{T}$ is a **greatest fixed-point terminology** of $\mathcal{L}$.

In order to state the meaning of such least and greatest fixed-point terminologies, $\mu\mathcal{T}$ and $\nu\mathcal{T}$, all models of $\mathcal{T}$ which agree in the interpretation of all undefined concept and role names of $\mathcal{T}$ must be compared to each other, hence the following definition.

Definition 11. Suppose $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ and $\mathcal{J} = \langle \Delta^{\mathcal{J}}, .^{\mathcal{J}}, \mathcal{W} \rangle$ are arbitrary interpretations such that $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}}$. If $\mathcal{L}$ is a set of concept and role names, then $\mathcal{J}$ is said to be **$\mathcal{L}$-compatible** with $\mathcal{I}$ if and only if for every concept and role name TN of $\mathcal{L}$, $TN^{\mathcal{J}} = TN^{\mathcal{I}}$.

In other words, $\mathcal{V}$ and $\mathcal{W}$ coincide at least on all elements of $\mathcal{L}$ if $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ and $\mathcal{J} = \langle \Delta^{\mathcal{J}}, .^{\mathcal{J}}, \mathcal{W} \rangle$ are $\mathcal{L}$-compatible.

Definition 12. Assume $\mathcal{T}$ is some terminology and assume $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ is an arbitrary interpretation. Then $\mathcal{I}$ is a **least fixed-point model** of $\mathcal{T}$ if and only if

it is a model of $\mathcal{T}$ and, additionally, for each other model $\langle \Delta^{\mathcal{I}}, .^{\mathcal{J}}, \mathcal{W} \rangle$ of $\mathcal{T}$ which is $undef(\mathcal{T})$-compatible with $\mathcal{I}$, it holds that $CN^{\mathcal{I}} \subseteq CN^{\mathcal{J}}$, for every CN defined in $\mathcal{T}$. The **greatest fixed-point models** of $\mathcal{T}$ are defined correspondingly by requiring $CN^{\mathcal{I}} \supseteq CN^{\mathcal{J}}$ instead of $CN^{\mathcal{I}} \subseteq CN^{\mathcal{J}}$. Furthermore, $\mathcal{I}$ is defined to be a **model** of $\mu\mathcal{T}$ (or $\nu\mathcal{T}$) if and only if it is a least (or greatest) fixed-point model of $\mathcal{T}$.

The motivation for the notion of a *fixed-point* model is the observation that a terminology $\mathcal{T}$ together with an $undef(\mathcal{T})$-valuation $\mathcal{V}_u$ over Δ induces an n-ary function $f : (2^{\Delta})^n \to (2^{\Delta})^n$, provided that $\mathcal{T}$ comprises exactly n concept introductions. Consider, for instance, the terminology $\{\mathsf{Tree} \doteq \mathsf{Node} \sqcap \forall \mathsf{branch}{:}\mathsf{Tree}\}$. This terminology together with an $undef(\mathcal{T})$-valuation $\mathcal{V}_u$ over Δ induces a function $nb : 2^{\Delta} \to 2^{\Delta}$, which can be thought of as mapping each subset S of Δ to all those *n*odes the only *b*ranches of which are among S. Formally, this function is defined for each $S \subseteq \Delta$ as follows:

$$nb(S) \;=\; \mathcal{V}_u(\mathsf{Node}) \cap \{d \in \Delta : \mathcal{V}_u(\mathsf{branch})(d) \subseteq S\}.$$

In general, this function will be defined in terms of an interpretation $\mathcal{I} = \langle \Delta, .^{\mathcal{I}}, \mathcal{V} \rangle$ which extends $\mathcal{V}_u$ in such a way that $\mathsf{Tree}^{\mathcal{I}}$ is S, i.e., $\mathcal{I}$ extends the $\mathcal{N}$-valuation $\mathcal{V}_u \cup \{\langle \mathsf{Tree}, S \rangle\}$. Resorting to this interpretation, $nb(S)$ can simply be defined to be $(\mathsf{Node} \sqcap \forall \mathsf{branch}{:}\mathsf{Tree})^{\mathcal{I}}$. This definition yields the intended function:

$$\begin{aligned} nb(S) \;&=\; (\mathsf{Node} \sqcap \forall \mathsf{branch}{:}\mathsf{Tree})^{\mathcal{I}} \\ &=\; \mathsf{Node}^{\mathcal{I}} \cap \{d \in \Delta^{\mathcal{I}} : \mathsf{branch}^{\mathcal{I}}(d) \subseteq \mathsf{Tree}^{\mathcal{I}}\} \\ &=\; \mathcal{V}_u(\mathsf{Node}) \cap \{d \in \Delta : \mathcal{V}_u(\mathsf{branch})(d) \subseteq S\}. \end{aligned}$$

Next we give the general definition of the function induced by $\mathcal{T}$ and an $undef(\mathcal{T})$-valuation.

Definition 13. Suppose $\mathcal{T}$ is a terminology of the form $\{CN_i \doteq C_i : 1 \leq i \leq n\}$, where $CN_1, ..., CN_n$ are ordered by some fixed total ordering on $\mathcal{N}_{\mathcal{C}}$. Suppose, furthermore, $\mathcal{V}_u$ is an $undef(\mathcal{T})$-valuation over Δ. Then the **function induced by** $\mathcal{T}$ and $\mathcal{V}_u$ is the function $f : (2^{\Delta})^n \to (2^{\Delta})^n$ defined as follows. Assume $S_1, ..., S_n$ are arbitrary subsets of Δ and $\langle \Delta, .^{\mathcal{I}}, \mathcal{V} \rangle$ is the interpretation which extends the $\mathcal{N}$-valuation $\mathcal{V}_u \cup \mathcal{V}_d$ over Δ, where $\mathcal{V}_d$ is $\{\langle CN_i, S_i \rangle : 1 \leq i \leq n\}$. Then $f(S_1, ..., S_n)$ is $\langle C_1^{\mathcal{I}}, ..., C_n^{\mathcal{I}} \rangle$.

It should be clear that $\mathcal{V}_u \cup \mathcal{V}_d$ is in fact an $\mathcal{N}$-valuation because it combines an $undef(\mathcal{T})$-valuation with a $def(\mathcal{T})$-valuation, so that each concept and role name is handled either by the former or by the latter. In the previous section, it has already been noted that the interpretation function $.^{\mathcal{I}}$ of every interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ is uniquely determined by the $\mathcal{N}$-valuation $\mathcal{V}$. The $C_i^{\mathcal{I}}$'s of the definition above are, therefore, uniquely defined.

Definition 14. Assume Δ is an arbitrary set and f is some n-ary function mapping $(2^\Delta)^n$ into $(2^\Delta)^n$. An element $\langle S_1, ..., S_n\rangle$ of $(2^\Delta)^n$ is called **fixed point** of f if and only if $f(S_1, ..., S_n)$ is $\langle S_1, ..., S_n\rangle$, and such a fixed point is a **least fixed point** of f if and only if for each other fixed point $\langle S'_1, ..., S'_n\rangle$ of f, it holds that $S_i \subseteq S'_i$, for every i $(1 \leq i \leq n)$. A **greatest fixed point** of f is defined correspondingly by requiring $S_i \supseteq S'_i$ instead of $S_i \subseteq S'_i$.

A moment's thought should convince the reader that there is a close connection between the fixed points of the function induced by a terminology and an *undef*$(\mathcal{T})$-valuation $\mathcal{V}_u$ on the one hand, and models of $\mathcal{T}$ on the other hand. Take, for instance, the by now familiar terminology $\mathcal{T} = \{\mathsf{Tree} \doteq \mathsf{Node} \sqcap \forall \mathsf{branch}{:}\mathsf{Tree}\}$ and the function nb induced by $\mathcal{T}$ and an arbitrary *undef*$(\mathcal{T})$-valuation $\mathcal{V}_u$. Every model $\langle \Delta^\mathcal{I}, .^\mathcal{I}, \mathcal{V}\rangle$ extending $\mathcal{V}_u$ is a model of $\mathcal{T}$ if and only if $\mathsf{Tree}^\mathcal{I}$ is a fixed point of the function nb. Clearly, exactly the same close relationship exists between the least (or greatest) fixed points of nb and the least (or greatest) fixed-point models of $\mathcal{T}$.

Proposition 1. *Suppose $\mathcal{T}$ is some terminology, $\mathcal{V}_u$ is an undef$(\mathcal{T})$-valuation over Δ, $\langle \Delta^\mathcal{I}, .^\mathcal{I}, \mathcal{V}\rangle$ is an arbitrary interpretation extending $\mathcal{V}_u$, and f is the function induced by $\mathcal{T}$ and $\mathcal{V}_u$. Assume, moreover, def$(\mathcal{T})$ is $\{CN_1, ..., CN_n\}$, where $CN_1, ..., CN_n$ are ordered by the same ordering as in the definition of f. Then $\mathcal{I}$ is a least (or greatest) fixed-point model of $\mathcal{T}$ if and only if $\langle CN_1^\mathcal{I}, ..., CN_n^\mathcal{I}\rangle$ is the least (or greatest) fixed point of the function f.*

To ensure the existence of least and greatest fixed-point models of a terminology $\mathcal{T}$ the function induced by $\mathcal{T}$ and an arbitrary *undef*$(\mathcal{T})$-valuation $\mathcal{V}_u$ should be monotonically increasing. As customary, an n-ary function $f : (2^\Delta)^n \to (2^\Delta)^n$ is said to be **monotonically increasing** (or **monotone**, for short) if and only if for every two elements $\overline{S}$ and $\overline{S'}$ of $(2^\Delta)^n$, it holds that $f(\overline{S}) \subseteq f(\overline{S'})$ whenever $\overline{S} \subseteq \overline{S'}$. It should be clear that in this case $\subseteq$ denotes the component-wise subset relation.

Monotonicity of the function induced by $\mathcal{T}$ and $\mathcal{V}_u$ can be achieved by requiring all occurrences of concept names which are defined in $\mathcal{T}$ to be positive, i.e., they must occur in the scope of an even number of negations. This restriction is called formal monotonicity.

Definition 15. A terminology, $\mathcal{T}$, of $\mathcal{ALC}$ is **formally** or **syntactically monotone** if and only if for every concept introduction $CN \doteq C$ of $\mathcal{T}$, it holds that C contains only *positive* occurrences of all those concept names which are defined in $\mathcal{T}$. This is to say, if C contains an occurrence of a concept name defined in $\mathcal{T}$, then this occurrence must appear in the scope of an even number of negation signs $\neg$. Clearly, $\mu\mathcal{T}$ and $\nu\mathcal{T}$ are defined to be formally monotone if and only if $\mathcal{T}$ is.

An immediate consequence of Theorem 3.2 of [Park, 1970] is the fact that for each formally monotone terminology $\mathcal{T}$ and for each *undef*$(\mathcal{T})$-valuation $\mathcal{V}_u$, the function

induced by $\mathcal{T}$ and $\mathcal{V}_u$ is actually monotone. Apart from the formal monotonicity, $\mathcal{T}$ is only required to be equivalent to some first-order formula. The latter condition is actually met by any terminology of $\mathcal{ALC}$. This follows immediately from the transformation outlined in the introduction on page 11.

Proposition 2. *Assume $\mathcal{T}$ is some formally monotone terminology of $\mathcal{ALC}$ and $\mathcal{V}_u$ is an arbitrary undef$(\mathcal{T})$-valuation over Δ. Then the function induced by $\mathcal{T}$ and $\mathcal{V}_u$ is monotone.*

According to the well-known Knaster-Tarski Theorem, monotone functions always have least and greatest fixed points [Tarski, 1955]. In addition, this theorem states that the least and greatest fixed points of every monotone function f are unique, and, in particular, the least fixed point of f is the intersection of all its fixed points, while the greatest fixed point of f is their union [Tarski, 1955]. The Knaster-Tarski Theorem will be presented in more detail in the subsequent chapter. Whenever $\mathcal{T}$ is a *formally monotone* terminology of $\mathcal{ALC}$, the Knaster-Tarski Theorem can be applied to the function induced by $\mathcal{T}$ and an arbitrary *undef*$(\mathcal{T})$-valuation in that the previous lemma assures the monotonicity of this function. According to Proposition 1 this result can be carried over to the corresponding least and greatest fixed-point models of $\mathcal{T}$:

Proposition 3. *Suppose $\mathcal{T}$ is some formally monotone terminology of $\mathcal{ALC}$. Then $\mathcal{T}$ has both a least as well as a greatest fixed-point model. In particular, an arbitrary interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ is a least (or greatest) fixed-point model of $\mathcal{T}$ if and only if for each CN which is defined in $\mathcal{T}$, $CN^{\mathcal{I}}$ is the intersection (or union) of all $CN^{\mathcal{J}}$ where $.^{\mathcal{J}}$ ranges over the interpretation functions of all models of $\mathcal{T}$ which are undef$(\mathcal{T})$-compatible with $\mathcal{I}$.*

In a nutshell, this means that the least and greatest fixed-point terminologies of a formally monotone terminology can be characterized in terms of its ordinary models.

In the introduction we argued that *single* least and greatest fixed-point terminologies are too limited in at least two respects. First, they do not allow for reasoning about different kinds of semantics. Second, we do need both least as well as greatest fixed-point terminologies of $\mathcal{ALC}$ whenever we want to express not only universal, but also existential quantification over the reflexive-transitive closure R^* of a role. In fact, we shall see in Section 2.5 that as far as formally monotone terminologies of $\mathcal{ALC}$ are concerned concepts like $\exists R^*{:}C$ can be defined solely by least fixed-point terminologies, while $\forall R^*{:}C$ can be defined solely by greatest fixed-point terminologies. At this very point it should be stressed again that it it is not possible to resort to the duality $\models \forall R^*{:}C \doteq \neg\exists R^*{:}\neg C$ in this context. To see this recall that $\nu\{A \doteq C \sqcap \forall R{:}A\}$ defines $\forall R^*{:}C$, but $\nu\{A \doteq C \sqcap \forall R{:}A, \overline{A} \doteq \neg A\}$ is neither formally monotone nor does it have any model.

We introduce complex fixed-point terminologies just to overcome these deficiencies. To do so, however, we have to extend the notion of mutual dependency to be applicable to terminologies as well.

Definition 16. Let $\Gamma = \{\mathcal{T}_1, .., \mathcal{T}_n\}$ be a set of terminologies. For every two terminologies, $\mathcal{T}_i$ and $\mathcal{T}_j$, of Γ with $i \neq j$ we say that $\mathcal{T}_i$ **directly uses** $\mathcal{T}_j$ if and only if at least one concept introduction of $\mathcal{T}_i$ directly uses a concept introduction of $\mathcal{T}_j$. Now, let Γ-*uses* denote the transitive closure of *directly uses* over Γ. Then the terminologies $\mathcal{T}_1, .., \mathcal{T}_n$ are said to be **mutually dependent** if and only if there are two terminologies $\mathcal{T}_i$ and $\mathcal{T}_j$ $(1 \leq i, j \leq n)$ with $i \neq j$ which Γ-use each other.

Definition 17. Assume $\mathcal{L}$ is a set of concepts. A finite set $\Gamma = \{\sigma_i \mathcal{T}_i : 1 \leq i \leq n, \sigma_i \in \{\mu, \nu\}\}$ of least and greatest fixed-point terminologies of $\mathcal{L}$ is called **complex fixed-point terminology** of $\mathcal{L}$ if and only if $def(\mathcal{T}_1), ..., def(\mathcal{T}_n)$ are pairwise disjoint, and $\mathcal{T}_1, .., \mathcal{T}_n$ are not mutually dependent.

Such a complex fixed-point terminology Γ is said to be **formally** or **syntactically monotone** if and only if all least and greatest fixed-point terminologies of Γ are formally monotone. Notably, formal monotonicity is required for each *single* fixed-point terminology of Γ rather than for the union of all involved terminologies. Consider, for instance, the following complex fixed-point terminology:

$$\{\nu\{A \doteq C \sqcap \forall R{:}A\}, \mu\{\overline{A} \doteq \neg A\}\}.$$

Here, C is assumed to be some concept not containing any occurrence of A. According to the definition just given, this complex fixed-point terminology is then formally monotone although the union of its two constituent terminologies is not.

Not very surprisingly, an interpretation is defined to be a **model** of a complex fixed-point terminology Γ if and only if it is a model of each least and greatest fixed-point terminology of Γ. It is straightforward to generalize the notion of a *defined* concept as well as semantic relations such as subsumption and equivalence to deal with complex fixed-point terminologies.

We have already seen that every formally monotone terminology of $\mathcal{ALC}$ does have both a least as well as a greatest fixed-point model. The question arises, however, whether this result also applies to an arbitrary formally monotone complex fixed-point terminology $\Gamma = \{\sigma_i \mathcal{T}_i : 1 \leq i \leq n, \sigma_i \in \{\mu, \nu\}\}$ of $\mathcal{ALC}$. According to the definition of complex fixed-point terminologies, the terminologies $\mathcal{T}_1, ..., \mathcal{T}_n$ must not be mutually dependent, so that there is at least one terminology $\mathcal{T}_i$ $(1 \leq i \leq n)$ which does not involve any occurrence of a concept name defined in any other terminology $\mathcal{T}_j$ if $j \neq i$. Moreover, $\mathcal{T}_i$ is formally monotone since Γ is. According to Proposition 3, $\mathcal{T}_i$ does have a least as well as a greatest fixed-point model. Induction on n then proves that Γ has in fact a model as well.

Proposition 4. *Each formally monotone complex fixed-point terminology of $\mathcal{ALC}$ has a model.*

2.4 The Fixed-Point Description Logic $\mathcal{ALC}\mu$

So far we dealt with least and greatest fixed points at the metalevel rather than on the concept level. In what follows, we shall introduce an extension of $\mathcal{ALC}$, called $\mathcal{ALC}\mu$, which comprises explicit least as well as greatest fixed-point operators. $\mathcal{ALC}\mu$ additionally comprises concepts of the form $\mu CN.\mathcal{T}$ and $\nu CN.\mathcal{T}$, where $\mathcal{T}$ stands for an arbitrary formally monotone terminology of $\mathcal{ALC}\mu$, i.e., $\mathcal{T}$ may involve not only concepts of $\mathcal{ALC}$, but also least and greatest fixed-point operators. The meaning of these concepts is given in terms of the function induced by $\mathcal{T}$ and an $undef(\mathcal{T})$-valuation $\mathcal{V}_u$. If $\mathcal{T}$ is a terminology such that $def(\mathcal{T})$ is $\{CN_1, ..., CN_n\}$ and $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ is an interpretation extending $\mathcal{V}_u$, then $(\mu CN_j.\mathcal{T})^{\mathcal{I}}$ and $(\nu CN_j.\mathcal{T})^{\mathcal{I}}$ represent the jth component of the least and the greatest fixed point of the function induced by $\mathcal{T}$ and $\mathcal{V}_u$. However, this is only the case if CN_j is actually defined in $\mathcal{T}$; otherwise $\mu CN_j.\mathcal{T}$ is equivalent to $\bot$, whereas $\nu CN_j.\mathcal{T}$ is equivalent to $\top$.

We consider this extension of $\mathcal{ALC}$ for various reasons. First, it is a rather simple extension to cope with least and greatest fixed points explicitly, and therefore it provides a unifying framework for all three kinds of semantics. Second, $\mathcal{ALC}\mu$ will turn out to be a notational variant of the so-called *propositional μ-calculus*. The propositional μ-calculus is well-understood in terms of its expressive power and computational complexity so that this correspondence will provide us with a better understanding of $\mathcal{ALC}\mu$. Last but not least, a certain fragment of $\mathcal{ALC}\mu$ will turn out to be a suitable framework for analyzing the expressive power and computational complexity of formally monotone complex fixed-point terminologies of $\mathcal{ALC}$.

Definition 18. The concepts and roles of $\boldsymbol{\mathcal{ALC}\mu}$ are inductively defined as follows.

1. Every role name is a role of $\mathcal{ALC}\mu$.
2. Every concept name, $\bot$ and $\top$ are a concepts of $\mathcal{ALC}\mu$.
3. If C and D are concepts of $\mathcal{ALC}\mu$ and RN is a role name, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall RN{:}C$ and $\exists RN{:}C$ are all concepts of $\mathcal{ALC}\mu$.
4. If CN is a concept name and $\mathcal{T}$ is some formally monotone terminology of $\mathcal{ALC}\mu$, then both $\mu CN.\mathcal{T}$ and $\nu CN.\mathcal{T}$ are concepts of $\mathcal{ALC}\mu$. Concepts of this form are called **least** and **greatest fixed-point operators** respectively.
5. These are the concepts and roles of $\mathcal{ALC}\mu$.

In $\mathcal{ALC}\mu$ the notion of formal monotonicity of terminologies is exactly the same as in $\mathcal{ALC}$. If $\mathcal{T}$ contains only one concept introduction, say, $CN \doteq C$, we may use the notation $\mu CN.C$ and $\nu CN.C$ instead of $\mu CN.\mathcal{T}$ and $\nu CN.\mathcal{T}$.

We next extend the notion of an **interpretation** $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ to cope with least and greatest fixed-point operators. We do so by additionally requiring that $(\mu CN.\mathcal{T})^{\mathcal{I}}$ is the intersection and $(\nu CN.\mathcal{T})^{\mathcal{I}}$ is the union of all $CN^{\mathcal{J}}$ where $.^{\mathcal{J}}$ ranges over the interpretation functions of all models of $\mathcal{T}$ which are $undef(\mathcal{T})$-compatible with $\mathcal{I}$. According to Proposition 3, this amounts to requiring that $(\mu CN_j.\mathcal{T})^{\mathcal{I}}$ denotes the jth component of the least fixed point of the function f induced by $\mathcal{T}$ and $\mathcal{V}_u$. For this statement to be true, of course, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ must be an interpretation extending the $undef(\mathcal{T})$-valuation $\mathcal{V}_u$ and $def(\mathcal{T})$ must be of the form $\{CN_1, ..., CN_n\}$ such that $1 \leq j \leq n$. Similarly, $(\nu CN_j.\mathcal{T})^{\mathcal{I}}$ denotes the jth component of the greatest fixed point of the function f.

In [Schild, 1991a], we have shown that $\mathcal{ALC}$ is a notational variant of the propositional multi-modal logic $\mathbf{K}_{(m)}$. The main observation is that the elements of the domain of an interpretation can be thought of as *worlds* or *states* rather than objects. Consequently, concept names can be viewed as propositional variables denoting the set of worlds in which they hold, and $\top$, $\bot$, $\sqcap$, $\sqcup$ and $\neg$ naturally correspond to the logical connectives *true*, *false*, $\wedge$, $\vee$ and to $\neg$. But then $\forall RN$: and $\exists RN$: become RN-indexed modalities of necessity $[RN]$ and of possibility $\langle RN \rangle$ respectively. This explains why $\mathcal{ALC}$ is a notational variant of the propositional multi-modal logic $\mathbf{K}_{(m)}$, for details, the reader is referred to [Schild, 1991a]. The *propositional μ-calculus* extends $\mathbf{K}_{(m)}$ by explicit least and greatest fixed point operators to reason about concurrent programs. The propositional version of the μ-calculus has been proposed by Kozen [1983], while Vardi and Wolper [1984] investigated the propositional μ-calculus with multiple fixed points. The fixed-point operators of the latter directly correspond to those of $\mathcal{ALC}\mu$. The only difference is that $\mu CN_j.\mathcal{T}$ and $\nu CN_j.\mathcal{T}$ are written as $\mu CN_j(CN_1, ..., CN_n):(C_1, ..., C_n)$ and as $\nu CN_j(CN_1, ..., CN_n):(C_1, ..., C_n)$, provided that $\mathcal{T}$ is of the form $\{CN_i \doteq C_i : 1 \leq i \leq n\}$.

Correspondence Theorem 1. *$\mathcal{ALC}\mu$ is a notational variant of the propositional μ-calculus with multiple fixed points.*

In view of of this correspondence, we assume henceforth that all results shown for the propositional μ-calculus and its variants are also shown for $\mathcal{ALC}\mu$ and the corresponding variants.

It is worth mentioning that the meaning of the concepts $\mu CN.\mathcal{T}$ and $\nu CN.\mathcal{T}$ is preserved by renaming any of those concept names which are defined in $\mathcal{T}$. This is because $(\mu CN.\mathcal{T})^{\mathcal{I}}$ and $(\nu CN.\mathcal{T})^{\mathcal{I}}$ always gives a local meaning to all concept names defined in $\mathcal{T}$. Of course, the local character of the concept names defined in $\mathcal{T}$ is just a consequence of the fact that $(\mu CN.\mathcal{T})^{\mathcal{I}}$ and $(\nu CN.\mathcal{T})^{\mathcal{I}}$ are to yield a denotation of

CN which is the result of a mutual minimalization or maximalization process of the denotations of all these concept names. The following lemma, due to Kozen [1983, Proposition 5.7(i)], is devoted to a renaming of concept names defined in $\mathcal{T}$.

Lemma 1. *Assume $\mathcal{T}$ is an arbitrary terminology such that CN_i is defined in this terminology, but there is no occurrence of the concept name A_i in $\mathcal{T}$. Suppose, moreover, $(\mu CN_j.\mathcal{T})_{CN_i/A_i}$ and $(\nu CN_j.\mathcal{T})_{CN_i/A_i}$ are obtained from $\mu CN_j.\mathcal{T}$ and from $\nu CN_j.\mathcal{T}$ by replacing each occurrence of CN_i with A_i. Then the following equivalences hold if $i \neq j$:*

$$\begin{aligned} \models \quad \mu CN_j.\mathcal{T} \quad &\doteq \quad (\mu CN_j.\mathcal{T})_{CN_i/A_i}, \\ \models \quad \nu CN_j.\mathcal{T} \quad &\doteq \quad (\nu CN_j.\mathcal{T})_{CN_i/A_i}. \end{aligned}$$

In view of this Lemma, we henceforth assume that in any concept, C, of $\mathcal{ALC}\mu$, no concept name is defined in more than one terminology occurring in C. This is to say, there are no two different occurrences of fixed-point operators in C such that there is at least one concept name which is defined in both of the two fixed-point's terminologies.

The reader may have wondered why there is only an indication of the least fixed-point operator in the name '$\mathcal{ALC}\mu$' although it extends $\mathcal{ALC}$ not only by least, but also by greatest fixed-point operators. The reason for this is that we can eliminate greatest fixed-point operators in favor of least fixed-point operators and vice versa. For instance, the least fixed-point operator $\mu A.\{A \doteq C \sqcup \exists R{:}A\}$ is equivalent to the negated greatest fixed-point operator $\neg\nu A.\{A \doteq \neg(C \sqcup \exists R{:}\neg A)\}$. It is important to realize that the terminology $\{A \doteq \neg(C \sqcup \exists R{:}\neg A)\}$ is syntactically monotone if and only if so is $\{A \doteq C \sqcup \exists R{:}A\}$. Park [1970] showed that this equivalence can be generalized as follows: Suppose $\tilde{C}_i$ is obtained from C_i by replacing all occurrences of concept names defined in $\mathcal{T}$ by their negated form. According to Theorem 2.3 of [Park, 1970], then the following equivalences hold:

$$\begin{aligned} &\models \neg\mu CN.\mathcal{T} \doteq \nu CN.\{CN_i \doteq \neg\tilde{C}_i : (CN_i \doteq C_i) \in \mathcal{T}\}, \\ &\models \neg\nu CN.\mathcal{T} \doteq \mu CN.\{CN_i \doteq \neg\tilde{C}_i : (CN_i \doteq C_i) \in \mathcal{T}\}. \end{aligned}$$

As $\neg\tilde{C}_i$ adds exactly two negations to the scope of each occurrence of a concept name which is defined in $\mathcal{T}$, the terminology $\{CN_i \doteq \neg\tilde{C}_i : (CN_i \doteq C_i) \in \mathcal{T}\}$ is clearly formally monotone if and only if $\mathcal{T}$ is formally monotone.

These equivalences are crucial to obtain a negation normal form. As usual, the negation normal form of a concept is an equivalent concept involving no negated compound concepts. In case of $\mathcal{ALC}$, it can be obtained by exploiting de Morgan's laws as well as the equivalences $\models \neg\forall R{:}C \doteq \exists R{:}\neg C$ and $\models \neg\exists R{:}C \doteq \forall R{:}\neg C$. In case of $\mathcal{ALC}\mu$, however, we additionally have to exploit the dualities for least and greatest fixed-point operators given above.

Definition 19. The function ***nnf*** maps concepts of $\mathcal{ALC}\mu$ to concepts of $\mathcal{ALC}\mu$. Applied to an arbitrary concept, *nnf* yields the concept obtained from the original concept by repeatedly applying the following substitution rules:

$$\begin{array}{lcl}
\neg\neg C & \leadsto & C, \\
\neg(C \sqcap D) & \leadsto & \neg C \sqcup \neg D, \\
\neg(C \sqcup D) & \leadsto & \neg C \sqcap \neg D, \\
\neg(\forall R{:}C) & \leadsto & \exists R{:}\neg C, \\
\neg(\exists R{:}C) & \leadsto & \forall R{:}\neg C, \\
\neg\mu CN.\mathcal{T} & \leadsto & \nu CN.\{CN_i \doteq \neg\tilde{C}_i : (CN_i \doteq C_i) \in \mathcal{T}\}, \\
\neg\nu CN.\mathcal{T} & \leadsto & \mu CN.\{CN_i \doteq \neg\tilde{C}_i : (CN_i \doteq C_i) \in \mathcal{T}\}.
\end{array}$$

As above, $\tilde{C}_i$ denotes the concept obtained from C by simultaneously replacing each occurrence of $CN_1, ..., CN_n$ with its negated form. We call $nnf(T)$ the **negation normal form** of T.

Lemma 2. *Every concept of $\mathcal{ALC}\mu$ is equivalent to its negation normal form.*

In view of the fact that many μ-calculi considered in the literature do not allow for mutual fixed points, we should clarify the actual role of these: By *mutual fixed points* we mean least or greatest fixed-point operators applied to terminologies comprising more than one concept introduction. It turns out that we can eliminate mutual fixed-points in favor of nested ones. Consider, for instance, the mutual fixed-point $\nu A.\{A \doteq \forall R{:}B, B \doteq \forall S{:}(A \sqcap B)\}$. This concept is in fact equivalent to $\nu A.\{A \doteq \forall R{:}\nu B.\{B \doteq \forall S{:}(A \sqcap B)\}\}$, which obviously does not contain any mutual fixed-point. The following lemma just generalizes this observation.

Lemma 3. *Assume $\mathcal{T}$ is a formally monotone terminology of $\mathcal{ALC}\mu$ which is of the form $\{CN_i \doteq C_i : 1 \leq i \leq n\}$. Assume, moreover, $\hat{C}_j$ and $\check{C}_j$ are obtained from C_j by simultaneously replacing all occurrences of every CN_l $(l \neq j)$ with $\mu CN_l.\{CN_l \doteq C_l\}$ and with $\nu CN_l.\{CN_l \doteq C_l\}$ respectively. Then $\mu CN_j.\mathcal{T}$ is equivalent to $\mu CN_j.\{CN_j \doteq \hat{C}_j\}$ and $\nu CN_j.\mathcal{T}$ is equivalent to $\nu CN_j.\{CN_j \doteq \check{C}_j\}$.*

A proof of this lemma is given inter alia in [de Bakker, 1980, Theorem 5.14.e]. Of course, a finite number of applications of this lemma eliminates all mutual fixed points.

Corollary 1. *Every concept of $\mathcal{ALC}\mu$ is equivalent to a concept of $\mathcal{ALC}\mu$ which involves solely terminologies which contain at most one concept introduction.*

Unfortunately, the size of the equivalent concept is not always bounded polynomially in the size of the original concept.

As Proposition 3 suggests, there exists a close relationship between formally monotone least and greatest fixed-point terminologies on the one hand, and least and greatest fixed-point operators of $\mathcal{ALC}\mu$ on the other hand. The least fixed-point terminology $\mu\{CN \doteq C\}$, for instance, has exactly the same models as the concept introduction $CN \doteq \mu A.\{A \doteq C_{CN/A}\}$, where $C_{CN/A}$ is obtained from C by replacing all occurrences of CN with A. Note, however, that the concept names which are defined in $\mathcal{T}$ have a *local* meaning in $\mu A.\mathcal{T}$ and in $\nu A.\mathcal{T}$, whereas those defined in fixed-point terminologies have a *global* meaning. This is due to the fact that according to Lemma 1 the meaning of the concepts $\mu A.\mathcal{T}$ and $\nu A.\mathcal{T}$ is preserved by *renaming* each concept name which is defined in $\mathcal{T}$. In contrast to this, renaming defined concepts does change the meaning of least and greatest fixed-point terminologies. Therefore, for *each* concept introduction $CN_i \doteq C_i$ of the least fixed-point terminology $\mu\mathcal{T}$, a concept introduction $CN_i \doteq \mu A_i.\mathcal{T}_{CN_i/A_i}$ is needed.

Proposition 5. *Assume $\mathcal{T}$ is some formally monotone terminology of $\mathcal{ALC}$ which is of the form $\{CN_i \doteq C_i : 1 \leq i \leq n\}$ and which does not contain any of the (pairwise distinct) concept names $A_1, ..., A_n$. Suppose, furthermore, $\mathcal{T}_A$ is obtained from $\mathcal{T}$ by replacing each occurrence of $CN_1, ..., CN_n$ with $A_1, ..., A_n$ respectively. Then $\mu\mathcal{T}$ has the same models as $\{CN_i \doteq \mu A_i.\mathcal{T}_A : 1 \leq i \leq n\}$ and $\nu\mathcal{T}$ has the same models as $\{CN_i \doteq \nu A_i.\mathcal{T}_A : 1 \leq i \leq n\}$.*

This proposition describes how to represent formally monotone least and greatest fixed-point terminologies of $\mathcal{ALC}$ by terminologies of $\mathcal{ALC}\mu$. It should be remarked that we could have taken also $\mu CN_i.\mathcal{T}$ and $\nu CN_i.\mathcal{T}$ instead of $\mu A_i.\mathcal{T}_A$ and $\nu A_i.\mathcal{T}_A$ because renaming defined concepts does not change the meaning of least and greatest fixed-point operators. However, we have taken the ones above because they end up with *acyclic* terminologies. Let us take a closer look at these terminologies. $\mathcal{T}_A$ is clearly a terminology of $\mathcal{ALC}$ since $\mathcal{T}$ is assumed to be a terminology of $\mathcal{ALC}$. This means neither $\mu A_i.\mathcal{T}_A$ nor $\nu A_i.\mathcal{T}_A$ involve any nested fixed-point operators. In fact, we shall see that we do not need the full power of $\mathcal{ALC}\mu$ to represent formally monotone fixed-point terminologies of $\mathcal{ALC}$. In particular, we do not need nested *alternating* least and greatest fixed-point operators interacting via a defined concept.

Definition 20. A concept C of $\mathcal{ALC}\mu$ is called **restricted** if and only if its negation normal form does not contain any least fixed-point operator $\mu CN.\mathcal{T}$ (or greatest fixed-point operator $\nu CN.\mathcal{T}$) which involves some greatest (or least) fixed-point operator in which a concept defined in $\mathcal{T}$ occurs. We denote with $\boldsymbol{\mathcal{ALC}\mu^{-}}$ the set of all restricted concepts of $\mathcal{ALC}\mu$.

Consider, for instance, the concept $\mu\boldsymbol{A}.\{\boldsymbol{A} \doteq \forall R{:}\nu B.\{B \doteq \boldsymbol{A} \sqcap \forall S{:}B\}\}$. First, it is already in negation normal form. Second, it comprises a greatest fixed-point

operator, viz. $\nu B.\{B \doteq \boldsymbol{A} \sqcap \forall S{:}B\}$, which is nested in a least fixed-point operator of the form $\mu \boldsymbol{A}.\mathcal{T}$. As there is an occurrence of $\boldsymbol{A}$ in the greatest fixed-point operator $\nu B.\{B \doteq \boldsymbol{A} \sqcap \forall S{:}B\}$, the concept above is not restricted and is therefore no concept of $\mathcal{ALC}\mu^-$. Observe, however, that $\mu A.\{A \doteq \forall R{:}\mu B.\{B \doteq A \sqcap \forall S{:}B\}\}$ *is* restricted.

Remarkably, both Lemma 3 and Corollary 1 hold for $\mathcal{ALC}\mu^-$ as well. This is because $\mathcal{ALC}\mu^-$ restricts only the interaction of nested *alternating* least and greatest fixed-points operators which are not needed to eliminate mutual fixed-points in favor of nested ones.

The following theorem is just an immediate consequence of the observation made above that the way Proposition 5 reprepresents formally monotone complex fixed-point terminologies of $\mathcal{ALC}$ in terms of acyclic terminologies of $\mathcal{ALC}\mu$ does not encounter any non-restricted concept. This is to say, acyclic terminologies of $\mathcal{ALC}\mu^-$ do suffice for this purpose.

Representation Theorem 1. *There is a function π which maps an arbitrary formally monotone complex fixed-point terminology Γ of $\mathcal{ALC}$ to some acyclic terminology $\pi(\Gamma)$ of $\mathcal{ALC}\mu^-$ in such a way that Γ and $\pi(\Gamma)$ have exactly the same models. Additionally, π is computable in polynomial time and the size of $\pi(\Gamma)$ is linearly bounded in the size of Γ.*

By **size** we mean the length when considered as a string over $\mathcal{N}$, $\sqcap$, $\sqcup$, $\neg$, $\forall$, $\exists$, μ and ν.

2.5 The Expressive Power of Recursion in $\mathcal{ALC}$

In this section we shall investigate the expressive power of fixed-point terminologies. In particular, we shall see that the concepts definable by formally monotone complex fixed-point terminologies of $\mathcal{ALC}$ are exactly those concepts which are equivalent to concepts of $\mathcal{ALC}\mu^-$. We then give a strict lower bound of the expressive power of $\mathcal{ALC}\mu^-$ and of full $\mathcal{ALC}\mu$ in terms of $\mathcal{ALC}$ augmented by regular and ω-regular role expressions. Of course, before engaging into details, we have to clarify what exactly is meant by *expressive power* and *definability*.

Definition 21. Suppose $\mathcal{L}$ and $\mathcal{L}'$ are two sets of concepts. Then $\mathcal{L}$ is **at least as strong in expressive power** as $\mathcal{L}'$, $\mathcal{L}' \leq \mathcal{L}$ for short, if and only if for each concept in $\mathcal{L}'$ there is at least one equivalent concept in $\mathcal{L}$, and $\mathcal{L}$ is **strictly stronger in expressive power** than $\mathcal{L}'$ if and only if $\mathcal{L}' \leq \mathcal{L}$, but it is not the case that $\mathcal{L} \leq \mathcal{L}'$. Furthermore, a concept C is **definable** by a set of complex fixed-point terminologies of $\mathcal{L}$ if and only if there is an element Γ of the set and there is a concept name CN which is defined in Γ such that $\Gamma \models CN \doteq C$.

This is to say, $\mathcal{L}$ is at least as strong in expressive power as $\mathcal{L}'$ just in case that for each concept in $\mathcal{L}'$ there is a concept in $\mathcal{L}$ which has exactly the same meaning, though the two concepts may differ in their syntax. If there is in addition a concept in $\mathcal{L}$ which is not equivalent to any concept of $\mathcal{L}'$, $\mathcal{L}$ is said to be strictly stronger in expressive power than $\mathcal{L}'$. For example, it can be shown that $\mathcal{ALC}$ augmented by the reflexive-transitive closure RN^* of a role name is strictly stronger in expressive power than $\mathcal{ALC}$. The definition of definability of concepts takes into account the fact that the definitional power of terminologies consists in the concept names which they define. The concept $\exists RN^*{:}C$, for instance, is definable by formally monotone complex fixed-point terminologies of $\mathcal{ALC}$ since $\mu\{A \doteq C \sqcup \exists RN{:}A\} \models A \doteq \exists RN^*{:}C$, provided that A is a concept name not occurring in C.

Expressiveness Theorem 1. *The concepts definable by formally monotone complex fixed-point terminologies of $\mathcal{ALC}$ are exactly those concepts equivalent to concepts of $\mathcal{ALC}\mu^-$.*

A corresponding result for the propositional μ-calculus was given in [Cleaveland and Steffen, 1991]. This result is of great importance in that it justifies $\mathcal{ALC}\mu^-$ (rather than full $\mathcal{ALC}\mu$) as a unifying framework for the least and the greatest fixed-point semantics. One part of the proof, viz. the proof that every concept of $\mathcal{ALC}\mu^-$ is definable by some formally monotone fixed-point terminology of $\mathcal{ALC}$, can be based on Proposition 5. In particular, it can be shown that every least fixed-point operator $\mu CN_i.\mathcal{T}$ of $\mathcal{ALC}\mu^-$ can be replaced with some fresh concept name A_i if the least fixed-point terminology $\mu\mathcal{T}_A$ is added and $\mathcal{T}_A$ is defined as in Proposition 5. Of course, every greatest fixed-point operator can be eliminated in an analogous way too. All fixed-point terminologies needed are then collected together in a complex fixed-point terminology. Of course, for this being possible all these fixed-point terminologies must not be mutually dependent. Nested fixed-point operators interacting via a defined concept, however, might cause problems in this respect. Eliminating the two fixed-point operators of $\mu\boldsymbol{A}.\{\boldsymbol{A} \doteq \forall R{:}\mu B.\{B \doteq \boldsymbol{A} \sqcap \forall S{:}B\}\}$, for instance, in the way just described would yield two mutually dependent fixed-point terminologies. This is why concepts such as the one above must be replaced by the equivalent concepts containing no nested fixed-point operator any more, at least if the nested fixed-point operators interact via a defined concept. The concept above, for instance, must be replaced by $\mu A.\{A \doteq \forall R{:}B, B \doteq A \sqcap \forall S{:}B\}$. According to Lemma 3, an elimination of nested fixed-point operators of this kind is always possible. Nested *alternating* fixed-point operators, however, cannot be eliminated in this way. But such concepts are, of course, not restricted.

Now, recall that Corollary 1 states that each concept of $\mathcal{ALC}\mu$ is equivalent to one which involves solely terminologies comprising at most one concept introduction. Recall furthermore that we already noted that this holds also for restricted concepts of $\mathcal{ALC}\mu$. But then we can immediately conclude the following theorem.

Expressiveness Theorem 2. *$\mathcal{ALC}\mu$ involving no terminology which comprises more than one concept introduction is at least as strong in expressive power as full $\mathcal{ALC}\mu$. The corresponding statement holds for $\mathcal{ALC}\mu^-$ as well.*

We next compare both $\mathcal{ALC}\mu^-$ and full $\mathcal{ALC}\mu$ with the regular and the ω-regular extension of $\mathcal{ALC}$ in their expressive power. For the **regular extension** of $\mathcal{ALC}$ see [Baader, 1991] or [Schild, 1991a]. It additionally comprises the reflexive-transitive closure R^* of a role, the composition $R \circ S$ and union $R \sqcup S$ of two roles, the identity role ϵ, as well as the role $R|C$ restricting the range of a role to a concept. The **ω-regular extension** of $\mathcal{ALC}$ extends its regular extension by the additional concept $\exists R^\omega$, which stipulates the existence of an infinite chain of the role R.

It is worth mentioning that this language can sometimes be used to clarify the actual meaning of fixed-point terminologies. For instance, Streett [1985, page 364] mentioned the following equivalences:

$$\begin{aligned} \models \quad & \mu A.\{A \doteq C \sqcap \forall R{:}A\} \quad \doteq \quad (\forall R^*{:}C) \sqcap \neg\exists R^\omega, \\ \models \quad & \nu A.\{A \doteq C \sqcup \exists R{:}A\} \quad \doteq \quad (\exists R^*{:}C) \sqcup \quad \exists R^\omega. \end{aligned}$$

Of course, A has to be some concept name not appearing in C. According to Proposition 5, both equivalences can be carried over directly to the corresponding fixed-point terminologies:

$$\begin{aligned} \mu\{A \doteq C \sqcap \forall R{:}A\} \quad \models \quad A \quad \doteq \quad (\forall R^*{:}C) \sqcap \neg\exists R^\omega, \\ \nu\{A \doteq C \sqcup \exists R{:}A\} \quad \models \quad A \quad \doteq \quad (\exists R^*{:}C) \sqcup \quad \exists R^\omega. \end{aligned}$$

This indicates that *some* concepts of the ω-regular extension of $\mathcal{ALC}$ are definable by formally monotone complex fixed-point terminologies of $\mathcal{ALC}$. The next theorem together with Expressiveness Theorem 1 implies that formally monotone complex fixed-point terminologies of $\mathcal{ALC}$ are in fact able to define *all* concepts of the *regular* extension of $\mathcal{ALC}$. They are even able to define concepts which are not equivalent to any concept of the regular extension of $\mathcal{ALC}$.

Expressiveness Theorem 3. *$\mathcal{ALC}\mu^-$ is* strictly *stronger in expressive power than the regular extension of $\mathcal{ALC}$, while $\mathcal{ALC}\mu$ is* strictly *stronger in expressive power than the ω-regular extension of $\mathcal{ALC}$.*

Proof. Consider the following equivalences, due to Kozen [1983], which presuppose that A is some concept name not occurring in C:

$$\begin{aligned}
&\models \forall R{:}C &&\doteq \neg\exists R{:}\neg C,\\
&\models \exists R^*{:}C &&\doteq \mu A.\{A \doteq C \sqcup \exists R{:}A\},\\
&\models \exists(R \circ S){:}C &&\doteq \exists R{:}\exists S{:}C,\\
&\models \exists(R \sqcup S){:}C &&\doteq (\exists R{:}C) \sqcup (\exists S{:}C),\text{[4]}\\
&\models \exists(R|C){:}D &&\doteq \exists R{:}(C \sqcap D),\\
&\models \exists\epsilon{:}C &&\doteq C,\\
&\models \exists R^\omega &&\doteq \nu A.\{A \doteq \exists R{:}A\}.
\end{aligned}$$

These equivalences can be used directly to prove by induction on the structure of concepts of the ω-regular extension of $\mathcal{ALC}$ that $\mathcal{ALC}\mu$ is at least as strong in expressive power as the ω-regular extension of $\mathcal{ALC}$. As A does not occur in C, all but the last equivalence yield restricted concepts in case that C and D are restricted. The last equivalence, however, may yield concepts which are not restricted. For instance, $\exists(R \circ S^*)^\omega$ is equivalent to $\nu\boldsymbol{A}.\{\boldsymbol{A} \doteq \exists R{:}\mu B.\{B \doteq \boldsymbol{A} \sqcup \exists S{:}B\}\}$, which is obviously not restricted. This concerns, however, solely the last of the above equivalences, so that $\mathcal{ALC}\mu^-$ is yet at least as strong in expressive power as the *regular* extension of $\mathcal{ALC}$. Now, according to Kozen [1983, Proposition 4.1], there is at least one concept of $\mathcal{ALC}\mu^-$ which is not equivalent to any concept of the regular extension of $\mathcal{ALC}$, viz. $\nu A.\{A \doteq \exists RN{:}A\}$. Despite the fact that the latter concept is equivalent to a concept of the ω-regular extension of $\mathcal{ALC}$, Niwinsky has shown that this does not apply to $\nu A.\{A \doteq \exists RN_1{:}A \sqcap \exists RN_2{:}A\}$ [Streett, 1985, Theorem 2.7]. □

2.6 The Computational Complexity of Recursion in $\mathcal{ALC}$

In what follows we shall see that as far as formally monotone terminologies of $\mathcal{ALC}$ are concerned, all three kinds of semantics essentially do not differ in the computational complexity of the corresponding subsumption relation. In each case, subsumption turns out to be complete for deterministic exponential time. To be more accurate, we investigate the computational complexity of the following three problems. For an arbitrary formally monotone terminology $\mathcal{T}$ of $\mathcal{ALC}$ and for an arbitrary primitive concept introduction $CN \sqsubseteq C$ of $\mathcal{ALC}$ decide whether (a) $\mathcal{T} \models CN \sqsubseteq C$, (b) $\mu\mathcal{T} \models CN \sqsubseteq C$, and (c) $\nu\mathcal{T} \models CN \sqsubseteq C$. It turns out that all three problems are hard for deterministic exponential time, even if $\mathcal{T}$ is restricted such that it contains at most one concept introduction. All these lower complexity bounds are obtained

[4] If we were concerned with linear length-boundedness, we could have taken the equivalent concept $\mu A.\{A \doteq (\exists R{:}B) \sqcup (\exists S{:}B), B \doteq C\}$ instead.

from a result of Fischer and Ladner [1979], which proves that the set of coherent concepts of the regular extension of $\mathcal{ALC}$ is hard for deterministic exponential time. Inspection of their proof reveals that the syntactic form of the concepts can be restricted considerably. In fact, the set of all coherent concepts of the form $C \sqcap \forall RN^*{:}D$ such that both C and D are concepts of $\mathcal{ALC}$ and RN is a role name is hard for deterministic exponential time as well. We shall prove this set to be polynomial-time (many-one) reducible to each of the three problems mentioned above. The proof involves the following reductions:

$$\begin{array}{lllll} & \models C \sqcap \forall RN^*{:}D \doteq \bot & & & \\ \textit{iff} & A \sqsubseteq D \sqcap \forall RN{:}A & \models & A & \sqsubseteq \neg C \\ \textit{iff} & \nu\{A \doteq D \sqcap \forall RN{:}A\} & \models & A & \sqsubseteq \neg C \\ \textit{iff} & \mu\{A \doteq \neg D \sqcup \exists RN{:}A\} & \models & \neg A & \sqsubseteq \neg C. \end{array}$$

The last two reductions are immediate consequences of the fact that the corresponding fixed-point terminologies entail $A \doteq \forall RN^*{:}D$ and $A \doteq \exists RN^*{:}\neg D$ respectively. The first reduction is, however, more involved. Anyway, Fischer and Ladner's result holds even if C is of the form $CN \sqcap C'$, where CN is a concept name, so that the axiom $\neg A \sqsubseteq \neg C$ can be shown to be equivalent to some *primitive* concept introduction, viz. $CN \sqsubseteq A \sqcup \neg C'$.

We shall also show that the entailment relation that integrates all three kinds of semantics is computable in deterministic exponential time. By that we mean the problem to decide whether $\mathcal{A} \cup \Gamma \models C \doteq D$, for arbitrary formally monotone complex fixed-point terminologies Γ of $\mathcal{ALC}$ and arbitrary finite sets $\mathcal{A} \cup \{C \doteq D\}$ of axioms of $\mathcal{ALC}\mu^-$. According to Proposition 5, all fixed-point terminologies of Γ can be represented by acyclic terminologies of $\mathcal{ALC}\mu^-$, so that we may assume Γ to be empty. Now, Vardi and Wolper [1984] showed the set of coherent concepts of $\mathcal{ALC}\mu^-$ to be computable in deterministic exponential time.[5] In the same paper Vardi and Wolper also showed that each concept C of $\mathcal{ALC}\mu^-$ is coherent if and only if there is a *tree-like* interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ such that the empty word is an element of $C^{\mathcal{I}}$. This ensures that any axiom $C \doteq D$ can be internalized within $\mathcal{ALC}\mu^-$ using the technique introduced independently by Baader [1991] and Schild [1991a]. This means Vardi and Wolper's result can be shown to hold also for subsumption with respect to finite sets of axioms of $\mathcal{ALC}\mu^-$.

Before going into details, the reader should recall some basic notions of structural complexity theory. First, problems are usually represented as *sets*. The first of the problems mentioned above thus will be represented by the set of all tuples $\langle \mathcal{T}, C \doteq D \rangle$ such that $\mathcal{T}$ ranges over all syntactically monotone terminologies of $\mathcal{ALC}$ and $C \doteq D$

[5] Concerning the computational complexity of full $\mathcal{ALC}\mu$, the following is known: Streett and Emerson [1989] gave an elementary upper time bound for accepting the set of coherent concepts of $\mathcal{ALC}\mu$, while Safra [1988] as well as Emerson and Jutla [1988] proved that this set is even computable in deterministic exponential time, at least when no mutual fixed points are involved.

ranges over all axioms of $\mathcal{ALC}$ with $\mathcal{T} \models C \doteq D$. The following fundamental notion of structural complexity theory states intuitively that some set S is at least as hard as another set T. T is called **polynomial-time *m*-reducible** to S if and only if there is a total function $\pi : T \to S$ computable in polynomial-time such that for all x, $x \in T$ if and only if $\pi(x) \in S$. If, in addition, there is some constant $c > 0$ such that for all x, $|\pi(x)| \leq c|x|$, then T is said to be **polynomial-time *lin*-reducible** to S. It can easily be seen that both polynomial-time m-reducibility and polynomial-time *lin*-reducibility are preorders, i.e., they constitute a reflexive and transitive relation on sets. Given a class $\mathcal{C}$, a set S is called **polynomial-time hard** for $\mathcal{C}$ if and only if every element of $\mathcal{C}$ is polynomial-time m-reducible to S, and a set is **polynomial-time complete** for $\mathcal{C}$ if and only if it is polynomial-time hard for $\mathcal{C}$ and it is a member of $\mathcal{C}$. We omit the corresponding definitions for log space reductions. For any function t with $t(n) > n$, we denote with **DTIME(*t*)** the class of all sets accepted by deterministic Turing machines whose running time is bounded above by $t(n)$, for each input of length n. Similarly, for any function s with $s(n) \geq 1$, **DSPACE(*s*)** denotes the class of all sets accepted by deterministic Turing machines whose work space is bounded above by $s(n)$. We define **P** as $\bigcup_{i \geq 0}$ DTIME(n^i), **DEXT** as $\bigcup_{c>0}$ DTIME(2^{cn}), **EXPTIME** as $\bigcup_{i \geq 0}$ DTIME(2^{n^i}), and **PSPACE** as $\bigcup_{i \geq 0}$ DSPACE(n^i). We shall make use of the fact that EXPTIME is closed under polynomial-time m-reducibility, whereas DEXT is closed under polynomial-time *lin*-reducibility. That is, a set is a member of EXPTIME if it is polynomial time m-reducible to some set in EXPTIME and a set is a member of DEXT if it is polynomial-time *lin*-reducible to some set in DEXT. Note, however, that DEXT is not closed under polynomial time m-reducibility. After all, for each function f, DTIME(f) is closed under complementation, i.e., a set is member of DTIME(f) if and only if its complement is a member of DTIME(f). This implies, for instance, that P = co-P and EXPTIME = co-EXPTIME and, therefore, any set is polynomial-time hard for EXPTIME if and only if it is polynomial-time hard for co-EXPTIME. According to the well-known linear speed-up theorem, it holds for each constant $c > 0$ that DTIME(2^{cn}) = $\bigcup_{d>0}$ DTIME($d2^{cn}$). For details the reader is referred to Chapter 3 of Balcázar *et al.* [1988].

For the complexity results to be presented, it is worth mentioning that P $\subseteq$ PSPACE $\subseteq$ EXPTIME and, moreover, P is *strictly* included in DEXT which in turn is *strictly* included in EXPTIME. It is also known that the class of all sets acceptable in deterministic linear space, i.e., $\bigcup_{c>0}$ DSPACE(cn) is included in DEXT. It is not known, though, whether the whole of PSPACE is included in DEXT or vice versa. The only fact that *is* known is that PSPACE $\neq$ DEXT. For all these results the reader is referred to Theorem 2.8, Proposition 3.1, and Exercise 14 of Chapter 3.9 in [Balcázar *et al.*, 1988].

Lower Bounds

For the lower complexity bounds, we shall utilize a result due to Fischer and Ladner [1979]. It states roughly that accepting the set of coherent concepts of the regular extension of $\mathcal{ALC}$ is polynomial-time hard for EXPTIME and requires more than deterministic time $c^{n/\log n}$, for some constant $c > 1$, even if the concepts contain at most one occurrence of * [Fischer and Ladner, 1979, Lemma 4.1 & Theorem 4.4]. Harel [1984] observed that Fischer and Ladner's proof also shows that this set is not acceptable in deterministic time 2^{cn}, for some constant $c > 0$ [Harel, 1984, Theorem 2.14]. Inspection of Fischer and Ladner's proof immediately reveals that the syntactic form of the concepts can be restricted further:

Proposition 6. *The set of all coherent concepts of the form $C \sqcap \forall RN^*{:}D$ such that both C and D are concepts of $\mathcal{ALC}$ and RN is a role name, henceforth denoted with* **FL**, *is polynomial-time hard for* EXPTIME. *There is moreover a constant $c > 0$ such that* FL *is* not *a member of* DTIME(2^{cn}). *Both results hold even if C is solely composed of concept names, their negations, as well as $\sqcap$ and, additionally, at least one concept name occurs in C positively.*

First, we give a lower bound for computing the subsumption relation with respect to the descriptive semantics.

Complexity Theorem 1. *The set of all $\langle \mathcal{T}, CN \sqsubseteq C\rangle$ such that $\mathcal{T}$ ranges over all syntactically monotone terminologies of $\mathcal{ALC}$ and $CN \sqsubseteq C$ ranges over all concept introductions of $\mathcal{ALC}$ with $\mathcal{T} \models CN \sqsubseteq C$ is polynomial-time hard for* EXPTIME. *Moreover, there is a constant $c > 0$ such that this set is* not *a member of* DTIME(2^{cn}). *Both results hold even if $\mathcal{T}$ may contain at most one concept introduction.*

Proof. In what follows, we shall prove the following. Assume C and D are arbitrary concepts which do not contain any occurrence of the concept name A. It then holds that:

$$\models C \sqcap \forall RN^*{:}D \doteq \bot \quad \textit{iff} \quad A \sqsubseteq D \sqcap \forall RN{:}A \models A \sqsubseteq \neg C. \tag{2.5}$$

Take, just for a moment, this reduction for granted. But then the *complement* of FL and the following set is polynomial-time *lin*-reducible to each other: The set of all $\langle \{A \sqsubseteq D \sqcap \forall RN{:}A\}, A \sqsubseteq \neg C\rangle$ such that C and D range over all concepts of $\mathcal{ALC}$ which do not contain any occurrence of A such that $A \sqsubseteq D \sqcap \forall RN{:}A \models A \sqsubseteq \neg C$. As FL is polynomial-time hard for EXPTIME, the latter set is hard for EXPTIME as well. Now, if this set were a member of $\bigcap_{c>0}$ DTIME(2^{cn}), the complement of FL and thus FL itself would be elements of $\bigcap_{c>0}$ DTIME(2^{cn}). This would contradict, however, Proposition 6 which states that for some constant $c > 0$, FL is *not* a member of DTIME(2^{cn}).

We shall prove both directions of (2.5) by contraposition. For the if-part, suppose $C \sqcap \forall RN^*{:}D$ is not equivalent to $\bot$. That is, there is at least one interpretation $\mathcal{I} = \langle \Delta^\mathcal{I}, .^\mathcal{I} \rangle$ such that $C^\mathcal{I} \cap (\forall RN^*{:}D)^\mathcal{I} \neq \bot^\mathcal{I} = \emptyset$. Clearly, $C^\mathcal{I} \cap (\forall RN^*{:}D)^\mathcal{I} \neq \emptyset$ holds exactly when

$$(\forall RN^*{:}D)^\mathcal{I} \quad \not\subseteq \quad \Delta^\mathcal{I} \setminus C^\mathcal{I} \quad = \quad \neg C^\mathcal{I}. \tag{2.6}$$

Suppose N is the set of all concept and role names occurring in $C \sqcap \forall RN^*{:}D$. Consider some interpretation $\mathcal{J} = \langle \Delta^\mathcal{I}, .^\mathcal{J} \rangle$ which is N-compatible with $\mathcal{I}$ such that $A^\mathcal{J} = (\forall RN^*{:}D)^\mathcal{I}$. As A occurs neither in C nor in D, such an interpretation $\mathcal{J}$ exists. It will turn out that $\mathcal{J}$ is a model of $A \sqsubseteq D \sqcap \forall RN{:}A$, but it is not a model of $A \sqsubseteq \neg C$, so that $\{A \sqsubseteq D \sqcap \forall RN{:}A\}$ does not entail $A \sqsubseteq \neg C$. The fact that $\mathcal{J}$ is not a model of $A \sqsubseteq \neg C$ is an immediate consequence of the assumption that $A^\mathcal{J} = (\forall RN^*{:}D)^\mathcal{I}$ together with (2.6) which states that $(\forall RN^*{:}D)^\mathcal{I} \not\subseteq \neg C^\mathcal{I}$. Since $\mathcal{J}$ is N-compatible with $\mathcal{I}$, the functions $.^\mathcal{I}$ and $.^\mathcal{J}$ map D to the same subset of $\Delta^\mathcal{I}$ and RN to the same binary relation over $\Delta^\mathcal{I}$. Together with the assumption that $A^\mathcal{J} = (\forall RN^*{:}D)^\mathcal{I}$ this proves that $\mathcal{J}$ is in fact a model of $A \sqsubseteq D \sqcap \forall RN{:}A$:

$$\begin{aligned}
& A^\mathcal{J} \\
= \; & (\forall RN^*{:}D)^\mathcal{I} \\
= \; & (D \;\sqcap\; \forall RN{:}\forall RN^*{:}D)^\mathcal{I} \\
= \; & D^\mathcal{I} \;\cap\; \{d \in \Delta^\mathcal{I} : RN^\mathcal{I}(d) \subseteq (\forall RN^*{:}D)^\mathcal{I}\} \\
= \; & D^\mathcal{J} \cap\; \{d \in \Delta^\mathcal{I} : RN^\mathcal{J}(d) \subseteq A^\mathcal{J}\} \\
= \; & (D \;\sqcap\; \forall RN{:}A)^\mathcal{J}.
\end{aligned}$$

To prove the only-if-part of (2.5), assume $\{A \sqsubseteq D \sqcap \forall RN{:}A\}$ does not entail $A \sqsubseteq \neg C$. That is, there is at least one interpretation, say $\mathcal{I} = \langle \Delta^\mathcal{I}, .^\mathcal{I} \rangle$, which is a model of $A \sqsubseteq D \sqcap \forall RN{:}A$, but which is no model of $A \sqsubseteq \neg C$. The latter means that $A^\mathcal{I} \not\subseteq \neg C^\mathcal{I}$, i.e., $A^\mathcal{I} \cap C^\mathcal{I} \neq \emptyset$, whereas the former means $A^\mathcal{I}$ must be a subset of $(D \sqcap \forall RN{:}A)^\mathcal{I}$. It is folklore that $(\forall RN{:}C_1)^\mathcal{I}$ is a subset of $(\forall RN{:}C_2)^\mathcal{I}$ if $C_1^\mathcal{I}$ is a subset of $C_2^\mathcal{I}$. Using this observation it is easy to see that $A^\mathcal{I}$ is a subset of $(D \sqcap \forall RN{:}D)^\mathcal{I}$:

$$\begin{aligned}
& A^\mathcal{I} \\
\subseteq \; & (D \sqcap \forall RN{:}A)^\mathcal{I} \\
= \; & D^\mathcal{I} \cap (\forall RN{:}A)^\mathcal{I} \\
\subseteq \; & D^\mathcal{I} \cap (\forall RN{:}(D \sqcap \forall RN{:}A))^\mathcal{I} \\
= \; & (D \sqcap \forall RN{:}D)^\mathcal{I} \cap (\forall RN{:}\forall RN{:}A)^\mathcal{I} \\
\subseteq \; & (D \sqcap \forall RN{:}D)^\mathcal{I}.
\end{aligned}$$

Induction on n proves that for any natural number n, $A^\mathcal{I}$ is a subset of $(D \sqcap (\forall RN^1{:}D) ... \sqcap (\forall RN^n{:}D))^\mathcal{I}$. As one might suspect, $\forall RN^n{:}D$ abbreviates $\forall RN{:}\forall RN^{n-1}{:}D$ if $n > 1$, whereas $\forall RN^1{:}D$ is $\forall RN{:}D$. This means, $A^\mathcal{I}$ is a subset of $(\forall RN^*{:}D)^\mathcal{I}$

and, therefore, $A^{\mathcal{I}} \cap C^{\mathcal{I}}$ is subset of $(\forall RN^*{:}D)^{\mathcal{I}} \cap C^{\mathcal{I}}$. As $A^{\mathcal{I}} \cap C^{\mathcal{I}}$ is assumed to be nonempty, $(\forall RN^*{:}D)^{\mathcal{I}} \cap C^{\mathcal{I}}$ must be nonempty too. But this is just to say that $C \sqcap \forall RN^*{:}D$ is coherent. □

Next, we give lower complexity bounds for the the least and the greatest fixed-point semantics.

Complexity Theorem 2. *The set of all $\langle \mu\mathcal{T}, CN \sqsubseteq C\rangle$ such that $\mathcal{T}$ ranges over all syntactically monotone terminologies of $\mathcal{ALC}$ and $CN \sqsubseteq C$ ranges over all concept introductions of $\mathcal{ALC}$ with $\mu\mathcal{T} \models CN \sqsubseteq C$ is hard for* EXPTIME. *There is moreover a constant $c > 0$ such that this set is* not *a member of* DTIME(2^{cn}). *Both results hold even if $\mathcal{T}$ may contain at most one concept introduction. The corresponding statements hold for greatest fixed-point terminologies as well.*

Proof. We have already seen that $\mu\{A \doteq C \sqcup \exists RN{:}A\}$ has exactly the same models as $A \doteq \exists RN^*{:}C$ and that $\nu\{A \doteq C \sqcap \forall RN{:}A\}$ has exactly the same models as $A \doteq \forall RN^*{:}C$. The only condition is that A does not occur in C. This means, if A occurs neither in C nor in D, then it holds that:

$$\begin{aligned} \models C \sqcap \forall RN^*{:}D \doteq \bot \quad &\textit{iff} \quad \nu\{A \doteq D \sqcap \forall RN{:}A\} \models C \sqcap A \doteq \bot \\ &\textit{iff} \quad \mu\{A \doteq \neg D \sqcup \exists RN{:}A\} \models C \sqcap \neg A \doteq \bot. \end{aligned}$$

To proceed, observe that $C \sqcap A \doteq \bot$ and the *primitive* concept introduction $A \sqsubseteq \neg C$ have exactly the same models. Similarly, if C is of the form $CN \sqcap C'$, $C \sqcap \neg A \doteq \bot$ and the *primitive* concept introduction $CN \sqsubseteq A \sqcup \neg C'$ have exactly the same models too. To summarize, if A occurs neither in C nor in D, and if C is of the form $CN \sqcap C'$, then it holds that:

$$\begin{aligned} \models C \sqcap \forall RN^*{:}D \doteq \bot \quad &\textit{iff} \quad \nu\{A \doteq D \sqcap \forall RN{:}A\} \models A \sqsubseteq \neg C \\ &\textit{iff} \quad \mu\{A \doteq \neg D \sqcup \exists RN{:}A\} \models CN \sqsubseteq A \sqcup \neg C'. \end{aligned}$$

This means, the sets whose lower complexity bounds we are about to prove are polynomial-time *lin*-reducible to the *complement* of FL and vice versa, at least when taking the restriction into consideration which is mentioned in Proposition 6. As FL is hard for EXPTIME, these sets are hard for EXPTIME as well. Furthermore, if they were a member of $\bigcap_{c>0}$ DTIME(2^{cn}), the complement of FL would be an element of $\bigcap_{c>0}$ DTIME(2^{cn}). As deterministic classes such as DTIME(2^{cn}) are closed under complementation, this would imply that not only the complement of FL, but also FL itself would be an element of $\bigcap_{c>0}$ DTIME(2^{cn}). This would contradict, however, Proposition 6. In fact, according to this proposition, there is a constant $c > 0$ such that FL is *not* a member of DTIME(2^{cn}). □

Upper Bounds

Streett and Emerson [1984, 1989] gave an elementary upper time bound for accepting the set of coherent concepts of full $\mathcal{ALC}\mu$. Vardi and Wolper [1984] show that the set of coherent concepts of $\mathcal{ALC}\mu^-$ is a member of EXPTIME. The next theorem generalizes Vardi and Wolper's result to subsumption with respect to finite sets of axioms of $\mathcal{ALC}\mu^-$.

Complexity Theorem 3. *The set of all* $\langle \mathcal{A}, C \doteq D \rangle$ *such that* $\mathcal{A} \cup \{C \doteq D\}$ *ranges over all finite sets of axioms of* $\mathcal{ALC}\mu^-$ *with* $\mathcal{A} \models C \doteq D$ *is a member of* EXPTIME.

Proof. As already mentioned, Vardi and Wolper [1984, Theorem 3] proved the claim for $\mathcal{A}$ being the empty set. In the same paper they also showed that each concept C of $\mathcal{ALC}\mu^-$ is coherent if and only if there is a tree interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}} \rangle$ such that the empty word λ is an element of $C^{\mathcal{I}}$ [Theorem 2]. According to Vardi and Wolper [1986, Page 197], a **tree interpretation** is an interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}} \rangle$ such that:

1. For some finite alphabet Σ, $\Delta^{\mathcal{I}}$ is a nonempty subset of all words over Σ.
2. For every $w \in \Sigma^*$ and for every $a \in \Sigma$, $wa \in \Delta^{\mathcal{I}}$ only if $w \in \Delta^{\mathcal{I}}$.
3. For each role name RN and all $w, w' \in \Sigma^*$, $\langle w, w' \rangle \in RN^{\mathcal{I}}$ only if for some $a \in \Sigma$, $w' = wa$ and, additionally, for each other role name RN', $\langle w, w' \rangle \notin RN'^{\mathcal{I}}$.

This ensures that any terminological axiom $C \doteq D$ can be internalized within $\mathcal{ALC}\mu^-$ using the technique introduced independently by Baader [1991] and by Schild [1991a]. This technique utilizes the concept $\forall_{\mathcal{R}}{:}C$ defined as follows if $\mathcal{R} = \{RN_i : 1 \leq i \leq n\}$ is a finite set of role names:

$$\forall_{\mathcal{R}}{:}C \stackrel{\text{def}}{=} \nu A.\{A \doteq C \sqcap \forall RN_1{:}A \ldots \sqcap \forall RN_n{:}A\}.$$

The concept name A must not occur in C. We have already seen that the concepts $\nu A.\{A \doteq C \sqcap \forall RN{:}A\}$ and $\forall RN^*{:}C$ are in fact equivalent. In analogy to this equivalence, $\forall_{\mathcal{R}}{:}C$ is equivalent to $\forall(RN_1 \ldots \sqcup RN_n)^*{:}C$, if $\mathcal{R}$ is as above. Suppose $\overline{C \doteq D}$ abbreviates the concept $(\neg C \sqcup D) \sqcap (\neg D \sqcup C)$. Then for each *tree* interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}} \rangle$, it holds that $\mathcal{I}$ is a model of $C \doteq D$ if and only if $\lambda \in (\forall_{\mathcal{R}}{:}\overline{C \doteq D})^{\mathcal{I}}$, provided $\mathcal{R}$ is the set of all role names appearing in C and D. In addition, it holds that the empty word λ is a element of $(\forall_{\mathcal{R}}{:}\overline{C \doteq D})^{\mathcal{I}}$ if and only if $(\forall_{\mathcal{R}}{:}\overline{C \doteq D})^{\mathcal{I}}$ is the full domain $\Delta^{\mathcal{I}}$, so that $C \doteq D \models C' \doteq D'$ if and only if $\models (\forall_{\mathcal{R}}{:}\overline{C \doteq D}) \sqsubseteq \overline{C' \doteq D'}$. This internalization is clearly computable in polynomial-time and it does not increase the size of the involved axioms more than linearly. Moreover, it preserves the restrictedness of the involved concepts. Induction on the cardinality of $\mathcal{A}$ shows

that the set of all tuples $\langle \mathcal{A}, C \doteq D \rangle$ such that $\mathcal{A} \cup \{C \doteq D\}$ ranges over all finite sets $\mathcal{A}$ of axioms of $\mathcal{ALC}\mu^-$ with $\mathcal{A} \models C \doteq D$ is polynomial-time m-reducible to the same set but with $\mathcal{A}$ restricted to be empty. As the latter is a member of EXPTIME, the former must be an element of EXPTIME too. □

It should be stressed that according to Representation Theorem 1, the last theorem actually gives an upper complexity bound for computing the subsumption relation which integrates all three kinds of semantics.

Corollary 2. *Consider the set of all* $\langle \mathcal{A} \cup \Gamma, C \doteq D \rangle$ *such that* Γ *ranges over all syntactically monotone complex fixed-point terminologies of* $\mathcal{ALC}$ *and* $\mathcal{A} \cup \{C \doteq D\}$ *ranges over all finite sets of axioms of* $\mathcal{ALC}\mu^-$ *with* $\mathcal{A} \cup \Gamma \models C \doteq D$. *This set is a member of* EXPTIME.

2.7 Discussion

Recursion is considered one of the most important means of definition, not only in description logics. In the setting of description logics recursion refers to terminologies containing concepts or roles the definition of which depend on each other. Such cyclic dependencies in terminologies are called *terminological cycles*. Of course, terminological cycles can be direct or indirect.

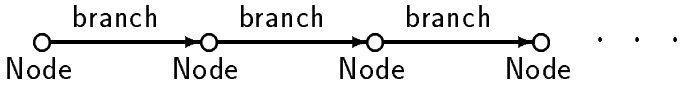
Many concepts are in fact most naturally defined by recursion. For instance, it is rather natural to define trees in the following recursive manner:[6]

$$\mathsf{Tree} \doteq \mathsf{Leaf} \sqcup (\mathsf{Node} \sqcap \forall \mathsf{branch}{:}\mathsf{Tree}).$$

One might expect the traditional semantics for concept introductions to work in the presence of such terminological cycles too. However, the problem with customary semantics is that in such cases it might give rise to ambiguities of the following type. Consider the recursive definition above. Suppose, the interpretation of Node, Leaf, as well as branch has already been fixed, but the extension of Tree is to be determined by the given recursive concept introduction. We would clearly expect something functioning as a definition to determine what it is supposed to define in a unique way. In this case, we would expect that the interpretations of Node, Leaf, and branch result in a unique interpretation of Tree. In general, this means that an interpretation of all primitive concepts and roles (i.e., all those concept and role names occurring in the concept introduction, except for the concept to be defined) should always determine a unique model. In contrast to acyclic concept introductions, those being

[6] Of course, for the definition's completeness, appropriate concept introductions for Leaf as well as Node must be given too. A Leaf can best be defined as a Node having no branch-successor, while in the context of a tree, a Node is any object having no more than one branch-predecessor.

recursive may violate this very characteristic of a definition. Such a situation, for instance, is encountered with the following infinite structure.



As far as this structure is concerned, the recursive concept introduction above does not uniquely determine an interpretation of Tree. In particular, there are two different ways of interpreting Tree, both in accordance with this concept introduction. One possibility is to interpret Tree as the set of all nodes, in which case it denotes the same set as Node. On the other hand, Tree can also be interpreted just as the empty set. This means that there are two rather different ways of satisfying this recursive definition, notwithstanding that all primitive parts of the definition have been fixed properly.

This is why the traditional notion of a model is no longer reasonable as soon as recursion enters the picture. In view of this problem, Nebel [1990a, 1991] refined the common idea of a model. Inspired by Lloyd's [1984] work on the semantic foundations of logic programming, Nebel put forward two alternative semantics. In analogy to Lloyd's terminology, he baptized these alternatives *least* and *greatest fixed-point semantics*. In a nutshell, fixed-point semantics does not consider all models as admissible, but only those which are the least or the greatest with respect to the interpretation of the concept to be defined. Of course, in order to give this minimization and maximization process reasonable limits, the terms *least* and *greatest* refer only to those models which have the same interpretation domain and agree in the interpretation of all primitive concepts and roles. These two different fixed-point semantics can be thought of as giving rise to *inductive* or *co-inductive* definitions. This is to say, the two alternative semantics capture different definitions along the line of the dichotomy "the least set such that..." versus "the greatest set such that...."

Fixed-point semantics turned out to be more suited for recursion than customary semantics. However, it was disputed for a while which of the two different fixed-point semantics should be taken, where the most substantial contribution to this debate is due to Baader [1990a]. He argued that careful inspection of recursive definitions reveals that recursion is often used to express the reflexive-transitive closure of a role. Consequently, he then argued, any semantics of recursion should take into account this fact. The question then arises what kind of fixed-point semantics captures this reading of recursion. Baader was able to prove that, as far as concept introductions fitting the pattern $CN \doteq C \sqcap \forall R{:}CN$ (without any occurrence of CN in C) are concerned, only the greatest fixed-point semantics supports this reading. Only the greatest fixed-point semantics aptures in this case exactly the meaning of $CN \doteq \forall R^*{:}C$. One can add that the recursive concept introduction for Tree fits into this scheme, too, at least when taking into account an appropriate concept introduction

for Leaf. For in this case Leaf $\sqcup$ (Node $\sqcap$ $\forall$branch:Tree) is actually equivalent to Node $\sqcap$ $\forall$branch:Tree.[7]

Taking for granted that recursion is commonly used to express regular role expressions such as the reflexive-transitive closure, Baader finally concluded that the greatest fixed-point semantics comes off best. As regards the very weakest description logic, this is, in fact, the only semantics supporting this reading of recursion.

Baader's result, however, is somewhat misleading in that its validity is strictly limited to the very weakest description logic. As a matter of fact, his result is just a consequence of the fact that he considered only concept conjunction as well as universal role quantification, but left out the dual concept-structuring primitives. We pointed out that for the dual concept-structuring primitives the situation is just the opposite. In particular, in order to express regular role expressions occurring within *existential* role quantification, we have to resort to *least* fixed-point semantics rather than to greatest fixed points. Take, for instance, the recursive concept introduction $CN \doteq C \sqcup \exists R{:}CN$. If this concept introduction is to capture the meaning of $CN \doteq \exists R^*{:}C$, the only way to enforce this reading is by least fixed-point semantics, which shows that both kinds of fixed-point semantics are needed.

The reader might object that in stronger languages such as $\mathcal{ALC}$, one could employ the duality between $\exists R{:}C$ and $\neg\forall R{:}\neg C$ anyway. Instead of expressing, say, $\exists R^*{:}C$, one could first capture $\forall R^*{:}\neg C$ with the help of the of $CN \doteq \neg C \sqcap \forall R{:}CN$. Of course, in order to achieve this, greatest fixed-point semantics must be invoked. If one would simply add the concept introduction $\overline{CN} \doteq \neg CN$, then $\overline{CN}$ should finally capture $\exists R^*{:}C$. Such a representation, however, does not work. This is because terminologies of the form $\{CN \doteq \neg C \sqcap \forall R{:}CN,\ \overline{CN} \doteq \neg CN\}$ do not always have a greatest fixed-point model, nor do they always have a least fixed-point model. To see this, observe that it is impossible to maximize (or to minimize) the interpretation of CN and its complement at the same time. This proves not only that both kinds of fixed-point semantics are needed, but that they are even needed in *coexistence*.

This is, however, not the only argument in support of the coexistence of the two different fixed-point semantics. Another argument attacks the claim that recursion is mostly used to express regular role expressions. This claim cannot be put into effect without a thorough investigation on what the least and greatest fixed-point semantics really express. Such an investigation was carried out in the present chapter. This was done with the help of a previously unknown, but nevertheless extremely useful concept-structuring primitive. In particular, we introduced new concepts of

[7] In fact, if the concept introduction Leaf $\doteq$ Node$\sqcap\neg\exists$branch is put into effect, then Leaf$\sqcup$(Node$\sqcap$ $\forall$branch:Tree) is obviously equivalent to (Node $\sqcap$ $\neg\exists$branch) $\sqcup$ (Node $\sqcap$ $\forall$branch:Tree). It is not hard to verify that the latter concept is in turn equivalent to the following concept:

(Node $\sqcap$ $\neg\exists$branch $\sqcap$ $\forall$branch:Tree) $\sqcup$ (Node $\sqcap$ $\exists$branch $\sqcap$ $\forall$branch:Tree).

The latter concept is finally equivalent to Node $\sqcap$ $\forall$branch:Tree.

the form $\exists R^{\omega}$, where R can be an arbitrary role. Such a concept denotes all those objects, d, such that there is at least one infinite R-chain emanating from d. The significance of a concept-structuring primitive capable of stipulating or forbidding (when negated) such infinite role chains should be evident. The addition of this new concept-structuring primitive to the regular extension of $\mathcal{ALC}$ results in a description logic which we refer to as the *ω-regular extension of $\mathcal{ALC}$*. With the help of this new description logic, the consequences of enforcing the different kinds of fixed-point semantics can be exactly stated. The analysis, however, is restricted to standard recursion following the patterns $CN \doteq C \sqcap \forall R{:}CN$ and $CN \doteq C \sqcup \exists R{:}CN$. The following table sums up the analysis carried out in the present chapter:

	least fixed-point semantics	*greatest fixed-point semantics*
$CN \doteq C \sqcap \forall R{:}CN$	$CN \doteq \forall R^{*}{:}C \sqcap \neg\exists R^{\omega}$	$CN \doteq \forall R^{*}{:}C$
$CN \doteq C \sqcup \exists R{:}CN$	$CN \doteq \exists R^{*}{:}C$	$CN \doteq \exists R^{*}{:}C \sqcup \exists R^{\omega}$

The table shows that the situation in least and greatest-fixed point semantics is completely symmetric. What this table also shows is that the question which of the two different fixed-point semantics should be preferred depends on what one intends to express in that particular case. For example, in the case of the concept introduction of a tree given above, it is just a question of whether or not trees of infinite depth are excluded. According to the table just given, the least fixed-point semantics does exclude them, whereas the greatest fixed-point semantics does not. Of course, this raises the question whether such an analysis is limited to recursion fitting into the standard patterns included in the table above.

We pointed out that questions like the latter can be tackled perfectly well in terms of the explicit fixed-point operators known from program logics, and so can recursion in $\mathcal{ALC}$ as a whole. This is particularly true in view of the need for coexistence of both kinds of fixed-point semantics. Explicit least and greatest fixed-point operators are used in logics of programs to state specific correctness properties not expressible by ordinary dynamic logics. They have been employed successfully to state deadlock freedom and starvation, see [Flon and Suzuki, 1978]. In the context of first-order logic, explicit fixed-point operators were investigated by Park [1970], Hitchcock and Park [1973], as well as de Bakker and de Roever [1973]. Regarding description logics, the propositional case is more interesting though. The propositional case was first investigated by Pratt [1981] and, more influentially, by Kozen [1983]. In both cases, least and greatest fixed-point operators are treated just as a new kind of formulae. This treatment makes it possible to express nested fixed-points operators easily. In the propositional case, the following notation is common. If x is an arbitrary propositional variable and α is an arbitrary formula, then $\mu x.\alpha$ is a least fixed-point operator, while $\nu x.\alpha$ is a greatest fixed-point operator. Fixed-point formulae of the form $\mu x.\alpha$ and $\nu x.\alpha$ are to be read as "the least x such that α" and "the greatest x

such that α."

From a semantic point of view, fixed-point formulae represent the least and the greatest fixed point of a certain function, hence the name *fixed-point* operator. This function can be best described with the following notation. If $\mathcal{M}$ is an arbitrary Kripke structure and α is a formula, then let $\alpha^{\mathcal{M}}$ denote the set of all those states, w, such that $\mathcal{M} \models_w \alpha$. This means that $\alpha^{\mathcal{M}}$ denotes exactly those states in which the proposition α holds. The function we have in mind then maps every set, S, of states to $\alpha^{\mathcal{M}_{x/S}}$. Here $\mathcal{M}_{x/S}$ is supposed to denote the Kripke structure which agrees with $\mathcal{M}$ except for the fact that $x^{\mathcal{M}}$ is S. This function is, of course, interesting only if α involves at least one occurrence of x; otherwise it would denote a constant function that always yields $\alpha^{\mathcal{M}}$.

Of course, the meaning of fixed-point operators is then given in terms of the least and greatest fixed points of the function just introduced. This would not be possible if the uniqueness of these least and greatest fixed points was not guaranteed. This is usually ensured by imposing a simple restriction on the possible syntactic shape of fixed-point formulae, referred to as *formal monotonicity*. This restriction requires every occurrence of x in $\mu x.\alpha$ and $\nu x.\alpha$ to be *positive*. This is to say, every occurrence of x must lie under an even number of negations. The function described above is then guaranteed to be monotonic. According to the well-known Knaster-Tarski Theorem, monotonicity of a function in turn ensures the existence and the uniqueness of its least as well as its greatest fixed point [Tarski, 1955].

It was Kozen [1983] who enriched the Hennessy-Milner Logic with exactly this kind of fixed-point formulae. The resulting logic is called *propositional μ-calculus*. Kozen already observed that this logic is at least as strong in expressive power as the propositional dynamic logic. The PDL formula $\langle \textbf{while } \alpha \textbf{ do } a \rangle \beta$, for instance, can be expressed in a recursive fashion with the help of the least fixed-point formula $\mu x.\Big((\neg\alpha \wedge \beta) \vee (\alpha \wedge \langle a \rangle x)\Big)$. Kozen additionally noted that there are even formulae of the propositional μ-calculus not expressible in propositional dynamic logic.

Having in mind that $\mathcal{ALC}$ is a notational variant of the Hennessy-Milner logic, we should expect that recursion can be captured in $\mathcal{ALC}$ in terms of explicit fixed-point operators because so can recursion in the Hennessy-Milner logic. In the context of $\mathcal{ALC}$, fixed-point operators are to be treated as concept-structuring primitives. In analogy to the μ-calculus, we chose the following syntax for the new concept-structuring primitives to be introduced: If C is an admissible concept, then so are $\mu X.C$ and $\nu X.C$. The variable X is treated as a special kind of a concept name, referred to as *concept variable*. Of course, formal monotonicity has to be imposed on C in this case too. But then an arbitrary recursive concept introduction, $CN \doteq C$, can directly be recast by one of the following two acyclic concept introductions:

$$\begin{aligned} CN &\doteq \mu X.C_{CN/X}, \\ CN &\doteq \nu X.C_{CN/X}. \end{aligned}$$

The only precondition that $CN \doteq C$ has to meet is that of formal monotonicity. This is to say, all occurrences of CN in C must be positive. In any case, the expression $CN_{CN/X}$ is supposed to denote the concept obtained from C by simultaneously replacing each occurrence of CN with X throughout C. Observe that this kind of representation enables us to capture recursion with the help of acyclic concept introductions rather than cyclic ones. Of course, the choice between the two alternatives given above depends on whether least or greatest fixed-point semantics is preferred.

If there is an indirect recursion leading through more than one concept introduction, we have to resort either to nested fixed-point operators or to fixed-point operators dealing with mutual recursion. Fixed-point operators of the latter type were investigated by Vardi and Wolper [1984] in the framework of the Hennessy-Milner logic. In the present chapter, we enriched $\mathcal{ALC}$ with explicit fixed-point operators in the style of Vardi and Wolper. We thereby obtained a new kind of description logic which is actually a notational variant of Vardi and Wolper's version of the propositional μ-calculus. We chose the name $\mathcal{ALC}\mu$ for the extended standard description logic $\mathcal{ALC}$.

Thanks to the one-to-one correspondence with the propositional μ-calculus, we can take advantage of a number of results established for the μ-calculus. Most importantly, this includes several complete decision procedures. The first decision procedure for full propositional μ-calculus is due to Kozen and Parikh [1983]. However, the upper time bound thus obtained was non-elementary. The first decision elementary procedure was given by Street and Emerson [1984, 1989]. The employed algorithm, however, still has a triply exponential worst-case time complexity. Exponential-time procedures were devised by Vardi and Wolper [1984], Emerson and Jutla [1988], as well as Safra [1988]. Vardi and Wolper's algorithm is capable of dealing with mutual recursion, but it works only with the possible occurrences of nested alternating fixed-point operators restricted. This restriction, however, does not affect those nested fixed-point operators which are of practical use. In contrast to this, the Emerson and Jutla's as well as Safra's algorithms capture the propositional μ-calculus to its full extent, but without mutual recursion. Axiomatics of full propositional μ-calculus has been a longstanding open problem. This problem was solved recently by Walukiewicz [1993].

Other interesting results concern the expressive power of the μ-calculus. One result states that the propositional μ-calculus is strictly stronger than PDL, even when there are no nested fixed-point operators available [Kozen, 1983]. One interpretation of this result is that regular role expressions are not an alternative for recursion in $\mathcal{ALC}$ because they are strictly weaker in expressive power. Another result states that even when PDL is augmented by so-called *repeat* formulae, full propositional μ-calculus is still strictly stronger in expressive power than this extension of PDL. This result is due to Niwinsky and was reported in [Streett, 1985], page 363. The interesting point about Niwinsky's result is that repeat formulae correspond in a one-

to-one fashion to concepts of the form $\exists R^{\omega}$ which we have met before. But then the question whether the exact meaning of recursion in $\mathcal{ALC}$ can always be characterized in terms of the ω-regular role expressions must be answered in the negative.

Chapter 3

The Universal Description Logic $\mathcal{U}\mu$ Incorporating Recursion

So far, we have seen that explicit fixed-point operators provide for a general and—in terms of computability—also feasible framework for recursion. The idea to use second-order operators was borrowed from the propositional μ-calculus, and so were many of the results presented on the computational complexity and expressive power. However, the correspondence with the propositional μ-calculus does not extend beyond the few concept-structuring primitives of $\mathcal{ALC}$. But $\mathcal{ALC}$ was advocated by its inventors because of its "pleasant mathematical properties" [Schmidt-Schauß and Smolka, 1991, page 3] rather than because of its practical significance for knowledge representation purposes. The question then arises whether fixed-point operators work equally well in the context of richer languages. It will turn out that the syntactic and semantic foundations of fixed-point operators developed in the previous chapter can be extended to deal with a description logic which can be called *universal* because it encompasses all existing concept and role-structuring primitives. The major complication will be the generalization of the notion of formal monotonicity to the additional concept and role-structuring primitives of $\mathcal{U}$. In addition, we shall extend fixed-point operators to be applicable not only to concepts, but also to roles such that roles can be defined by recursion as well. The resulting description logic, called $\mathcal{U}\mu$, is expressive enough to define standard data structures such as lists, directed acyclic graphs, trees, binary trees, balanced binary trees, and those AND-OR-graphs for which there is at least one well-founded solution. Such great expressive power, however, can only be gained in return for losing computability in principle. This issue will be tackled later on in Chapter 4.

3.1 The Universal Description Logic $\mathcal{U}$

Patel-Schneider [1987] presented a general framework for terminological knowledge representation systems in his PhD Thesis. In particular, one of his goals was to define the syntax and semantics of a description logic which encompasses all known extensions. As a result he came up with a general-purpose, universal description logic, which he called $\mathcal{U}$ [Patel-Schneider, 1987, Chapter 5.1]. Of course, $\mathcal{U}$ comprises all those concept-structuring primitives which are provided by the standard concept language $\mathcal{ALC}$. That is to say, in addition to Boolean operations on concepts, $\mathcal{U}$ contains both existential and universal role quantification of the form $\forall R{:}C$ and $\exists R{:}C$. In contrast to $\mathcal{ALC}$, however, $\mathcal{U}$ also allows for *number restrictions*. Number restrictions are indispensable to most modeling tasks, and are therefore supported by nearly all terminological knowledge representation systems in praxis.

There are two kinds of number restrictions, qualified and unqualified ones. *Qualified number restrictions* are either of the form $\exists^{\geq n} R{:}C$ or $\exists^{\leq m} R{:}C$, where both $m \geq 0$ and $n \geq 1$ can be arbitrary natural numbers. These concepts represent all those objects, d, such that the possible number of instances of C related to d by R is restricted by norm. *Unqualified number restrictions* can be viewed as a special kind of qualified ones in which C is always the universal concept. In this case we write $\exists^{\geq n} R$ and $\exists^{\leq m} R$ in lieu of $\exists^{\geq n} R{:}\top$ and $\exists^{\leq m} R{:}\top$. Another important concept-structuring primitive of $\mathcal{U}$ is called *role-value map* and has been taken from KL-ONE's repository [Brachman and Schmolze, 1985, Section 9.1]. This concept-structuring primitive relates the fillers of two roles to each other. If R and S are two arbitrary roles, then $R \leq S$ constitutes such a role-value map. It represents all those objects, d, such that every object related to d by R is also related to d by S. Role-value maps permit us to express such a concept as $\mathsf{Person} \sqcap (\mathsf{has\_staff\text{-}member} \leq \mathsf{has\_friend})$. This concept represents all those persons whose staff members are all their friends. There is another concept-structuring primitive of $\mathcal{U}$, called *structural description*, also borrowed from KL-ONE (cf. [Brachman and Schmolze, 1985], Section 9.2). It is certainly the least obvious construct of $\mathcal{U}$. Structural descriptions are a way to describe concepts in a rather abstract manner by interrelating its role fillers with role fillers of some other concept. The following concept, for instance, involves such a structural description:

$$\mathsf{Person} \quad \sqcap \quad \exists \mathsf{Person}{:}\Big((\mathsf{has\_friend} \oslash \mathsf{has\_foe}) \sqcap (\mathsf{has\_friend} \oslash\!\!> \mathsf{has\_foe})\Big).$$

This concept is to represent those persons for whom there exists a second person such that all friends of the former are foes of the latter and, vice versa, all foes of the latter are friends of the former. In the most general case, structural descriptions are of the form $\exists C{:}(RB_1 \sqcap \ldots \sqcap RB_n)$ with $n \geq 1$ such that C is an arbitrary concept and the RB_i's are so-called *role bindings*. Such a role binding is a role formed out of two other roles. If R and S are two arbitrary roles, then both $R \oslash S$ and $R \oslash\!\!> S$ are role bindings. As the example given above suggests, the concept $\exists C{:}(RB_1 \sqcap \ldots \sqcap RB_n)$ represents all those objects, d, for which there is at least one instance of C, say, e, such

that the ordered pair $\langle d, e\rangle$ is an instance of each of the role bindings RB_1,...,RB_n. A role binding of the form $R \circledleft S$ represents all those ordered pairs of objects such that every object related to the first object by R is related to the second by S. In contrast to this, $R \circledright S$ represents all those ordered pairs of objects such that every object related to the second object by S is related to the first by R. Role-value maps can be viewed as structural descriptions of some special kind. It is instructive to verify that the role-value map $R \leq S$ is equivalent to the structural description $\exists\top{:}((R \circledleft S) \sqcap (\epsilon \circledleft \epsilon))$. The main stumbling block we have to overcome is the fact that the role binding $\epsilon \circledleft \epsilon$ is equivalent to the identity role ϵ itself.

These powerful concept-structuring primitives of $\mathcal{U}$ give rise to rather involved subsumption and equivalence relations. A nontrivial inference of this kind is, for instance, the fact that the concept $\mathsf{Person} \sqcap (\mathsf{has\_staff\text{-}member} \leq \mathsf{has\_friend})$ is equivalent to the following structural description:

$$\exists \mathsf{Person}\text{:}\ \Big((\mathsf{has\_friend} \circledleft \mathsf{has\_friend}) \sqcap (\mathsf{has\_friend} \circledright \mathsf{has\_friend}) \sqcap (\mathsf{has\_staff\text{-}member} \circledleft \mathsf{has\_friend})\Big).$$

The first two role bindings state that for each person which is an instance of the structural description above there exists a second person having exactly the same friends. The third role binding requires in addition that all staff members of the first are all friends of the second. Because the second person has exactly the same friends as the first, the structural description above denotes just all those persons whose staff members are all their friends. But this is exactly the class of persons represented by $\mathsf{Person} \sqcap (\mathsf{has\_staff\text{-}member} \leq \mathsf{has\_friend})$.

What is perhaps most important is the fact that $\mathcal{U}$ encompasses a great variety of rather powerful role-structuring primitives, these include such constructs as the identity role, ϵ, Boolean operations on roles, the inverse, R^{-1}, of a role, the composition, $R \circ S$, of two roles, as well as the transitive closure, R^{+}, and the reflexive-transitive closure, R^{*}, of a role. In addition, $\mathcal{U}$ allows a role to be formed out of two concepts, say, C and D. The resulting role is supposed to represent all those ordered pairs such that the first component is an instance of C and the second is an instance of D. Such a role is denoted by $C \times D$ and is referred to as a *product*. Apart from some minor changes in syntax, the definition given so far agrees with that given in Chapter 5.1 of [Patel-Schneider, 1987].

In one respect, however, we shall deviate from $\mathcal{U}$'s original definition. This concerns the inclusion of individual concepts, a special kind of concepts composed of single individual names, which were mentioned already in the introduction. If Bill, for instance, is such an individual name, then $\{\mathsf{Bill}\}$ forms an admissible concept of $\mathcal{U}$, called an individual concept. Concepts of this kind denote singleton sets containing the very object associated with the individual name that the individual concept is composed of. With such individual concepts at our disposal, classes such as the

friends of Bill can be captured: The concept $\exists \mathsf{has\_friend}^{-1}{:}\{\mathsf{Bill}\}$ does represent exactly this class. The original definition of $\mathcal{U}$ does not include individual concepts, although they are present in the definition of KL-ONE [Brachman and Schmolze, 1985, Section 8].

For the following formal definition of the syntax of $\mathcal{U}$, we assume $\boldsymbol{\mathcal{N}_C}$, $\boldsymbol{\mathcal{N}_R}$, and $\boldsymbol{\mathcal{N}_I}$ to be countably infinite,[1] pairwise disjoint sets which contain neither $\top$ nor ϵ. The elements of these sets are called **concept names**, **role names**, and **individual names** respectively. Moreover, $\boldsymbol{\mathcal{N}}$ is assumed to be the union of $\mathcal{N}_C$, $\mathcal{N}_R$, and $\mathcal{N}_I$, that is to say, it denotes the set of all concept names, role names, and individual names. The elements of this set are called **term names**.

Definition 22. The **concepts and roles of $\mathcal{U}$** are simultaneously defined as:

1. All concept names, individual concepts, as well as the universal concept $\top$ are concepts of $\mathcal{U}$.
2. All role names as well as the identity role $\boldsymbol{\epsilon}$ are roles of $\mathcal{U}$.
3. Suppose C and D are concepts of $\mathcal{U}$, R and S are roles of $\mathcal{U}$, and $n \geq 1$ and $m \geq 0$ are any natural numbers. Then $\boldsymbol{C \sqcap D}$, $\boldsymbol{C \sqcup D}$, $\boldsymbol{\neg C}$, $\boldsymbol{\exists^{\geq n} R{:}C}$, and $\boldsymbol{\exists^{\leq m} R{:}C}$ are concepts of $\mathcal{U}$, while $\boldsymbol{R \sqcap S}$, $\boldsymbol{R \sqcup S}$, $\boldsymbol{\neg R}$, $\boldsymbol{R \circ S}$, $\boldsymbol{R^{+}}$, $\boldsymbol{R^{-1}}$, and $\boldsymbol{C \times D}$ are roles of $\mathcal{U}$.
4. These are the concepts and roles of $\mathcal{U}$.

The concepts and roles of $\mathcal{U}$ are divided into atomic and compound ones, as with the concepts and roles of $\mathcal{ALC}$. All concept and role names, all individual concepts, the universal concept, as well as the indentity role are said to be **atomic**. All other concepts and roles of $\mathcal{U}$ are **compound**.

For convenience sake, we introduce the remaining concept and role-structuring primitives as abbreviations in terms of those explicitly included in $\mathcal{U}$'s formal definition:

$$
\begin{aligned}
\boldsymbol{\exists R{:}C} &\stackrel{\text{def}}{=} \exists^{\geq 1} R{:}C,\\
\boldsymbol{\forall R{:}C} &\stackrel{\text{def}}{=} \exists^{\leq 0} R{:}\neg C,\\
\boldsymbol{\exists^{\geq n} R} &\stackrel{\text{def}}{=} \exists^{\geq n} R{:}\top,\\
\boldsymbol{\exists^{\leq m} R} &\stackrel{\text{def}}{=} \exists^{\leq m} R{:}\top,\\
\boldsymbol{R|C} &\stackrel{\text{def}}{=} R \sqcap (\top \times C),\\
\boldsymbol{R \leq S} &\stackrel{\text{def}}{=} \exists^{\leq 0} (R \sqcap \neg S){:}\top,
\end{aligned}
$$

[1] All these sets should be acceptable in deterministic polynomial time, for instance, by means of some finite state automaton.

$$
\begin{aligned}
\boldsymbol{R^*} &\stackrel{\text{def}}{=} R^+ \sqcup \epsilon, \\
\boldsymbol{\exists C{:}(RB_1 \sqcap \ldots \sqcap RB_n)} &\stackrel{\text{def}}{=} \exists^{\geq 1}(RB_1 \sqcap \ldots \sqcap RB_n){:}C, \\
\boldsymbol{R \oslash S} &\stackrel{\text{def}}{=} \neg(R \circ \neg S^{-1}), \\
\boldsymbol{R \ominus S} &\stackrel{\text{def}}{=} \neg(\neg R \circ S^{-1}).
\end{aligned}
$$

The long form of role-value maps is exactly the same as that mentioned in [Patel-Schneider, 1987], page 72. All other long forms should be straightforward, with the exception of role bindings, whose long forms are taken from [Schmidt, 1991], page 74.

Of course, the definition of an interpretation and in particular of an interpretation function has to be generalized in order to cope with the additional concept and role-structuring primitives of $\mathcal{U}$. The other syntactic and semantic notions remain unchanged, with the notable exception of formal monotonicity that will be treated in Section 3.4.

Definition 23. Assume $\mathcal{L}$ is a set of concepts and roles, whereas Δ is an arbitrary set. An **$\mathcal{L}$-valuation** over Δ is a function, $\mathcal{V}$, which maps every concept of $\mathcal{L}$ to a subset of Δ and each role of $\mathcal{L}$ to a binary relation over Δ. In addition, $\mathcal{V}$ maps each individual concept of $\mathcal{L}$ to a singleton set containing exactly one element of Δ.

Definition 24. An **interpretation**, $\mathcal{I}$, is a triple $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$, where $\Delta^{\mathcal{I}}$ is an arbitrary set, $\mathcal{V}$ is a $\mathcal{N}$-valuation over $\Delta^{\mathcal{I}}$, and $.^{\mathcal{I}}$ is a function extending $\mathcal{V}$ to deal with arbitrary concepts and roles of $\mathcal{U}$. The set $\Delta^{\mathcal{I}}$ is called the **domain** of $\mathcal{I}$, and $.^{\mathcal{I}}$ is said to be the **interpretation function** of $\mathcal{I}$. The latter is inductively defined as follows. First, $\top^{\mathcal{I}}$ is $\Delta^{\mathcal{I}}$ and $\epsilon^{\mathcal{I}}$ is $\{\langle d, d \rangle : d \in \Delta^{\mathcal{I}}\}$. Then $TN^{\mathcal{I}}$ is $\mathcal{V}(TN)$ whenever TN is an atomic concept or role other than $\top$ or ϵ. For the induction step, suppose that $.^{\mathcal{I}}$ applied to the concepts C and D as well as the roles R and S has already been defined. Then $.^{\mathcal{I}}$ applied to concepts and roles of $\mathcal{U}$ composed of C, D, R, and S is defined as follows:

$$
\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}, \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}}, \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \\
(\exists^{\geq n} R{:}C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : \sharp |R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}| \geq n\}, \\
(\exists^{\leq m} R{:}C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} : \sharp |R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}| \leq m\}, \\
(R \sqcap S)^{\mathcal{I}} &= R^{\mathcal{I}} \cap S^{\mathcal{I}}, \\
(R \sqcup S)^{\mathcal{I}} &= R^{\mathcal{I}} \cup S^{\mathcal{I}}, \\
(\neg R)^{\mathcal{I}} &= (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}, \\
(R \circ S)^{\mathcal{I}} &= R^{\mathcal{I}} \circ S^{\mathcal{I}}, \\
(R^+)^{\mathcal{I}} &= \bigcup_{i \geq 1} (R^{\mathcal{I}})^i, \\
(C \times D)^{\mathcal{I}} &= C^{\mathcal{I}} \times D^{\mathcal{I}}.
\end{aligned}
$$

As far as the formal syntax and semantics is concerned, this completes the description of the universal description logic $\mathcal{U}$. For a better understanding of this language, however, it may be better to flesh out some of its most important concept and role-structuring primitives. In order to do so, we shall take an example drawn from the domain of directed graphs. In particular, we shall single out directed acyclic graphs (DAGs for short), trees, binary trees, and lists by means of a terminology of $\mathcal{U}$.

Example 1 (DAGs, trees, binary trees, and lists). In the following discussion, we shall confine ourselves to those directed graphs that are *connected*. We presuppose that there is an edge relation, say, E, which all graphs have in common. This makes it possible to represent a graph by its vertices only. More precisely, an object, d, represents the directed graph $\langle V_d, E_d \rangle$ generated by this object and E. This graph is given as follows. V_d is the set of all those vertices accessible by d, while E_d is the binary relation obtained from E by restricting both its domain and its range to V_d, i.e., E_d is $E \cap (V_d \times V_d)$. Of course, *accessible* is to be understood in this connection in the sense of the reflexive, transitive, and symmetric closure of the edge relation, E. We represent the edge relation by the role name edge, so that the compound role $(\mathsf{edge} \sqcup \mathsf{edge}^{-1})^*$ defines exactly this accessibility relation. A vertex which is a member of an edge-cycle can then be represented in $\mathcal{U}$ in a very concise manner by requiring that it must be related to itself by the transitive closure of edge. The unqualified number restriction $\exists^{\geq 1}(\mathsf{edge}^+ \sqcap \epsilon)$ represents this very situation. On the other hand, a vertex can be precluded from an edge-cycle just by requiring that all vertices related to it by the transitive closure of edge have to be different from the vertex itself, i.e., they must not be related to the vertex itself by the identity role ϵ. This condition is captured by the role-value map $\mathsf{edge}^+ \leq \neg\epsilon$. Hence, the directed graph represented by an arbitrary object, say, d, is *acyclic* if and only if all those vertices which are accessible by d are instances of the concept $\mathsf{edge}^+ \leq \neg\epsilon$. The corresponding concept introduction is depicted in Figure 3.1. Trees are then easily defined in terms of DAGs. The only characteristic distinguishing trees from DAGs is that in contrast to the vertices of a DAG, no vertex of a tree is allowed to have more than one edge leading to it. The unqualified number restriction $\exists^{\leq 1}\mathsf{edge}^{-1}$ obviously corresponds to this very characteristic of trees. If trees are assumed to be represented by their root rather than by each of their vertices, the accessibility relation of trees can be captured by the role edge^* rather than by $(\mathsf{edge} \sqcup \mathsf{edge}^{-1})^*$. Therefore, a tree can simply be defined as $\mathsf{DAG} \sqcap \forall \mathsf{edge}^* . \exists^{\leq 1}\mathsf{edge}^{-1}$. The corresponding concept introduction can be found in Figure 3.1, and so can the definitions of binary trees and lists. It is important to note, however, that the definitions given there do not exclude *infinite acyclic* structures from being either lists or trees. In order to do so, we have to use fixed-point operators.

$$
\begin{array}{rcl}
\mathsf{edge\text{-}Cycle} & \doteq & \exists^{\geq 1}(\mathsf{edge}^{+} \sqcap \epsilon) \\
\mathsf{No\_edge\text{-}Cycle} & \doteq & \mathsf{edge}^{+} \leq \neg\epsilon \\
\mathsf{DAG} & \doteq & \forall \mathsf{accessible}{:}\mathsf{No\_edge\text{-}Cycle} \\
\mathsf{accessible} & \doteq & (\mathsf{edge} \sqcup \mathsf{edge}^{-1})^{*} \\
\mathsf{Tree} & \doteq & \mathsf{DAG} \sqcap \forall \mathsf{edge}^{*}{:}\exists^{\leq 1}\mathsf{edge}^{-1} \\
\mathsf{BinaryTree} & \doteq & \mathsf{Tree} \sqcap \forall \mathsf{edge}^{*}{:}\exists^{\leq 2}\mathsf{edge} \\
\mathsf{List} & \doteq & \forall \mathsf{tail}^{*}{:}\Big((\mathsf{Nil} \sqcup \mathsf{NonemptyList}) \sqcap (\mathsf{tail}^{+} \leq \neg\epsilon)\Big) \\
\mathsf{Nil} & \doteq & \exists^{\leq 0}(\mathsf{head} \sqcup \mathsf{tail}) \\
\mathsf{NonemptyList} & \doteq & \exists^{\leq 1}\mathsf{head} \sqcap \exists^{\leq 1}\mathsf{tail} \sqcap \exists^{\geq 2}(\mathsf{head} \sqcup \mathsf{tail})
\end{array}
$$

Figure 3.1: A terminology defining DAGs, trees, binary trees, and lists

3.2 The Expressive Power of $\mathcal{U}$

We shall provide a *strict* lower bound of $\mathcal{U}$'s expressive power in terms of the well-known *calculus of binary relations* [Tarski, 1941]. This result is of importance in at least two respects. First, the calculus of relations has been studied thoroughly as a query language for relational databases, at least when generalized to relations of arbitrary rank [Codd, 1971]. This will be of particular significance for Chapter 4. The main subject of that chapter will be the investigation of a fixed-point extension of $\mathcal{U}$ as a query language for a special kind of knowledge bases which can be viewed as relational databases. Second, the expressive power of the calculus of binary relations can in turn be characterized precisely by means of a certain fragment of first-order logic, called the *elementary theory of binary relations*. The only restriction imposed on formulae of this theory is that they do not involve any nonlogical symbols other than binary predicate symbols. The calculus of binary relations comprises six fundamental operations on binary relations, but no variables and no quantifiers. This calculus is, however, expressively equivalent to the three-variable fragment of the elementary theory of binary relations, usually referred to as L_3 [Tarski and Givant, 1987, Chapter 3.10]. In combination with the strict lower expressiveness bound mentioned at the very beginning of this section, this implies that $\mathcal{U}$ is *strictly* stronger in expressive power than L_3.

We begin the discussion by defining the syntax and semantics of the calculus of relations. The foundations of this calculus were laid by De Morgan, Peirce, and Schröder during the second half of the nineteenth century. A brief overview of the history of the calculus of relations including comprehensive references is given, for instance, in the preface of [Tarski and Givant, 1987], pages XV-XVII.

Definition 25. The **calculus of binary relations** (also called **calculus of relations** or simply $\mathfrak{R}$ for short) consists of two different types of formulae, called *relation designations* and *sentences*. The set of **relation designations of $\mathfrak{R}$** is defined as follows: First, the **relation variables** R_1,R_2,..., the **universal relation** 1, as well as the **identity relation** $\mathring{1}$ are relation designations of $\mathfrak{R}$. Moreover, if R and S are both relation designations of $\mathfrak{R}$, then so are $\overline{\boldsymbol{R}}$, $\breve{\boldsymbol{R}}$, $\boldsymbol{R} \cdot \boldsymbol{S}$, $\boldsymbol{R} + \boldsymbol{S}$, $\boldsymbol{R} \odot \boldsymbol{S}$, as well as $\boldsymbol{R} \oplus \boldsymbol{S}$. The respective operator signs are called the **complement**, **converse**, **absolute product**, **absolute sum**, **relative product**, and **relative sum**. There are no other relation designations of $\mathfrak{R}$. If R and S are arbitrary relation designations of $\mathfrak{R}$, then the expression $\mathbf{R} = \mathbf{S}$ is called a **sentence of $\mathfrak{R}$**.

In what follows, we shall present two different (but ultimately equivalent) ways of giving meanings to relation designations and sentences of the calculus of relations. The first translates them into first-order formulae of a special kind, that is, those drawn from the elementary theory of relations. The second maps them into an algebra of binary relations comprising model-theoretic operations such as complementation, intersection, union, as well as the composition of binary relations. Of course, this is exactly the way meanings are given to roles of $\mathcal{U}$. In fact, we shall define this mapping by means of interpretations borrowed from Section 3.1.

Definition 26. The **elementary theory of binary relations** (also called **elementary theory of relations** for short) is the set of all those first-order formulae with equality which involve no nonlogical symbols other than binary predicate symbols. All those formulae of the elementary theory of relations which do not contain any occurrence of a free variable are called **sentences**.

Definition 27. The function $\widetilde{\cdot}$ maps relation designations of $\mathfrak{R}$ along with ordered pairs of variables (drawn from the elementary theory of relations) to formulae of the elementary theory of relations. In particular, an arbitrary relation designation, R, and two variables, say, X and Y, are mapped to a formula defined by induction as follows:

$$\widetilde{\boldsymbol{R}}(\boldsymbol{X},\boldsymbol{Y}) \;=\; \begin{cases} R(X,Y) & \text{if } R \text{ is a relation variable;} \\ \mathit{true} & \text{if } R \text{ is } 1; \\ X = Y & \text{if } R \text{ is } \mathring{1}; \\ \neg\, \widetilde{R'}(X,Y) & \text{if } R \text{ is } \overline{R'}; \\ \widetilde{R'}(Y,X) & \text{if } R \text{ is } \breve{R'}; \\ \widetilde{R'}(X,Y) \wedge \widetilde{R''}(X,Y) & \text{if } R \text{ is } R' \cdot R''; \\ \widetilde{R'}(X,Y) \vee \widetilde{R''}(X,Y) & \text{if } R \text{ is } R' + R''; \\ \exists Z.\, \widetilde{R'}(X,Z) \wedge \widetilde{R''}(Z,Y) & \text{if } R \text{ is } R' \odot R''; \\ \forall Z.\, \widetilde{R'}(X,Z) \vee \widetilde{R''}(Z,Y) & \text{if } R \text{ is } R' \oplus R''. \end{cases}$$

Of course, this definition presupposes that relation variables can be viewed as binary predicate symbols of the elementary theory of relations. The particular cases handling relative sums and products both assume Z to be an arbitrary chosen variable which differs from X and Y.

In addition, the operation sign $\tilde{\cdot}$ stands also for a function mapping sentences of $\Re$ to formulae of the elementary theory of relations as follows. If X and Y are arbitrarily chosen but fixed variables of the elementary theory of relations such that $X \neq Y$, then $\tilde{\cdot}$ maps each sentence, $R = S$, of $\Re$ to the formula $\forall X.\forall Y.\, \tilde{R}(X,Y) \leftrightarrow \tilde{S}(X,Y)$. We henceforth denote this formula by $\widetilde{(\boldsymbol{R} = \boldsymbol{S})}$. Finally, we set $\widetilde{\Re}$ to be the set of all those sentences, α, of the elementary theory of relations for which there is at least one sentence, $R = S$, of $\Re$ such that $\widetilde{(R = S)}$ yields α.

Having this mapping at our disposal, the meaning of the calculus of relations is unambiguously determined in terms the elementary theory of relations. A semantics in this spirit can also be found, for example, in [Tarski, 1941], pages 75-76. Take, for instance, the following sentence, which forces the relation designation R to be a one-to-one function mapping the interpretation domain onto itself:

$$1 \;=\; (\overline{R \odot \breve{R}} + \mathring{1}) \cdot (\overline{\breve{R} \odot R} + \mathring{1}) \cdot (R \odot 1) \cdot (1 \odot R).$$

If the translation mapping $\tilde{\cdot}$ is applied to this sentence of $\Re$, it yields the sentence of the elementary theory of relations shown below. Here, we allow ourselves the freedom of using $\alpha \rightarrow \beta$ instead of $\neg\alpha \vee \beta$:

$$\begin{aligned} \forall X.\forall Y.\; & (\forall Z.(R(X,Z) \wedge R(Y,Z)) \rightarrow X = Y \quad \wedge \\ & (\forall Z.(R(Z,X) \wedge R(Z,Y)) \rightarrow X = Y \quad \wedge \\ & \exists Z.R(X,Z) \wedge \mathit{true} \qquad\qquad\qquad\quad \wedge \\ & \mathit{true} \wedge \exists Z.R(Z,Y). \end{aligned}$$

However, as already mentioned, this is not the only way of giving meaning to relation designations and sentences of $\Re$. Another possibility is to extend the definition of an interpretation, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$, as given in Section 3.1, so as to cope with both relation designations and sentences of $\Re$. It is convenient to assume that $\mathcal{N}_{\mathcal{R}}$ (i.e., the set of all role names) contains at least all relational variables as its members. Consequently, $R^{\mathcal{I}}$ is then defined to be $\mathcal{V}(R)$ whenever R is a relational variable. The interpretation function, $.^{\mathcal{I}}$, applied to all other relation designations of $\Re$ is defined by induction as follows. First, when applied to 1 and $\mathring{1}$, it yields the full domain, $\Delta^{\mathcal{I}}$, and the indentity relation over $\Delta^{\mathcal{I}}$ respectively. For the induction step assume that $.^{\mathcal{I}}$ applied to some relation designations, say, R and S, has already been defined. Then $(\overline{\boldsymbol{R}})^{\boldsymbol{\mathcal{I}}}$, $(\breve{\boldsymbol{R}})^{\boldsymbol{\mathcal{I}}}$, $(\boldsymbol{R} \cdot \boldsymbol{S})^{\boldsymbol{\mathcal{I}}}$, and $(\boldsymbol{R} \odot \boldsymbol{S})^{\boldsymbol{\mathcal{I}}}$ are defined to be $(\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus R^{\mathcal{I}}$, $\{\langle e,d \rangle : \langle d,e \rangle \in R^{\mathcal{I}}\}$, $R^{\mathcal{I}} \cap S^{\mathcal{I}}$, and $R^{\mathcal{I}} \circ S^{\mathcal{I}}$ respectively. Absolute and relative sum are to be interpreted such that they reflect the duals of the corresponding products, that is, $(\boldsymbol{R} + \boldsymbol{S})^{\boldsymbol{\mathcal{I}}}$

denotes $R^{\mathcal{I}} \cup S^{\mathcal{I}}$, whereas $(\boldsymbol{R} \oplus \boldsymbol{S})^{\mathcal{I}}$ is $\{\langle d, e\rangle \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} : \forall f \in \Delta^{\mathcal{I}},$ either $\langle d, f\rangle \in R^{\mathcal{I}}$ or $\langle f, e\rangle \in S^{\mathcal{I}}\}$. Finally, $.^{\mathcal{I}}$ applied to an arbitrary sentence, $R = S$, of $\Re$ yields $\Delta^{\mathcal{I}}$ if $R^{\mathcal{I}} = S^{\mathcal{I}}$, and the empty set otherwise.

The following fact states that the first-order semantics for the calculus of relations and its model theoretical counterpart are ultimatively equivalent in the following strong sense.

Fact 1. Assume $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$ is an arbitrary interpretation, and note that the $\mathcal{N}$-valuation $\mathcal{V}$ can always be extended to a first-order interpretation over $\Delta^{\mathcal{I}}$. If $\mathcal{N}$ contains at least all binary predicate symbols of $\widetilde{\Re}$, then this first-order interpretation can deal with any formula of $\widetilde{\Re}$. For every such first-order interpretation, $[\![.]\!]^{\mathcal{V}}$, and every relation designation, R, of $\Re$, $[\![\widetilde{R}]\!]^{\mathcal{V}}$ denotes exactly the same set as $R^{\mathcal{I}}$. Therefore, for every sentence, $R = S$, of $\Re$, $(\widetilde{R = S})$ evaluates to true in $[\![.]\!]^{\mathcal{V}}$ if and only if $(R = S)^{\mathcal{I}}$ yields $\Delta^{\mathcal{I}}$.

Now, returning to the issue of expressive power: It is not hard to see that three different variables rather than an arbitrary number of variables are sufficient for the function $\tilde{}$: Consider, for instance, $\widetilde{R \oplus S}(X, Y)$. This expression is defined to be $\exists Z.\widetilde{R}(X, Z) \wedge \widetilde{S}(Z, Y)$, where Z is a variable different from both X and Y. The first time when the recursive evaluation of $\widetilde{R}(X, Z)$ meets a relative product or sum, the variable Y can be reused within the new quantification which has to be introduced then. Similarly, the variable X can be reused in the course of performing $\widetilde{S}(Z, Y)$. This is why the function $\tilde{}$ can manage with three different variables rather than an arbitrary number of variables. But then $\widetilde{\Re}$ is clearly a sublanguage of the three-variable fragment of the elementary theory of relations.

Definition 28. The **three-variable fragment** of the elementary theory of relations, called $\boldsymbol{L_3}$ for short, is the set of all those formulae of the elementary theory of relations which involve at most three different variables.

As a matter of fact, $\widetilde{\Re}$ captures not only a sublanguage of L_3, but the whole extent of L_3. However, before going into detail, we have to generalize the notion of the expressive power so as to deal with arbitrary languages that have at least the notion of equivalence in common.

Definition 29. Suppose $\mathcal{L}$ and $\mathcal{L}'$ are two languages with a common notion of equivalence. Then $\mathcal{L}$ is said to be **at least as strong in expressive power as** $\mathcal{L}'$, in symbols $\boldsymbol{\mathcal{L}' \leq \mathcal{L}}$, if and only if for each syntactically well-formed expression of $\mathcal{L}'$ there is an equivalent one of $\mathcal{L}$. On the other hand, $\mathcal{L}$ is **strictly stronger in expressive power than** $\mathcal{L}'$ if and only if $\mathcal{L}' \leq \mathcal{L}$, but it is not the case that $\mathcal{L} \leq \mathcal{L}'$. Finally, $\mathcal{L}$ and $\mathcal{L}'$ are said to be **equipollent in means of expressive power** if and only if $\mathcal{L}' \leq \mathcal{L}$ and $\mathcal{L} \leq \mathcal{L}'$.

Expressiveness Theorem 4 (Tarski & Givant). *$\widetilde{\mathfrak{R}}$ and L_3 are equipollent in means of expressive power.*

This result was reported in [Tarski and Givant, 1987], page 90, item (i). Strictly speaking, Tarski and Givant prove a slightly weaker theorem. They only show that $\widetilde{\mathfrak{R}}$ and L_3 are equipollent in means of expressive power if both involve no nonlogical symbols other than a *single* binary predicate symbol. Careful inspection of their proof sketch, however, reveals that their result remains valid when more than one binary predicate is present. Instead of repeating their proof sketch, or even providing a clumsy complete proof, we confine ourselves to justifying this generalization by an example. This unsoundness is justified because in the remainder of the thesis nothing will depend on the validity of this claim. Rather, we mentioned this claim just to give an impression of the high expressive power $\widetilde{\mathfrak{R}}$ (and therefore also $\mathfrak{R}$) provides.

Example 2 ($L_3 \leq \widetilde{\mathfrak{R}}$). Consider the following sentence of the three-variable fragment of the elementary theory of relations:

$$\forall X.\exists Y.\forall Z.R_1(X,Y) \wedge R_2(X,Z) \rightarrow \exists X.X \neq Z \wedge R_3(Y,Z) \wedge R_4(X,X).$$

It should be obvious that this sentence can be rewritten as follows without affecting its meaning:

$$\forall X.\exists Y.\forall Z.\overbrace{\neg R_1(X,Y) \vee \neg R_2(X,Z) \vee \exists X.\underbrace{\underbrace{R_4(X,X) \wedge X \neq Z}_{\stackrel{\text{def}}{=}\gamma} \wedge R_3(Y,Z)}_{\stackrel{\text{def}}{=}\beta}}^{\stackrel{\text{def}}{=}\alpha}.$$

In what follows, we shall see that there is a sentence of $\widetilde{\mathfrak{R}}$ which is equivalent to this three-variable sentence. In fact, the $\sim$-image of the following sentence of $\mathfrak{R}$ proves to be such an equivalent sentence of $\widetilde{\mathfrak{R}}$:

$$\overbrace{\underbrace{\left(\overline{R_1} + \left(\overline{R_2} \oplus \left(\underbrace{\left(\left(\left(1 \odot \left(R_4 \cdot \mathring{1}\right)\right) \cdot \breve{\mathring{1}}\right)}_{\stackrel{\text{def}}{=}R_\gamma} \odot 1\right) \cdot \breve{R}_3\right)\right)\right)}_{\stackrel{\text{def}}{=}R_\beta} \odot 1}^{\stackrel{\text{def}}{=}R_\alpha} = 1.$$

When resorting to the abbreviations $\alpha, \beta, \gamma, R_\alpha, R_\beta$, and R_γ, it is not hard to verify that the $\sim$-image of the sentence just given is equivalent to exactly that sentence of L_3 we aim to express, that is to say, $(R_\alpha \widetilde{\odot 1} = 1) \equiv \forall X.\exists Y.\alpha$. This can be done step

by step as follows:

$$\begin{aligned}
(R_\alpha \,\widetilde{\odot}\, 1 = 1) &= \forall X.\forall Z.(\exists Y.\, \widetilde{R_\alpha}(X,Y) \wedge true \leftrightarrow true) \\
&\equiv \forall X.\exists Y.\, \widetilde{R_\alpha}(X,Y); \\
\widetilde{R_\alpha}(X,Y) &= \neg R_1(X,Y) \vee \forall Z.\neg R_2(X,Z) \vee \widetilde{R_\beta}(Z,Y); \\
\widetilde{R_\beta}(Z,Y) &= \exists X.\widetilde{R_\gamma}(Z,X) \wedge true \wedge R_3(Y,Z) \\
&\equiv \exists X.\widetilde{R_\gamma}(Z,X) \wedge R_3(Y,Z); \\
\widetilde{R_\gamma}(Z,X) &= \exists U.true \wedge R_4(U,X) \wedge U = X \wedge X \neq Z \\
&\equiv R_4(X,X) \wedge X \neq Z.
\end{aligned}$$

We should not close this part without mentioning that a result similar to the Expressiveness Theorem 4 has been put forward independently by Borgida [1993]. In effect, he claims that what we shall call the first-order fragment of $\mathcal{U}$ is expressively equivalent to the three-variable fragment of first-order logic with equality. Of course, the three-variable fragment must not involve any nonlogical symbol other than unary and binary predicate symbols or constants. The **first-order fragment of $\mathcal{U}$** refers to all those concepts and roles of $\mathcal{U}$ which do not involve any occurrence of the transitive or the reflexive-transitive closure of roles. His proof in support of this claim, however, contains a major error.[2]

What we know for sure is that $\Re$ is a strict lower bound of $\mathcal{U}$'s expressive power and even of $\mathcal{U}$'s first-order fragment.

Expressiveness Theorem 5. *The first-order fragment of the universal description logic $\mathcal{U}$ is strictly stronger in expressive power than the calculus of relations $\Re$.*

Proof. The following straightforward equivalences form the basis for a proof of the

[2]One crucial argument in Borgida's [1993] line of reasoning is that certain three-variable subformulae can be transformed into disjunctive normal form. On page 11 of [Borgida, 1993], it is claimed that the subformula $\Phi(X,Y,Z)$ can be transformed into an equivalent formula of the form $\bigvee_i \bigwedge_j \Psi_{i,j}$. However, this can be accomplished only if $\Phi(X,Y,Z)$ does not contain any quantifier, which is not necessarily the case. In the presence of quantifiers, $\Phi(X,Y,Z)$ has to be transformed into prenex normal form. That is, it must be transformed in a way that it is of the form $Q_1X_1.\ldots Q_nX_n.\phi$, where the Q_i's are either $\exists$ or $\forall$ and ϕ is quantifier free. Such a transformation, however, might cause serious problems because it possibly introduces new variables and, thereby, might lead out of the three-variable fragment.

fact that the first-order fragment of $\mathcal{U}$ is at least as strong in expressive power as $\mathfrak{R}$:

$$\begin{array}{lll} \models & 1 & \doteq \top\times\top, \\ \models & \mathring{1} & \doteq \epsilon, \\ \models & \overline{R} & \doteq \neg R, \\ \models & \breve{R} & \doteq R^{-1}, \\ \models & R\cdot S & \doteq R\sqcap S, \\ \models & R+S & \doteq R\sqcup S, \\ \models & R\odot S & \doteq R\circ S, \\ \models & R\oplus S & \doteq \neg(\neg R\circ\neg S), \\ \models & R=S & \doteq \forall(\top\times\top){:}\Big((R\leq S)\sqcap(S\leq R)\Big). \end{array}$$

These equivalences induce a proof on the structure of relation designations and sentences of $\mathfrak{R}$ showing that for every such a relation designation and sentence there exists at least one equivalent concept or role of the first-order fragment of $\mathcal{U}$.

In order to prove that the first-order fragment of $\mathcal{U}$ is even *strictly* stronger in expressive power than $\mathfrak{R}$, we can make use of the following fact. It is known that there is no formula of $\widetilde{\mathfrak{R}}$ which is equivalent to this sentence of the elementary theory of relations:

$$\exists X.\exists Y.\exists Z.\forall U.(X=U\vee Y=U\vee Z=U).$$

This observation is due to Korselt, and was reported in [Löwenheim, 1915], page 448. In view of Fact 1, this result can be recast as follows. There exists no sentence (and clearly no relation designation) of the calculus of relations such that for every interpretation, $\langle\Delta^{\mathcal{I}},.^{\mathcal{I}},\mathcal{V}\rangle$, it holds that $.^{\mathcal{I}}$ applied to this sentence would yield $\Delta^{\mathcal{I}}$ if and only if its cardinality, $\sharp|\Delta^{\mathcal{I}}|$, is greater than 0, but lower than or equal to 3. The following concept of $\mathcal{U}$, however, does have this very property:

$$\exists(\top\times\top){:}\exists^{\leq 3}(\top\times\top).$$

We thereby have fixed a concept of $\mathcal{U}$ which does not involve any occurrence of a transitive closure of a role, but which is not equivalent to any sentence and relation designation of the calculus of relations. □

Of course, this alone proves the first-order fragment of $\mathcal{U}$ to be undecidable because so is the calculus of binary relations [Tarski, 1941]. Another immediate corollary is the following.

Corollary 3. *The first-order fragment of the universal description logic $\mathcal{U}$ is strictly stronger in expressive power than L_3, the three-variable fragment of the elementary theory of binary relations.*

3.3 Recursion in $\mathcal{U}$

In what follows we will argue that explicit fixed-point operators gain their full definitional power only in combination with the concept and role-structuring primitives of $\mathcal{U}$. In particular, number restrictions as well as the inverse of a role will be essential for characterizing lists, trees, and binary trees. This is not surprising because the corresponding iterative definition of Section 3.1 uses these constructs too. In contrast to the iterative definitions, however, fixed-point operators are able to single out those lists, trees, and binary trees whose depth is *finite*. This is an improvement, but there are concepts for the definition of which fixed-point operators seem to be just indispensable: The solvable vertices of an AND-OR graph are an example. We shall furthermore argue that additional expressive power can yet be gained when not only concepts, but also roles are allowed to be defined by means of recursion. It will turn out that fixed-point operators on roles are able to capture a new role-structuring primitive which can be thought of as a generalization of the reflexive-transitive closure of a role. This role-structuring primitive captures the union of all role chains of the form $R^n S^n$ such that n ranges over all natural numbers greater than or equal to 0. A practical application of this new construct will be the definition of a balanced binary tree.

In Section 3.1 we already dealt with DAGs, trees, binary trees, and lists, but all these concepts were defined by *iteration* rather than by *induction* or *recursion*. However, with the exception of DAGs, all these concepts are usually defined by induction, and it seems to be most natural to do so. Take, for instance, lists. Their customary definition is by induction. A list is either the empty list, also called *nil*, or else there is exactly one head and one tail such that the latter refers to a list again. When resorting to the concepts Nil and NonemptyList, already defined in Figure 3.1 of Section 3.1, the corresponding recursive concept introduction looks as follows:

$$\mathsf{List} \doteq \mathsf{Nil} \sqcup (\mathsf{NonemptyList} \sqcap \forall \mathsf{tail}{:}\mathsf{List}).$$

In Chapter 2 we have also seen that in order to avoid ambiguities, such cyclic concept introductions must be given either least or greatest fixed-point semantics. The most flexible way to do so is to recast recursion in terms of explicit fixed-point operators, borrowed from the μ-calculus. In the case of the recursive concept introduction shown above, we have to make a decision between one of following two possibilities:

$$\begin{aligned} \mathsf{List} &\doteq \mu X.\big(\mathsf{Nil} \sqcup (\mathsf{NonemptyList} \sqcap \forall \mathsf{tail}{:}X)\big), \\ \mathsf{List} &\doteq \nu X.\big(\mathsf{Nil} \sqcup (\mathsf{NonemptyList} \sqcap \forall \mathsf{tail}{:}X)\big). \end{aligned}$$

The first is to be thought of as representing the least set, S, containing the empty list and those objects which have exactly one head as well as exactly one tail such that the tail in turn points to a member of S. The second concept introduction represents the greatest such set. The difference between the meanings of these two concept

$$\begin{aligned}
\mathsf{FiniteList} &\doteq \mu X.\big(\mathsf{Nil} \sqcup (\mathsf{NonemptyList} \sqcap \forall \mathsf{tail}{:}X)\big)\\
\mathsf{FiniteDepthTree} &\doteq \mu X.(\exists^{\leq 1}\mathsf{edge}^{-1} \sqcap \forall \mathsf{edge}{:}X)\\
\mathsf{FiniteBinaryTree} &\doteq \mu X.(\exists^{\leq 1}\mathsf{edge}^{-1} \sqcap \exists^{\leq 2}\mathsf{edge} \sqcap \forall \mathsf{edge}{:}X)
\end{aligned}$$

Figure 3.2: A terminology defining lists and trees of finite depth

introductions is that the former requires the recursion to terminate, whereas the latter does not. In particular, recursion must terminate by satisfying the condition of its base, i.e., the condition that the current object is an empty list. In Chapter 2 these two different readings were distinguished from each other by means of a new concept-structuring primitive, $\exists R^{\omega}$, stipulating that for each object it represents there is at least one infinite chain of the role R emanating from that object. In terms of concepts of this kind, the difference between $\mu X.(C \sqcap \forall R{:}X)$ and $\nu X.(C \sqcap \forall R{:}X)$ reduces to the fact that the former is equivalent to $\forall R^{*}{:}C \sqcap \neg\exists R^{\omega}$, while the latter is equivalent to $\forall R^{*}{:}C$. The concepts $\mu X.(\mathsf{Nil} \sqcup (\mathsf{NonemptyList} \sqcap \forall \mathsf{tail}{:}X))$ as well as $\nu X.(\mathsf{Nil} \sqcup (\mathsf{NonemptyList} \sqcap \forall \mathsf{tail}{:}X))$ fit into this scheme, too, at least when the concept introduction $\mathsf{Nil} \doteq \exists^{\leq 0}(\mathsf{head} \sqcup \mathsf{tail})$ is taken into account. The explanation is that Nil is equivalent to $\mathsf{Nil} \sqcap \forall \mathsf{tail}{:}X$ with respect to any terminology containing the aforementioned concept introduction. But then the same equivalence relationship holds for $(\mathsf{Nil} \sqcup (\mathsf{NonemptyList} \sqcap \forall \mathsf{tail}{:}X))$ and $(\mathsf{Nil} \sqcup \mathsf{NonemptyList}) \sqcap \forall \mathsf{tail}{:}X$ as well. Obviously, the latter concept fits into the scheme $C \sqcap \forall R{:}X$. Hence, we obtain the following equivalences relative to any terminology comprising at least the concept introduction $\mathsf{Nil} \doteq \exists^{\leq 0}(\mathsf{head} \sqcup \mathsf{tail})$:

$$\begin{aligned}
\mathcal{T} &\models \mu X.(\mathsf{Nil} \sqcup (\mathsf{NonemptyList} \sqcap \forall \mathsf{tail}{:}X)) \doteq \forall \mathsf{tail}^{*}{:}(\mathsf{Nil} \sqcup \mathsf{NonemptyList}) \sqcap \neg\exists \mathsf{tail}^{\omega},\\
\mathcal{T} &\models \nu X.(\mathsf{Nil} \sqcup (\mathsf{NonemptyList} \sqcap \forall \mathsf{tail}{:}X)) \doteq \forall \mathsf{tail}^{*}{:}(\mathsf{Nil} \sqcup \mathsf{NonemptyList}).
\end{aligned}$$

The common idea of lists is captured in what the least fixed-point operator expresses, especially when infinite and recursive lists (i.e., those lists having a tail-chain referring to themselves) are to be excluded. The greatest fixed-point operator, however, does include such nonstandard lists. A terminology defining finite, nonrecursive lists with the help of a least fixed-point is depicted in Figure 3.2. Recall that in contrast to this, the terminology fixed in Figure 3.1 of Section 3.1 defines a List in such a way that it excludes recursive lists, but not acyclic, infinite ones. This is why List actually *subsumes* FiniteList, but the subsumption does not hold the other way round. In this sense FiniteList is strictly stronger than List. On the other hand, a corresponding definition of FiniteList with the help of a greatest fixed-point operator is in the same sense strictly weaker than List because a greatest-fixed point operator even permits recursive lists.

It should be clear that trees and binary trees whose depth is finite can be captured in the spirit of a FiniteList. The corresponding concept introductions are also included

$$
\begin{aligned}
\mathsf{AND\text{-}OR\text{-}Graph} &\doteq \forall \mathsf{accessible}{:}\mathsf{AND\text{-}OR\text{-}Vertex} \\
\mathsf{AND\text{-}OR\text{-}Vertex} &\doteq \mathsf{AND\text{-}Vertex} \sqcap \neg \mathsf{OR\text{-}Vertex} \\
&\quad \sqcup \ \mathsf{OR\text{-}Vertex} \ \sqcap \neg \mathsf{AND\text{-}Vertex} \\
\mathsf{Leaf} &\doteq \mathsf{AND\text{-}OR\text{-}Vertex} \sqcap \exists^{\leq 0}\mathsf{edge} \\
\mathsf{SolvableVertex} &\doteq \mu X.\big(\mathsf{Leaf} \sqcup \\
&\qquad (\mathsf{AND\text{-}Vertex} \sqcap \forall \mathsf{edge}{:}X) \sqcup \\
&\qquad (\mathsf{OR\text{-}Vertex} \ \sqcap \exists \mathsf{edge}{:}X)\big)
\end{aligned}
$$

Figure 3.3: A terminology defining AND-OR graphs

in Figure 3.2. In this connection it is important to note that all these concept introductions do not make any use of the role-value map $\mathsf{edge}^+ \leq \neg\epsilon$ or equivalent concepts such as $\exists^{\leq 0}(\mathsf{edge}^+ \sqcap \epsilon)$ and alike. In the case of DAGs, however, it seems to be impossible to manage without such concepts. In particular, $\exists R^\omega$, expressed with the help of a fixed-point operator, cannot distinguish between R-chains which are cyclic and those which are acyclic but infinite. This confirms our impression that the transitive closure is a role structuring primitive in its own right, even with fixed-point operators at our disposal. But least and greatest fixed-point operators are concept structuring primitives in their own right, too, even when the transitive closure as well as concepts such as $\exists R^\omega$ are available. By the Expressiveness Theorem 3 of the previous chapter, it is known that this is definitely true in the case of the ω-regular extension of $\mathcal{ALC}$. In the case of $\mathcal{U}$, we confine ourselves to provide the following instructive example for which an equivalent handling in $\mathcal{U}$ itself does not seem not to work either.

Example 3 (AND-OR graphs). This example involves a special class of connected, directed graphs, known as *AND-OR graphs*. The vertices of these graphs are exhaustively partioned into two disjoint sets, one which contains so-called *AND vertices* and another containing *OR vertices*. Either kind of vertex is to be thought of as representing a certain problem whose solution can be decomposed into solving others. In particular, a problem associated with an AND vertex is solved when a solution to all those problems has been accomplished which are associated with its edge-successors. On the other hand, a problem associated with an OR vertex is solved when at least one of the problems associated with its edge-successors has been solved. Figure 3.3 shows a terminology defining such AND-OR graphs. Of course, we should not run into a an infinite decomposition sequence in the course of solving a problem, no matter whether the infinite decomposition sequence is cyclic or not. We define, therefore, the *solvable vertices* of an AND-OR graph, G, to be exactly those of its vertices which span an acyclic and finite subgraph, G_s, of G such that every AND vertex of G_s has

exactly those edges that it has in G. In addition, every OR vertex of G_s has to have at least one of those edges that it has in G. The fact that a least fixed-point operator is used rather than a greatest ensures that only those subgraphs count for problem decomposition which are both acyclic and finite.

The fixed-point operators considered so far are not the only ones we shall deal with: We consider not only those applied to concepts, but also those applied to *roles*. Such least and greatest fixed-point operators are role-structuring primitives of the form $\mu X.R$ or $\nu X.R$, where in either case R can be an arbitrary role, while X has to be a distinguished role name, referred to as a *role variable*. These roles are interpreted along much the same line as the corresponding fixed-point operators on concepts. In particular, $\mu X.R$ can be thought of as representing the least fixed point of a certain function on X induced by R, whereas $\nu X.R$ represents its greatest fixed point. A precise definition of the semantics of such fixed-point operators, however, must be postponed to Sections 3.4 and 3.5. Having these role-structuring primitives at our disposal, the reflexive-transitive closure of a role becomes definable in a very concise manner, as the following equivalence proves:

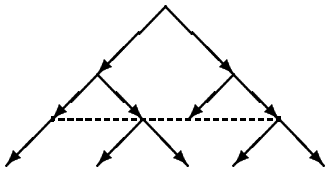
$$\models \quad R^* \doteq \mu X.(\epsilon \sqcup (R \circ X)).$$

The role $\mu X.(\epsilon \sqcup (R \circ X))$ represents the least binary relation which includes the identity relation and is closed under composition with the binary relation denoted by R; therefore it expresses the reflexive-transitive closure of R.

Not only role-structuring primitives (which are supported in $\mathcal{U}$ anyway) can be expressed in terms of fixed-point operators on roles, but also new ones such as $\boldsymbol{R^n \circ S^n}$. Roughly speaking, this role denotes all those ordered pairs of objects such that the first is related to the second by an arbitrary role chain of the shape $R^i S^i$ with i being any positive integer greater than or equal to zero.[3] The following equivalence holds:

$$\models \quad R^n \circ S^n \quad \doteq \quad \mu X.(\epsilon \sqcup (R \circ X \circ S)).$$

In the same spirit, we can capture the relation between nodes of a tree connecting one node with another just in case both are situated exactly at the same level of the tree. Given a particular node, we can access the single neighbor of that node including the node itself by the role $\epsilon \sqcup (\mathsf{edge}^{-1} \circ \mathsf{edge})$. However, in order to access all other nodes which are situated at the same level too, we actually have to retreat to recursion, as the following tree exemplifies:



[3]Technically speaking, $(\boldsymbol{R^n \circ S^n})^{\mathcal{I}}$ is defined to be the union of all binary relations $(R^{\mathcal{I}})^i \circ (S^{\mathcal{I}})^i$ such that i is allowed to range over all positive integers greater than or equal to zero.

$$
\begin{array}{rcl}
\mathsf{at\_same\_level} & \doteq & \mu X.\Big(\epsilon \sqcup (\mathsf{edge}^{-1} \circ X \circ \mathsf{edge})\Big) \\
\mathsf{BalancedBinaryTree} & \doteq & \mathsf{BinaryTree} \\
& \sqcap & \forall \mathsf{edge}^{*}\!:\!\Big((\exists^{\leq 0}\mathsf{edge} \circ \mathsf{edge}) \sqcup \forall \mathsf{at\_same\_level}\!:\!\exists^{\geq 2}\mathsf{edge}\Big)
\end{array}
$$

Figure 3.4: A terminology defining balanced binary trees

If, for instance, the left most node appearing on the dashed line is to be accessed from the right most one on that line, we first have to follow two edges up to the root and, then, follow two edges down to the left most node on the dashed line. The corresponding role introduction defining the role at_same_level with the help of a least fixed-point operator on a role is depicted in Figure 3.4.

Having the definition of the role at_same_level on hand, that of a so-called height-balanced binary tree becomes almost trivial. A binary tree is said to be *height-balanced* if and only if at any node in the tree, all those maximal paths which emanate from that node at its left-hand side and its right-hand side differ in length by at most one, see e.g. [Ralston and Reilly, 1993], page 1188. This means that a binary tree is height-balanced just in case all nodes that are situated at one level of the tree meet the following condition: If at least one of the nodes at this level has a path emanating from it whose length is greater than one, then both the right as well as the left paths emanating from any other node at this level are all nonempty. The corresponding concept introduction can be found in Figure 3.4.

3.4 Formal Monotonicity in $\mathcal{U}$

We are now going to lay the foundations of least and greatest fixed-point operators. As usual, the very foundation of such fixed-point operators, $\mu X.T$ and $\nu X.T$, is the function on X induced by T and a $\mathcal{N}$-valuation $\mathcal{V}$. The least and greatest fixed points of this function are then taken as denotations of the relevant fixed-point operators. Compared with its treatment in Chapter 2, however, this notion has to be generalized in the following sense. In order to cope also with fixed-point operators on roles, the case where X is a role variable and T is an arbitrary role has to be taken into account. As a matter of fact, all permutations of X being either a concept or role variable and of T being either a concept or a role have to be dealt with. All these cases are necessary in that $\mathcal{U}$ permits of concepts appearing in roles and vice versa. But then an inductive definition of the function on X induced by T and $\mathcal{V}$ may invoke corresponding functions of any of the four different types just described.

Notation 2. Assume $\mathcal{V}$ is an arbitrary $\mathcal{L}$-valuation over some set, say, Δ. If CN is a concept name of $\mathcal{L}$ and S is an arbitrary subset of Δ, then $\mathcal{V}_{\langle CN,S\rangle}$ is the $\mathcal{L}$-valuation over Δ which agrees with $\mathcal{V}$, except for the fact that $\mathcal{V}(CN)$ is S. That is to say, $\mathcal{V}_{\langle CN,S\rangle}$ denotes the $\mathcal{L}$-valuation $(\mathcal{V} \setminus \{\langle CN, \mathcal{V}(CN)\rangle\}) \cup \{\langle CN, S\rangle\}$. For RN being a role name of $\mathcal{L}$ and S being a binary relation over Δ, the valuation $\mathcal{V}_{\langle RN,S\rangle}$ is defined correspondingly.

Definition 30. Suppose T is an arbitrary concept or role, TN is a concept or role name, and $\mathcal{L}$ is a set of concept and role names as well as individual concepts which comprise as its elements at least those occurring in T. Moreover, assume that Δ is an arbitrary set and that Γ is Δ if TN is a concept name, and $\Delta \times \Delta$ otherwise. Similarly, let Θ be Δ if T is a concept, and $\Delta \times \Delta$ otherwise. Then for every $\mathcal{L}$-valuation, $\mathcal{V}$, over Δ, the **function on** TN **induced by** T and $\mathcal{V}$ is a function mapping subsets of Γ to subsets of Θ. This is defined as follows: Applied to an arbitrary subset of Γ, say, S, this function yields the subset of Θ denoted by $T^{\mathcal{I}}$. Here, $.^{\mathcal{I}}$ is the interpretation function of an arbitrary chosen interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}'\rangle$ such that $\Delta^{\mathcal{I}} = \Delta$ and $\mathcal{V}_{\langle TN,S\rangle} \subseteq \mathcal{V}'$.

As in Definition 13, it is important to note that this function is actually well-defined, inspite of the fact that it is left open which particular interpretation of the form $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}'\rangle$ such that $\Delta^{\mathcal{I}} = \Delta$ and $\mathcal{V}_{\langle TN,S\rangle} \subseteq \mathcal{V}'$ is to be taken. It can easily be seen that the value of $T^{\mathcal{I}}$ does not depend on which of the possible interpretations is actually chosen. The explanation is that these different $\mathcal{L}$-valuations may differ only in the way they handle those term names which do *not* occur in $\mathcal{L}$ and, therefore, do not occur in T either. The value of $T^{\mathcal{I}}$, however, does not depend on the these term names. Anyway, $\mathcal{V}'$ is uniquely determined whenever $\mathcal{L}$ coincides with $\mathcal{N}$, i.e., it contains all existing concept and role names as well as individual concepts whatsoever. This is due to the fact that in this case $\mathcal{V}'$ coincides with $\mathcal{V}_{\langle TN,S\rangle}$.

But even with the function on X induced by T and $\mathcal{V}$ defined for the general case, the meanings of fixed-point operators in terms of the least and greatest fixed points of this function are not necessarily well-defined. This is because neither the existence nor the uniqueness of these fixed points can be taken for granted. From Chapter 2, however, we already know that this is ensured by Knaster and Tarski's fixed-point theorem for all those functions which are monotonically increasing. Therefore, it just remains to guarantee that the function on X induced by T and $\mathcal{V}$ is always monotonically increasing, no matter which particular $\mathcal{N}$-valuation, $\mathcal{V}$, is taken as a basis. The notion of syntactic or formal monotonicity developed in Chapter 2 has been devised for this purpose. It remains to generalize this notion to cope with the additional concept and role-structuring primitives of $\mathcal{U}$.

Definition 31. Assume T is an arbitrary concept or role, while TN is a concept or role name. The **positive** and **negative occurrences** of TN in T are defined as

follows. An occurrence of TN in T is said to be positive (or negative) if and only if this occurrence appears in an even (or odd) number of different subconcepts and subroles of T which are either of the form $\neg C$, $\neg R$, or $\exists^{\leq m} R{:}C$, where C and R can be in each particular case an arbitrary concept and role respectively. Moreover, T is said to be **syntactically** or **formally monotonic** in TN if and only if there is no negative occurrence of TN in T, whereas T is **syntactically** or **formally anti-monotonic** in TN if and only if there is no positive occurrence of TN in T.

A similar notion has been developed by Park [1970] and Moschovakis [1974] for full first-order logic, see [Park, 1970], page 67, and Chapter 1B of [Moschovakis, 1974]. In view of our definition, it is important to keep in mind that a universal role restriction of the form $\forall R{:}C$ is treated as an abbreviation of $\exists^{\leq 0} R{:}\neg C$. But then this concept is formally monotonic in an arbitrary term name, say, TN, just in case C is formally monotonic in TN, whereas R has to be formally anti-monotonic in TN. Similar comments apply to role-value maps and structural descriptions. More precisely, because of the fact that the role-value map $R \leq S$ abbreviates to $\exists^{\leq 0}(R \sqcap \neg S){:}\top$, it is formally monotonic in TN just in case S is formally monotonic in TN, but R has to be formally anti-monotonic in TN. So far as structural descriptions are concerned, it should suffice to consider those of the form $\exists C{:}(R_1 \circledcirc S_1 \sqcap R_2 \circledcirc S_2)$. Such a structural description abbreviates to $\exists^{\geq 1}(\neg(R_1 \circ \neg S_1^{-1}) \sqcap \neg(\neg R_2 \circ S_2^{-1})){:}C$ and, therefore, it is formally monotonic in TN just in case all concept and roles among C, S_1, and R_2 are formally monotonic in TN, but both R_1 and S_2 must be formally anti-monotonic in TN.

Definition 32. A function, f, mapping subsets of some set, say, Γ, to subsets of Γ is called **monotonically increasing**, or **monotonic** for short, if and only if for every two subsets, S and S', of Δ such that $S \subseteq S'$, it holds that $f(S) \subseteq f(S')$. Moreover, f is **anti-monotonic** if and only if for every such two subsets, it holds that $f(S) \supseteq f(S')$.

Proposition 7. *Assume T is a concept or role of $\mathcal{U}$, TN is a concept or role name of $\mathcal{U}$, and $\mathcal{V}$ is an arbitrary $\mathcal{N}$-valuation. If T is formally monotonic in TN, then the function on TN induced by T and $\mathcal{V}$ is monotonic, whereas it is anti-monotonic whenever T is formally anti-monotonic in TN.*

Proof. Let Δ be the set over which $\mathcal{V}$ operates, that is, $\mathcal{V}$ is a $\mathcal{N}$-valuation over Δ. Let f_T denote the function on TN induced by T and $\mathcal{V}$. The proof that f_T is monotonic (or anti-monotonic) proceeds by induction on the structure of the concept or role T.

Induction base: We have to consider the cases where T is atomic, i.e., it is either a concept or role name, an individual concept, the universal concept, $\top$, or the identity role, ϵ. In each of these cases T is formally monotonic in TN. It therefore has to be

verified that the function f_T is actually monotonic. It is readily seen that in each of these cases f_T denotes a constant function, with the only exception of T coinciding with TN. If, for instance, TN is a concept name different from T and the latter is a concept rather than a role, then f_T denotes the constant function that maps every subset of Δ to some fixed subset of Δ. In other words, there is an $S_0 \subseteq \Delta$ such that for all $S \subseteq \Delta$, $f_T(S)$ is S_0. Ignoring the possibility that T may coincide with TN, this means that f_T is trivially monotonic. So there remains the case when T coincides with TN. If T is TN, then f_T denotes the identity function mapping either every subset of Δ to itself or every binary relation over Δ to itself, depending on whether TN is a concept or role name respectively. It should be obvious that every such an identity function is, in fact, monotonic.

The induction step: Let T be of the form $T_1 \sqcap T_2$, $T_1 \sqcup T_2$, $\neg T_1$, $\exists^{\geq n} R{:}C$, $\exists^{\leq m} R{:}C$, $R \circ S$, R^{-1}, R^{+}, or $C \times D$. In what follows, we shall adopt the convention that for every concept and role, T', $f_{T'}$ denotes the function on TN induced by T' and $\mathcal{V}$, where TN and $\mathcal{V}$ are taken to be exactly those which we have considered so far. It is not hard to verify that if all of the functions f_{T_1}, f_{T_1}, f_C, f_D, f_R, and f_S are monotonic (or anti-monotonic), then so are $f_{T_1 \sqcap T_2}$, $f_{T_1 \sqcup T_2}$, $f_{R \circ S}$, $f_{R^{-1}}$, $f_{R^{+}}$, as well as $f_{C \times D}$. On the other hand, $f_{\neg T_1}$ is anti-monotonic whenever f_{T_1} is monotonic, whereas it is monotonic whenever f_{T_1} is anti-monotonic. Bearing these facts in mind, all of the induction steps mentioned above are entirely straightforward, except for those involving qualified number restrictions.

So let us consider the case when T is of the form $\exists^{\leq m} R{:}C$. In order to do so, first assume that T is formally monotonic in TN. That is to say, there is no negative occurrence of TN in $\exists^{\leq m} R{:}C$. In view of Definition 31, this implies that there is neither in C nor in R any *positive* occurrence of TN, i.e., both C as well as R are formally *anti-monotonic* in TN. According to the induction hypothesis, both f_C as well as f_R are, therefore, anti-monotonic. We are now going to prove that $f_{\exists^{\leq m} R:C}$ is monotonic because f_C and f_R are anti-monotonic, where we shall adopt the following convention. Assume Γ denotes either Δ or $\Delta \times \Delta$, dependent on whether TN is a concept or a role name, so that $f_{\exists^{\leq m} R:C}$ can be described as a function mapping subsets of Γ to subsets of Δ. Recall that for an arbitrary subset, S, of Γ, $f_C(S)$ and $f_R(S)$ are defined to be $C^{\mathcal{I}}$ and $R^{\mathcal{I}}$ respectively, where $.^{\mathcal{I}}$ is an interpretation function of $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}' \rangle$ such that $\Delta^{\mathcal{I}} = \Delta$ and $\mathcal{V}'_{\langle TN, S \rangle} = \mathcal{V}$. Resorting to this interpretation function $.^{\mathcal{I}}$, we can state that the following equations are satisfied, for every subset, S, of Γ:

$$\begin{aligned} f_{\exists^{\leq m} R:C}(S) &= (\exists^{\leq m} R{:}C)^{\mathcal{I}} \\ &= \{d \in \Delta : \sharp |R^{\mathcal{I}}(d) \cap C^{\mathcal{I}}| \leq m\} \\ &= \{d \in \Delta : \sharp |f_R(S)(d) \cap f_C(S)| \leq m\}. \end{aligned} \tag{3.1}$$

If we take two arbitrary subsets of Γ, say, S_1 and S_2, such that S_1 is a subset of S_2, then we can come up with the conclusion that $f_{\exists^{\leq m} R:C}(S_1)$ is a subset of $f_{\exists^{\leq m} R:C}(S_2)$,

too, as the following equations prove:

$$\begin{array}{lll} & f_{\exists\leq^m R:C}(S_1) & \\ = & \{d \in \Delta : \sharp|f_R(S_1)(d) \cap f_C(S_1)| \leq m\} & \text{(according to (3.1))} \\ \subseteq & \{d \in \Delta : \sharp|f_R(S_2)(d) \cap f_C(S_2)| \leq m\} & \text{(because } f_C \text{ and } f_R \text{ are anti-monotonic)} \\ = & f_{\exists\leq^m R:C}(S_2) & \text{(according to (3.1)).} \end{array}$$

But this means that $f_{\exists\leq^m R:C}$ is monotonic, as was to be shown. The third line above, however, may call for comment. As f_C and f_R are known to be anti-monotonic, it is also known that $f_C(S_1) \supseteq f_C(S_2)$ and $f_R(S_1) \supseteq f_R(S_2)$. An immediate consequence is then that for every $d \in \Delta^{\mathcal{I}}$, $f_R(S_1)(d) \supseteq f_R(S_2)(d)$. From this we can conclude that for every $d \in \Delta^{\mathcal{I}}$, $f_R(S_1)(d) \cap f_C(S_1)$ is a superset of $f_R(S_2)(d) \cap f_C(S_2)$. This means that the cardinality of $f_R(S_1)(d) \cap f_C(S_1)$ is greater than or equal to the cardinality of $f_R(S_2)(d) \cap f_C(S_2)$. But then for every natural number, m, all those $d \in \Delta$ such that the cardinality of $f_R(S_1)(d) \cap f_C(S_1)$ is not greater than m are, in fact, contained in the corresponding set with S_2 substituted for S_1.

The remaining cases can be shown analogously. □

Definition 33. Assume f is an arbitrary function mapping subsets of some set, say, Γ, to subsets of Γ. A subset, S, of Γ is said to be a **fixed point** of f if and only if $f(S) = S$, and such a fixed point of f is a **least fixed point** of f if and only if each fixed point of f is a superset of it. Similarly, a fixed point of f is a **greatest fixed point** of f if and only if each fixed point of f is a subset of it. Finally, a subset, S, of Γ is said to be a **prefixed point** of f if and only if $f(S) \subseteq S$, while it is a **postfixed point** of f if and only if $f(S) \supseteq S$.

Note that the models $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ of a concept or role introduction of the form $TN \doteq T$ can be recast by means of the fixed points of the function of on TN induced by T and $\mathcal{V}$ and, accordingly, so can the models of $TN \sqsubseteq T$ and $T \sqsubseteq TN$ in terms of its postfixed and prefixed points respectively.

Fact 2. Assume $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ is an arbitrary interpretation. Then $\mathcal{I}$ is a model of the concept or role introduction $TN \doteq T$ if and only if $TN^{\mathcal{I}}$ is a fixed point of the function on TN induced by T and $\mathcal{V}$, while $\mathcal{I}$ is a model of $TN \sqsubseteq T$ (or $T \sqsubseteq TN$) if and only if $TN^{\mathcal{I}}$ is a postfixed (or prefixed) point of this function.

As already mentioned, Knaster and Tarski's fixed-point theorem establishes not only the existence of unique least as well as unique greatest fixed points of monotonic functions, but it gives also rise to a characterization of these least and greatest fixed points in terms of ordinary fixed points, or, alternatively, by means of prefixed and postfixed points. In its historical formulation, reported in [Tarski, 1955], however,

the theorem refers to *complete lattices* rather than to functions of type $2^{\Delta} \to 2^{\Delta}$ or $2^{\Delta\times\Delta} \to 2^{\Delta\times\Delta}$ which are monotonically increasing with respect to the subset relation. Clearly, these functions can also be viewed as functions of type $2^{\Gamma} \to 2^{\Gamma}$, at least when Γ is taken to be $\Delta \cup (\Delta \times \Delta)$. It is well-known that for *every* set Γ (not necessarily of the latter kind), the set of all subsets of Γ, 2^{Γ}, forms a complete lattice along with the subset relation over Γ.

Theorem 1 (Knaster & Tarski). *Let Γ be a set. Assume f is a monotonic function mapping subsets of Γ to subsets of Γ. Then the intersection of all fixed points of f coincides with the intersection of all its prefixed points, which is in turn a fixed point of f. Similarly, the union of all fixed points of f coincides with the union of all its postfixed points, which is also a fixed point of f.*

It should be clear that any fixed point of f which coincides with the intersection of all its fixed points is necessarily a *least* fixed point of f and, correspondingly, any fixed point of f which coincides with the union of all its fixed points is necessarily a *greatest*.

Corollary 4. *Assume f is a monotonic function mapping subsets of Γ to subsets of Γ. Then there is both a unique least and a unique greatest fixed point of f. In particular, the former coincides with the intersection of all prefixed points of f, whereas the latter coincides with the union of all its postfixed points.*

3.5 The Fixed-Point Description Logic $\mathcal{U}\mu$

In this section we shall define the syntax and semantics of a description logic, called $\mathcal{U}\mu$, which augments $\mathcal{U}$ by least and greatest fixed-point operators both on concepts as well as on roles. For technical reasons, we add such fixed-point operators not to $\mathcal{U}$ directly, but to a slight modification of it. We shall omit the transitive closure of roles in that this role-structuring primitive will turn out to be expressible by least fixed-point operators on roles along with the remaining role-structuring operators of $\mathcal{U}$. The transitive closure of an arbitrary role, R, free of any occurrence of the role variable X is, in fact, equivalent to $\mu X.(R \sqcup (R \circ X))$. The reader may object that the latter is not always polynomially bounded in the length of R^+. However, R^+ is equivalent to $\mu X.(R \sqcup (X \circ X))$ too. The length of this role is always linearly bounded in the length of R^+. Hence, we may treat R^+ as an abbreviation of $\mu X.(R \sqcup (X \circ X))$ without running into problems. The second modification of $\mathcal{U}$ concerns the addition of a role-structuring operator of the form $\mathbf{R} \oplus \mathbf{S}$, called **relative sum**, taken from the calculus of relations. Having this new role-structuring primitive at hand, we can represent, for instance, the complement of the grandparent relation in terms of the complement of the parent relation, along much the same lines the grandparent

relation can be defined by the composition of the parent relation with itself. In fact, $\neg$parent_of $\oplus$ $\neg$parent_of can be thought of as representing the complement of the grandparent relation. More precisely, it represents all those ordered pairs of objects, $\langle d, e\rangle$, such that for each object, f, it holds that either f is not related to d by the role parent_of or e is not related to f by parent_of. In general, this binary operator on roles is to be interpreted in such a way that for each interpretation, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$, $(\boldsymbol{R} \oplus \boldsymbol{S})^{\mathcal{I}}$ denotes the set of all those ordered pairs, $\langle d, e\rangle$, of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that for every $f \in \Delta^{\mathcal{I}}$, either $\langle d, f\rangle$ is a member of $R^{\mathcal{I}}$ or $\langle f, e\rangle$ is a member of $S^{\mathcal{I}}$. As a matter of fact, this additional role-structuring operator does not add anything to the expressive power of $\mathcal{U}$ in that $R \oplus S$ is equivalent to $\neg(\neg R \circ \neg S)$. However, this new role-structuring primitive will play an essential role in obtaining the so-called *negation normal form* of roles in the sense that it is indispensable for moving role negations inwards as long as they are applied to noncompound roles.

Definition 34. Assume $\boldsymbol{\mathcal{X}}$ to be a nonempty set of distinguished concept and role names, called **concept variables** and **role variables** respectively. The **concepts and roles of $\boldsymbol{\mathcal{U}\mu}$** are simultaneously defined as:

1. All concept names, all individual concepts, all concept variables, and $\top$ are concepts of $\mathcal{U}\mu$.
2. All role names, all role variables, and ϵ are roles of $\mathcal{U}\mu$.
3. Suppose C and D are concepts of $\mathcal{U}\mu$, R and S are roles of $\mathcal{U}\mu$, and $n \geq 1$ and $m \geq 0$ are any natural numbers. Then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\exists^{\geq n} R{:}C$, and $\exists^{\leq m} R{:}C$ are concepts of $\mathcal{U}\mu$, whereas $R \sqcap S$, $R \sqcup S$, $\neg R$, $R \circ S$, $R \oplus S$, R^{-1}, and $C \times D$ are roles of $\mathcal{U}\mu$.
4. Suppose X is a concept variable and C is a concept of $\mathcal{U}\mu$ formally monotonic in X. Then both $\boldsymbol{\mu X.C}$ and $\boldsymbol{\nu X.C}$ are concepts of $\mathcal{U}\mu$.
5. Suppose X is a role variable and R is a role of $\mathcal{U}\mu$ formally monotonic in X. Then both $\boldsymbol{\mu X.R}$ and $\boldsymbol{\nu X.R}$ are roles of $\mathcal{U}\mu$.
6. These are the concepts and roles of $\mathcal{U}\mu$.

Concepts and roles of the form $\mu X.T$ and $\nu X.T$ are called **least** and **greatest fixed-point operators** respectively.

As already suggested, we shall henceforth treat the transitive closure of a role as an abbreviation defined as follows, where X is always assumed to be a fresh role variable not occurring in R:

$$\boldsymbol{R^{+}} \stackrel{\text{def}}{=} \mu X.(R \sqcup (X \circ X)).$$

Of course, the definition of an interpretation, as given in Section 3.1, has to be extended to deal with the additional concept and role-structuring primitives of $\mathcal{U}\mu$. The way relative sums are to be interpreted has already been described: $(R \oplus S)^{\mathcal{I}}$ is $\{\langle d, e\rangle$: for all $f \in \Delta^{\mathcal{I}}, \langle d, f\rangle \in R^{\mathcal{I}}$ or $\langle f, e\rangle \in S^{\mathcal{I}}\}$. It therefore remains to ascribe meanings to least and greatest fixed-point operators. For an arbitrary interpretation, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$, the definition of both $(\mu X.T)^{\mathcal{I}}$ and $(\mu X.T)^{\mathcal{I}}$ is by induction. Suppose, f_T denotes the function on X induced by T and $\mathcal{V}$. Notably, in so doing, it is implicitly presupposed that $T^{\mathcal{J}}$ has already been defined. As a matter of fact, this is presupposed for each interpretation, $\langle \Delta^{\mathcal{J}}, .^{\mathcal{J}}, \mathcal{W}\rangle$ such that $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}}$ and $\mathcal{W} = \mathcal{V}_{\langle X,S\rangle}$, where S ranges here over all subsets of $\Delta^{\mathcal{I}}$ or $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, depending on whether X is a concept or role variable. Then $.^{\mathcal{I}}$ applied to $\mu X.T$ and $\nu X.T$ is defined as follows:

$$\begin{aligned} (\boldsymbol{\mu X.T})^{\boldsymbol{\mathcal{I}}} &= \bigcap\{S \subseteq \Gamma : f_T(S) \subseteq S\}; \\ (\boldsymbol{\nu X.T})^{\boldsymbol{\mathcal{I}}} &= \bigcup\{S \subseteq \Gamma : f_T(S) \supseteq S\}. \end{aligned}$$

That is to say, $(\mu X.T)^{\mathcal{I}}$ denotes the intersection of all prefixed points of the function on X induced by T and $\mathcal{V}$, while $(\nu X.T)^{\mathcal{I}}$ denotes the union of all its postfixed points. If we were allowed at this stage to apply Knaster and Tarski's theorem, we would know that $(\mu X.T)^{\mathcal{I}}$ and $(\nu X.T)^{\mathcal{I}}$ denote the least and the greatest fixed point of the function on X induced by T and $\mathcal{V}$, as was intended. In order to do so, however, it must be ensured that this function is *monotonic*. Of course, Proposition 7 does guarantee exactly this, but solely in case of T being a concept or role of $\mathcal{U}$ rather than of $\mathcal{U}\mu$. Proposition 7, therefore, remains to be generalized in order to deal with the additional concept and role-structuring operators of $\mathcal{U}\mu$.

Proposition 7b. *Assume T is a concept or role of $\mathcal{U}\mu$, TN is a concept or role name, and $\mathcal{V}$ is an arbitrary $\mathcal{N}$-valuation over some set, say, Δ. If T is formally monotonic in TN, then the function on TN induced by T and $\mathcal{V}$ is monotonic, whereas it is anti-monotonic whenever T is formally anti-monotonic in TN.*

Proof. Throughout the proof, we shall adopt the following convention. For every concept or role name, TN, every concept or role, T, and every $\mathcal{N}$-valuation, $\mathcal{V}$, the function on TN induced by T and $\mathcal{V}$ is denoted by $f^{TN}_{T,\mathcal{V}}$. In terms of this notation, we have to show that, when TN, T and $\mathcal{V}$ are given as in the proposition we are about to prove, $f^{TN}_{T,\mathcal{V}}$ is monotonic whenever T is formally monotonic in TN, whereas it is anti-monotonic whenever T is formally anti-monotonic in TN. Just like Proposition 7, the proof proceeds by induction on the structure of T. The induction base as well as the induction steps are, of course, the very same, except for the additional cases when T is of the form $R \oplus S$, $\mu X.T'$, or $\nu X.T'$. As far as the first case is concerned, it suffices to verify the fact that if both $f^{TN}_{R,\mathcal{V}}$ and $f^{TN}_{S,\mathcal{V}}$ are monotonic (or anti-monotonic), then so is $f^{TN}_{R\oplus S,\mathcal{V}}$. The corresponding proof is straightforward and is, therefore, left to the reader.

So let us consider the more difficult case when T is of the form $\mu X.T'$. Assume $\mu X.T'$ is formally monotonic in TN, so that T' is formally monotonic in TN too. According to the induction hypothesis, we know that for every $\mathcal{N}$-valuation, $\mathcal{W}$, over Δ, the function $f^{TN}_{T',\mathcal{W}}$ on TN induced by T and $\mathcal{W}$ is monotonic. In the sequel, we shall prove that $f^{TN}_{T,\mathcal{V}}$ is monotonic because so is $f^{TN}_{T',\mathcal{W}}$, no matter which particular $\mathcal{N}$-valuation is denoted by $\mathcal{W}$. In order to do so, consider the set Γ which is assumed to be either Δ or $\Delta \times \Delta$, depending on whether T is a concept or a role. Moreover, take two arbitrary subsets, S_1 and S_2, of Γ such that the former is a subset of the latter. As T is assumed to be a least fixed-point operator of the form $\mu X.T'$, $f^{TN}_{T,\mathcal{V}}$ applied to S_i with i being either 1 or 2 yields the following:

$$f^{TN}_{T,\mathcal{V}}(S_i) \;=\; \bigcap\{S \subseteq \Gamma : f^{X}_{T',\mathcal{V}_{\langle TN,S_i\rangle}}(S) \subseteq S\}. \tag{3.2}$$

In what follows, it will be important that for each subset, S, of Γ, the function $f^{X}_{T',\mathcal{V}_{\langle TN,S_i\rangle}}$ applied to S yields exactly the same value as $f^{TN}_{T',\mathcal{V}_{\langle X,S\rangle}}$ applied to S_i. We shall also make use of the fact that according to the induction hypothesis the latter function is monotonic. Bearing this in mind, we can reason as follows:

$$\begin{array}{lll}
 & f^{TN}_{T,\mathcal{V}}(S_1) & \\
= & \bigcap\{S \subseteq \Gamma : f^{X}_{T',\mathcal{V}_{\langle TN,S_1\rangle}}(S) \subseteq S\} & \text{(according to (3.2))} \\
= & \bigcap\{S \subseteq \Gamma : f^{TN}_{T',\mathcal{V}_{\langle X,S\rangle}}(S_1) \subseteq S\} & \text{(because } f^{X}_{T',\mathcal{V}_{\langle TN,S_1\rangle}}(S) = f^{TN}_{T',\mathcal{V}_{\langle X,S\rangle}}(S_1)\text{)} \\
\subseteq & \bigcap\{S \subseteq \Gamma : f^{TN}_{T',\mathcal{V}_{\langle X,S\rangle}}(S_2) \subseteq S\} & \text{(because } f^{TN}_{T',\mathcal{V}_{\langle X,S\rangle}} \text{ is monotonic)} \\
= & \bigcap\{S \subseteq \Gamma : f^{X}_{T',\mathcal{V}_{\langle TN,S_2\rangle}}(S) \subseteq S\} & \text{(because } f^{TN}_{T',\mathcal{V}_{\langle X,S\rangle}}(S_2)) = f^{X}_{T',\mathcal{V}_{\langle TN,S_2\rangle}}(S) \\
= & f^{TN}_{T,\mathcal{V}}(S_2) & \text{(according to (3.2)).}
\end{array}$$

This just means that $f^{TN}_{\mathcal{V},}$ is monotonic, as was to be shown. The involved subset relation given above is not so obvious as one might think at first glance. It yet can be checked using solely basic set theory.

The other case when T is formally anti-monotonic in TN as well as the case that T is of the form $\nu X.T'$ can be shown analogously. □

Corollary 5. *Assume that $\mu X.T$ and $\nu X.T$ are any concepts or roles of $\mathcal{U}\mu$ and $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$ is an arbitrary interpretation. Then $(\mu X.T)^{\mathcal{I}}$ is the least fixed point of the function on X induced by T and $\mathcal{V}$, whereas $(\nu X.T)^{\mathcal{I}}$ is its greatest fixed point.*

We are now returning to issues of syntax. In particular, we are going to introduce some useful restrictions which can be imposed on the syntactic shape of concepts and roles of $\mathcal{U}\mu$ without giving up any bit of expressive power. Chapter 4 will make use of both of these restrictions.

Without loss of generality, we can confine ourselves to the case when each concept and role variable, X, is bounded by at most one fixed-point operator. That is to say,

there are no two concepts or roles of $\mathcal{U}\mu$ of the form $\mu X.T_1$ and $\nu X.T_2$ such that for every two concepts or roles either of the form $\mu X.T_1$ and $\mu X.T_2$ or of the form $\nu X.T_1$ and $\nu X.T_2$, it must be the case that T_1 coincides with T_2. The fact below justifies that this simplification can actually be put into effect without loss of generality. Hence, we do so in the remainder of the thesis.

Definition 35. Assume T is a concept or role of $\mathcal{U}\mu$, while X is a concept or role variable. Then an occurrence of X in T is said to be **bounded** if and only if this very occurrence appears in at least one subconcept or subrole of T being either of the form $\mu X.T'$ or $\nu X.T'$; otherwise it is **unbounded**. Moreover, T is called **closed** whenever there is no unbounded occurrence concept or role variable in T.

Fact 3. Assume $\mu X.T$ and $\nu X.T$ are concepts or roles of $\mathcal{U}\mu$ not involving any occurrence of Y. Here, Y is supposed to be a concept variable if X is one; else it is a role variable. Assume, moreover, $T_{X/Y}$ denotes the concept or role obtained from T by simultaneously substituting Y for each unbounded occurrence of X in T. Then $\mu X.T$ is equivalent to $\mu Y.T_{X/Y}$ and, similarly, $\nu X.T$ is equivalent to $\nu Y.T_{X/Y}$.

The second simplification of $\mathcal{U}\mu$'s syntax concerns its negation normal form and is merely a generalization of the corresponding notion for $\mathcal{ALC}\mu$, introduced in Chapter 2, page 36.

Definition 36. The function ***nnf*** maps concepts or roles of $\mathcal{U}\mu$ to concepts or roles of $\mathcal{U}\mu$. Applied to an arbitrary concept or role, *nnf* yields the concept or role obtained from the original one by repeatedly applying the following substitution rules to all subconcepts and subroles until no rule is applicable any more:

$$
\begin{array}{rcl}
\neg\neg T & \leadsto & T, \\
\neg(T_1 \sqcap T_2) & \leadsto & \neg T_1 \sqcup \neg T_2, \\
\neg(T_1 \sqcup T_2) & \leadsto & \neg T_1 \sqcap \neg T_2, \\
\neg\exists^{\geq n} R{:}C & \leadsto & \exists^{\leq n-1} R{:}C, \\
\neg\exists^{\leq m} R{:}C & \leadsto & \exists^{\geq m+1} R{:}C, \\
\neg(R \circ S) & \leadsto & \neg R \oplus \neg S, \\
\neg(R \oplus S) & \leadsto & \neg R \circ \neg S, \\
\neg(R^{-1}) & \leadsto & (\neg R)^{-1}, \\
\neg(C \times D) & \leadsto & (\neg C \times \top) \sqcup (\top \times \neg D), \\
\neg\mu X.T & \leadsto & \nu X.\neg\tilde{T}, \\
\neg\nu X.T & \leadsto & \mu X.\neg\tilde{T}.
\end{array}
$$

In the last two substitution rules, $\tilde{T}$ denotes the concept or role obtained from T by simultaneously replacing each unbounded occurrence of X in T with $\neg X$. We call

the result of applying *nnf* to an arbitrary concept or role, T, of $\mathcal{U}\mu$, the **negation normal form** of T.

It should be clear that $\neg\tilde{T}$ is formally monotonic in X when T is also. Hence, $\mu X.\neg\tilde{T}$ and $\nu X.\neg\tilde{T}$ obey the syntactic restrictions imposed on least and greatest fixed-point operators of $\mathcal{U}\mu$ whenever $\mu X.T$ and $\nu X.T$ also do so.

Inspection of the substitution rules employed to obtain the negation normal form immediately reveals that the negation normal form of each concept or role of $\mathcal{U}\mu$ can be computed in deterministic polynomial time. In addition, the negation normal form is always linearly bounded in the length of the original concept or role. Another basic property of the negation normal form of $\mathcal{U}\mu$ is that it never involves any occurrence of a negation sign $\neg$ applied to a compound concept or role. However, its most important property is the following justifying the use of the term *normal form* after all.

Proposition 8. *Every concept and role of $\mathcal{U}\mu$ is equivalent to its negation normal form.*

Proof. As far as all but the last three substitution rules for obtaining the negation normal form are concerned, it should be obvious that they constitute *equivalence transformations*. That is, these substitution rules replace subconcepts and subroles by equivalent ones. In order to show that this is also the case for the substitution rule dealing with roles of the form $\neg(C\times D)$, it has to be verified whether $\neg(C\times D)$ is actually equivalent to $(\neg C\times\top)\sqcup(\top\times\neg D)$, no matter which particular concepts of $\mathcal{U}\mu$ are denoted by C and D. The following line of reasoning proves that this is actually the case:

$$\begin{aligned}
\neg(C\times D)^{\mathcal{I}} &= (\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}})\setminus(C^{\mathcal{I}}\times D^{\mathcal{I}})\\
&= (\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}})\setminus\{\langle d,e\rangle\in\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}} : d\in C^{\mathcal{I}},\ e\in D^{\mathcal{I}}\}\\
&= \{\langle d,e\rangle\in\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}} : d\notin C^{\mathcal{I}}\ \textit{or}\ e\notin D^{\mathcal{I}}\}\\
&= \{\langle d,e\rangle\in\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}} : d\in\neg C^{\mathcal{I}}\ \textit{or}\ e\in\neg D^{\mathcal{I}}\}\\
&= \{\langle d,e\rangle\in\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}} : d\in\neg C^{\mathcal{I}}\}\cup\{\langle d,e\rangle\in\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}} : e\in\neg D^{\mathcal{I}}\}\\
&= (\neg C^{\mathcal{I}}\times\Delta^{\mathcal{I}})\cup(\Delta^{\mathcal{I}}\times\neg D^{\mathcal{I}})\\
&= ((\neg C\times\top)\sqcup(\top\times\neg D))^{\mathcal{I}}.
\end{aligned}$$

In the case of the next to the last substitution rule, the proof proceeds as follows. We assume as usual that Γ is either $\Delta^{\mathcal{I}}$ or $\Delta^{\mathcal{I}}\times\Delta^{\mathcal{I}}$, dependent on whether $\mu X.T$ is a concept or role. Moreover, we adopt the convention that f_T (or $f_{\neg\tilde{T}}$) denotes the function on X induced by T (or $\neg\tilde{T}$) and $\mathcal{V}$:

$$(\neg\mu X.T)^{\mathcal{I}} = \Gamma\setminus\bigcap\{S\subseteq\Gamma : f_T(S)\subseteq S\}$$

$$
\begin{aligned}
&= \bigcup\{\overline{S} \subseteq \Gamma : f_T(\Gamma \setminus \overline{S}) \subseteq \Gamma \setminus \overline{S}\} \\
&= \bigcup\{\overline{S} \subseteq \Gamma : \Gamma \setminus f_T(\Gamma \setminus \overline{S}) \supseteq \overline{S}\} \\
&= \bigcup\{\overline{S} \subseteq \Gamma : f_{\neg\tilde{T}}(\overline{S}) \supseteq \overline{S}\} \\
&= (\nu X.\neg\tilde{T})^{\mathcal{I}}.
\end{aligned}
$$

The fact that $\neg\nu X.T$ is equivalent to $\mu X.\neg\tilde{T}$ can be shown analogously. □

3.6 Discussion

With $\mathcal{U}\mu$ we have a description logic that is capable of defining nontrivial concepts frequently emerging in artificial intelligence and computer science. These concepts include lists, DAGs, trees, binary trees, and balanced binary trees, and we can even define those AND-OR graphs for which there exist at least one well-founded solution. The origin of $\mathcal{U}\mu$'s considerable expressive power is twofold. On the one hand, it encompasses all traditional concept and role-structuring primitives that are known from the literature. This part is covered by Patel-Schneider's universal description logic $\mathcal{U}$. On the other hand, $\mathcal{U}\mu$ incorporates a general means of recursion which provides not only for recursive definitions of concepts, but also for recursive definitions of roles. This part of $\mathcal{U}\mu$ is covered by least and greatest fixed-point operators, which can be viewed to be more procedural in flavor.

The only thing that $\mathcal{U}\mu$ seems to lack is a means of mutual recursion as included in $\mathcal{ALC}\mu$. However, it is important to recall that in Chapter 2 it was established that mutual recursion can be captured by nested fixed-point operators, which *are* admissible in $\mathcal{U}\mu$. As a matter of fact, $\mathcal{U}\mu$ is not only able to capture mutual recursion among concepts, but also among roles, and even among concepts and roles in a mixed fashion. To illustrate how this can be realized, suppose we want to define the universal concept $\top$ as well as the universal role $\top\times\top$ by mutual recursion. This can be done by defining the universal role as $\mathsf{UnivConcept}\times\mathsf{UnivConcept}$ and to define $\mathsf{UnivConcept}$ as the projection of the universal role to its, say, first component. One way to capture the latter definition is to require that $\mathsf{UnivConcept}$ is the set of all those objects, d, for which there exists at least one object, e, such that the ordered pair $\langle d, e\rangle$ is an instance of the universal role. Because of the fact that the object e is not restricted and because $\mathsf{UnivConcept}$ is supposed to include all objects, e can be required to be an instance of this concept. One possible representation of the projection of the universal role to its first component is therefore $\exists\mathsf{UnivRole{:}UnivConcept}$. The whole mutual recursion is captured by the following terminology of $\mathcal{U}$:

$$
\begin{aligned}
\mathsf{UnivRole} &\doteq \mathsf{UnivConcept}\times\mathsf{UnivConcept}, \\
\mathsf{UnivConcept} &\doteq \exists\mathsf{UnivRole{:}UnivConcept}.
\end{aligned}
$$

Of course, this terminology must be given a greatest fixed-point semantics in that

we are interested only in those models which correspond to solutions which are maximal with respect to the subset relation rather than minimal. We can obviously replace throughout the right-hand sides of the concept and role introductions UnivRole and UnivConcept with their defining parts, that is, UnivConcept$\times$UnivConcept and $\exists$UnivRole:UnivConcept respectively. The resulting terminology, then, looks as follows:

$$\begin{aligned} \mathsf{UnivRole} &\doteq (\exists \mathsf{UnivRole}{:}\mathsf{UnivConcept}) \times (\exists \mathsf{UnivRole}{:}\mathsf{UnivConcept}), \\ \mathsf{UnivConcept} &\doteq \exists (\mathsf{UnivConcept} \times \mathsf{UnivConcept}){:}\mathsf{UnivConcept}. \end{aligned}$$

The models of this terminology coincide clearly with those of the original. But then we have already eliminated mutual recursion. So it just remains to recast the greatest fixed-point semantics in terms of greatest fixed-point operators, resulting in the following acyclic terminology:

$$\begin{aligned} \mathsf{UnivRole} &\doteq \nu X.\Big(\big(\exists X{:}\nu Y.\exists (Y \times Y){:}Y\big) \times \big(\exists X{:}\nu Y.\exists (Y \times Y){:}Y\big)\Big), \\ \mathsf{UnivConcept} &\doteq \nu Y.\exists (Y \times Y){:}Y. \end{aligned}$$

This example shows how even mutual recursion among concepts and roles can be captured in $\mathcal{U}\mu$ by means of nested fixed-point operators. In this particular case, there are actually no nested fixed operators needed at all because $\nu Y.\exists (Y \times Y){:}Y$ occurring in the scope of $\nu X.$ does not involve any occurrence of the role variable X, so that we can take the following terminology instead of the above:

$$\begin{aligned} \mathsf{UnivRole} &\doteq \nu X.\Big((\exists X{:}\mathsf{UnivConcept}) \times (\exists X{:}\mathsf{UnivConcept})\Big), \\ \mathsf{UnivConcept} &\doteq \nu Y.\exists (Y \times Y){:}Y. \end{aligned}$$

It is well known that the expressive power of $\mathcal{U}$ can be gained only at the expense of losing decidability [Schild, 1988]. In this sense everything presented in this chapter would be only of theoretical interest if we were not be able to show how mechanized reasoning can take place in the framework of $\mathcal{U}\mu$. The next chapter is devoted to this very issue.

Chapter 4

$\mathcal{U}\mu$ as a Query Language for Knowledge & Data Bases

In the previous chapter the expressive power of description logics was extended such that complex data structures can be represented. This was accomplished, however, at the expense of losing the chance to mechanize the form of reasoning involved. But from the correspondence with the propositional μ-calculus, it is known that tractable reasoning is possible in provably intractable concept languages such as $\mathcal{ALC}\mu^-$. In particular, tractability can be achieved by guaranteeing that all primitive concepts and roles are extensionally specified by means of a finite semantic structure, where such a semantic structure can be viewed as a relational database. Hence this shows that $\mathcal{ALC}\mu^-$ can serve as a tractable query language for databases. But at least from a database point of view, this statement is of theoretical interest only because it seems to be far fetched to advocate a query language having an expressive power equal to that of $\mathcal{ALC}\mu^-$. The challenge that will be pursued in the present chapter is whether an analogous tractability result can be obtained for $\mathcal{U}\mu$. This question will be answered in the positive. From the knowledge representation point of view, however, it would be desirable to have more flexible databases allowing for some kind of incomplete knowledge as well. A uniform approach to knowledge and data bases will therefore be developed covering the whole range from relational databases to traditional knowledge bases. It will turn out that even in the case of a knowledge base which is unrestricted except for the fact that it must have a finite domain, a worst-case complexity can be attained which does not exceed that of the very weakest description logic's setting incorporating only acyclic terminologies. Technically speaking, in this case co-NP-completeness (and thus decidability) can be attained. In view of the undecidability of the original setting, this is a promising and surprising result.

4.1 Model Checking Versus Theorem Proving

At the end of the last chapter we were left with the dilemma that on the one hand we have with $\mathcal{U}\mu$ a powerful description logic, but on the other hand such high expressive power was gained at the expense of losing the possibility to mechanize reasoning. In particular, deciding subsumption between two concept or roles of $\mathcal{U}\mu$ is undecidable, because so is subsumption between two roles of $\mathcal{U}$, even with no terminology taken into account [Schild, 1988]. Even in very small sublanguages of $\mathcal{U}\mu$ and $\mathcal{U}$, subsumption with respect to empty terminologies is known to be undecidable too. This applies, for instance, to a sublanguage of $\mathcal{U}$ considered in [Schild, 1991a] which augments the regular extension of $\mathcal{ALC}$ by the inverse of roles as well as functional roles, where functional roles can be simulated using number restrictions of the form $\exists^{\leq 1}$ [Schild, 1991a, Proposition 9]. The same applies to sublanguages of $\mathcal{U}$ being part of that description logic attributed to the very system which has attracted most attention in the realm of terminological knowledge representation, viz. the KL-ONE-system. The corresponding undecidability proofs are due to Patel-Schneider [1989b] and Schmidt-Schauss [1989]. Notably, the sublanguage for which Schmidt-Schauss established undecidability of subsumption between two concepts includes in addition to the concept-structuring primitives of the very weakest description logic just role-value maps and composition of roles.

In a sense things become even worse when terminologies are taken into account. In Chapter 2, for instance, it is proved that in case of the standard concept language $\mathcal{ALC}$, every algorithm capable of deciding whether or not one concept subsumes another uses exponential time on infinitely many inputs if there is at least one recursive concept introduction. Notably, this result holds no matter which of the usual kinds of semantics for recursive concept introductions is presupposed, viz. either descriptive semantics or least or greatest fixed-point semantics. It is also known that in case of the very weakest description logic (comprising no concept and role-structuring primitives other than concept conjunction and universal quantification over role names), there exists no polynomial-time algorithm to decide, with respect to acyclic terminologies, whether or not one concept subsumes another, unless P = NP [Nebel, 1990b]. As Woods and Schmolze [1992] put it, "the surfeit of intractability results seems to have reached its logical end with the conclusion that practically everything of any use is intractable (in the worst case)."

Recently, Halpern and Vardi [1991] proposed a possible solution to this problem: As a starting point, they re-examined the traditional approach to knowledge representation, going back to McCarthy [1968]. According to McCarthy the world to be modeled should be represented by a finite collection of formulae drawn from some given logic, preferably first-order logic. If a query to be answered is then formulated within the same logic, the answer depends on whether or not this formula is a *logical consequence* of the given collection of formulae. In other words, a check

$$\left\{\begin{array}{l} \forall x.block(x) \Leftrightarrow x = a \vee x = b, \\ a \neq b, a \neq table, b \neq table, \\ \forall x \forall y.on(x,y) \\ \Leftrightarrow \quad (x = a \wedge y = b) \\ \vee \quad (x = b \wedge y = table) \end{array}\right\} \stackrel{?}{\models} \quad block(b) \wedge \neg \exists x.block(x) \wedge on(x,b)$$

Figure 4.1: Representing the sample blocks world by first-order formulae

is made whether *every* semantic structure which is a model of each of the formulae representing the world is also a model of the query.

We shall illustrate this traditional approach to knowledge representation by an example borrowed from the blocks world. Suppose, we want to represent the following blocks world and, then, would like to know whether b is a top block or not:

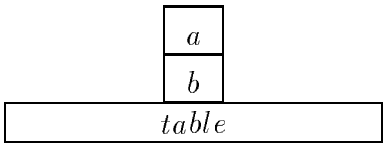


Figure 4.1 gives a representation of this situation in terms of first-order logic in the spirit of McCarthy. Halpern and Vardi pointed out that such a formalization gives rise to the problem that the need to represent all facts about the world in terms of some logic necessitates the use of very expressive logics such as full first-order logic. One can add to this argument that not only all things that hold in the given world have to be made explicit, but also all those things which do *not* hold. This is, of course, an immediate consequence of the so-called *open-world assumption*, which is at the very heart of McCarthy's approach. In contrast to the closed-world assumption, the open-world assumption does *not* assume that all those facts that are not explicitly mentioned (or that cannot be inferred) are considered to be false. For instance, it obviously would not suffice to take just the formulae $on(a,b)$ and $on(b,table)$ instead of the more clumsy formula for on given in Figure 4.1. In fact, in addition to $on(a,b)$ and $on(b,table)$ it has to be stated that these are the only instances for which the two-place predicate symbol on is true. A predicate completion of the following form expresses this fact:

$$\forall x \forall y.on(x,y) \Rightarrow \Big((x = a \wedge y = b) \vee (x = b \wedge y = table)\Big).$$

It should be clear that the formula for on given in Figure 4.1 just combines this formula with $on(a,b)$ and $on(b,table)$ because the conjunction of the latter two results in a formula equivalent to the following one, which represents the opposite direction of predicate completion given above:

$$\forall x \forall y.\Big((x = a \wedge y = b) \vee (x = b \wedge y = table)\Big) \Rightarrow on(x,y).$$

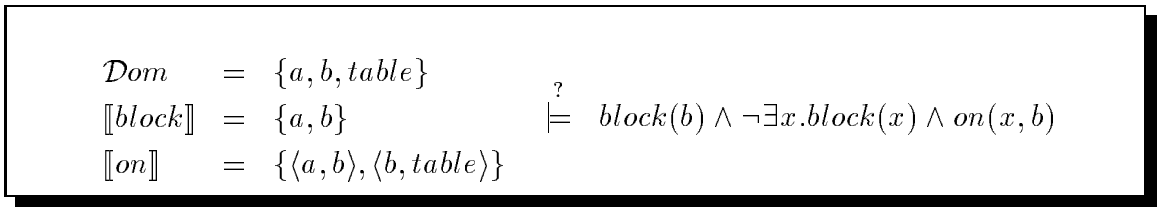


Figure 4.2: Representing the sample blocks world by a semantic structure

It seems to be the case that the use of very expressive logics is particularly necessitated by the need to make also those things explicit which do not hold. But once a powerful logic such as full first-order logic is chosen as a representation language, difficulties arise immediately, not the least is its undecidability, which is the case even with only finite interpretation domains taken into consideration [Trahtenbrot, 1963].

At this point Halpern and Vardi [1991] put forward that in many cases the natural representation of a world to be modeled is a *semantic structure* rather than a collection of formulae. If, as in the traditional approach, queries are represented by single formulae of a given logic, queries can still be decided, but in this case on the ground of whether or not the query is true in the *given* semantic structure. This is to say, in this case it is sufficient to check whether or not the given semantic structure is a model of the formula corresponding to the query. The fact that a (closed) formula α is true in a semantic structure $\mathcal{M}$ is usually indicated by $\mathcal{M} \models \alpha$. Resorting to this convention, Figure 4.2 gives such an alternative representation of the blocks world considered above.

In many cases, model checking has tremendous benefits compared with troublesome theorem proving, at least in terms of computational complexity. For instance, checking the truth of an arbitrary closed first-order formula, α, in a finite semantic structure which fixes the interpretation of all predicate symbols and constants occurring in α is known to be *decidable* in polynomial space, by Theorem 6 of [Chandra and Merlin, 1977].[1] Recall that in contrast to this, there exists no algorithm at all capable of deciding in each particular case whether a formula of this kind is a logical consequence of a finite set of first-order formulae, even with only finite interpretation domains taken into account. On the other hand, it is also known that first-order model checking is still at least as hard as *any other* problem solvable using polynomial space, hence this problem is still very hard [Chandra and Merlin, 1977, Theorem 6]. Restricting the syntactic shape of first-order formulae may not always help in this respect either. This applies, for instance, to the well-known *conjunctive Boolean queries*, i.e., first-order formulae of the form $\exists X_1 ... \exists X_n.\alpha_1 \wedge ... \wedge \alpha_m$, where the α_i's are restricted to be positive literals, all whose variables are among $X_1, ..., X_n$. In fact, the problem of checking the truth of conjunctive Boolean queries in a finite semantic structure is still as hard as any other problem solvable in nondeterministic polyno-

[1] To be accurate, α must not involve any function symbol other than constants.

mial time [Chandra and Merlin, 1977, Theorem 7]. Anyway, Halpern and Vardi's intention was to forge a new approach to knowledge representation rather than to give concrete instances.

It should be clear that terminological knowledge representation as it stands is committed to the traditional theorem-proving approach rather than to model checking, although theorem proving takes place in case of terminological reasoning in the setting of restricted languages. Let us exemplify this fact by the blocks world considered above. At first, the terminology inherent in the blocks world domain should be fixed. For the present purpose, it is sufficient to treat the class of all blocks as well as the binary relation "lying on" as not further specified. But then they can be represented by concept and role names, such as Block and on. The only relevant term to be defined in this domain is that of a top block. It can be characterized as a block on which there is no block lying. This definition can easily be captured in $\mathcal{ALC}$ along with the inverse of a role:

$$\mathsf{TopBlock} \doteq \mathsf{Block} \sqcap \neg\exists \mathsf{on}^{-1}{:}\mathsf{Block}.$$

The usual procedure undertaken then is that the concrete world at hand is described by providing concepts and roles with particular instances supposed to reflect the situation in the world. This is done by means of so-called *assertions*, each of which assigns either a particular individual name with a concept or else a particular ordered pair of individual names with a role. The assertions describing our example blocks world look as follows:

$$\begin{aligned}&\mathsf{a}{:}\mathsf{Block},\\ &\mathsf{b}{:}\mathsf{Block},\\ &\mathsf{table}{:}\neg\mathsf{Block},\\ &(\mathsf{a},\mathsf{b}){:}\mathsf{on},\\ &(\mathsf{b},\mathsf{table}){:}\mathsf{on}.\end{aligned}$$

Assertions of this kind are given meaning by defining an arbitrary interpretation, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$, as a model of $a{:}C$ if and only if $a^{\mathcal{I}} \in C^{\mathcal{I}}$. It is moreover a model of $\langle a, b\rangle{:}R$ if and only if $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$. Here, $a^{\mathcal{I}}$ and $b^{\mathcal{I}}$ are assumed to denote the single elements of $\Delta^{\mathcal{I}}$ which are contained in $\mathcal{V}(\{a\})$ and $\mathcal{V}(\{b\})$ respectively. It is not unusual to impose a general unique-name assumption upon individual names, meaning that $a^{\mathcal{I}}$ and $b^{\mathcal{I}}$ denote distinct elements of the domain whenever a and b are distinct individual names. A finite set of assertions then constitutes a so-called *knowledge base*. It should not be surprising that a model of such a knowledge base is an interpretation which is a model of each of the assertions of the knowledge base. A *query*, which can either be an assertion or an axiom, is then said to hold in such a knowledge base with respect to a terminology if and only if every interpretation which is a model of the knowledge base and the terminology is also a model of the query. This just defines the notion of logical consequence within this specific framework.

$$\left\{\begin{array}{l} \mathsf{a{:}Block}, \mathsf{b{:}Block}, \mathsf{table{:}}\neg\mathsf{Block}, \\ (\mathsf{a},\mathsf{b}){:}\mathsf{on}, (\mathsf{b},\mathsf{table}){:}\mathsf{on}, \\ \mathsf{a{:}}\exists^{\leq 0}\mathsf{on}^{-1}, \mathsf{a{:}}\exists^{\leq 1}\mathsf{on}, \\ \mathsf{b{:}}\exists^{\leq 1}\mathsf{on}^{-1}, \mathsf{b{:}}\exists^{\leq 1}\mathsf{on}, \\ \mathsf{table{:}}\exists^{\leq 1}\mathsf{on}^{-1} \end{array}\right\} \quad \stackrel{?}{\models_{\mathcal{T}}} \quad \mathsf{b{:}TopBlock}$$

$$\mathcal{T} = \{\mathsf{TopBlock} \doteq \mathsf{Block} \sqcap \neg\exists\mathsf{on}^{-1}{:}\mathsf{Block}\}$$

Figure 4.3: Representing the sample blocks world by a knowledge base

Note that this means particularly that knowledge bases are to be interpreted under an open-world assumption. It is for this reason why the knowledge base given above is *incomplete* in the sense that it solely states that block a lies on block b, while the latter in turn lies on the table, but it is left open whether there is any other block lying on a, b, or on the table. A knowledge base which is in this sense complete is depicted in Figure 4.3.

Recall that in general inferences of the kind as shown in Figure 4.3 are tractable only if P = NP because the same applies to subsumption in the very weakest description logic, even with only acyclic terminologies are taken into account. In the presence of at least one recursive concept introduction, inferences of the kind shown in Figure 4.3 are even provably intractable, at least in the general case.

Therefore, it could be promising to accommodate Halpern and Vardi's model checking approach to the case of terminological reasoning, and the present chapter will be devoted to this. It will turn out that even in the case of a slightly restricted version of $\mathcal{U}\mu$, answering queries become tractable just by replacing the usual kind of knowledge bases with single semantic structures. This semantic structure has to fix not only a finite domain, but also the interpretation of all *primitive* concepts and roles (i.e., those concept and role names which are mentioned somewhere in the terminology or in the query, but which are not defined).

Figure 4.4 modifies the traditional representation just considered in the spirit of the model checking approach. A finite semantic structure is shown there, which fixes the interpretation of each primitive concept and role of $\mathcal{T}$, that is, it fixes the interpretation of Block and on.

The syntactic representation of such a semantic structure is called a *vivid* or *physical knowledge base*, emphasizing the fact that it is intended to replace customary knowledge bases. Such a knowledge base consists essentially of two parts. One component consists of a set, $\mathcal{D}$, of individual names, supposed to give an upper bound for the possible interpretation domains of models. The second consists of a finite set of assertions of the form $TN \doteq \mathcal{S}$, where $\mathcal{S}$ is a unary relation over $\mathcal{D}$ if TN is a

$$\begin{array}{lcl} \mathcal{D}om & = & \{\mathsf{a},\mathsf{b},\mathsf{table}\} \\ [\![\mathsf{Block}]\!] & = & \{\mathsf{a},\mathsf{b}\} \\ [\![\mathsf{on}]\!] & = & \{\langle \mathsf{a},\mathsf{b}\rangle,\langle \mathsf{b},\mathsf{table}\rangle\} \\ \mathcal{T} = \{\mathsf{TopBlock} \doteq \mathsf{Block} \sqcap \neg\exists \mathsf{on}^{-1}{:}\mathsf{Block}\} & & \end{array} \qquad \stackrel{?}{\models_{\mathcal{T}}} \quad \mathsf{b{:}TopBlock}$$

Figure 4.4: Representing the sample blocks world by a vivid knowledge base

concept name, and a binary relation over $\mathcal{D}$ if TN is a role name. An interpretation, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$, is a model of such an assertion just in case $TN^{\mathcal{I}}$ equals $\{a^{\mathcal{I}} : a \in \mathcal{S}\}$ and $\{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle : \langle a, b\rangle \in \mathcal{S}\}$, respectively, depending on whether TN is a concept or a role name. Compared with customary knowledge bases, the models a knowledge base of the kind just described have to meet is the following additional condition: If d is any element of $\Delta^{\mathcal{I}}$, then it has to have at least one syntactic counterpart in $\mathcal{D}$, that is, there must be at least one $a \in \mathcal{D}$ such that $a^{\mathcal{I}} = d$. The definition of $\mathcal{KB} \models_{\mathcal{T}} Q$ then can remain unchanged.

One of the main results of the present chapter is that, even in $\mathcal{U}\mu$ $\mathcal{KB} \models_{\mathcal{T}} Q$ can be decided in deterministic polynomial time in the sense of the Vardi's [1982] notion of *combined complexity*, i.e., it can be decided in deterministic time polynomial in the sum of the sizes of $\mathcal{KB}$, $\mathcal{T}$, and Q. For this positive result to hold, however, some restrictions must be invoked: First of all, we have to restrict the syntax of $\mathcal{U}\mu$ slightly, but this restriction concerns solely the possible occurrences of nested *alternating* fixed-point operators. The restriction does not affect those nested fixed-point operators which are of practical use. In particular, all terminologies given in the last chapter do not extend beyond the restricted version of $\mathcal{U}\mu$. The corresponding restricted version of $\mathcal{U}\mu$ is called $\mathcal{U}\mu^{-}$.

More importantly, the tractability result additionally presupposes that each of the following three conditions is met.

(a) The knowledge base $\mathcal{KB}$ fixes an interpretation domain, say, $\mathcal{D}$, consisting of a finite number of individual names with a general unique-name assumption imposed on.

(b) At least all those concept and role names are specified by means of a unary or binary relation over $\mathcal{D}$ which occur in Q or $\mathcal{T}$, except for those which are defined in $\mathcal{T}$. In addition, $\mathcal{D}$ has to contain at least all those individual names which occur in $\mathcal{T}$ or Q.

(c) The terminology $\mathcal{T}$ is acyclic.

Each of the conditions above calls for comment: First, condition (c) does not exclude

recursive concept or role definitions because the fixed-point operators of $\mathcal{U}\mu^-$ admit of encoding them by acyclic, nonrecursive concept or role introductions, at least as far as least and greatest fixed-point semantics is concerned. Anyway, the most important conditions are the first two ensuring all primitive concepts and roles to be specified extensionally by means of a unary or binary relation. This restriction does make sense since these concepts and roles are exactly those which are not further specified according to the denotational semantics or the terminology. It can easily be verified that the sample query of Figure 4.4 obeys each of the conditions above.

This result is significant both from Halpern and Vardi's [1991] model checking point of view and from the viewpoint of description logics. From Halpern and Vardi's point of view, our result singles out a useful fragment of first-order logic which gives rise to tractable model checking, even when this fragment is enriched by fixed-point operators. As opposed to the corresponding tractability results for fixed-point languages based on full first-order logic [Vardi, 1982], ours is to be understood in terms of *combined complexity* rather than the far weaker notion of *data complexity*. The crucial difference between these two different complexity measures is that in contrast to combined complexity, data complexity presupposes an arbitrary but *fixed* query and in our case also an arbitrary but *fixed* terminology. The difference between these two complexity measures is perhaps best explained by means of an example. Suppose that we came up, for instance, with a upper bound of $|\mathcal{KB}|^{O(|\mathcal{T}|+|Q|)}$. This bound is not only exponential, but even *superexponential* in terms of combined complexity. In the sense of data complexity, however, this would yet be a *polynomial* upper bound. In fact, for *fixed* $\mathcal{T}$'s and Q's, $|\mathcal{KB}|^{O(|\mathcal{T}|+|Q|)}$ constitutes a polynomial bound. It is for this reason that if our tractability result is to be understood in the sense of data complexity, it may be possible for it still to be an exponential lower bound in the sense of the combined complexity. This actually happens to be the case for the fixed-point languages based on full first-order logic investigated by [Vardi, 1982]. Anyway, it should be obvious that the notion of data complexity is not tailored for terminological reasoning because it seems far fetched to presuppose that terminological reasoning takes place with some fixed terminology or even with some fixed query.

The description logic's point of view suggests the following interpretation: If we are able to abandon any incomplete knowledge from our knowledge base, we gain not only decidability, but also tractability, even in case of the most powerful description logic considered in the literature. Thus our result suggests that the main source of computational complexity of terminological reasoning is the ability to express incomplete knowledge. This finding was confirmed by another result which will be presented in this chapter too. It explores the consequences of a limited form of incomplete knowledge by 'unknown' individual names corresponding to Reiter's [1984] null values in databases. The result states that admitting of individual names without any general unique name assumption imposed on causes intractability, unless P = NP, even in case of the standard description logic $\mathcal{ALC}$. Nevertheless, based on Vardi's [1986] work on null values in databases, we shall be in a position to give

an algorithm capable of dealing with such 'null values' soundly if approximately. The resulting algorithm runs not only in polynomial time, but turns out to be also *complete* in case no such 'null value' is present.

Notice that any finite semantic structure can be viewed as a relational database, and so can any vivid knowledge base. From this point of view, one interpretation of our tractability result is that $\mathcal{U}\mu^-$ can serve as a powerful, but tractable query language for databases. This is the reason why we shall compare the query power of $\mathcal{U}\mu^-$ with traditional database query languages, especially with that of fixed-point languages based on full predicate calculus, which are known to be intractable in the sense of combined complexity [Vardi, 1982]).

However, we shall not argue that model checking can replace ordinary terminological reasoning. On the contrary, we believe that one should try to mediate between two extremes on a wide-range spectrum between abandoning any kind of incomplete knowledge on the one hand and allowing for an unlimited use of incomplete knowledge on the other. One possible compromise in this direction is indicated by another outcome of the present chapter. In particular, it turned out that even when relaxing conditions (a) and (b) in such a way that $\mathcal{KB}$ is solely required to fix a domain consisting of a finite number of individual names, the problem of deciding $\mathcal{KB} \models_{\mathcal{T}} Q$ is still decidable in $\mathcal{U}\mu^-$. We shall prove that in this case the computational complexity is essentially the same as that of deciding ordinary subsumption between two concepts with respect to acyclic terminologies in the *weakest* concept language. Technically speaking, in this case deciding $\mathcal{KB} \models_{\mathcal{T}} Q$ in $\mathcal{U}\mu^-$ is co-NP-complete. It will turn out that the same applies to general terminological reasoning in $\mathcal{U}\mu^-$ if a simple domain-closure axiom is put into effect.

4.2 A Uniform Approach to Knowledge & Data Bases

This section is devoted to knowledge bases as a means of describing application domains by providing generic concepts and roles with specific instances. Special attention will be paid to knowledge bases which can be called in Levesque's [1986] sense *vivid*. According to Levesque vivid knowledge bases are those whose models have the same structure as the knowledge bases themselves.

In the realm of description logics, *knowledge bases* are finite sets of so-called *assertions*, each of which associates either a particular individual name with some concept or an ordered pair of individual names with some role. Typical assertions of this kind are, for instance, a:Block and $\langle a, b\rangle$:on. As we have not only knowledge bases, but also terminologies at our disposal, we can assume without loss of generality that those concepts and roles out of which assertions can be formed are simple concept

or role names. This is because any other concept and role can be replaced by a new concept or role name if an appropriate concept or role introduction is added. For instance, in the presence of the concept introduction $\mathsf{TopBlock} \doteq \mathsf{Block} \sqcap \neg\exists \mathsf{on}^{-1}{:}\mathsf{Block}$, $a{:}\mathsf{TopBlock}$ expresses exactly the same assertion as $a{:}(\mathsf{Block} \sqcap \neg\exists \mathsf{on}^{-1}{:}\mathsf{Block})$. Knowledge bases are usually reduced to finite sets of simple assertions of this sort. In our account of knowledge bases, however, we shall adopt several deviations from this prevailing view. The nonstandard view of knowledge bases is needed if traditional databases are to be covered in that framework too.

- First, we do not implicitly impose any general unique-name assumption on individual names. Instead, explicit *uniqueness axioms* of the form $a \neq b$ can be stated whenever it is convenient to do so. Such a uniqueness axiom enables us to impose the restriction on two individual names that they always denote *distinct* objects of the domain just by need.

- In addition to common assertions of the form $a{:}C$ and $\langle a, b\rangle{:}R$ stipulating specific instances of C and R, we introduce assertions enumerating *all* their instances. Typical examples of assertions of this kind are $\mathsf{Block} \doteq \{a, b, table\}$ as well as $\mathsf{on} \doteq \{\langle a, b\rangle, \langle b, table\rangle\}$. Both kinds of these new assertions make it possible to succinctly express complete knowledge about the extensions of concepts and roles whenever it is convenient to do so. In presence of such assertions, we say that the corresponding concepts and roles are *specified extensionally.*

- Usually the set of concept and role names out of which assertions can be formed is left implicit. In our account of knowledge bases, however, this set is made explicit and is usually denoted by $\mathcal{S}ig$. We thereby fix the set of those concept and role names we are interested in.

- The most important deviation from the traditional view of knowledge bases predominating in description logics, however, is the fact that in our account each knowledge base fixes a certain set of individual names, usually denoted by the letter $\mathcal{D}$. This set is supposed to limit not only those individual names about which assertions can be made, but also the possible interpretation domains of models. This is because each domain element is assumed to have at least one syntactic counterpart in $\mathcal{D}$. The effect of this restriction is that of a domain-closure axiom, although domain-closure axioms are more restrictive. Of course, an explicit domain-closure axiom can be stated only in those cases in which $\mathcal{D}$ contains solely a finite number of individual names, say, a_1,...,a_n. In such a case, we can retreat to an axiom of the form $\top \sqsubseteq \{a_1\} \sqcup ... \sqcup \{a_n\}$. On the other hand, such a representation is impossible if $\mathcal{D}$ contains an infinite number of individual names. Anyway, in case $\mathcal{D}$ is the set of *all* individual names (of which there are countably many), the restriction that all elements of the interpretation domain have to have at least one syntactic counterpart in $\mathcal{D}$ can clearly be ignored. In this case, only those domains are excluded which

are not countable. This restriction does not bear any practical importance in most applications.

The latter deviation from the common view of knowledge bases predominating in description logics is indispensable for restricting knowledge bases in such a way that they become vivid. According to [Levesque, 1986], page 93, *vivid* knowledge bases are exactly those for which there exists always a one-to-one correspondence with their models, at least as far as the relevant parts of those models are concerned. In particular, there has to be a one-to-one correspondence between a certain class of symbols of the knowledge base and the interpretation domain of the model. In addition, for every simple relationship of interest, there has to be a type of connection among symbols in the knowledge base such that the relationship holds among a group of objects in the model if and only if the corresponding connection holds among the corresponding symbols in the knowledge base. Applied to knowledge bases of the particular sort discussed so far, this means that every model, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$, of such a vivid knowledge base has to satisfy the following two criteria. First, there must be a one-to-one correspondence between $\mathcal{D}$ and the domain $\Delta^{\mathcal{I}}$. In addition, for every concept, C, of $\mathcal{S}ig$, it must be the case that an element of $\Delta^{\mathcal{I}}$ is a member of $C^{\mathcal{I}}$ just in case the knowledge base asserts the corresponding individual name to be an instance of C. This is to say, the knowledge base contains the assertion $a{:}C$. An analogous relationship must hold, of course, for all roles of $\mathcal{S}ig$ too. Because in our account of knowledge bases each object of the domain has to have a syntactic counterpart in $\mathcal{D}$, the first criterion can be achieved just by requiring that for each distinct pair of individual names of $\mathcal{D}$, there is at least one uniqueness axiom. Only a finite number of such uniqueness axioms is actually needed whenever $\mathcal{D}$ is finite. One way to accomplish the second criterion is to force the knowledge base to contain exactly one of the two assertions $a{:}CN$ and $a{:}\neg CN$, for each individual name, a, of $\mathcal{D}$ and each concept name, CN, of $\mathcal{S}ig$, and to impose a corresponding restriction on all ordered pairs of individual names of $\mathcal{D}$ and all role names contained in $\mathcal{S}ig$ as well. A more succinct way is to specify all concept and role names of $\mathcal{S}ig$ extensionally.

The main reason to investigate such vivid knowledge bases is the fact that they can be viewed as a means of syntactically representing parts of interest of single interpretations or, which amounts to the same thing, as relational databases comprising relations at most of rank one or two. Hence, vivid knowledge bases provide a general framework for investigating not only the model checking approach to description logics, but also description logics as a new kind of query language for relational databases. Over and above that, our account of knowledge bases is flexible enough to handle a wide range of knowledge bases which are more general than the vivid ones: An example of such knowledge bases are those incorporating unknown values, also known as *null values* in database theory. This class of knowledge bases can be captured by relaxing the first criterion characterizing vivid ones, while still retaining the second. In this case, we just force knowledge bases to specify each concept and

role name of $\mathcal{S}ig$ extensionally. The presence of any uniqueness axiom is not generally presupposed, but is not forbidden either. Knowledge bases of this type will be called *closed*.

In any case, it will be important to impose a certain restriction on the lower bounds of $\mathcal{S}ig$ and $\mathcal{D}$. Both sets should always be large enough to enable knowledge bases to make assertions about all term names occurring in the terminology or query, except perhaps for those concept and role names which are defined in the terminology. In this case, the terminology and the query will be called *compatible* with the knowledge base. This restriction will be of particular significance both from the model checking as well as the query language point of view.

In the remainder of this section, we shall develop all notions discussed so far, but elucidate also the power of null values by an example. The example tackles the chromatic number of a given graph, that is, the problem whether for a given number of k colors, all vertices of the graph can be colored in such a way that each vertex is colored with exactly one of the k different colors and, moreover, every two vertices connected by at least one edge do not have the same color.

4.2.1 Finite, Closed, & Vivid Knowledge Bases

Definition 37. Assume $\mathcal{L}$ is a set of concepts and roles, while $\mathcal{D}$ is a set of individual names. If C is a concept of $\mathcal{L}$, R is a role of $\mathcal{L}$, a and b are both individual names of $\mathcal{D}$, $\mathcal{S}_1$ is a set of individual names of $\mathcal{D}$, and $\mathcal{S}_2$ is a set of ordered pairs of individual names of $\mathcal{D}$, then $\boldsymbol{a}{:}\boldsymbol{C}$, $\langle \boldsymbol{a},\boldsymbol{b}\rangle{:}\boldsymbol{R}$, $\boldsymbol{C \doteq \mathcal{S}_1}$, as well as $\boldsymbol{R \doteq \mathcal{S}_2}$ are all said to be **$\mathcal{L}$-assertions over $\mathcal{D}$**.

We may speak simply of *$\mathcal{L}$-assertions* whenever $\mathcal{D}$ is understood, and may even use the term *assertion* alone whenever $\mathcal{L}$ is understood as well.

For convenience, it is presupposed from now on that for every interpretation, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$, and every individual name, a, the expression $\boldsymbol{a^{\mathcal{I}}}$ denotes exactly that element of $\Delta^{\mathcal{I}}$ which is the only member of the singleton subset of $\Delta^{\mathcal{I}}$ denoted by $\mathcal{V}(\{a\})$. This means that $\mathcal{V}(\{a\})$ is always $\{a^{\mathcal{I}}\}$.

Definition 38. Assume $a{:}C$, $\langle a,b\rangle{:}R$, $C \doteq \mathcal{S}_1$, and $R \doteq \mathcal{S}_2$ are arbitrary assertions, while $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$ is any interpretation. We define the restrictions to be imposed on $\mathcal{I}$ to be a model of the assertion as follows.

- $\mathcal{I}$ is a **model of $\boldsymbol{a{:}C}$** if and only if $a^{\mathcal{I}} \in C^{\mathcal{I}}$.
- $\mathcal{I}$ is a **model of $\boldsymbol{\langle a,b\rangle{:}R}$** if and only if $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$.
- $\mathcal{I}$ is a **model of $\boldsymbol{C \doteq \mathcal{S}_1}$** if and only if $C^{\mathcal{I}} = \{a^{\mathcal{I}} : a \in \mathcal{S}_1\}$.

- $\mathcal{I}$ is a **model of** $\boldsymbol{R \doteq \mathcal{S}_2}$ if and only if $R^{\mathcal{I}} = \{\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle : \langle a, b\rangle \in \mathcal{S}_2\}$.

Definition 39. If $\mathcal{D}$ is a set of individual names and a and b are two elements of this set such that $a \neq b$, then the assertion $\langle a, b\rangle{:}\neg\epsilon$ is said to be a **uniqueness axiom** over $\mathcal{D}$. For the sake of readability, we may use $\boldsymbol{a \neq\!\!\!\doteq b}$ in lieu of $\langle a, b\rangle{:}\neg\epsilon$.

Note that an arbitrary interpretation, whose interpretation function is $.^{\mathcal{I}}$, is a model of such a uniqueness axiom if and only if $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

Definition 40. Assume $\mathcal{S}ig$ is a nonempty finite set of concept and role names, while $\mathcal{D}$ is a nonempty set of individual names. Suppose, moreover, $\mathcal{A}$ is a set of $\mathcal{S}ig$-assertions over $\mathcal{D}$ which may contain, of course, also uniqueness axioms over $\mathcal{D}$. Then the triple $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$ is said to be a **knowledge base**.

Definition 41. Assume $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$ is an arbitrary interpretation and $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$ is a knowledge base. Then $\mathcal{I}$ is said to be a **model of** $\boldsymbol{\mathcal{KB}}$ if and only if it is a model of each element of $\mathcal{A}$ and, moreover, $\Delta^{\mathcal{I}}$ is a nonempty subset of $\{a^{\mathcal{I}} : a \in \mathcal{D}\}$.

This is to say, $\mathcal{I}$ is a model of a $\mathcal{KB}$ just in case $\mathcal{I}$ is a model of all assertions and uniqueness axioms contained in $\mathcal{A}$ and, in addition, for each $d \in \Delta^{\mathcal{I}}$, there is at least one individual name of $\mathcal{D}$, say, a, such that $a^{\mathcal{I}} = \{d\}$. Roughly speaking, this means that each semantic object of the domain of $\mathcal{I}$ has to have a syntactic counterpart in $\mathcal{D}$.

Definition 42. Assume $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$ is an arbitrary knowledge base. Then $\mathcal{KB}$ is said to be **finite** if and only if $\mathcal{D}$ is finite. A concept or role, T, is **extensionally specified** in $\mathcal{KB}$ if and only if $\mathcal{A}$ contains exactly one $\{T\}$-assertion over $\mathcal{D}$ and this very assertion is of the form $T \doteq \mathcal{S}$ such that $\mathcal{S}$ is either a set of individuals of $\mathcal{D}$ or a set of ordered pairs of individuals of $\mathcal{D}$, depending on whether T is a concept or role. Furthermore, $\mathcal{KB}$ is said to be **closed** if and only if it is finite and all concept and role names of $\mathcal{S}ig$ are extensionally specified in $\mathcal{KB}$.

In view of this definition of a finite knowledge base, recall that $\mathcal{S}ig$ is a set of concept and role names which is always finite, regardless of whether the corresponding knowledge base is finite or not. It is for this reason that $\mathcal{A}$ is finite whenever the corresponding knowledge base is finite in the sense of the definition above.

The following fact concerning knowledge bases which are closed is due to the fact that in such a knowledge base all concept and role names of $\mathcal{S}ig$ are specified extensionally exactly once. It is therefore impossible to state any contradictory assertions. The

same applies to uniqueness axioms in that closed knowledge bases are able to state solely inequalities between different individual names with the help of uniqueness axioms, but it is impossible to stipulate any equalities by means of assertions of the form $\langle a, b\rangle{:}\epsilon$ or $\epsilon \doteq \{\langle a, b\rangle\}$ and alike.[2]

Fact 4. Each closed knowledge base has at least one model.

We can be more specific and fix a class of straightforward models of closed knowledge bases. This class of interpretations is characterized in such a way that their interpretation domains are identical to $\mathcal{D}$ and, moreover, all individual concepts are mapped to themselves, while all concept and role names of $\mathcal{S}ig$ are dealt with in exactly the way as suggested by the assertions of $\mathcal{A}$.

Definition 43. Assume $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$ is an arbitrary closed knowledge base. The interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$ is then said to be an **initial interpretation of $\mathcal{KB}$** if and only if $\Delta^{\mathcal{I}}$ is $\mathcal{D}$ and, moreover, $\mathcal{V}$ deals with an arbitrary individual name, $a \in \mathcal{D}$, and an arbitrary concept or role name, $TN \in \mathcal{S}ig$, in such a way that $a^{\mathcal{I}}$ is $\{a\}$ and, if $\mathcal{V}(TN)$ is $\mathcal{S}$, then $TN \doteq \mathcal{S}$ is a member of $\mathcal{A}$.

Fact 5. Each initial interpretation of a closed knowledge base is a model of it.

Definition 44. Assume $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$ is an arbitrary knowledge base. Then $\mathcal{KB}$ is said to be **vivid** or **physical** if and only if it is closed and for every two distinct individual names, a and b, of $\mathcal{D}$, at least one of the uniqueness axioms $a \neq b$ or $b \neq a$ is a member of $\mathcal{A}$.

Note that both observations about closed knowledge bases apply to vivid ones, too, because vivid knowledge bases are defined as a special kind of closed ones.

Example 4 (Vivid knowledge base). Consider the knowledge base $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$ such that $\mathcal{D}, \mathcal{S}ig, \mathcal{A}$ are given as follows:

$$\begin{aligned} \mathcal{D} &= \{\mathsf{a}, \mathsf{b}, \mathsf{table}\}, \\ \mathcal{S}ig &= \{\mathsf{Block}, \mathsf{on}\}, \\ \mathcal{A} &= \{\mathsf{Block} \doteq \{\mathsf{a}, \mathsf{b}\}, \mathsf{a} \neq \mathsf{b}, \mathsf{on} \doteq \{\langle \mathsf{a}, \mathsf{b}\rangle, \langle \mathsf{b}, \mathsf{table}\rangle\}, \mathsf{table} \neq \mathsf{a}, \mathsf{table} \neq \mathsf{b}\}. \end{aligned}$$

This knowledge base is vivid.

[2] Recall that, as far as closed knowledge bases are concerned, assertions of the form $\langle a, b\rangle{:}\epsilon$ are not admissible in any case, while an assertion of the form $\epsilon \doteq \{\langle a, b\rangle\}$ would be admissible only if ϵ were a member of $\mathcal{S}ig$. The latter, however, is required to be a set of concept and role names and, therefore, must not contain the identity role ϵ.

The difference between vivid and closed knowledge bases consists in the fact that models of the latter may have some freedom in dealing with individual names. In particular, if a and b are two individual names of $\mathcal{D}$ such that $\mathcal{A}$ contains neither the uniqueness axiom $a \not\equiv b$ nor $b \not\equiv a$, then a model, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$, of a corresponding closed knowledge base is allowed to handle a and b in such a way that $a^{\mathcal{I}} = b^{\mathcal{I}}$. A corresponding vivid knowledge base, however, is not allowed to do so in that it always has to contain at least one of the uniqueness axioms $a \not\equiv b$ or $b \not\equiv a$.

Definition 45. Assume $\mathcal{L}$ is a set of concepts and roles, $\mathcal{D}$ is a set of individual names, while both $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ as well as $\mathcal{J} = \langle \Delta^{\mathcal{J}}, .^{\mathcal{J}}, \mathcal{W} \rangle$ are any interpretations. Then $\mathcal{I}$ and $\mathcal{J}$ are said to be **($\mathcal{L}$,$\mathcal{D}$)-isomorphic** if and only if there exists a one-to-one function, h, mapping $\Delta^{\mathcal{I}}$ onto $\Delta^{\mathcal{J}}$ such that for every individual name, a, of $\mathcal{D}$, every concept, C, of $\mathcal{L}$, and every role, R, of $\mathcal{L}$, all the following equations are satisfied:

$$\begin{aligned} a^{\mathcal{J}} &= h(a^{\mathcal{I}}), \\ C^{\mathcal{J}} &= \{h(d) : d \in C^{\mathcal{I}}\}, \\ R^{\mathcal{J}} &= \{\langle h(d), h(e) \rangle : \langle d, e \rangle \in R^{\mathcal{I}}\}. \end{aligned}$$

Fact 6. Assume $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ is a vivid knowledge base. Then every two models of $\mathcal{KB}$ are ($\mathcal{S}ig$,$\mathcal{D}$)-isomorphic.

Roughly speaking, this means that the models of any physical knowledge base, $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$, are *unique* in the sense that both their domains as well as the way they handle those term names contained in $\mathcal{S}ig$ and $\mathcal{D}$ are identical, at least modulo renaming the elements of their domains.

4.2.2 Knowledge Base Queries

So far, we have specified what is meant by knowledge bases and their models and we have fixed some basic properties of closed and vivid knowledge bases. It remains, however, to describe how such knowledge bases can be queried and, then, how the answers are determined. On top of that, we shall see that in case of vivid knowledge bases, evaluating queries reduces, in effect, to model checking.

Definition 46. Let $\mathcal{L}$ be a set of concepts and roles. A **query** of $\mathcal{L}$ is either an axiom of $\mathcal{L}$ or an $\mathcal{L}$-assertion over some set of individuals.

Definition 47. Assume $\mathcal{KB}$ is an arbitrary knowledge base, $\mathcal{T}$ is a terminology, and Q is a query. Then Q is said to **hold** in $\mathcal{KB}$ with respect to $\mathcal{T}$, in symbols $\mathcal{KB} \models_{\mathcal{T}} Q$,

if and only if each interpretation which is a model of both $\mathcal{KB}$ and $\mathcal{T}$ is a model of Q as well. Moreover, we write $\mathcal{KB} \not\models_{\mathcal{T}} Q$ if and only if Q does *not* hold in $\mathcal{KB}$ with respect to $\mathcal{T}$.

Fact 7. Assume $\mathcal{KB}$ is a knowledge base, $\mathcal{T}$ is a terminology, and $a{:}C$ as well as $(a,b){:}R$ are any $\mathcal{U}\mu$-assertions. Then $\mathcal{KB} \not\models_{\mathcal{T}} a{:}\neg C$ if and only if there is at least one interpretation which is a model of $\mathcal{KB}$, $\mathcal{T}$, and $a{:}C$. Similarly, $\mathcal{KB} \not\models_{\mathcal{T}} (a,b){:}\neg R$ if and only if there is at least one interpretation which is a model of $\mathcal{KB}$, $\mathcal{T}$, and $(a,b){:}R$.

So far we gave meanings to queries posed to any knowledge base along with a terminology. In doing so we have ignored the fact that only certain queries make sense when a particular knowledge base and a particular terminology is given and that terminologies should go with the given knowledge base too. It does make sense, of course, to restrict attention to those queries and terminologies which are built up from term names each of which is drawn from either $\mathcal{S}ig$, $\mathcal{D}$, or the set of those concept and role names which are defined in the terminology. Whenever the terminology as well as the query satisfy this condition, we say that both are *compatible* with the given knowledge base.

Definition 48. Assume $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ is an arbitrary knowledge base, $\mathcal{T}$ is a terminology, and Q is a query. Then $\mathcal{T}$ and Q are said to be a **compatible with** $\mathcal{KB}$ if and only if all term names occurring in $\mathcal{T}$ and Q are among $\mathcal{S}ig \cup \mathcal{D}$, except for those concept and roles names which are defined in $\mathcal{T}$.

We have already seen that all initial interpretations of closed knowledge bases are models and that all models of single vivid knowledge bases are $(\mathcal{S}ig, \mathcal{D})$-isomorphic to each other. But then all models of a vivid knowledge base are known to be $(\mathcal{S}ig, \mathcal{D})$-isomorphic to any of its initial interpretations. This means that each vivid knowledge base can be viewed as one possible syntactic representation of a certain part of any of its initial interpretations, namely exactly that part which all initial interpretations of it have in common. In this connection recall that all initial interpretations of a vivid knowledge base differ only in their handling of those term names which are neither contained in $\mathcal{S}ig$ nor in $\mathcal{D}$. This indicates that there is an intimate relationship between model checking and querying vivid knowledge bases, at least as far as terminologies and queries compatible with the knowledge base are concerned. However, there still remains a minor difference. The difference is that the definition of the semantic relation $\mathcal{KB} \models_{\mathcal{T}} Q$ universally quantifies over *all* interpretations which are models of both $\mathcal{KB}$ and $\mathcal{T}$ and, of course, there can be more than one such interpretation, even when $\mathcal{KB}$ is a vivid knowledge base. Therefore, it remains to be shown that in this case, it suffices to choose an *arbitrary* interpretation which is a model of both $\mathcal{KB}$ and $\mathcal{T}$ (for example, an initial interpretation of $\mathcal{KB}$ being a model

of $\mathcal{T}$) and, then, to check whether or not this particular interpretation is a model of Q as well. In other words, it must be shown that querying vivid knowledge bases, in effect, reduces to model checking. To prove this fact, the following lemma will be useful.

Lemma 4. *Assume $\mathcal{L}$ is a set of concepts or roles of $\mathcal{U}\mu$, while $\mathcal{L}_0$ is a set of concept and role names and $\mathcal{D}$ is a set of individual names such that the union of $\mathcal{L}_0$ with $\mathcal{D}$ contains all those term names which occur in at least one concept or role of $\mathcal{L}$. Then every two interpretations which are $(\mathcal{L}_0, \mathcal{D})$-isomorphic are also $(\mathcal{L}_0 \cup \mathcal{L}, \mathcal{D})$-isomorphic.*

This lemma can be proved by induction on the structure of the concepts and roles of $\mathcal{L}$.

Proposition 9. *Assume $\mathcal{KB}$ is a vivid knowledge base. Assume, moreover, that $\mathcal{T}$ is an* acyclic *terminology of $\mathcal{U}\mu$ and Q is a query of $\mathcal{U}\mu$ such that $\mathcal{T}$ and Q are compatible with $\mathcal{KB}$. Finally assume $\mathcal{I}$ is an* arbitrary *chosen model of $\mathcal{KB}$ and $\mathcal{T}$. Then $\mathcal{KB} \models_{\mathcal{T}} Q$ if and only if $\mathcal{I}$ is a model of Q.*

We may, of course, chose any initial interpretation of $\mathcal{KB}$ being a model of $\mathcal{T}$ as one particular model of $\mathcal{KB}$ and $\mathcal{T}$.

Proof. Suppose Q is of the form $T_1 \doteq T_2$ and $\mathcal{KB}$ is $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$. It clearly suffices to prove that if $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ and $\mathcal{J} = \langle \Delta^{\mathcal{J}}, .^{\mathcal{J}}, \mathcal{W} \rangle$ are two interpretations which are models of both $\mathcal{KB}$ and $\mathcal{T}$, then $\mathcal{I}$ and $\mathcal{J}$ are $(\{T_1, T_2\}, \mathcal{D})$-isomorphic, so that $\mathcal{I}$ is a model of $T_1 \doteq T_2$ if and only if $\mathcal{J}$ is also a model of $T_1 \doteq T_2$. Recall, according to Fact 6, $\mathcal{I}$ and $\mathcal{J}$ are already known to be $(\mathcal{S}ig, \mathcal{D})$-isomorphic. As $\mathcal{T}$ is assumed to be acyclic, $\mathcal{I}$ and $\mathcal{J}$ can be shown to be not only $(\mathcal{S}ig, \mathcal{D})$-isomorphic, but also $(\mathcal{S}ig \cup \mathit{def}(\mathcal{T}), \mathcal{D})$-isomorphic. But then the last lemma immediately yields that $\mathcal{I}$ and $\mathcal{J}$ are $(\{T_1, T_2\}, \mathcal{D})$-isomorphic. Lemma 4 can actually be applied to this case by assigning $\mathcal{L}_0$ with $\mathcal{S}ig \cup \mathit{def}(\mathcal{T})$ and $\mathcal{L}$ with $\{T_1, T_2\}$. All preconditions of the lemma are then met in that $\mathcal{T}$ and $T_1 \doteq T_2$ are assumed to be compatible with $\mathcal{KB}$. The term names occurring in T_1 and T_2 are therefore among $\mathcal{S}ig \cup \mathcal{D} \cup \mathit{def}(\mathcal{T})$.

So it remains to give a proof of the fact that $\mathcal{I}$ and $\mathcal{J}$ are not only $(\mathcal{S}ig, \mathcal{D})$-isomorphic, but also $(\mathcal{S}ig \cup \mathit{def}(\mathcal{T}), \mathcal{D})$-isomorphic. The proof proceeds by a straightforward induction on the number of concept and role introductions of $\mathcal{T}$. More precisely, we are going to show that for each set of concept and role names, $\mathcal{L}$, containing at least those of $\mathcal{S}ig$, $\mathcal{I}$ and $\mathcal{J}$ are $(\mathcal{L} \cup \mathit{def}(\mathcal{T}), \mathcal{D})$-isomorphic whenever they are $(\mathcal{L}, \mathcal{D})$-isomorphic. As regards the induction base, one has to consider the case when $\mathcal{T}$ is the empty terminology, in which case the claim to be proved holds trivially in that $\mathit{def}(\mathcal{T})$ is in this case empty. For the induction step, choose an arbitrary concept or role introduction of $\mathcal{T}$, say, $TN \doteq T$, such that T does not involve any

concept or role name which is defined in $\mathcal{T}$. The only term names occurring in T are therefore among $\mathcal{L} \cup \mathcal{D}$. As $\mathcal{T}$ is assumed to be acyclic, the existence of such a concept or role introduction is actually guaranteed. According to the last lemma, it immediately follows that $\mathcal{I}$ and $\mathcal{J}$ are also $(\mathcal{L} \cup \{T\}, \mathcal{D})$-isomorphic and, therefore, also $(\mathcal{L} \cup \{TN\}, \mathcal{D})$-isomorphic. The induction hypothesis then can be applied to the terminology $\mathcal{T}' = \mathcal{T} \setminus \{TN \doteq T\}$ and the set of concept and role names $\mathcal{L}' = \mathcal{L} \cup \{TN\}$ with the result that $\mathcal{I}$ and $\mathcal{J}$ are $(\mathcal{L}' \cup def(\mathcal{T}'), \mathcal{D})$-isomorphic too. But then we are already done because $\mathcal{L}' \cup def(\mathcal{T}')$ coincides with $\mathcal{L} \cup \{TN\} \cup def(\mathcal{T} \setminus \{TN \doteq T\})$ and, therefore, also with $\mathcal{L} \cup def(\mathcal{T})$. □

Now let us look at the example of the chromatic number of graphs, which demonstrates the power of 'null values' of closed knowledge bases. First, let us define the notion of the k-colorability of graphs. A directed graph, $\langle V, E \rangle$, is said to be ***k*-colorable** if and only if there is a total function $f : V \to \{1, ..., k\}$ such that $f(v) \neq f(w)$ whenever $\langle v, w \rangle \in E$, see e.g. [Garey and Johnson, 1979], page 191. In the following example we are going to explain how to construct, given an arbitrary directed graph, a closed knowledge base along with a query such that the given graph is k-colorable if and only if the query holds in the closed knowledge base with respect to the empty terminology. The part of the knowledge base which represents the given graph can be thought of as being vivid. The knowledge base constructed contains also k different individual names, each representing a single color, including appropriate uniqueness axioms imposed on each pair of distinct 'colors'. Because of the fact that there is no uniqueness axiom at all between a vertex and a color, the resulting knowledge base is closed rather than vivid. Each individual name representing a color can, therefore, be viewed as a null value. The query, then, forces each vertex of the graph to take one of the k colors such that no two vertices of the graph take the same color whenever they are connected by at least one edge. We shall confine ourselves in the example to the case of three colors. The reduction, however, works for any other number of colors in an entirely analogous way.

Example 5 (3-colorable graphs). Consider an arbitrary directed graph, $G = \langle V, E \rangle$, such that V is a finite set of individual names. Assume Vertex, $\mathsf{Color}_1, \mathsf{Color}_2, \mathsf{Color}_3$ are pairwise distinct concept names, while edge and $\mathsf{accessible}$ are pairwise distinct role names. Finally, assume red, green, as well as yellow to be distinct individual names not contained in V. Let $\mathcal{KB}_G$ be the knowledge base $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ such that $\mathcal{D}$, $\mathcal{S}ig$, and $\mathcal{A}$ are given as follows.

(a) $\mathcal{D} = V \cup \{\mathsf{red}, \mathsf{green}, \mathsf{yellow}\}$;

(b) $\mathcal{S}ig = \{\mathsf{Vertex}, \mathsf{edge}, \mathsf{accessible}, \mathsf{Color}_1, \mathsf{Color}_2, \mathsf{Color}_3\}$;

(c) $\mathcal{A}$ is the set containing all and only the following $\mathcal{S}ig$-assertions and uniqueness axioms over $\mathcal{D}$:

$$\mathsf{Vertex} \doteq V,$$

$$\begin{aligned}
\mathsf{edge} &\doteq E,\\
\mathsf{accessible} &\doteq \{\langle a,b\rangle : a,b \in V\},\\
\mathsf{Color}_1 &\doteq \{\mathsf{red}\},\\
\mathsf{Color}_2 &\doteq \{\mathsf{green}\},\\
\mathsf{Color}_3 &\doteq \{\mathsf{yellow}\},\\
\mathsf{red} &\not\doteq \mathsf{green},\\
\mathsf{red} &\not\doteq \mathsf{yellow},\\
\mathsf{green} &\not\doteq \mathsf{yellow}.
\end{aligned}$$

The graph G is then 3-colorable if and only if for an arbitrary but fixed individual name of V, say, a_0, the following $\mathcal{ALC}$-assertion does *not* hold in $\mathcal{KB}_G$ with respect to the empty terminology $\emptyset$:

$$a_0{:}\neg\forall\mathsf{accessible}{:}\bigsqcup_{i=1}^{3}(\mathsf{Color}_i \sqcap \forall\mathsf{edge}{:}\neg\mathsf{Color}_i).$$

As one might think, $\bigsqcup_{i=1}^{3}(\mathsf{Color}_i \sqcap \forall\mathsf{edge}{:}\neg\mathsf{Color}_i)$ is shorthand for the concept $(\mathsf{Color}_1 \sqcap \forall\mathsf{edge}{:}\neg\mathsf{Color}_1) \sqcup \ldots \sqcup (\mathsf{Color}_3 \sqcap \forall\mathsf{edge}{:}\neg\mathsf{Color}_3)$.

We are going to give a proof of this claim because it will serve later on as a key argument in providing a lower computational complexity bound for querying closed knowledge bases. Throughout the proof we use $\mathsf{3Colorable}$ in lieu of $\forall\mathsf{accessible}{:}\bigsqcup_{i=1}^{3}(\mathsf{Color}_i \sqcap \forall\mathsf{edge}{:}\neg\mathsf{Color}_i)$. First, let us prove the if-part. In order to do so, assume that $\mathcal{KB}_G \not\models_\emptyset a_0{:}\neg\mathsf{3Colorable}$. We have to show that this assumption leads to the conclusion that the graph G is 3-colorable. According to Fact 7, there is at least one model of $\mathcal{KB}_G$, say, $\mathcal{I} = \langle\Delta^\mathcal{I}, .^\mathcal{I}, \mathcal{V}\rangle$, such that $a_0^\mathcal{I} \in \mathsf{3Colorable}^\mathcal{I}$. The fact that $\mathcal{I}$ is a model of $\mathcal{KB}_G$ and, therefore, also one of the uniqueness axioms $\mathsf{red} \not\doteq \mathsf{green}$, $\mathsf{red} \not\doteq \mathsf{yellow}$, and $\mathsf{green} \not\doteq \mathsf{yellow}$, all included in $\mathcal{A}$, forces $\mathsf{red}^\mathcal{I}$, $\mathsf{green}^\mathcal{I}$, and $\mathsf{yellow}^\mathcal{I}$ to denote pairwise distinct elements of $\Delta^\mathcal{I}$, say, 1, 2, and 3 respectively. The fact that $\mathcal{I}$ is a model of $\mathsf{Color}_1 \doteq \{\mathsf{red}\}$, $\mathsf{Color}_2 \doteq \{\mathsf{green}\}$, and of $\mathsf{Color}_3 \doteq \{\mathsf{yellow}\}$, also all contained in $\mathcal{A}$, forces $\mathsf{Color}_1^\mathcal{I}$, $\mathsf{Color}_2^\mathcal{I}$, $\mathsf{Color}_3^\mathcal{I}$ to denote the sets $\{1\}$, $\{2\}$, and $\{3\}$ respectively. But then $a_0^\mathcal{I}$ is an element of $\mathsf{3Colorable}^\mathcal{I}$ just in case the following subset relation is satisfied:

$$\begin{aligned}
\underbrace{\mathsf{accessible}^\mathcal{I}(a_0^\mathcal{I})}_{=\ \{a^\mathcal{I}:a\in V\}} &\subseteq (\bigsqcup_{i=1}^{3}(\mathsf{Color}_i \sqcap \forall\mathsf{edge}{:}\neg\mathsf{Color}_i))^\mathcal{I}\\
&= \bigcup_{i=1}^{3}\{i : \mathsf{edge}^\mathcal{I}(i) \subseteq \Delta^\mathcal{I} \setminus \{i\}\} \qquad (4.1)\\
&= \bigcup_{i=1}^{3}\{i : \textit{for every } j \textit{ such that } \langle i,j\rangle \in \mathsf{edge}^\mathcal{I},\ j \neq i\}.
\end{aligned}$$

Because of the fact that $\mathsf{accessible}^{\mathcal{I}}(a_0^{\mathcal{I}})$ is $\{a^{\mathcal{I}} : a \in V\}$, this subset relation means in particular that for each $a \in V$, $a^{\mathcal{I}}$ is a member of $\{1,2,3\}$.

In order to continue, consider the total function $f : V \to \{1,2,3\}$ such that for every $a \in V$, $f(a)$ is defined to be $a^{\mathcal{I}}$. This function is well-defined in that for each $a \in V$, $a^{\mathcal{I}}$ is actually a member of $\{1,2,3\}$, as we just remarked. To prove that G is 3-colorable, it remains to verify that $f(a) \neq f(b)$ whenever $\langle a,b\rangle \in E$. Hence, take two arbitrary elements of V, say, a and b, such that $\langle a,b\rangle \in E$. As $\mathcal{I}$ is a model of $\mathsf{edge} \doteq E$, $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle$ is, therefore, known to be an element of $\mathsf{edge}^{\mathcal{I}}$ and, as $f(a)$ and $f(b)$ are defined to be $a^{\mathcal{I}}$ and $b^{\mathcal{I}}$ respectively, $\langle f(a), f(b)\rangle$ must be an element of $\mathsf{edge}^{\mathcal{I}}$ too. According to the subset relation (4.1) given above, it follows immediately that $f(a) \neq f(b)$. This completes the if-part.

For the only-if-part assume G is 3-colorable. This is to say, it is assumed that there exists a total function $f : V \to \{1,2,3\}$ such that $f(a) \neq f(b)$ whenever $\langle a,b\rangle \in E$. In what follows, we shall define an interpretation which is a model of $\mathcal{KB}_G$, but which will turn out to be *no* model of $a_0{:}\neg\mathsf{3Colorable}$. This will prove that $\mathcal{KB}_G \not\models a_0{:}\neg\mathsf{3Colorable}$.

The basis of this interpretation can be any $\mathcal{N}$-valuation, $\mathcal{V}$, over Δ satisfying the following constraints:

$$\begin{aligned}
\Delta &= \{1,2,3\},\\
\mathcal{V}(\mathsf{Vertex}) &= \{f(a) : a \in V\},\\
\mathcal{V}(\mathsf{edge}) &= \{\langle f(a), f(b)\rangle : \langle a,b\rangle \in E\},\\
\mathcal{V}(\mathsf{accessible}) &= \{\langle f(a), f(b)\rangle : a,b \in V\},\\
\mathcal{V}(\mathsf{Color}_1) &= \mathcal{V}(\{\mathsf{red}\}) \quad = \{1\},\\
\mathcal{V}(\mathsf{Color}_2) &= \mathcal{V}(\{\mathsf{green}\}) \ = \{2\},\\
\mathcal{V}(\mathsf{Color}_3) &= \mathcal{V}(\{\mathsf{yellow}\}) = \{3\},\\
\mathcal{V}(\{a\}) &= \{f(a)\} \ \textit{whenever } a \in V.
\end{aligned}$$

It can easily be seen that whenever $\mathcal{V}$ is such a valuation, the interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$ with $\Delta^{\mathcal{I}} = \Delta$ is always a model of $\mathcal{KB}_G$. It thus remains to prove that this interpretation is no model of the assertion $a_0{:}\neg\mathsf{3Colorable}$. That is, it must be shown that $a_0^{\mathcal{I}} \notin \neg\mathsf{3Colorable}^{\mathcal{I}}$ or, equivalently, that $a_0^{\mathcal{I}} \in \mathsf{3Colorable}^{\mathcal{I}}$. The latter clearly holds just in case the following subset relation is satisfied:

$$\begin{aligned}
\mathsf{accessible}^{\mathcal{I}}(a_0^{\mathcal{I}}) &\subseteq (\bigsqcup_{i=1}^{3}(\mathsf{Color}_i \sqcap \forall\mathsf{edge}{:}\neg\mathsf{Color}_i))^{\mathcal{I}}\\
&= \bigcup_{i=1}^{3}\{i : \mathsf{edge}^{\mathcal{I}}(i) \subseteq \{1,2,3\} \setminus \{i\}\}\\
&= \bigcup_{i=1}^{3}\{i : \textit{for all } j \textit{ such that } \langle i,j\rangle \in \mathcal{V}(\mathsf{edge}),\ j \neq i\}.
\end{aligned}$$

We shall see that this subset relation is trivially satisfied in that the set $\bigcup_{i=1}^{3}\{i$: *for all* j *such that* $\langle i,j\rangle \in \mathcal{V}(\mathsf{edge}),\ j \neq i\}$ does contain *each* element of the domain of $\mathcal{I}$, i.e., it contains each element of the set $\{1,2,3\}$. In order to do so, it suffices to verify that $j \neq i$ whenever $\langle i,j\rangle$ is an element of $\mathcal{V}(\mathsf{edge})$. Consider an arbitrary ordered pair, say, $\langle i,j\rangle$, which is a member of $\mathcal{V}(\mathsf{edge})$. Inspection of the definition of $\mathcal{V}(\mathsf{edge})$ immediately reveals that there have to exist two elements of V, say, a and b, such that $f(a)$ is i, $f(b)$ is j, and $\langle a,b\rangle \in E$. But then the assumption made about f implies that $f(a)$ cannot be equal to $f(b)$. But then $i \neq j$, as was to be shown. □

4.3 Querying Vivid Knowledge Bases

In this and the following section, we shall explore the computational complexity of querying all kinds of knowledge bases, i.e., vivid, closed, and finite ones. One of the main results will be that only vivid knowledge bases give rise to tractable query answering. However, evaluating queries posed to vivid knowledge bases and acyclic terminologies will turn out to be among the hardest problems that are solvable in polynomial time, that is, it is P-complete, even in case of the standard description logic $\mathcal{ALC}$.

4.3.1 A Polynomial-Time Algorithm

We are now going to present a polynomial-time algorithm for evaluating queries posed to vivid knowledge bases and acyclic terminologies. More precisely, the algorithm computes for an arbitrary query, Q, an acyclic terminology, $\mathcal{T}$, and a vivid knowledge base, $\mathcal{KB}$, a Boolean value supposed to reflect whether or not Q holds in $\mathcal{KB}$ with respect to $\mathcal{T}$. Of course, the algorithm deals solely with those $\mathcal{T}$'s and Q's which are compatible with $\mathcal{KB}$. The main characteristic of the algorithm is that it is sound and complete, at least as far as a certain fragment of $\mathcal{U}\mu$ is concerned. Another important characteristic of the algorithm is that its running time is polynomially bounded in the length of the sum of the sizes of the given knowledge base, the terminology, and the query. The fragment of $\mathcal{U}\mu$ for which the algorithm will turn out to be both sound and complete, called $\mathcal{U}\mu^{-}$, merely restricts the possible occurrences of nested *alternating* fixed-point operators. This restriction, however, does not affect practical usability.

In case fixed-point operators are absent, the algorithm is quite straightforward. At its very heart there is a procedure, called *ext*, which computes for an arbitrary concept or role, T, a set which corresponds to the value of $T^{\mathcal{I}}$, where $.^{\mathcal{I}}$ is the interpretation function of a certain model of $\mathcal{KB}$. If $\mathcal{KB}$ is $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$, then according to Fact 6, all models of $\mathcal{KB}$ are pairwise $(\mathcal{S}ig, \mathcal{D})$-isomorphic. But then it does not matter which particular model is actually chosen. For convenience's sake, we decided to take one

of those which we called on page 98 *initial*, this is to say, a model $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ such that $\Delta^{\mathcal{I}}$ is $\mathcal{D}$ and, moreover, for each individual name, $a \in \mathcal{D}$, $\mathcal{V}(\{a\})$ is $\{a\}$. Given such a valuation over $\mathcal{D}$, the procedure computes what corresponds exactly to the result of applying $.^{\mathcal{I}}$ to T, where $.^{\mathcal{I}}$ is part of an arbitrary chosen interpretation, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}' \rangle$, such that $\Delta^{\mathcal{I}} = \mathcal{D}$ and $\mathcal{V} \subseteq \mathcal{V}'$. The actual value of $T^{\mathcal{I}}$ does not depend on which specific initial interpretation of $\mathcal{KB}$ is taken as a basis because only those T's will be dealt with whose only term names are among $\mathcal{S}ig \cup \mathcal{D}$. This means that the possible interpretations, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}' \rangle$, may differ only in their handling of those term names which do not occur in T. The value of $T^{\mathcal{I}}$ is, of course, independent of the interpretation of these term names.

The value of $T^{\mathcal{I}}$ is computed by *ext* in a recursive fashion. In case of T being an atomic concept or role, it yields the value $\mathcal{V}(T)$ specified by $\mathcal{V}$, unless T is either $\top$ or ϵ, in which case it just yields $\mathcal{D}$ itself and the identity relation over $\mathcal{D}$ respectively. In case of compound concepts or roles, the value of $T^{\mathcal{I}}$ is computed with the help of the corresponding values of the subconcepts and subroles, mimicing the semantics of the particular concept or role-structuring primitive involved. This can be done employing operations of basic set theory only. In particular, each of these operations can be computed by at most $O(\sharp|\mathcal{D}|^2)$ steps, at least as far as $\mathcal{U}$ is concerned.

Acyclic terminologies are dealt with as follows. Whenever *ext* is called with T being a concept or role name which is among those defined in $\mathcal{T}$, it computes the result of applying $.^{\mathcal{I}}$ to the right-hand side of the corresponding concept or role introduction of $\mathcal{T}$. In order to avoid recomputations, however, for each call of *ext* its result has to be stored. This means that in particular those concept and role names are evaluated solely once which are among those defined in $\mathcal{T}$. For this purpose a memory space of size $O(n^2 + n\sharp|\mathcal{D}|^2)$ will do if n is the sum of the sizes of T and $\mathcal{T}$. To see this, realize that there can be at most $O(n)$ different memory entries, the size of each cannot exceed $O(n + \sharp|\mathcal{D}|^2)$. Figure 4.3.1 depicts the recursive procedure *ext* in full details.

Difficulties arise, however, as soon as fixed-point operators enter the picture. The reader may object that, as is pretty well-known, both least and greatest fixed points of monotonic functions can be computed by iteration. If the domain of the monotonic function is finite, then even a finite number of iteration steps will do. We are going to explain this point in some detail because fixed points will actually be computed by *ext* with the help of finite iteration, though not exactly in the standard way. So take an arbitrary monotonic function mapping subsets of some set, say, Γ, to subsets of Γ. No matter whether Γ is finite or not, the least and the greatest fixed point of f can be characterized by means of the so-called ordinal powers of f. *Ordinals* can be thought of as sets constructed out of the empty set. The first finite ordinal, 0, corresponds to $\emptyset$, the second, 1, to $\{\emptyset\}$, the third, 3, to $\{\emptyset, \{\emptyset\}\}$, and so on. In fact, the *successor*, $\alpha + 1$, of an arbitrary ordinal, α, can always be treated as the set $\alpha \cup \{\alpha\}$. The first infinite ordinal is then ω, the set of all positive integers greater than or equal to 0. It is also the first *limit* ordinal in the sense that it is not the successor of any other

algorithm : $ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V})$;
global variable : E; % $\mathcal{L}_{evaluated}$-*valuation over* $\mathcal{D}$
let *for every concept,* C, $top(C)$ *be* $\mathcal{D}$ *and for every role,* R, $top(R)$ *be* $\mathcal{D} \times \mathcal{D}$;
if $E(T)$ *is defined* **then begin** $S := E(T)$; **goto** $L2$ **end**;
if $\mathcal{V}(T)$ *is defined* **then begin** $S := \mathcal{V}(T)$; **goto** $L2$ **end**;
if $T \in def(\mathcal{T})$ & $(T \doteq T') \in \mathcal{T}$ **then begin** $S := ext(T', \mathcal{T}, \mathcal{D}, \mathcal{V})$; **goto** $L1$ **end**;
begin case T **of** :

$\top$: $S := \mathcal{D}$;

ϵ : $S := \{\langle a, a\rangle : a \in \mathcal{D}\}$;

$T_1 \sqcap T_2$: $S_1 := ext(T_1, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S_2 := ext(T_2, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S := S_1 \cap S_2$;

$T_1 \sqcup T_2$: $S_1 := ext(T_1, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S_2 := ext(T_2, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S := S_1 \cup S_2$;

$\neg T_1$: **if** T_1 *is compound*
then $S := ext(nnf(\neg T_1), \mathcal{T}, \mathcal{D}, \mathcal{V})$
else if $T_1 \in def(\mathcal{T})$ & $(T_1 \doteq T_2) \in \mathcal{T}$
then $S := ext(nnf(\neg T_2), \mathcal{T}, \mathcal{D}, \mathcal{V})$
else begin $S_1 := ext(T_1, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S := top(T_1) \setminus S_1$ **end**;

$\exists^{\geq n} R{:}C$: $S_R := ext(R, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S_C := ext(C, \mathcal{T}, \mathcal{D}, \mathcal{V})$;
$S := \{a \in \mathcal{D} : \sharp|S_R(a) \cap S_C| \geq n\}$;

$\exists^{\leq m} R{:}C$: $S_R := ext(R, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S_C := ext(C, \mathcal{T}, \mathcal{D}, \mathcal{V})$;
$S := \{a \in \mathcal{D} : \sharp|S_R(a) \cap S_C| \leq m\}$;

$\mu X.T_1$: **if** $\mathcal{V}(X)$ *is not defined* **then** $\mathcal{W} := \mathcal{V} \cup \{\langle X, \emptyset\rangle\}$ **else** $\mathcal{W} := \mathcal{V}$;
$S := fixp(T_1, X, \mathcal{T}, \mathcal{D}, \mathcal{W})$;

$\nu X.T_1$: **if** $\mathcal{V}(X)$ *is not defined* **then** $\mathcal{W} := \mathcal{V} \cup \{\langle X, top(T_1)\rangle\}$ **else** $\mathcal{W} := \mathcal{V}$;
$S := fixp(T_1, X, \mathcal{T}, \mathcal{D}, \mathcal{W})$;

$R \circ S$: $S_R := ext(R, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S_S := ext(S, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S := S_R \circ S_S$;

$R \oplus S$: $S_R := ext(R, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S_S := ext(S, \mathcal{T}, \mathcal{D}, \mathcal{V})$;
$S := \{\langle d, e\rangle :$ *for every* $f \in \mathcal{D}$, $\langle d, f\rangle \in S_R$ *or* $\langle f, e\rangle \in S_S\}$;

R^{-1} : $S_R := ext(R, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S := \{\langle a, b\rangle : \langle b, a\rangle \in S_R\}$;

$C \times D$: $S_C := ext(C, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S_D := ext(D, \mathcal{T}, \mathcal{D}, \mathcal{V})$; $S := S_C \times S_D$

end case;
$L1$: **if** T *is closed and not equal to* $\top$ *or* ϵ **then** $E := E \cup \{\langle T, S\rangle\}$;
$L2$: **return** S;

Figure 4.5: The algorithm *ext*

ordinal. With the help of ω, we can again construct an infinite sequence of ordinals, $\omega + 1$, $\omega + 2$, and so on, the collection of which together forms another limit ordinal,

denoted by $\omega 1$. This construction process continues indefinitely. We can specify an ordering, $<$, on the collection of all ordinals by defining $\alpha < \beta$ if and only if $\alpha \in \beta$, where both α and β are arbitrary ordinals. For a formal account of ordinals confer [Halmos, 1974]. The *ordinal powers* of f are finally defined by induction as follows. First, $\boldsymbol{f}^{\uparrow\, 0}$ and $\boldsymbol{f}^{\downarrow 0}$ are $\emptyset$ and Γ respectively. Moreover, $\boldsymbol{f}^{\uparrow\, \alpha+1}$ is defined to be $f(f^{\uparrow\alpha})$, while $\boldsymbol{f}^{\downarrow\alpha+1}$ is $f(f^{\downarrow\alpha})$. Whenever α is a limit ordinal, $\boldsymbol{f}^{\uparrow\, \alpha}$ and $\boldsymbol{f}^{\downarrow\alpha}$ are $\bigcup\{f^{\uparrow\beta} : \beta < \alpha\}$ and $\bigcap\{f^{\downarrow\beta} : \beta < \alpha\}$ respectively.

As already mentioned, the least as well as the greatest fixed point of f can be characterized by means of ordinal powers of f, no matter whether Γ is finite or not. The following proposition, usually attributed to Kleene, describes how such a characterization appears.

Proposition 10 (Kleene). *Assume f is an arbitrary monotonic function mapping subsets of some set, say, Γ, to subsets of Γ. Then there exist ordinals, α and β, such that $f^{\uparrow\alpha}$ and $f^{\downarrow\beta}$ coincide with the least and the greatest fixed point of f respectively.*

When Γ is finite, the least and the greatest fixed point of f can be obtained by finite iteration. The explanation is as follows. Due to the monotonicity of f, the infinite sequence $f^{\uparrow 0}, f^{\uparrow 1}, ...$ increases with respect to the subset relation, whereas $f^{\downarrow 0}, f^{\downarrow 1}, ...$ is decreasing. Whenever Γ is finite and has a cardinality of n, this clearly means that both sequences have to be stabilized, so to speak, when $f^{\uparrow n}$ and $f^{\downarrow n}$ are reached, or even before that. This is to say, if Γ has a finite cardinality of n, then $f^{\uparrow n} = f^{\uparrow n+1}$ and $f^{\downarrow n} = f^{\downarrow n+1}$. Both $f^{\uparrow n}$ as well as $f^{\downarrow n}$ are therefore always fixed points of f. But then $f^{\uparrow\omega}$ is $f^{\uparrow 0} \cup ... \cup f^{\uparrow n}$, which in turn coincides with $f^{\uparrow n}$ itself, and therefore so does $f^{\uparrow\omega}$. By induction it can be shown that for every ordinal, α, it holds that $f^{\uparrow\alpha}$ equals $f^{\uparrow n}$ whenever $n < \alpha$. Now, according to the previous proposition, it is known that there exists an ordinal, say, β, such that the least fixed point of f is exactly $f^{\uparrow\beta}$. As we have just seen, the latter coincides with $f^{\uparrow n}$ if $n < \beta$. On the other hand, if $\beta < n$, then $f^{\uparrow\beta}$ coincides also with $f^{\uparrow n}$ because $f^{\uparrow\beta}$ is already a fixed point of f. This means that $f^{\uparrow n}$ is in any case not only a fixed point of f, but also the least such fixed point because so is $f^{\uparrow\beta}$. The proof of the fact that $f^{\downarrow n}$ is the greatest fixed point of f proceeds analogously. This proves the following proposition.

Proposition 11. *Assume f is an arbitrary monotonic function mapping subsets of some set, say, Γ, to subsets of Γ. If Γ is finite and its cardinality is n, then the least and the greatest fixed point of f coincide with $f^{\uparrow n}$ and $f^{\downarrow n}$ respectively.*

It should be clear how this proposition provides for an alternative characterization of $(\mu X.T)^{\mathcal{I}}$ and of $(\nu X.T)^{\mathcal{I}}$ whenever $.^{\mathcal{I}}$ is part of an interpretation, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$, such that $\Delta^{\mathcal{I}}$ has a finite cardinality of n. In particular, if f_T is the function of X induced by T and $\mathcal{V}$, then according to the last proposition, the following two equations are

always met:

$$\begin{aligned} \mu X.T^{\mathcal{I}} &= {f_C}^{\uparrow n}, \\ \nu X.T^{\mathcal{I}} &= {f_C}^{\downarrow n}. \end{aligned}$$

At a first glance, this seems to give rise to an efficient method for computing the result of applying $.^{\mathcal{I}}$ to least and greatest fixed-point operators if the interpretation domain is finite. As a matter of fact, in the presence of nested fixed-point operators, the worst-case complexity of the induced method can be exponential and even superexponential. To see this, consider the following concept:

$$\mu X.\underbrace{(CN \sqcup \mu Y.\overbrace{(X \sqcup \exists RN{:}Y)}^{\stackrel{\text{def}}{=} D})}_{\stackrel{\text{def}}{=} C}.$$

The computation of $\mu X.C^{\mathcal{I}}$ with C being $(CN \sqcup \mu Y.D)$ and with D being $(X \sqcup \exists RN{:}Y)$ proceeds with the help of the last proposition as follows. As a first iteration step, the value of $C^{\mathcal{I}}$ has to be computed with $\mathcal{V}(X)$ being set to $\emptyset$. The result will correspond to $f_C(\emptyset) = {f_C}^{\uparrow 1}$. Before this value can be computed, $\mu Y.D^{\mathcal{I}}$ with $\mathcal{V}(X)$ being set to $\emptyset$ has to be computed by iteration too. This is done by computing First $D^{\mathcal{I}}$ with $\mathcal{V}(X) = \emptyset$ and $\mathcal{V}(Y) = \emptyset$. The second step of this inner iteration loop then computes the value of $D^{\mathcal{I}}$ with $\mathcal{V}(X) = \emptyset$ and $\mathcal{V}(Y)$ being set to the result of the previous step. After at most $\sharp|\mathcal{D}| + 1$ iteration steps, $.^{\mathcal{I}}$ applied to $\mu Y.D$ with $\mathcal{V}(X) = \emptyset$ and $\mathcal{V}(Y)$ being assigned with the result of the previous step will eventually yield the same value as the previous step. At this stage, the least fixed point of the function of Y induced by D and $\mathcal{V}_{\langle X,\emptyset\rangle}$ has been obtained and its union with $\mathcal{V}(CN)$ immediately yields the set corresponding to $f_C(\emptyset)$.

The second step of the outer loop then computes $f_C(f_C(\emptyset)) = {f_C}^{\uparrow 2}$ in the same manner. In the course of computing this value, the result of applying $.^{\mathcal{I}}$ to $\mu Y.D$ has to be computed once more, but this time with $\mathcal{V}(X)$ being set to ${f_C}^{\uparrow 1}$ rather than ${f_C}^{\uparrow 0}$. The computation of the least fixed point of the function of Y induced by D and $\mathcal{V}_{\langle X,{f_C}^{\uparrow 1}\rangle}$ may again require $\sharp|\mathcal{D}| + 1$ iteration steps. In the worst case, the result of applying $.^{\mathcal{I}}$ to $\mu Y.D$ has to be computed for a number of $\sharp|\mathcal{D}| + 1$ different values assigned to $\mathcal{V}(X)$. The computation of the latter may require in each case again $\sharp|\mathcal{D}| + 1$ iteration steps. All in all there can be $\mathrm{O}((\sharp|\mathcal{D}| + 1)^2|\mu X.C|)$ recursive calls in the course of computing $\mu X.C^{\mathcal{I}}$. In the general case, there can even be $\mathrm{O}((\sharp|\mathcal{D}| + 1)^{d+1}|\mu X.C|)$ such calls with d being the maximal nesting depth of fixed-point operators in $\mu X.C$.[3] In the presence of fixed-point operators on roles, there can be even $\mathrm{O}((\sharp|\mathcal{D}|^2 + 1)^{d+1}|\mu X.C|)$ calls of *ext* in that the computation of the least and greatest fixed points can require in this case a number of $\sharp|\mathcal{D} \times \mathcal{D}| + 1$ iteration

[3] *Nesting depth* is to be understood in such a way that, for instance, $\mu X.(CN \sqcup \mu Y.(X \sqcup \exists RN{:}Y))$ has a maximal nesting depth of 1.

steps rather than just $\sharp|\mathcal{D}| + 1$. Because of the fact that the upper bound of d is determined by the size of the concept $\mu X.C$, in any case, this method would give rise to an upper bound of the number of recursive calls of *ext* which is *superexponential* in $|\mu X.C| + \sharp|\mathcal{D}|$.

In the special case of the concept $\mu X.C$ with C being $(CN \sqcup \mu Y.D)$, however, the inner loop need not reset $\mathcal{V}(Y)$ to $\emptyset$ each time the outer loop invokes the computation of $.^{\mathcal{I}}$ applied to $\mu Y.D$ with a new value assigned to $\mathcal{V}(X)$. As a matter of fact, whenever newly invoked, the inner loop can continue with the same value of $\mathcal{V}(Y)$ as that it was assigned to when the previous call of the inner loop terminated. The justification for doing so is as follows. Suppose the outer loop invokes the inner with $\mathcal{V}(X) = {f_C}^{\uparrow i}$. The inner loop then computes the least fixed point of the function of Y induced by D and $\mathcal{V}_{\langle X, {f_C}^{\uparrow i}\rangle}$ and will eventually terminate with $\mathcal{V}(Y)$ being assigned with exactly this least fixed point. The next time the inner loop is called, it will be called with $\mathcal{V}(X) = {f_C}^{\uparrow i+1}$ and, therefore, it will have to compute the least fixed point of the function of Y induced by D and $\mathcal{V}_{\langle X, {f_C}^{\uparrow i+1}\rangle}$. Now the old value of $\mathcal{V}(Y)$ is always a subset of the current least fixed point to be computed. This can be seen as follows. The old value of $\mathcal{V}(Y)$ represents the least fixed point of the function of Y induced by D and $\mathcal{V}_{\langle X, {f_C}^{\uparrow i}\rangle}$. This least fixed point is obviously a subset of the least fixed point of the function of Y induced by D and $\mathcal{V}_{\langle X, {f_C}^{\uparrow i+1}\rangle}$ in that D is formally monotonic in X and ${f_C}^{\uparrow i} \subseteq {f_C}^{\uparrow i+1}$. The next proposition states that in order to compute a least fixed point by iteration, it is actually sufficient to start with an arbitrary subset of the least fixed point to be computed rather than $\emptyset$.

The proposition will make use of the following notational convention. Assume that for every natural number $i \geq 0$ and every subset, S, of Γ, $\boldsymbol{f^i(S)}$ is defined to be S if $i = 0$, and $f(f^{i-1}(S))$ otherwise.

Proposition 12. *Assume f is an arbitrary monotonic function mapping subsets of some set, say, Γ, to subsets of Γ. Suppose, moreover, Γ has a finite cardinality of n. Then for every subset, S_0, of the least fixed point of f and for every superset, S_1, of its greatest, the least and the greatest fixed point of f coincide with $f^n(S_0)$ and $f^n(S_1)$ respectively.*

Notably, the last but one proposition is just a special case of the last one. The former restricts S_0 and S_1 to $\emptyset$ and Γ respectively. In this special case, $f^n(S_0)$ is $f^{\uparrow n}$ and $f^n(S_1)$ is $f^{\downarrow n}$.

Proof. According to Proposition 11, the least and the greatest fixed points of f coincide with $f^{\uparrow n} = f^n(\emptyset)$ and $f^{\downarrow n} = f^n(\Gamma)$ respectively. Consider two arbitrary subsets of Γ, say, S_0 and S_1, such that $S_0 \subseteq f^n(\emptyset)$ and $S_1 \supseteq f^n(\Gamma)$. It is easily seen that $f^n(S_0)$ and $f^n(S_1)$ are equal to $f^n(\emptyset)$ and $f^n(\Gamma)$ respectively. Indeed, the fact that f is monotonic along with the assumption that $S_0 \subseteq f^n(\emptyset)$ and $S_1 \supseteq f^n(\Gamma)$

algorithm : *fixp*$(T, X, \mathcal{T}, \mathcal{D}, \mathcal{V})$;
$S_0 := \mathcal{V}(X)$; $i := 0$;
repeat $j := i + 1$; $S_j := ext(C, \mathcal{T}, \mathcal{D}, \mathcal{V}_{\langle X, S_i \rangle})$ **until** $S_j = S_i$;
return S_i;

Figure 4.6: The algorithm *fixp*

entails that $f^n(S_0) \subseteq f^n(f^n(\emptyset)) = f^n(\emptyset)$ and that $f^n(S_1) \supseteq f^n(f^n(\Gamma)) = f^n(\Gamma)$. On the other hand, the monotonicity of f together with the fact that $\emptyset \subseteq S_0$ and $\Gamma \supseteq S_1$ implies also that $f^n(\emptyset) \subseteq f^n(S_0)$ and that $f^n(\Gamma) \supseteq f^n(S_1)$. But then $f^n(S_0) \subseteq f^n(\emptyset) \subseteq f^n(S_0)$ and $f^n(S_1) \supseteq f^n(\Gamma) \supseteq f^n(S_1)$. We can conclude that $f^n(S_0)$ and $f^n(S_1)$ are equal to $f^n(\emptyset)$ and $f^n(\Gamma)$ respectively, as was to be shown. □

At least as far as the special case of the concept $\mu X.C$ with C being $CN \sqcup \mu Y.D$ is concerned, this justifies that concept variables have to be initialized only once and, then, can be assigned new values in a monotonically increasing fashion. This enables us to compute the value of $(\mu X.C)^{\mathcal{I}}$ by a number of at most $\mathrm{O}(2(\sharp|\mathcal{D}| + 1)|\mu X.C|)$ calls of *ext* rather than $\mathrm{O}((\sharp|\mathcal{D}| + 1)^2|\mu X.C|)$ such calls. In the general case, $\mathrm{O}((d + 1)(\sharp|\mathcal{D}| + 1)|\mu X.C|)$ are sufficient with d being again the maximal nesting depth of fixed point operators. In the presence of fixed-point operators on roles, a maximal number of $\mathrm{O}((d + 1)(\sharp|\mathcal{D}|^2 + 1)|\mu X.C|)$ calls will do.

Figure 4.6 shows a straightforward procedure, called *fixp*, computing for any concept or role, T, which is formally monotonic in X, the least and the greatest fixed point of the function, f_T, on X induced by T and $\mathcal{V}$. If $\mathcal{V}$ is a valuation over a finite set having a cardinality of n, then *fixp* computes a set corresponding to $f_T^n(\mathcal{V}(X))$. If $\mathcal{V}(X)$ is initialized properly, that is, if it is either a subset of the least fixed point of f_T or a superset of its greatest, $f_T^n(\mathcal{V}(X))$ yields either the least or the greatest fixed point of f_T.

However, it should be clear that, if the method just described for initializing the value of $\mathcal{V}(Y)$ only once was encountered with a concept of the form $\mu X.(CN \sqcup \nu Y.D)$ rather than $\mu X.(CN \sqcup \mu Y.D)$, it would produce unsound results. This can be explained as follows. Suppose the inner loop is about to compute the greatest fixed point of the function on Y induced by C and $\mathcal{V}_{\langle X, {f_C}^{\uparrow i+1} \rangle}$. This means that at the previous time when the inner loop was invoked, it must have terminated with $\mathcal{V}(Y)$ being assigned with the greatest fixed point of the function of Y induced by C and $\mathcal{V}_{\langle X, {f_C}^{\uparrow i} \rangle}$. This old value of $\mathcal{V}(Y)$ is, however, not necessarily a *superset* of the current fixed point the inner loop is about to compute. On the contrary, the former is always a *subset* of the latter in that D is formally monotonic in X and, moreover, ${f_C}^{\uparrow i+1}$ is a subset of ${f_C}^{\uparrow i}$. In other words, the preconditions of the last proposition are in this case violated. But

the method of initializing concept and role variables only once does work for a large number of concepts and roles of $\mathcal{U}\mu$. Consider the following definition of restricted concepts and roles of $\mathcal{U}\mu$, just generalizing the corresponding one for $\mathcal{ALC}\mu$ given in Chapter 2, page 37.

Definition 49. A concept or role of $\mathcal{U}\mu$ is called **restricted** if and only if its negation normal form does not contain any subconcept or subrole of the form $\mu X.T$ (respectively $\nu X.T$) which involves in turn at least one subconcept or subrole of the form $\nu Y.T'$ (respectively $\mu Y.T'$) such that $X \neq Y$ and X occurs in T'. We denote with $\mathcal{U}\mu^{-}$ the set of all and only those concepts and roles of $\mathcal{U}\mu$ which are restricted.

In terms of Emerson and Lei [1986] restricted concepts correspond exactly to those having *alternation depth 1*. It is important to realize that $\mathcal{U}\mu^{-}$ seems to include all those fixed-point operators and even nested ones which are of practical use. It does exclude only those concepts and roles which involve a greatest fixed-point operator nested in a least fixed-point operator or vice versa such that the inner contains at least one occurrence of the concept or role variable quantified by the outer. Note in this connection that in almost all cases it is not clear at all what concept or roles which are not restricted really express. For instance, it is instrumental to try to find out what the concept $\mu X.(CN \sqcup \nu Y.(X \sqcup \exists RN{:}Y))$ is supposed to mean.

The following lemma explores in detail the worst-case complexity of the algorithm we have presented and shows that it is always polynomial in nature.

Lemma 5. *Assume $\mathcal{T}$ is an acyclic terminology of $\mathcal{U}\mu^{-}$, Q is a query of $\mathcal{U}\mu^{-}$, and $\mathcal{V}$ is a valuation over a nonempty set, say, $\mathcal{D}$. Then $ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V})$ terminates and it does so after performing at most $\mathrm{O}(n^3 \sharp|\mathcal{D}|^2 + n^2 \sharp|\mathcal{D}|^4 + n \sharp|\mathcal{D}|^2 |\mathcal{V}|)$ steps, where n is the sum of the sizes of T and $\mathcal{T}$.*

Proof. Whenever $ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V})$ is called, it first checks whether or not $E(T)$ is defined and retrieves the value if it actually exists. That is, if T was evaluated previously, the result as stored in E is retrieved. As we have argued before, the size of E is bounded above by $\mathrm{O}(n^2 + n\sharp|\mathcal{D}|^2)$. This piece of code therefore requires not more than a number of $\mathrm{O}(n^2 + n\sharp|\mathcal{D}|^2)$ steps. If this test fails, $ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V})$ checks whether $\mathcal{V}(T)$ is defined and, if this is not the case, whether T is defined in $\mathcal{T}$. Both tests can obviously be carried out by $\mathrm{O}(|\mathcal{V}|)$ and $\mathrm{O}(|\mathcal{T}|)$ steps respectively. Dependent on the syntactic shape of T, the procedure then computes the value of S by applying certain operations of basic set theory to recursively computed subsets of $\mathcal{D}$ and $\mathcal{D} \times \mathcal{D}$, where a possible call of *fixp* is to be thought of as being replaced by its procedure body. If T is of the form $\neg T_1$, for some concept or role, T_1, an additional check must be made whether there is some T_2 such that $(T_1 \doteq T_2) \in \mathcal{T}$ and then, depending on the outcome, either the negation normal form of $\neg T_1$ or that of $\neg T_2$ must be computed too.

algorithm : $holds(Q, \mathcal{KB}, \mathcal{T})$ **with** $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ *being vivid*;
global variable : E;
$E := \emptyset$;
$\mathcal{V} := \{\langle TN, \mathcal{S} \rangle : (TN \doteq \mathcal{S}) \in \mathcal{A},\ TN \notin def(\mathcal{T})\} \cup \{\langle \{a\}, \{a\} \rangle : a \in \mathcal{D}\}$;
if *there is some* $Q_{\mathcal{A}} \in \mathcal{A}$ *such that* $holds(Q_{\mathcal{A}}, \mathcal{D}, \mathcal{V}, \mathcal{T}) = false$
then **return** *true*
else **return** $holds(Q, \mathcal{D}, \mathcal{V}, \mathcal{T})$;

Figure 4.7: The algorithm $holds/3$

In each particular case, the operations of basic set theory mentioned can be computed by not more than $O(\sharp|\mathcal{D}|^2)$ steps. If T is of the form $\neg T_1$, then an additional number of $O(n)$ steps has to be taken into account for the computation of the negation normal form. On top of this, there are in each case not more than $O((\sharp|\mathcal{D}|^2 + 1)n)$ recursive calls of ext, where the factor of $\sharp|\mathcal{D}|^2 + 1$ is due to fixed-point operators. All in all, the number of steps performed by $ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V})$ is thereby bounded above by the following order of magnitude:

$$\begin{aligned} & O\Big(\big((\sharp|\mathcal{D}|^2 + 1)n\big)\big((n^2 + n\sharp|\mathcal{D}|^2) + |\mathcal{V}| + |\mathcal{T}| + (\sharp|\mathcal{D}|^2 + n)\big)\Big) \\ \leq\ & O\Big(\big((\sharp|\mathcal{D}|^2 + 1)n\big)\big(n^2 + n\sharp|\mathcal{D}|^2 + |\mathcal{V}|\big)\Big) \\ \leq\ & O(n^3\sharp|\mathcal{D}|^2 + n^2\sharp|\mathcal{D}|^4 + n\sharp|\mathcal{D}|^2|\mathcal{V}|). \end{aligned}$$

Concerning the above line of reasoning, note that $n = |T| + |\mathcal{T}|$ is always greater than 0 and it is obviously also greater than $|\mathcal{T}|$. Moreover, $\sharp|\mathcal{D}|$ is by assumption greater than 0 as well. □

Having the algorithm ext on hand, which is now guaranteed to terminate in polynomial time, it is not hard to give the remaining top level parts of an algorithm which is capable of checking $\mathcal{KB} \models_{\mathcal{T}} Q$ in polynomial time whenever $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ is vivid. As already explained, at first a valuation, $\mathcal{V}$, over $\mathcal{D}$ has to be computed which corresponds to an initial model of $\mathcal{KB}$. Because the valuation computed should not only correspond to an initial model of $\mathcal{KB}$, but should also serve as a basis for a model $\mathcal{T}$, only those assertions of $\mathcal{A}$ have to be taken into account which do not specify any concept or role name defined in $\mathcal{T}$. Of course, one must then check whether any model of $\mathcal{T}$ based on $\mathcal{V}$ is actually a model of $\mathcal{KB}$. But then it just remains to check whether such a model is a model of Q as well. Both of the latter tests can easily be done with the help of ext. The corresponding top level algorithms can be found in Figures 4.7 and 4.8.

algorithm : $holds(Q, \mathcal{D}, \mathcal{V}, \mathcal{T})$;
begin case Q **of** :

$a{:}C$	:	**return** $a \in ext(C, \mathcal{T}, \mathcal{D}, \mathcal{V})$;
$\langle a, b\rangle{:}R$	:	**return** $\langle a, b\rangle \in ext(R, \mathcal{T}, \mathcal{D}, \mathcal{V})$;
$C \doteq \{a_1, ..., a_n\}$	:	**return** $ext(C, \mathcal{T}, \mathcal{D}, \mathcal{V}) = \{a_1, ..., a_n\}$;
$R \doteq \{\langle a_1, b_1\rangle, ..., \langle a_n, b_n\rangle\}$	:	**return** $ext(R, \mathcal{T}, \mathcal{D}, \mathcal{V}) = \{\langle a_1, b_1\rangle, ..., \langle a_n, b_n\rangle\}$;
$T_1 \doteq T_2$	:	**return** $ext(T_1, \mathcal{T}, \mathcal{D}, \mathcal{V}) = ext(T_2, \mathcal{T}, \mathcal{D}, \mathcal{V})$;

end case;

Figure 4.8: The algorithm *holds*/4

Proposition 13 (runtime of *holds*). *There is a polynomial, p, such that whenever $holds(Q, \mathcal{KB}, \mathcal{T})$ is called with $\mathcal{KB}$ being a vivid knowledge base and with $\mathcal{T}$ being an acyclic terminology of $\mathcal{U}\mu^-$ and Q a query of $\mathcal{U}\mu^-$ such that $\mathcal{T}$ and Q are compatible with $\mathcal{KB}$, $holds(Q, \mathcal{KB}, \mathcal{T})$ will terminate after performing a number of steps bounded above in the sum of the sizes of Q, $\mathcal{KB}$, and $\mathcal{T}$ by p.*

Proof. Throughout the proof it is assumed that $\mathcal{KB}$ is of the form $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$. With the help of this knowledge base, $holds(Q, \mathcal{KB}, \mathcal{T})$ first computes a certain $(\mathcal{S}ig \setminus def(\mathcal{T}))$-valuation, $\mathcal{V}$, over $\mathcal{D}$. This is be done by performing not more than $O(|\mathcal{A}| + |\mathcal{T}|)$ steps and the size of the resulting valuation is bounded above by $O(|\mathcal{S}ig| \sharp |\mathcal{D}|^2)$. The procedure then calls $holds(Q_i, \mathcal{D}, \mathcal{V}, \mathcal{T})$ with a number of $\sharp|\mathcal{A}|+1$ different queries, Q_i, each of which has has a size not exceeding the maximum of the sizes of Q and $\mathcal{A}$. In each particular case, $holds(Q_i, \mathcal{D}, \mathcal{V}, \mathcal{T})$ calls in turn $ext(T_i, \mathcal{T}, \mathcal{D}, \mathcal{V})$ with some concept or role, T_i, the size of which is always bounded above by that of Q_i. Apart from this, $holds(Q_i, \mathcal{D}, \mathcal{V}, \mathcal{T})$ does not perform any computation which is relevant due to its computational costs. According to the last lemma, each call of $ext(T_i, \mathcal{T}, \mathcal{D}, \mathcal{V})$ takes not more than a number of $O(n_i^3 \sharp |\mathcal{D}|^2 + n_i^2 \sharp |\mathcal{D}|^4 + n_i \sharp |\mathcal{D}|^2 |\mathcal{V}|)$ steps, provided that n_i is $|T_i| + |\mathcal{T}|$. Note that $|T_i| + |\mathcal{T}|$ is known to be lower than or equal to $|Q_i| + |\mathcal{T}| \leq max(|Q|, |\mathcal{A}|) + |\mathcal{T}| \leq |Q| + |\mathcal{A}| + |\mathcal{T}|$. All in all the number of steps performed by $holds(Q, \mathcal{KB}, \mathcal{T})$ is bounded above by the following order of magnitude if n is the sum of the sizes of Q, $\mathcal{A}$, and $\mathcal{T}$:

$$O\Big(\big(n + |\mathcal{S}ig| \sharp |\mathcal{D}|^2\big) + \big((|\mathcal{A}| + 1)(n^3 \sharp |\mathcal{D}|^2 + n^2 \sharp |\mathcal{D}|^4 + n \sharp |\mathcal{D}|^2 |\mathcal{V}|)\big)\Big).$$

Note that for each knowledge base, $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$, $\mathcal{D}$ as well as $\mathcal{S}ig$ are supposed to be nonempty. Because of the fact that in our particular case, $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$ is vivid, $\mathcal{A}$ must be nonempty, too, since it has to contain at least one assertion extensionally specifying a member of $\mathcal{S}ig$. The above order of magnitude is bounded above by the

following one:

$$\begin{aligned} & O(n + |\mathcal{S}ig|\sharp|\mathcal{D}|^2 + n^3\sharp|\mathcal{D}|^4|\mathcal{A}| + n^2\sharp|\mathcal{D}|^4|\mathcal{A}| + n\sharp|\mathcal{D}|^2|\mathcal{V}||\mathcal{A}|) \\ \leq\ & O(n + |\mathcal{S}ig|\sharp|\mathcal{D}|^2 + n^3\sharp|\mathcal{D}|^4|\mathcal{A}| + n^2\sharp|\mathcal{D}|^4|\mathcal{A}| + n\sharp|\mathcal{D}|^4|\mathcal{S}ig||\mathcal{A}|) \\ \leq\ & O(n^3\sharp|\mathcal{D}|^4|\mathcal{A}| + n\sharp|\mathcal{D}|^4|\mathcal{S}ig||\mathcal{A}|). \end{aligned}$$

This just means that there is a polynomial, p, such that the maximal number of steps performed by $holds(Q, \mathcal{KB}, \mathcal{T})$ does not exceed $p(|Q| + |\mathcal{KB}| + |\mathcal{T}|)$. □

4.3.2 Correctness of the Algorithm

In this section we shall establish the soundness and completeness of $holds(Q, \mathcal{KB}, \mathcal{T})$ with respect to $\mathcal{KB} \models_{\mathcal{T}} Q$. That is to say, we will prove that the procedure $holds(Q, \mathcal{KB}, \mathcal{T})$ terminates returning the value "true" if and only if $\mathcal{KB} \models_{\mathcal{T}} Q$. Of course, the correctness of the subprocedures *fixp* and *ext* must be established first. The following notion will be a crucial prerequisite for the correctness of *fixp*. In effect, it states that a valuation, $\mathcal{V}$, initializes a concept or role variable, X, in such a way that enables *fixp* to compute the least or the greatest fixed point of the function on X induced by T and $\mathcal{V}$.

Definition 50. Assume $\mu X.T$ and $\nu X.T$ are fixed-point operators of $\mathcal{U}\mu$. Assume, moreover, $\mathcal{V}$ is a $\mathcal{L}$-valuation over a set, Δ. $\mathcal{L}$ has to contain at least all those atomic concepts and roles which occur in $\mu X.T$ or $\nu X.T$, except for $\top$ and ϵ. Then $\mathcal{V}$ is said to **initialize $\mu X.T$** if and only if $\mathcal{V}(X)$ is a subset of the least fixed point of the function on X induced by T and $\mathcal{V}$. Furthermore, $\mathcal{V}$ **initializes $\nu X.T$** if and only if $\mathcal{V}(X)$ is a superset of the greatest fixed point of the function on X induced by T and $\mathcal{V}$.

The following lemma establishes the correctness of $fixp(T, X, \emptyset, \mathcal{D}, \mathcal{V})$ under the assumption that $ext(T, \emptyset, \mathcal{D}, \mathcal{V}_{\langle X,S\rangle})$ works correctly. When this assumption is put into effect, $fixp(T, X, \emptyset, \mathcal{D}, \mathcal{V})$ yields exactly the least fixed point of the function f if $\mathcal{V}$ initializes $\mu X.T$ and f is the function on X induced by T and $\mathcal{V}$. If $\mathcal{V}$ initializes $\nu X.T$, then $fixp(T, X, \emptyset, \mathcal{D}, \mathcal{V})$ yields exactly the greatest fixed point of this function. This partial correctness of *fixp* is an immediate consequence of Proposition 11.

Lemma 6 (Partial Correctness of *fixp*). *Assume $\mu X.T$ is any fixed-point operator of $\mathcal{U}\mu^-$. Assume, moreover, $\mathcal{V}$ is a $\mathcal{L}$-valuation over a* finite *set, $\mathcal{D}$, such that $\mathcal{V}$ initializes $\mu X.T$. Finally, assume that for every $S \supseteq \mathcal{V}(X)$ such that $\mathcal{V}_{\langle X,S\rangle}$ initializes $\mu X.T$, $ext(T, \emptyset, \mathcal{D}, \mathcal{V}_{\langle X,S\rangle})$ returns $f(S)$ if f denotes the function on X induced by T and $\mathcal{V}$. Then $fixp(T, X, \emptyset, \mathcal{D}, \mathcal{V})$ returns the least fixed point of f. This statement holds also with $\mu X.T$ replaced with $\nu X.T$, but in this case, $fixp(T, X, \emptyset, \mathcal{D}, \mathcal{V})$ returns the greatest fixed point of f.*

Lemma 7 (Partial Correctness of *ext* for empty terminologies). *Assume T is a concept or role of $\mathcal{U}\mu^-$. Suppose, furthermore, $\mathcal{V}$ is a $\mathcal{L}$-valuation over a* finite *set, $\mathcal{D}$, such that $\mathcal{L}$ initializes all least and greatest fixed-point operators occurring in T. Finally, assume that $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}' \rangle$ is an arbitrary interpretation such that $\Delta^{\mathcal{I}} = \mathcal{D}$ and $\mathcal{V} \subseteq \mathcal{V}'$. Then $ext(T, \emptyset, \mathcal{D}, \mathcal{V})$ returns $T^{\mathcal{I}}$ whenever it is called with a value of the global variable E such that for every $\langle T', S_{T'} \rangle$ in E, $S_{T'} = T'^{\mathcal{I}}$. After termination, the new value of E is always a superset of the old and still obeys the latter condition.*

Proof. Assume that the value of the global variable E meats the condition that for every $\langle T', S' \rangle$ in E, $S' = T'^{\mathcal{I}}$. We then have to prove that $ext(T, \emptyset, \mathcal{D}, \mathcal{V})$ returns $T^{\mathcal{I}}$. The proof proceeds by induction on the structure of T.

Induction Base. As regards the induction base, we have to consider the cases when T is $\top$, ϵ, or a concept or role name.

1. Consider the case $T = \top$. By definition $ext(\top, \emptyset, \mathcal{D}, \mathcal{V})$ returns in this case $\mathcal{D}$. Recall that $\top^{\mathcal{I}}$ is defined to be $\Delta^{\mathcal{I}}$ and that $\Delta^{\mathcal{I}}$ and $\mathcal{D}$ denote exactly the same sets. It immediately follows that $ext(\top, \emptyset, \mathcal{D}, \mathcal{V})$ returns $\top^{\mathcal{I}}$.

2. Consider the case $T = \epsilon$. By definition, $ext(\epsilon, \emptyset, \mathcal{D}, \mathcal{V})$ returns in this case $\{\langle a, a \rangle : a \in \mathcal{D}\}$. Recall that $\epsilon^{\mathcal{I}}$ is defined to be $\{\langle a, a \rangle : a \in \Delta^{\mathcal{I}}\}$. It immediately follows that $ext(\epsilon, \emptyset, \mathcal{D}, \mathcal{A}, \mathcal{V})$ returns $\epsilon^{\mathcal{I}}$.

3. Consider the case $T \in \mathcal{N}$. First, recall that $\mathcal{V}$ is assumed to initialize all fixed-point operators of T. This presupposes that $\mathcal{V}$ is a $\mathcal{L}$-valuation over $\mathcal{D}$ such that $\mathcal{L}$ contains at least all atomic concepts and roles occurring in T, except for $\top$ and ϵ. In particular, this implies that $\mathcal{V}(T)$ is defined. But then $ext(T, \emptyset, \mathcal{D}, \mathcal{V})$ returns $\mathcal{V}(T)$. Because T is a concept or role name, $T^{\mathcal{I}}$ is defined to be $\mathcal{V}(T)$. Therefore, $ext(T, \emptyset, \mathcal{D}, \mathcal{V})$ returns $T^{\mathcal{I}}$.

Induction Step. All cases are trivial, except for T being a fixed-point operator of the form $\mu X.T'$ or $\nu X.T'$. We shall concentrate on least fixed-point operators since the the proof for greatest fixed-point operators proceeds analogously. In the case of a least fixed-point operator, we have to show that $ext(\mu X.T', \emptyset, \mathcal{D}, \mathcal{V})$ returns $(\mu X.T')^{\mathcal{I}}$.

If there exists a S_T such that $\langle \mu X.T', S_T \rangle$ is a member of E, then the returned value is S_T. According to the conditions imposed on the value of E, S_T is equal to $(\mu X.T')^{\mathcal{I}}$, as was to be shown.

It therefore remains to consider the case when there exists no S_T such that $\langle \mu X.T', S_T \rangle$ is a member of E. Recall that $\mathcal{V}$ is initializes all least and greatest fixed-point operators occurring in $\mu X.T'$. In particular, this implies that $\mathcal{V}(X)$ is a subset of the least fixed point of f if f be the function on X induced by T' and $\mathcal{V}$. But then $\mathcal{V}(X)$ must be defined. In this case, $ext(\mu X.T, \emptyset, \mathcal{D}, \mathcal{V})$ calls $fixp(T, X, \emptyset, \mathcal{D}, \mathcal{V})$ and then

returns the result of this call. Here, we would like to apply Lemma 6 because it would immediately prove that $\mathit{fixp}(T, X, \emptyset, \mathcal{D}, \mathcal{V})$ yields the least fixed point of f. However, in order to apply this lemma, it must be guaranteed that for every $S \supseteq \mathcal{V}(X)$ such that $\mathcal{V}_{\langle X,S\rangle}$ initializes $\mu X.T'$, $\mathit{ext}(T', \emptyset, \mathcal{D}, \mathcal{V}_{\langle X,S\rangle})$ returns $f(S)$. This looks as if the induction hypothesis can be applied to each of these calls. The induction hypothesis, however, is applicable just in case that for each of the S above, $\mathcal{V}_{\langle X,S\rangle}$ initializes all least and greatest fixed point-operators occurring in T'. Lemma 8, given below, states that this precondition is actually satisfied. According to the induction hypothesis, $\mathit{ext}(T', \emptyset, \mathcal{D}, \mathcal{V}_{\langle X,S\rangle})$ therefore returns $T'^{\mathcal{J}}$ whenever $.^{\mathcal{J}}$ is the interpretation function of an arbitrary interpretation, $\langle \Delta^{\mathcal{J}}, .^{\mathcal{J}}, \mathcal{W}\rangle$, such that $\Delta^{\mathcal{J}} = \mathcal{D}$ and $\mathcal{V}_{\langle X,S\rangle} \subseteq \mathcal{W}$. It should be clear that $f(S)$ agrees with $T'^{\mathcal{J}}$. But then we can conclude that for for every $S \supseteq \mathcal{V}(X)$ such that $\mathcal{V}$ initializes $\mu X.T'$, $\mathit{ext}(T, \emptyset, \mathcal{D}, \mathcal{V}_{\langle X,S\rangle})$ returns $f(S)$. $\mathit{ext}(\mu X.T', \emptyset, \mathcal{D}, \mathcal{V})$ then calls $\mathit{fixp}(X, T', \emptyset, \mathcal{D}, \mathcal{V})$. According to Lemma 6, this call returns the least fixed point of the function f. But then $\mathit{ext}(\mu X.T', \emptyset, \mathcal{D}, \mathcal{V})$ returns $(\mu X.T')^{\mathcal{I}}$, as was to be shown. □

Lemma 8. *Assume $\mu X.T$ is a fixed-point operator of $\mathcal{U}\mu^-$ in negation normal form. Let $\mathcal{V}$ be a $\mathcal{L}$-valuation over a set, Δ, such that $\mathcal{V}$ initializes all least and greatest fixed-point operators occurring in $\mu X.T$, including $\mu X.T$ itself. Then for every $S \supseteq \mathcal{V}(X)$ such that $\mathcal{V}_{\langle X,S\rangle}$ initializes $\mu X.T$, it holds that not only $\mathcal{V}$, but also $\mathcal{V}_{\langle X,S\rangle}$ initializes all least and greatest fixed point-operators occurring in T. An analogous statement holds for $\nu X.T$ as well, at least when requiring $S \subseteq \mathcal{V}(X)$ instead of $S \supseteq \mathcal{V}(X)$.*

Proof. We shall concentrate on the case for $\mu X.T$ as the proof for $\nu X.T$ proceeds analogously. Throughout the proof, we shall adopt the convention that the least and greatest fixed point of any monotonic function, f, is denoted by $\mathit{lfp}(f)$ and $\mathit{gfp}(f)$ respectively. In addition, we adopt the convention that $f^X_{T,\mathcal{V}}$ always denotes the function on X induced by T and $\mathcal{V}$, not only for the specific T, X, and $\mathcal{V}$ given above. The fact that $\mathcal{V}$ initializes $\mu X.T$ can then simply be expressed by $\mathcal{V}(X) \subseteq \mathit{lfp}(f^X_{T,\mathcal{V}})$.

To begin with, consider some arbitrary $S \supseteq \mathcal{V}(X)$ such that $\mathcal{V}_{\langle X,S\rangle}$ initializes $\mu X.T$. This means that $\mathcal{V}_{\langle X,S\rangle}(X) = S \subseteq \mathit{lfp}(f^X_{T,\mathcal{V}_{\langle X,\rangle}})$. To continue consider an arbitrary least or greatest fixed-point operator of the form $\mu Y.T'$ or $\nu Y.T'$ occurring in T. We have to show that not only $\mathcal{V}$, but also $\mathcal{V}_{\langle X,S\rangle}$ initializes $\mu Y.T'$ (or $\nu Y.T'$). That is to say, in the case of $\mu Y.T'$, it must be shown that $\mathcal{V}_{\langle X,S\rangle}(Y) \subseteq \mathit{lfp}(f^Y_{T',\mathcal{V}_{\langle X,S\rangle}})$. In the case of $\nu Y.T'$, $\mathcal{V}_{\langle X,S\rangle}(Y) \supseteq \mathit{gfp}(f^Y_{T',\mathcal{V}_{\langle X,S\rangle}})$ must be established.

First, consider the case $\nu Y.T'$. We have to show that $\mathcal{V}_{\langle X,S\rangle}(Y) \supseteq \mathit{gfp}(f^Y_{T',\mathcal{V}_{\langle X,S\rangle}})$. Recall that $\mu X.T$ (and therefore also T) is not only in negation normal form, but it is also restricted. According to the definition of restrictedness, this means that T' must not involve any occurrence of X. By assumption, $\mathcal{V}$ initializes $\nu Y.T'$, that is to say, $\mathcal{V}(Y) \supseteq \mathit{gfp}(f^Y_{T',\mathcal{V}})$. It should be obvious that the function $f^Y_{T',\mathcal{V}}$ coincides with

$f^Y_{T',\mathcal{V}_{\langle X,S\rangle}}$ because T' does not involve any occurrence of X. Therefore, $gfp(f^Y_{T',\mathcal{V}})$ agrees with $gfp(f^Y_{T',\mathcal{V}_{\langle X,S\rangle}})$ too. But then we have already achieved the desired conclusion:

$$\mathcal{V}_{\langle X,S\rangle}(Y) \;=\; \mathcal{V}(Y) \;\supseteq\; gfp(f^Y_{T',\mathcal{V}}) \;=\; gfp(f^Y_{T',\mathcal{V}_{\langle X,S\rangle}}).$$

The other case when there is an occurrence of $\mu Y.T'$ in T is somewhat more involved. In this case, the definition of restrictedness states that T' might involve an occurrence of X. The proof of $\mathcal{V}_{\langle X,S\rangle}(Y) \subseteq lfp(f^Y_{T',\mathcal{V}_{\langle X,S\rangle}})$ proceeds as follows:

$$\begin{array}{lll}
 & lfp(f^Y_{T',\mathcal{V}_{\langle X,S\rangle}}) & \\
= & \bigcap\{S' : f^Y_{T',\mathcal{V}_{\langle X,S\rangle}}(S') \subseteq S'\} & (according\ to\ Proposition\ 11) \\
= & \bigcap\{S' : f^X_{T',\mathcal{V}_{\langle Y,S'\rangle}}(S) \subseteq S'\} & (f^Y_{T',\mathcal{V}_{\langle X,S\rangle}}(S') = f^X_{T',\mathcal{V}_{\langle Y,S'\rangle}}(S)) \\
\supseteq & \bigcap\{S' : f^X_{T',\mathcal{V}_{\langle Y,S'\rangle}}(\mathcal{V}(X)) \subseteq S'\} & (because\ f^X_{T',\mathcal{V}_{\langle Y,S'\rangle}}\ monotonic\ \&\ \mathcal{V}(X) \subseteq S) \\
= & \bigcap\{S' : f^Y_{T',\mathcal{V}_{\langle X,\mathcal{V}(X)\rangle}}(S') \subseteq S'\} & (because\ f^X_{T',\mathcal{V}_{\langle Y,S'\rangle}}(\mathcal{V}(X)) = f^Y_{T',\mathcal{V}_{\langle X,\mathcal{V}(X)\rangle}}(S')) \\
= & \bigcap\{S' : f^Y_{T',\mathcal{V}}(S') \subseteq S'\} & (because\ \mathcal{V}_{\langle X,\mathcal{V}(X)\rangle} = \mathcal{V}) \\
= & lfp(f^Y_{T',\mathcal{V}}) & (according\ to\ Proposition\ 11) \\
\supseteq & \mathcal{V}(Y) & (\mathcal{V}\ initializes\ \mu Y.T') \\
= & \mathcal{V}_{\langle X,S\rangle}(Y) & (because\ Y\ is\ distinct\ from\ X).
\end{array}$$

The last line might call for comment. It should be clear that that $\mathcal{V}_{\langle X,S\rangle}(Y)$ does not differ from $\mathcal{V}(Y)$ if Y is distinct from X. According to our general assumption that no concept or role variable is quantified twice (see Page 80), this means particularly that Y must be distinct from X. But then $\mathcal{V}_{\langle X,S\rangle}(Y)$ actually agrees with $\mathcal{V}(Y)$. This completes the proof of Lemma 8. □

The following example is to demonstrate that the validness of the last lemma is strictly limited to *restricted* concepts and roles of $\mathcal{U}\mu$.

Example 6. Consider the following concept of $\mathcal{U}\mu$, where CN is a concept name distinct from both X and Y:

$$\mu X.(\underbrace{CN \sqcup \nu Y.\overbrace{(X \sqcap Y)}^{\stackrel{\text{def}}{=} D}}_{\stackrel{\text{def}}{=} C}).$$

It should be clear that this concept is *not* restricted because $\nu Y.D$ involves a free occurrence of X. Let $\mathcal{V}$ be an arbitrary $\mathcal{L}$-valuation over a nonempty set, Δ, such that $\mathcal{V}(CN) = \Delta$ and $\mathcal{V}(X) = \mathcal{V}(Y) = \emptyset$. It will turn out that $\mathcal{V}_{\langle X,S\rangle}$ with $S = \Delta$ does *not* initializes $\nu Y.D$, although $\mathcal{V}$ initializes both $\mu X.C$ and $\nu Y.D$, and although $\mathcal{V}_{\langle X,S\rangle}$ initializes $\mu X.C$.

It can easily be verified that the greatest fixed point of the function $f^Y_{D,\mathcal{V}}$ is $\mathcal{V}(X) = \emptyset$, while the least fixed point of $f^X_{C,\mathcal{V}}$ is $\mathcal{V}(CN) = \Delta$. The fact that $\mathcal{V}(X)$ is $\emptyset$ implies that $\mathcal{V}(X)$ is trivially a subset of the least fixed point of $f^X_{C,\mathcal{V}}$. But then $\mathcal{V}$ initializes $\mu X.C$. On the other hand, $\mathcal{V}$ initializes also $\nu Y.D$. This is due to the fact that $\mathcal{V}(Y)$ is trivially a superset of the greatest fixed point of $f^Y_{D,\mathcal{V}}$ because both are equal to the empty set. This means that $\mathcal{V}$ together with $\mu X.C$ satisfies all preconditions of Lemma 8. The conclusion of this lemma, however, is violated. In particular, $\mathcal{V}_{\langle X,S\rangle}$ does *not* initialize $\nu Y.D$ if $S = \Delta$, but $\mathcal{V}_{\langle X,S\rangle}$ initializes $\mu X.C$. To see this, first observe that $\mathcal{V}_{\langle X,S\rangle}$ initializes $\mu X.C$ because $\mathcal{V}_{\langle X,S\rangle}(X) = S = \Delta$ is trivially a subset of $lfp(f^X_{T,\mathcal{V}}) = \Delta$. On the other hand, $\mathcal{V}_{\langle X,S\rangle}(Y) = \mathcal{V}(Y) = \emptyset$ cannot be a superset of $gfp(f^Y_{T',\mathcal{V}_{\langle X,S\rangle}}) = \mathcal{V}_{\langle X,S\rangle}(X) = S = \Delta$, unless $\Delta = \emptyset$. But $\Delta = \emptyset$ contradicts the assumption that Δ is nonempty.

It remains to generalize the partial correctness of $.^{\mathcal{I}}$ to nonempty terminologies. We shall make use of the well-known fact that we can rid of terminologies in the following way; for details confer Chapter 3.2.5 of [Nebel, 1990a].

Notation 3. Assume T is an arbitrary concept or role of $\mathcal{U}\mu$ and assume $\mathcal{T}$ is an acyclic terminology of $\mathcal{U}\mu$. Then $T_{\mathcal{T}}$ stands for the concept or role of $\mathcal{U}\mu$ obtained from T by simultaneously substituting T_i for TN_i whenever $TN_i \doteq TN_i$ is a concept and role introduction of $\mathcal{T}$. This substitution process is continued until $T_{\mathcal{T}}$ contains no occurrence of $def(T)$ any more.

Fact 8. Assume T is an arbitrary concept or role, while $\mathcal{T}$ is an acyclic terminology. Then for every model, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$, of $\mathcal{T}$, it holds that $T^{\mathcal{I}} = (T_{\mathcal{T}})^{\mathcal{I}}$.

Based of this fact, the following lemma can be shown by a straightforward induction on the number of recursive calls of $.^{\mathcal{I}}$

Lemma 9. *Assume T is a concept or role of $\mathcal{U}\mu^-$, while $\mathcal{T}$ is an acyclic terminology of $\mathcal{U}\mu^-$. Assume, moreover, that $\mathcal{V}$ is a $\mathcal{L}$-valuation over a finite set, $\mathcal{D}$, such that $\mathcal{L}$ comprises at least all atomic concepts and roles, except for $\top$ and ϵ and except for those which are defined in $\mathcal{T}$. Then $ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V})$ returns exactly the same value as $ext(T_{\mathcal{T}}, \emptyset, \mathcal{D}, \mathcal{V})$. Of course, both procedures must be called with the same value of the global variable E.*

Proposition 14 (Correctness of *ext*). *Assume T is a concept or role of $\mathcal{U}\mu^-$ and assume $\mathcal{T}$ is an acyclic terminology of $\mathcal{U}\mu^-$. Moreover, suppose $\mathcal{V}$ is a $\mathcal{L}$-valuation over a set, $\mathcal{D}$, such that $\mathcal{L}$ is a set of atomic concepts and roles comprising at least all those occurring in T, except for $\top$ and ϵ. In addition, $\mathcal{L}$ is assumed to contain no concept variable. Finally, it is assumed that $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}'\rangle$ is an arbitrary model of $\mathcal{T}$*

such that $\Delta^{\mathcal{I}} = \mathcal{D}$ *and* $\mathcal{V} \subseteq \mathcal{V}'$. *Then* $ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V})$ *returns* $T^{\mathcal{I}}$ *whenever it is called with a value of the global variable* E *such that for every* $\langle T', S' \rangle$ *in* E *and* $S' = T'^{\mathcal{I}}$. *After termination, the new value of* E *is a superset of the old and still meets the latter condition.*

Proof. According to Lemma 9, $ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V})$ returns exactly the same value as $ext(T_{\mathcal{T}}, \emptyset, \mathcal{D}, \mathcal{V})$. Because $\mathcal{I}$ is assumed to be a model of $\mathcal{T}$, Proposition 14 in turn ensures that the returned value is exactly $(T_{\mathcal{T}})^{\mathcal{I}}$. But according to Observation 8, $(T_{\mathcal{T}})^{\mathcal{I}}$ is equal to $T^{\mathcal{I}}$. □

Correctness Theorem 1 (for *holds*). *Assume* $\mathcal{KB}$ *is a vivid knowledge base. Assume, moreover,* $\mathcal{T}$ *is an acyclic terminology of* $\mathcal{U}\mu^{-}$ *and* Q *is a query of* $\mathcal{U}\mu^{-}$ *such that* $\mathcal{T}$ *and* Q *are compatible with* $\mathcal{KB}$. *Then the procedure* $holds(Q, \mathcal{KB}, \mathcal{T})$ *terminates returning "true" if and only if* $\mathcal{KB} \models_{\mathcal{T}} Q$.

Proof. We concentrate on the case when Q is a query of the form $T_1 \doteq T_2$. The other cases proceed analogously. Assume $\mathcal{KB}$ is $\langle \mathcal{D}, Sig, \mathcal{A} \rangle$. Consider a $\mathcal{L}$-valuation $\mathcal{V}$ over $\mathcal{D}$ as constructed by $holds(Q, \mathcal{KB}, \mathcal{T})$. $\mathcal{L}$ contains here exactly all concept and role names of $\mathcal{S}ig$ as well as all those individual concepts composed of an individual name of $\mathcal{D}$. Then $holds(Q, \mathcal{KB}, \mathcal{T})$ returns "true" if and only if $ext(T_1, \mathcal{T}, \mathcal{D}, \mathcal{V})$ and $ext(T_2, \mathcal{T}, \mathcal{D}, \mathcal{V})$ return exactly the same values. Take an arbitrary interpretation, $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}' \rangle$, such that $\Delta^{\mathcal{I}} = \mathcal{D}$ and $\mathcal{V} \subseteq \mathcal{V}'$. It should be obvious that $\mathcal{I}$ is a model of $\mathcal{KB}$. In addition, assume that for every concept or role introduction, $TN \doteq T$, of $\mathcal{T}$, $TN^{\mathcal{I}}$ denotes exactly the same set as $T^{\mathcal{I}}$. We have the freedom to make this assumption because $\mathcal{I}$ is solely required to agree with the $\mathcal{L}$-valuation $\mathcal{V}$, but $\mathcal{L}$ does not contain any concept or role defined in $\mathcal{T}$. But then $\mathcal{I}$ is not only a model of $\mathcal{KB}$, but also a model of $\mathcal{T}$. According to Proposition 9, $\mathcal{KB} \models_{\mathcal{T}} Q$ holds if and only if $\mathcal{I}$ is a model of Q. Therefore, it remains to show that $ext(T_1, \mathcal{T}, \mathcal{D}, \mathcal{V})$ returns exactly the same value as $ext(T_2, \mathcal{T}, \mathcal{D}, \mathcal{V})$ if and only if $\mathcal{I}$ is a model of Q. According to Proposition 14, $ext(T_1, \mathcal{T}, \mathcal{D}, \mathcal{V})$ and $ext(T_2, \mathcal{T}, \mathcal{D}, \mathcal{V})$ return $T_1^{\mathcal{I}}$ and $T_2^{\mathcal{I}}$ respectively. Therefore, $holds(Q, \mathcal{KB}, \mathcal{T})$ returns "true" if and only if $T_1^{\mathcal{I}}$ and $T_2^{\mathcal{I}}$ denote exactly the same sets, as was to be shown. □

4.3.3 P-Completeness of Vivid Knowledge Bases

Complexity Theorem 4. *Consider the set of all and only those triples of the form* $\langle \mathcal{KB}, \mathcal{T}, Q \rangle$ *such that* $\mathcal{KB} \models_{\mathcal{T}} Q$. *Here, it is assumed that* $\mathcal{KB}$ *ranges over all* vivid *knowledge bases,* $\mathcal{T}$ *ranges over all acyclic terminologies of* $\mathcal{U}\mu^{-}$, *and* Q *ranges over all queries of* $\mathcal{U}\mu^{-}$ *such that* $\mathcal{T}$ *and* Q *are compatible with* $\mathcal{KB}$. *This set is also log space complete for* P, *even with* $\mathcal{T}$ *additionally restricted to be a terminology of* $\mathcal{ALC}$ *and with* Q *as a particular fixed* $\mathcal{ALC}$*-assertion.*

Proof. The fact that the set under consideration is a member of P is an immediate consequence of Proposition 13 when combined with Correctness Theorem 1. As to show that it is log space hard for P, too, we shall employ an entirely straightforward reduction from a problem which is among those most commonly known to be log space hard for P, viz. the *circuit value problem.*

Loosely speaking, a *circuit* is a finite sequence of equations each of which assigns a new propositional variable with some Boolean expression built up from those propositional variables the values of which have been determined by a previous equation of the circuit. Propositional variables are drawn from a predefined set, say, $\{X_1, ..., X_m\}$, such that m is some natural number greater than or equal to 1. In a formally precise sense a **circuit** is a finite sequence of equations of the form $(X_1 = E_1, ..., X_n = E_n)$ such that $n \leq m$. For each i with $1 \leq i \leq n$, the equation $X_i = E_i$ has to obey one of the following four syntactic shapes:

$$
\begin{aligned}
X_i &= \mathit{true};\\
X_i &= \mathit{false};\\
X_i &= \neg X_j, \text{ for some } j \text{ such that } 1 \leq j < i;\\
X_i &= X_j \wedge X_k, \text{ for some } j,k \text{ such that } 1 \leq j < i \text{ and } 1 \leq k < i.
\end{aligned}
$$

The restriction imposed on the indices of X_j and X_k is to avoid cyclic dependencies among propositional variables. Given such a circuit, the truth values assigned to the propositional variables are uniquely determined in the order $X_1, ..., X_n$. The **circuit value problem**, **CVP** for short, then, consists of the set of all those circuits for which the propositional variable, X_m, with the highest index, m, evaluates to *true.* This well-known problem was shown by Ladner [1975] to be log space hard for P.

When an arbitrary circuit, $\mathcal{C}$, is given, it is not hard to construct an acyclic terminology, $\mathcal{T}_\mathcal{C}$, of $\mathcal{ALC}$ along with a vivid knowledge base, $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$, such that the following close relationship holds: $\mathcal{C}$ is a member of CVP if and only if the query $\mathsf{t}{:}X_m$ holds in $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ with respect to $\mathcal{T}_\mathcal{C}$. This can be accomplished by assigning $\mathcal{D}$ with $\{\mathsf{t}, \mathsf{f}\}$, $\mathcal{S}ig$ with $\{\mathsf{True}, \mathsf{False}, X_1, ..., X_m\}$, and $\mathcal{A}$ with $\{\mathsf{True} \doteq \{\mathsf{t}\}, \mathsf{False} \doteq \{\mathsf{f}\}, \mathsf{t} \not\doteq \mathsf{f}\}$. Moreover, each element of $\mathcal{T}_\mathcal{C}$ is obtained from the corresponding equation in $\mathcal{C}$ by simultaneously replacing all occurrences of the symbols *true*, *false* , $\wedge$, and $=$ by True, False, $\sqcap$, and $\doteq$ respectively. Not only that the resulting knowledge base is vivid, but $\mathcal{T}_\mathcal{C}$ and $\mathsf{t}{:}X_m$ are also compatible with this knowledge base. It should be obvious that only constant memory space is needed for computing $\mathcal{T}_\mathcal{C}$ as well as $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$. We thereby have already established that CVP is log space reducible to the set under consideration. But then the latter is log space hard for P because so is CVP. □

4.4 Querying Closed & Finite Knowledge Bases

After having fixed the exact computational complexity of vivid knowledge bases, we shall now explore closed and finite ones. We will come up with a lower complexity bound for closed knowledge bases and an upper bound for finite ones. The lower bound states that closed knowledge bases give rise to a worst-case complexity which is essentially not better than that of determining validity in propositional logic: Querying closed knowledge bases will turn out to be co-NP-hard. This negative result holds even for a single *fixed* query of the standard description logic $\mathcal{ALC}$ and with no terminology taken into account. On the basis of Vardi's [1986] work on null values in databases, we shall show how queries to closed knowledge bases can be evaluated efficiently though. In particular, we will devise an algorithm for $\mathcal{U}\mu^-$ which is not only *sound* and runs in polynomial time, but also *complete* for vivid knowledge bases.

We then will give an upper complexity bound for knowledge bases without any restriction but finiteness imposed on. In particular, we shall prove that in this case the worst-case complexity is *not* greater than that of determining validity of propositional formulae. This is to say, the problem of evaluating queries is in this case a member of co-NP. The complementary problem is therefore solvable in nondeterministic polynomial time. From this we will finally conclude that deciding in $\mathcal{U}\mu^-$ all traditional kinds of terminological inferences becomes a member of co-NP just by adding an arbitrary domain-closure axiom. This result remains valid even if arbitrary finite sets of axioms and assertions are allowed.

4.4.1 Co-NP-Hardness of Closed Knowledge Bases

To begin with, we provide a lower computational complexity bound for querying closed knowledge bases.

Complexity Theorem 5. *Consider the set of all those triples of the form $\langle \mathcal{KB}, \mathcal{T}, Q \rangle$ such that $\mathcal{KB} \models_{\mathcal{T}} Q$. Here, it is assumed that $\mathcal{KB}$ ranges over all closed knowledge bases, $\mathcal{T}$ ranges over all acyclic terminologies of $\mathcal{U}\mu^-$, and Q ranges over all queries of $\mathcal{U}\mu^-$ such that $\mathcal{T}$ and Q are compatible with $\mathcal{KB}$. This set is polynomial-time hard for co-*NP*, even if $\mathcal{T}$ is empty and Q is some* fixed *$\mathcal{ALC}$-assertion.*

Proof. The claim that the set under consideration is polynomial-time hard for co-NP is an immediate consequence of Example 5, presented at the very end of the next to the last section. The essence of this example was that for an arbitrary but fixed natural number, k, the problem of k-colorability of directed finite graphs is polynomial-time reducible to the *complement* of set that is considered here. The fact that Example 5 actually gives rise to such a polynomial-time reduction can

easily be checked: The closed knowledge base constructed is obviously computable in deterministic time bounded above linearly in the size of the given graph. The size of the relevant $\mathcal{ALC}$-query, however, is only for an arbitrary but fixed k polynomially bounded in k. The restriction that k must be fixed does not matter though. In fact, according to Garey *et al.* [1976], the problem of k-colorability of finite graphs is polynomial-time hard for NP for every fixed $k \geq 3$, see also [Garey and Johnson, 1979], page 191. The complement of the set considered here is, therefore, polynomial-time hard for NP because so is the k-colorability of graphs for any fixed $k \geq 3$. □

4.4.2 A Co-NP Upper Bound for Finite Knowledge Bases

The next theorem provides an upper complexity bound for querying knowledge bases in $\mathcal{U}\mu^-$ without any restriction but finiteness imposed on knowledge bases.

Complexity Theorem 6. *Consider the set of all those triples of the form $\langle \mathcal{KB}, \mathcal{T}, Q \rangle$ such that $\mathcal{KB} \models_{\mathcal{T}} Q$. Here, it is assumed that $\mathcal{KB}$ ranges over all* finite *knowledge bases, $\mathcal{T}$ ranges over all acyclic terminologies of $\mathcal{U}\mu^-$, and Q ranges over all queries of $\mathcal{U}\mu^-$ such that $\mathcal{T}$ and Q are compatible with $\mathcal{KB}$. This set is a member of co-*NP.

Notably, this upper bound matches the lower bound for closed knowledge bases. In other words, despite the fact that the only restriction that finite knowledge bases have to meet is that they fix a finite upper bound for the interpretation domain, they essentially give rise to the same worst-case complexity as the far more restricted closed ones.

Proof. The fact that the set under consideration is a member of co-NP can be gathered with some effort from Complexity Theorem 5. To demonstrate how this can accomplished, suppose we are given an arbitrary finite knowledge base, $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$, an acyclic terminology, $\mathcal{T}$, of $\mathcal{U}\mu^-$, as well as a query, Q, of $\mathcal{U}\mu^-$. $\mathcal{T}$ and Q are, of course, assumed to be compatible with $\mathcal{KB}$. We shall devise a nondeterministic algorithm which guesses, in effect, an interpretation being a model of both $\mathcal{KB}$ and $\mathcal{T}$ and, then, checks whether or not this very interpretation is a model of Q as well. If the interpretation turns out to be *no* model of Q, then it is known that $\mathcal{KB} \not\models_{\mathcal{T}} Q$. On the other hand, the existence of such an interpretation is always guaranteed if $\mathcal{KB} \not\models_{\mathcal{T}} Q$. This gives rise to a nondeterministic method by which the *complement* of the set under consideration can be accepted in polynomial time.

As a matter of fact, the algorithm guesses a syntactic representation of the parts of interest of an interpretation rather than the interpretation itself. This syntactic representation is a vivid knowledge base of the form $\mathcal{KB}' = \langle \mathcal{D}', \mathcal{S}ig', \mathcal{A}' \rangle$: $\mathcal{D}'$ is a nonempty subset of $\mathcal{D}$ and $\mathcal{S}ig'$ is the set of all those concept and role names of $\mathcal{S}ig$ which are not defined in $\mathcal{T}$, i.e., $\mathcal{S}ig'$ is $\mathcal{S}ig \setminus def(\mathcal{T})$. For this it suffices to guess

a nonempty subset, $\mathcal{D}'$, of $\mathcal{D}$ and, then, to guess for each $TN \in \mathcal{S}ig'$ an assertion of the form $TN \doteq \mathcal{S}$. Here, $\mathcal{S}$ is some subset of $\mathcal{D}'$ if TN is a concept name, and some subset of $\mathcal{D}' \times \mathcal{D}'$ otherwise. The size of each such an assertion is obviously bounded above by $O(|TN| + |\mathcal{D}' \times \mathcal{D}'|)$. Thus all in all the size of $\mathcal{A}'$ is bounded above by $O(|\mathcal{S}ig'| + \sharp|\mathcal{S}ig'||\mathcal{D}' \times \mathcal{D}'| + |\mathcal{D}'|^2)$, including a number of not more than $\sharp|\mathcal{D}'|^2$ uniqueness axioms. This means that the size of the vivid knowledge base guessed is polynomially bounded in the length of $\mathcal{KB}$.

The particular vivid knowledge base which was guessed can clearly be viewed as a syntactic representation of a particular $\mathcal{S}ig'$-valuation over $\mathcal{D}'$. Note that it may be the case that $\mathcal{D}'$ is a *proper* subset of $\mathcal{D}$. At the same time, $\mathcal{A}$, $\mathcal{T}$, as well as Q may yet contain some individual name of $\mathcal{D}$, say, a, which is not contained in $\mathcal{D}'$. But then a has to be mapped to some element of $\mathcal{D}'$. This is why it does not suffice to guess just $\mathcal{KB}'$: A mapping, $.^*$, for the individual names not contained in $\mathcal{D}'$ has to be guessed as well. This mapping maps all individual names of $\mathcal{D} \setminus \mathcal{D}'$ to individual names of $\mathcal{D}'$. Such a mapping can obviously be represented by a subset of $(\mathcal{D} \setminus \mathcal{D}') \times \mathcal{D}'$, the size of which is polynomially bounded in the size of $\mathcal{KB}$. We assume this mapping being canonically extended to assertions, axioms, and sets of assertions and axioms. This is to say that $.^*$ applied to such an item yields the original one, except that every individual name, a, of $\mathcal{D} \setminus \mathcal{D}'$ is replaced with a^*.

$\mathcal{T}^*$ and Q^* can then easily be seen to be compatible with $\mathcal{KB}'$ because neither $\mathcal{T}^*$ nor Q^* involves any occurrence of an individual name of $\mathcal{D} \setminus \mathcal{D}'$ any more. But then, according to Complexity Theorem 5, we can check in deterministic polynomial time whether or not each assertion of the $\mathcal{A}^*$ holds in $\mathcal{KB}'$ with respect to $\mathcal{T}^*$. If the outcome is positive, then $\mathcal{KB}'$ can be thought of as representing together with the mapping $.^*$ the parts of interest of one possible model of $\mathcal{KB}$. Because $\mathcal{S}ig'$ does not contain any concept or role name which is defined in $\mathcal{T}$, this model of $\mathcal{KB}$ can be extended, so to speak, to a model of $\mathcal{T}$. According to Complexity Theorem 5, we can check also in deterministic polynomial time whether or not Q^* holds in $\mathcal{KB}'$ with respect to $\mathcal{T}^*$. If the outcome of this test is negative, whereas that of the first is positive, then $\mathcal{KB}'$ represents together with $.^*$ a model of $\mathcal{KB}$ and $\mathcal{T}$ which is no model of Q. In this sense $\mathcal{KB}'$ constitutes together with $.^*$ a counterexample falsifying $\mathcal{KB} \models_{\mathcal{T}} Q$.

The foundations of this nondeterministic procedure are laid by Lemma 10, shown below. Assume that $\mathcal{KB} \models_{\mathcal{T}} Q$ does not hold. Lemma 10 then guarantees the existence of a specific vivid knowledge base, $\mathcal{KB}' = \langle \mathcal{D}', \mathcal{S}ig', \mathcal{A}' \rangle$ with $\mathcal{D}' \subseteq \mathcal{D}$ and $\mathcal{S}ig' = \mathcal{S}ig \setminus def(\mathcal{T})$, along with a mapping, $.^* : \mathcal{D} \setminus \mathcal{D}' \to \mathcal{D}'$. Together $\mathcal{KB}'$ and $.^*$ fulfill the following property: Each assertion of $\mathcal{A}^*$ holds in $\mathcal{KB}'$ with respect to $\mathcal{T}^*$, but Q^* does not hold in $\mathcal{KB}'$. According to Complexity Theorem 5, the latter conditions can altogether be checked in deterministic time bound above by polynomial in the sum of the sizes of $\mathcal{A}^*$, Q^*, $\mathcal{KB}'$, and $\mathcal{T}^*$. Because the size of $\mathcal{KB}'$ is itself bounded above by a polynomial in the size of $\mathcal{KB}$, and so is obviously the

algortihm : *nondet_holds*$(Q, \mathcal{KB}, \mathcal{T})$ *with* $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ *being finite*;
guess *a vivid knowledge base* $\mathcal{KB}' = \langle \mathcal{D}', \mathcal{S}ig', \mathcal{A}' \rangle$
such that $\mathcal{D}' \subseteq \mathcal{D}$ *and* $\mathcal{S}ig' = \mathcal{S}ig \setminus def(\mathcal{T})$;
guess *a mapping*, $.^*$, *consistently replacing each occurrence of all individual names of* $\mathcal{D} \setminus \mathcal{D}'$ *with some individual names of* $\mathcal{D}'$;
if *for every* $Q_{\mathcal{A}} \in \mathcal{A}$, $holds(Q^*_{\mathcal{A}}, \mathcal{KB}', \mathcal{T}^*) = true$
and $holds(Q^*, \mathcal{KB}', \mathcal{T}^*) = false$
then return *false*;

Figure 4.9: The nondeterministic algorithm *nondet_holds*

size of $\mathcal{A}^*$, we end up with a nondeterministic upper time bound, polynomial in the sum of the sizes of Q, $\mathcal{KB}$, and $\mathcal{T}$. The corresponding nondeterministic algorithm can be found in Figure 4.9. □

Lemma 10. *Assume* $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ *is a closed knowledge base. Assume, moreover,* $\mathcal{T}$ *is an arbitrary acyclic terminology of* $\mathcal{U}\mu$ *and* Q *is a query of* $\mathcal{U}\mu$ *such that* $\mathcal{T}$ *and* Q *are compatible with* $\mathcal{KB}$. *Then the following two statements are equivalent.*

(a) It does not hold that $\mathcal{KB} \models_{\mathcal{T}} Q$.

(b) There exists a vivid knowledge base, $\mathcal{KB}' = \langle \mathcal{D}', \mathcal{S}ig', \mathcal{A}' \rangle$, *with* $\mathcal{D}' \subseteq \mathcal{D}$ *and* $\mathcal{S}ig' = \mathcal{S}ig \setminus def(\mathcal{T})$ *and there exists a mapping,* $.^* : \mathcal{D} \setminus \mathcal{D}' \to \mathcal{D}'$ *such that* $\mathcal{KB}' \models_{\mathcal{T}^*} Q^*_{\mathcal{A}}$ *holds for every* $Q_{\mathcal{A}} \in \mathcal{A}$, *but* $\mathcal{KB}' \models_{\mathcal{T}^*} Q^*$ *does not hold.*

Again, $.^*$ is assumed to be canonically extended to assertions, axioms, as well as sets of assertions and axioms.

Proof that (a) implies (b). Assume $\mathcal{KB} \not\models_{\mathcal{T}} Q$. This is to say, there is at least one interpretation, say, $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$, which is a model of both $\mathcal{KB}$ and $\mathcal{T}$, but which is no model of Q. According to the definition of a model of a knowledge base, this implies that $\mathcal{I}$ is also a model of $\mathcal{A}$ and $\Delta^{\mathcal{I}}$ is a nonempty subset of $\{a^{\mathcal{I}} : a \in \mathcal{D}\}$. Without loss of generality, we can clearly make the additional assumption that $\Delta^{\mathcal{I}}$ is a subset of $\mathcal{D}$. In the sequel we are going to consider a vivid knowledge base, $\mathcal{KB}' = \langle \mathcal{D}', \mathcal{S}ig', \mathcal{A}' \rangle$, defined in terms of $\mathcal{I}$ as follows:

$$\begin{aligned} \mathcal{D}' &\stackrel{\text{def}}{=} \Delta^{\mathcal{I}}, \\ \mathcal{S}ig' &\stackrel{\text{def}}{=} \mathcal{S}ig \setminus def(\mathcal{T}), \\ \mathcal{A}' &\stackrel{\text{def}}{=} \{TN \doteq \mathcal{S} : TN \in \mathcal{S}ig',\ \mathcal{V}(TN) = \mathcal{S}\}. \end{aligned}$$

It should be obvious that $\mathcal{I}$ is a model of this vivid knowledge base.

We are going to consider a mapping $.^* : \mathcal{D} \setminus \mathcal{D}' \to \mathcal{D}'$ and its canonical extension to assertions, axioms, and sets of assertions and axioms. This mapping is defined such that for every two individual names, $a, b \in \mathcal{D}$, a^* and b^* yield exactly the same individual name of $\mathcal{D}'$ if and only if $a^{\mathcal{I}} = b^{\mathcal{I}}$. A moment's thought should convince the reader that such a mapping preserves $\mathcal{I}$'s modelhood of $\mathcal{A}$, $\mathcal{T}$, and Q. That is, $\mathcal{I}$ is a model of $\mathcal{A}^*$ and $\mathcal{T}^*$ because it is a model of $\mathcal{A}$ and $\mathcal{T}$ and, moreover, $\mathcal{I}$ is no model of Q^* because it is no model of Q either. Therefore, $\mathcal{I}$ is not only a model of $\mathcal{KB}'$, but also a model of $\mathcal{T}^*$ and $Q^*_{\mathcal{A}}$, for every $Q_{\mathcal{A}} \in \mathcal{A}$. At the same time, $\mathcal{I}$ is not a model of Q^*. According to Proposition 9, this single interpretation already proves that $\mathcal{KB}' \models_{\mathcal{T}^*} Q^*_{\mathcal{A}}$ holds for every $Q_{\mathcal{A}} \in \mathcal{A}$, but $\mathcal{KB}' \models_{\mathcal{T}^*} Q^*$ does not hold. We thereby succeeded in fixing a particular vivid knowledge base $\mathcal{KB}'$, along with a mapping $.^*$, which together meet exactly the conditions of statement (b).

Proof that (b) implies (a). Consider a vivid knowledge base, say, $\mathcal{KB}' = \langle \mathcal{D}', \mathcal{S}ig', \mathcal{A}' \rangle$, such that $\mathcal{D}' \subseteq \mathcal{D}$ and $\mathcal{S}ig' = \mathcal{S}ig \setminus def(\mathcal{T})$, along with a mapping, $.^* : \mathcal{D} \setminus \mathcal{D}' \to \mathcal{D}'$. Assume $\mathcal{KB}' \models_{\mathcal{T}^*} Q^*_{\mathcal{A}}$ holds for every $Q_{\mathcal{A}} \in \mathcal{A}$, but $\mathcal{KB}' \models_{\mathcal{T}^*} Q^*$ does not hold. All conditions of statement (b) are therefore met.

According to Proposition 9, an arbitrarily chosen interpretation, $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$, being a model of both $\mathcal{KB}'$ and $\mathcal{T}^*$ is a model of each assertion of $\mathcal{A}^*$, but is no model of Q^*. Because neither $\mathcal{A}^*$, $\mathcal{T}^*$, nor Q^* involve any occurrence of some individual name of $\mathcal{D}$ not contained in $\mathcal{D}'$, without loss of generality, we can assume that $\mathcal{I}$ is chosen such that $a^{\mathcal{I}}$ is $(a^*)^{\mathcal{I}}$ whenever a is an individual name of $\mathcal{D} \setminus \mathcal{D}'$. Such an interpretation can be thought of as simulating the given mapping $.^*$. But then it is not necessary any more to apply $.^*$ to $\mathcal{A}$, $\mathcal{T}$, and Q in advance. This is why in this case $\mathcal{I}$ is not only a model of $\mathcal{A}^*$ and $\mathcal{T}^*$, but also a model of $\mathcal{A}$ and $\mathcal{T}$. For the same reason, $\mathcal{I}$ is no model of Q because it is no model of Q^* either. If we were able to show that $\mathcal{I}$ is also a model of $\mathcal{KB}$, then, according to Proposition 9, we would already have reached the desired conclusion that $\mathcal{KB} \not\models_{\mathcal{T}} Q$.

The conclusion that $\mathcal{I}$ is not only a model of $\mathcal{A}$, but also one of $\mathcal{KB}$, is readily gathered from the fact that $\mathcal{I}$ is a model of $\mathcal{KB}'$. $\Delta^{\mathcal{I}}$ is therefore a nonempty subset of $\{a^{\mathcal{I}} : a \in \mathcal{D}'\}$. But then $\Delta^{\mathcal{I}}$ is also a nonempty subset of $\{a^{\mathcal{I}} : a \in \mathcal{D}\}$ because $\mathcal{D}'$ is a subset of $\mathcal{D}$. This means that all conditions for $\mathcal{I}$ being a model of $\mathcal{KB}$ are actually met, that is, $\mathcal{I}$ is a model of $\mathcal{A}$ and $\Delta^{\mathcal{I}}$ is a nonempty subset of $\{a^{\mathcal{I}} : a \in \mathcal{D}\}$. This completes the proof of Complexity Theorem 6. □

4.4.3 A Co-NP Upper Bound for Domain-Closed Reasoning

As far as the proof just given and Complexity Theorem 6 is concerned, $\mathcal{A}$ is restricted to a set of assertions each of which extensionally specifies some concept and role name

of $\mathcal{S}ig$. This restriction is due to the fact that $\mathcal{A}$ is part of a finite knowledge base. Inspection of the proofs, however, reveals that in any of the two proofs nothing essential depends on this restriction. As a matter of fact, both proofs remain valid even with $\mathcal{A}$ generalized to an arbitrary finite set of $\mathcal{U}\mu^-$-assertions and axioms of $\mathcal{U}\mu^-$. One important restriction has to be imposed on $\mathcal{A}$ though. This restriction concerns the second condition all models of finite knowledge bases have to meet: Their interpretation domain must always be a subset of $\{a^{\mathcal{I}} : a \in \mathcal{D}\}$, for some given finite set of individual names $\mathcal{D}$. In effect, this is the crucial point in both proofs. However, such an upper bound for the possible interpretation domains can also be provided by a simple *domain-closure axiom*.

Definition 51. Assume $\mathcal{A}$ is an arbitrary set of axioms and assertions. Let $\mathcal{D} = \{a_1, ..., a_n\}$ be a finite set of individual names which contains at least all those individual names occurring in $\mathcal{A}$. Then the inclusion axiom $\top \sqsubseteq \{a_1\} \sqcup ... \sqcup \{a_n\}$ is said to be a **domain-closure axiom for $\mathcal{A}$**.

Note that for every finite set of individual names, $\mathcal{D} = \{a_1, ..., a_n\}$, and for every interpretation, $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$, it holds that $\mathcal{I}$ is a model of the domain-closure axiom $\top \sqsubseteq \{a_1\} \sqcup ... \sqcup \{a_n\}$ just in case $\Delta^{\mathcal{I}}$ is a subset of $\{a_i^{\mathcal{I}} : a_i \in \mathcal{D}\}$. This is exactly what we already identified as the crucial point of the last two proofs. This is why Complexity Theorem 6 can be extended to the more general case when $\mathcal{A}$ is a finite set of $\mathcal{U}\mu^-$-assertions and axioms of $\mathcal{U}\mu^-$ comprising at least one domain-closure axiom for it. Of course, the definition of $\mathcal{KB} \models_{\mathcal{T}} Q$ must be extended so as to deal also with $\mathcal{KB}$ being replaced by such an $\mathcal{A}$.

Definition 52. Assume $\mathcal{A} \cup \{Q\}$ is an arbitrary set of assertions and axioms. Then Q is said to be **entailed by** $\mathcal{A}$, in symbols $\mathcal{A} \models Q$, if and only if every interpretation which is a model of each element of $\mathcal{A}$ is a model of Q as well.

Recall that terminologies are nothing but finite sets of axioms of some special kind. A terminology can therefore be incorporated into the set $\mathcal{A}$ itself rather than giving its own index as in $\mathcal{KB} \models_{\mathcal{T}} Q$.

Corollary 6. *Consider the set of all those tuples of the form $\langle \mathcal{A}, Q \rangle$ such that $\mathcal{A} \models Q$. Here, Q ranges over all queries of $\mathcal{U}\mu^-$. $\mathcal{A}$ ranges over all finite sets of axioms of $\mathcal{U}\mu^-$ and $\mathcal{U}\mu^-$-assertions such that $\mathcal{A}$ contains at least one domain-closure axiom for $\mathcal{A} \cup \{Q\}$. This set is a member of co-*NP.

The inferences captured by this corollary include problems such as whether *all* undirected graphs are 3-colorable having at most 6 vertices such that at least 3 of them have an outdegree lower than or equal to 2. Of course, we should exclude all those graphs which are trivially not colorable because their edge relation is not irreflexive

(that is, those graphs which involve at least one edge emanating from a vertex leading back to the same vertex). The inference identified by the following example does satisfy all preconditions of the corollary above.

Example 7 (3-colorability). Assume $\mathcal{A}$ comprises all the following assertions and axioms:

$$\begin{array}{lcl}
\mathsf{Vertex} & \doteq & \{\mathsf{v1}, \mathsf{v2}, \mathsf{v3}, \mathsf{v4}, \mathsf{v5}, \mathsf{v6}\}, \\
\mathsf{edge} & \sqsubseteq & (\mathsf{Vertex} \times \mathsf{Vertex}) \sqcap \neg\epsilon, \\
\mathsf{edge}^{-1} & \doteq & \mathsf{edge}, \\
\top & \sqsubseteq & \exists^{\geq 3}(\top \times \mathsf{Vertex}){:}\exists^{\leq 2}\mathsf{edge}, \\
\top & \sqsubseteq & \{\mathsf{v1}\} \sqcup \{\mathsf{v2}\} \sqcup \{\mathsf{v3}\} \sqcup \{\mathsf{v4}\} \sqcup \{\mathsf{v5}\} \sqcup \{\mathsf{v6}\} \sqcup \{1\} \sqcup \{2\} \sqcup \{3\}, \\
1 \neq 2, & & \\
1 \neq 3, & & \\
2 \neq 3. & &
\end{array}$$

Then $\mathcal{A} \models \mathsf{Vertex} \sqsubseteq \bigsqcup_{i=1}^{n}(i \sqcap \forall\mathsf{edge}{:}\neg i)$ holds if and only if every undirected graph, $\langle V, E\rangle$, with the following properties is 3-colorable: Its edge relation, E, is irreflexive and V comprises at most 6 vertices of which at least 3 have an outdegree lower than or equal to 2.

Notably, the inclusion axiom $\top \sqsubseteq \exists^{\geq 3}(\top \times \mathsf{Vertex}){:}\exists^{\leq 2}\mathsf{edge}$ simulates a *cardinality restriction* in the sense of [Baader *et al.*, 1994]. In particular, it simulates $(\geq 3\ (\mathsf{Vertex} \sqcap \exists^{\leq 2}\mathsf{edge}))$. As a matter of fact, all kinds of cardinality restrictions considered by [Baader *et al.*, 1994], that is, those of the form $(\geq n\ C)$ and $(\leq m\ C)$, can be simulated in this way too. Meaning is given to such cardinality restrictions in a straightforward way by defining that an arbitrary interpretation, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$, is a model of $(\geq n\ C)$ if and only if $\sharp|C^{\mathcal{I}}| \geq n$, while it is a model of $(\leq m\ C)$ if and only if $\sharp|C^{\mathcal{I}}| \leq m$. But then cardinality restrictions of the form $(\geq n\ C)$ and $(\leq m\ C)$ are at least in a weak sense equivalent to the inclusion axioms $\top \sqsubseteq \exists^{\geq n}(\top \times \top){:}C$ and $\top \sqsubseteq \exists^{\leq m}(\top \times \top){:}C$ respectively. By being equivalent in a weak sense we refer to the restriction that they denote the same models only so far as models with nonempty domains are concerned.

4.4.4 An Approximate Algorithm for Closed Knowledge Bases

Despite the discouraging result on the lower computational complexity bound for querying closed knowledge bases, there is yet a possibility to circumvent the worst-case complexity resulting from such knowledge bases. One possibility is indicated by the work of Vardi [1986] on null values in databases: He devised an algorithm capable of dealing with them approximately. Vardi's algorithm is not only sound,

but is also complete whenever no null value is present in the database. Details can be found in Section 5 of [Vardi, 1986]. It turns out that Vardi's ideas can also be incorporated into our procedure *holds* so as to enable it to handle closed knowledge bases at least in an approximate fashion. In order to see how this can be realized, recall, if $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ is a vivid knowledge base, then $holds(Q, \mathcal{KB}, \mathcal{T})$ at first computes a particular $\mathcal{L}$-valuation, $\mathcal{V}$, over $\mathcal{D}$. This valuation is then used as a basis for subsequent computations, especially when subprocedures such as $ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V})$ are called. $\mathcal{V}$ is computed in such a way that, if $\mathcal{A}$ contains as its member the assertion $TN \doteq \mathcal{S}$ and TN is not defined in $\mathcal{T}$, then $\mathcal{V}(TN)$ is $\mathcal{S}$. In addition, $\mathcal{V}$ maps every individual concept $\{a\}$ with $a \in \mathcal{D}$ to the set $\{a\}$, so that $\mathcal{V}$ corresponds to an initial interpretation of $\mathcal{KB}$.

If $\mathcal{KB}$ is closed rather than vivid, things become somewhat more involved. Consider, for instance, a closed knowledge base, $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$, such that $\mathcal{D}$ is $\{a, b, c, x\}$, $\mathcal{S}ig$ contains as its member the concept name CN, and $\mathcal{A}$ denotes the set $\{CN \doteq \{a, b\}, a \neq b, a \neq c, b \neq c\}$. Because of the lack of any uniqueness axiom for the individual name x, this knowledge base is closed rather than vivid. The valuation $\mathcal{V}$ as computed by $holds(Q, \mathcal{KB}, \mathcal{T})$ would then yield $\{a, b\}$ as the value of $\mathcal{V}(CN)$. If $\mathcal{V}$ was part of an arbitrary model, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$, of $\mathcal{KB}$, then $\mathcal{V}$ could assign $\mathcal{V}(\{x\})$ with either $\mathcal{V}(\{a\})$, $\mathcal{V}(\{b\})$, or $\mathcal{V}(\{c\})$, or else to none of them. In the former two cases $\mathcal{V}(\{x\})$ would belong to $\mathcal{V}(CN)$, in the latter two cases it would belong to $\mathcal{D} \setminus \mathcal{V}(CN)$. Nevertheless, $\mathcal{V}(CN) = \{a, b\}$ turns out to be a good *approximation* in that it contains all those individual names which are in this sense *necessarily* contained in $\mathcal{V}(CN)$. This indicates the possibility that the original algorithm as it stands can deal with closed knowledge bases as well, at least approximately. It is not hard to see that approximations of this kind are preserved by concept-structuring primitives which are monotonic in nature such as concept conjunction and disjunction. However, this does not apply to concept and role negation. To see this, suppose in the course of performing $holds(Q, \mathcal{KB}, \mathcal{T})$, $ext(\neg CN, \mathcal{T}, \mathcal{D}, \mathcal{V})$ is called with $\mathcal{V}(CN)$ being approximated by $\{a, b\}$. As CN is some concept name of $\mathcal{S}ig$ which is not defined in $\mathcal{T}$, $ext(\neg CN, \mathcal{T}, \mathcal{D}, \mathcal{V})$ would yield $\mathcal{D} \setminus \mathcal{V}(CN)$, that is, the set $\{c, x\}$. The question is whether this is a good approximation too. Because of the presence of the uniqueness axioms $a \neq c$ and $b \neq c$, $\mathcal{V}(\{c\})$ can be identified neither with $\mathcal{V}(\{a\})$ nor with $\mathcal{V}(\{b\})$. But then $\mathcal{V}(\{c\})$ is, in fact, necessarily contained in $\mathcal{D} \setminus \mathcal{V}(CN)$. In contrast to this, it is possible to identify $\mathcal{V}(\{x\})$ either with $\mathcal{V}(\{a\})$ or with $\mathcal{V}(\{b\})$: In this case $\mathcal{V}(\{x\})$ would belong to $\mathcal{V}(CN)$ rather than to $\mathcal{D} \setminus \mathcal{V}(CN)$. This suggests that $ext(\neg CN, \mathcal{T}, \mathcal{D}, \mathcal{V})$ should yield the set $\{c\}$ as an approximation because c is the only individual name necessarily belonging to $\mathcal{D} \setminus \mathcal{V}(CN)$. This can be achieved by modifying $ext(\neg CN, \mathcal{T}, \mathcal{D}, \mathcal{V})$ in such a way that it returns $\mathcal{V}(\neg CN)$ rather than $\mathcal{D} \setminus \mathcal{V}(CN)$. Of course, an appropriate approximation must then assigned with $\mathcal{V}(\neg CN)$. In particular, for every individual name, TN, of $\mathcal{S}ig$ which is not among those defined in $\mathcal{T}$, $holds(Q, \mathcal{KB}, \mathcal{T})$ has to compute not only an approximation of $\mathcal{V}(TN)$, but also one for $\mathcal{V}(\neg TN)$. The former is computed in exactly the same way as in the origi-

nal version of *holds*; to explain how $\mathcal{V}(\neg TN)$ is computed in the general case, some additional conceptual machinery will be useful.

Definition 53. Assume $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ is an arbitrary knowledge base. If a and b are two distinct individual names of $\mathcal{D}$, then we say that ***a* disagrees in $\mathcal{KB}$ with *b*** if and only if either $a \doteq b$ or $b \doteq a$ is a member of $\mathcal{A}$. Similarly, if $\langle a_1, b_1 \rangle$ and $\langle a_2, b_2 \rangle$ are two distinct ordered pairs of individual names of $\mathcal{D}$, then $\langle \boldsymbol{a_1, b_1} \rangle$ **disagrees in $\mathcal{KB}$ with $\langle \boldsymbol{a_2, b_2} \rangle$** if and only if $\mathcal{A}$ contains as its member at least one of the uniqueness axiom $a_1 \doteq a_2$, $a_2 \doteq a_1$, $b_1 \doteq b_2$, or $b_2 \doteq b_1$.

The modification of *holds* to be made can be described as follows, where the resulting approximate versions of *holds*/3 as well as *holds*/4 will henceforth be referred to as ***approx_holds***. Whenever TN is a concept or role name of $\mathcal{S}ig \setminus def(\mathcal{T})$ such that $\mathcal{A}$ contains an assertion of the form $TN \doteq \mathcal{S}$, then $approx\_holds(Q, \mathcal{KB}, \mathcal{T})$ assigns not only $\mathcal{V}(TN)$ with $\mathcal{S}$, but additionally it assigns with $\mathcal{V}(\neg TN)$ the set of all those individual names of $\mathcal{D}$ (or ordered pairs of individual names of $\mathcal{D}$ if TN is a role name) which disagree in $\mathcal{KB}$ with *all* elements of $\mathcal{S}$. It should be obvious that if $\mathcal{KB}$ is vivid, then $\mathcal{V}(\neg TN)$ coincides with $\mathcal{D} \setminus \mathcal{S}$ (or with $(\mathcal{D} \times \mathcal{D}) \setminus \mathcal{S}$ if TN is a role name). In addition, for every individual name, $a \in \mathcal{D}$, $\mathcal{V}(\neg\{a\})$ is assigned with the set of all those individual names which disagree in $\mathcal{KB}$ with a. On the other hand, $\mathcal{V}(\{a\})$ is the set $\{a\}$, just as in the original version of *holds*. Again, if $\mathcal{KB}$ is vivid, $\mathcal{V}(\neg\{a\})$ coincides with $\mathcal{D} \setminus \mathcal{V}(\{a\})$. Only two modifications of *ext* remain to be made. The resulting approximate version of *ext* will be referred to as ***approx_ext***. First, as already explained above, $approx\_ext(\neg T, \mathcal{T}, \mathcal{D}, \mathcal{V})$ has to yield $\mathcal{V}(\neg T)$ rather than $top(T) \setminus \mathcal{V}(T)$ whenever T is an atomic concept or role not defined in $\mathcal{T}$. The second modification concerns number restrictions of the form $\exists^{\leq m} R{:}C$, implicitly involving a negation in the sense that they are equivalent to $\neg\exists^{\geq m+1} R{:}C$. The original version of $ext(\exists^{\leq m} R{:}C, \mathcal{T}, \mathcal{D}, \mathcal{V})$ would return $\{d \in \mathcal{D} : \sharp|S_R(d) \cap S_C| \leq m\}$, where S_C and S_R are assigned with $ext(C, \mathcal{T}, \mathcal{D}, \mathcal{V})$ and $ext(R, \mathcal{T}, \mathcal{D}, \mathcal{V})$ respectively. Because of the fact that both S_C as well as S_R are themselves only approximations, $ext(\exists^{\leq m} R{:}C, \mathcal{T}, \mathcal{D}, \mathcal{V})$ might include individual names not necessarily belonging to it. This is why $approx\_ext(\exists^{\leq m} R{:}C, \mathcal{T}, \mathcal{D}, \mathcal{V})$ returns the set of all those individual names which disagree in $\mathcal{KB}$ with all individual names of $\{d \in \mathcal{D} : \sharp|S_R(d) \cap S_C| \geq m + 1\}$. This does make sense because the latter set is exactly what $ext(\exists^{\geq m+1} R{:}C, \mathcal{T}, \mathcal{D}, \mathcal{V})$ would yield.

Not only does the modified algorithm share with the original one its polynomial running time, no matter whether applied to vivid or closed knowledge bases, but when applied to vivid knowledge bases, its behavior agrees with that of the original. In the case of a vivid knowledge base, $approx\_holds(Q, \mathcal{KB}, \mathcal{T})$ is therefore both sound and complete with respect to $\mathcal{KB} \models_{\mathcal{T}} Q$, at least with the same conditions imposed on Q, $\mathcal{KB}$, and $\mathcal{T}$ as in Correctness Theorem 1. At least for a certain kind of basic queries, $approx\_holds(Q, \mathcal{KB}, \mathcal{T})$ turns out also to be sound with respect to

$\mathcal{KB} \models_{\mathcal{T}} Q$. In particular, it is sound when Q is an arbitrary assertion of the form $a{:}C$ or $\langle a, b\rangle{:}R$. For such basic assertions, *approx_holds* works very well. However, for queries of the form $C \doteq D$, for instance, *approx_holds* may come up with the result that the set of those individual names necessarily belonging to C and D are exactly the same, although the two concepts may differ in the *potential* individual names belonging to them.

Soundness Theorem 1 (for *approx_holds*). *Assume $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$ is a closed knowledge base. Assume, moreover, $\mathcal{T}$ is an acyclic terminology of $\mathcal{U}\mu^{-}$ and Q is a query of $\mathcal{U}\mu^{-}$ such that $\mathcal{T}$ and Q are compatible with $\mathcal{KB}$. Finally, assume that $\mathcal{S}ig$ does not contain any concept or role name as its member which is defined in $\mathcal{T}$. If Q is either of the form $a{:}C$ or $\langle a, b\rangle{:}R$, then the procedure $approx\_holds(Q, \mathcal{KB}, \mathcal{T})$ terminates returning "true" only if $\mathcal{KB} \models_{\mathcal{T}} Q$.*

Proof. Assume $\mathcal{V}$ is the valuation as computed by $approx\_holds(Q, \mathcal{KB}, \mathcal{T})$. Consider an arbitrary vivid knowledge base, $\mathcal{KB}' = \langle \mathcal{D}', \mathcal{S}ig, \mathcal{A}'\rangle$, with $\mathcal{D}' \subseteq \mathcal{D}$. Let $.^* : \mathcal{D} \setminus \mathcal{D}' \rightarrow \mathcal{D}'$ be an arbitrary mapping such that for every query, $Q_{\mathcal{A}} \in \mathcal{A}$, $Q^*_{\mathcal{A}}$ holds in $\mathcal{KB}'$ with respect to $\mathcal{T}^*$. Finally, let $\mathcal{V}'$ be the valuation as computed by the original version of $holds(Q^*, \mathcal{KB}', \mathcal{T}^*)$. Then the following can be proven.

> Assume *approx_ext* and *ext* are called with global variables E and E' respectively. It is assumed that for every concept, C, and every individual name, a, $a^* \in E'(C)$ if $a \in E(C)$. We assume also that for every role, R, and every two individual names, a and b, $\langle a^*, b^*\rangle \in E'(R)$ if $\langle a, b\rangle \in E(R)$. Then for every concept, C, of $\mathcal{U}\mu^{-}$, and for every individual name, a, of $\mathcal{D}$, $a \in approx\_ext(C, \mathcal{T}, \mathcal{D}, \mathcal{V})$ only if $a^* \in ext(C^*, \mathcal{T}^*, \mathcal{D}', \mathcal{V}')$. Similarly, for very role, R, of $\mathcal{U}\mu^{-}$, and for every two individual names, a and b, of $\mathcal{D}$, $\langle a, b\rangle \in approx\_ext(R, \mathcal{T}, \mathcal{D}, \mathcal{V})$ only if $\langle a^*, b^*\rangle \in ext(R^*, \mathcal{T}^*, \mathcal{D}', \mathcal{V}')$.

Take, just for a moment, this claim for granted. The proof will be given later on.

Now, suppose $approx\_holds(Q, \mathcal{KB}, \mathcal{T})$ returns the value "true." We then will have to come up with the conclusion that $\mathcal{KB} \models_{\mathcal{T}} Q$. Because $\mathcal{S}ig$ is assumed contains as its member no concept or role name which is defined in $\mathcal{T}$, $approx\_holds(Q_{\mathcal{A}}, \mathcal{D}, \mathcal{V}, \mathcal{T})$ returns "true" for every $Q_{\mathcal{A}} \in \mathcal{A}$. Therefore, "true" must also be the outcome of $approx\_holds(Q, \mathcal{D}, \mathcal{V}, \mathcal{T})$. This is to say, if Q is of the form $a{:}C$, for instance, then a must be a member of $approx\_ext(C, \mathcal{T}, \mathcal{D}, \mathcal{V})$. But then, according to the claim above, a^* is known to be a member of $ext(C^*, \mathcal{T}^*, \mathcal{D}', \mathcal{V}')$. This why $holds(Q^*, \mathcal{KB}', \mathcal{T}^*)$ yields in this case "true." The same reasoning applies to queries of the form $\langle a, b\rangle{:}R$ as well. Because $\mathcal{KB}'$ is a vivid knowledge base with which $\mathcal{T}^*$ and Q^* are compatible, Correctness Theorem 1 then guarantees that $\mathcal{KB}' \models_{\mathcal{T}^*} Q^*$. At this stage it is important to recall that in all the reasoning done so far, $\mathcal{KB}'$ is assumed to be an *arbitrary* vivid knowledge base, $\mathcal{KB}' = \langle \mathcal{D}', \mathcal{S}ig', \mathcal{A}'\rangle$, such that $\mathcal{D}' \subseteq \mathcal{D}$, $\mathcal{S}ig' = \mathcal{S}ig \setminus def(\mathcal{T}) = \mathcal{S}ig$,

and $Q^*_{\mathcal{A}}$ holds in $\mathcal{KB}'$ with respect to $\mathcal{T}^*$, for every query, $Q_{\mathcal{A}} \in \mathcal{A}$. But then, according to Lemma 10, we can conclude that $\mathcal{KB} \models_{\mathcal{T}} Q$.

So it remains to prove the claim given above. The proof is by induction on the number of concept and role introductions of $\mathcal{T}$. The nontrivial part of the induction is its induction base, in which case $\mathcal{T}$ is empty. The induction base is established by induction, too, but this time by induction on the structure of T. We can clearly assume without loss of generality that T is already in negation normal form because *approx_ext* always invokes the computation of the negation normal form of negated compound concepts and roles.

As regards the induction base, we have to consider the case when T is atomic. In this case T can be $\top$, ϵ, or another atomic concept or role such that either $E(T)$ or $\mathcal{V}(T)$ is defined. All but the last case are trivial. So consider, for instance, the case when T is an atomic concept other than $\top$ such that $\mathcal{V}(T)$ is $\mathcal{S}$.

If T is an individual concept, say, $\{b\}$, then $approx\_ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V})$ yields the set $\{b\}$. But then $a \in approx\_ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V}) = \{b\}$ is possible only if $a = b$. This is why $a \in approx\_ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V}) = \{b\}$ implies that $a^* \in ext(T^*, \mathcal{T}^*, \mathcal{D}', \mathcal{V}') = \mathcal{V}'(\{b^*\}) = \{a^*\}$.

If T is a concept name, say, CN, then $approx\_ext(T, \mathcal{T}, \mathcal{D}, \mathcal{V})$ yields $\mathcal{V}(CN)$. The latter is assigned with a set of individual names, $\mathcal{S}$, such that $\mathcal{A}$ contains the assertion $CN \doteq \mathcal{S}$. On the other hand, the correctness of $ext(CN^*, \mathcal{T}^*, \mathcal{D}', \mathcal{V}')$, shown in the previous section, guarantees that this call of *ext* returns $\mathcal{S}^*$ because $Q^*_{\mathcal{A}}$ holds in $\mathcal{KB}'$ with respect to $\mathcal{T}^*$ for *every* $Q_{\mathcal{A}} \in \mathcal{A}$, including the particular case of $Q_{\mathcal{A}}$ being $CN \doteq \mathcal{S}$. But then $a \in approx\_holds(Q, \mathcal{KB}, \mathcal{T}) = \{b\}$ implies that $a^* \in ext(CN^*, \mathcal{T}^*, \mathcal{D}', \mathcal{V}') = \mathcal{S}^*$.

As regards the induction step, the only interesting cases are when T is of the form $\neg T_1$ and $\exists^{\leq m} R{:}C$. In all other cases, roughly speaking, one can make use of the monotonic nature of the relevant concept and role-structuring primitives in a straightforward way. So, first consider the case when T is a concept of the form $\neg C$. Because of the fact that T is assumed to be in negation normal form, C must be atomic. But then $approx\_ext(\neg C, \mathcal{T}, \mathcal{D}, \mathcal{V})$ returns exactly that value which is assigned with $\mathcal{V}(\neg C)$. We have to show that $a \in approx\_ext(\neg C, \mathcal{T}, \mathcal{D}, \mathcal{V}) = \mathcal{V}(\neg C)$ implies $a^* \in ext(\neg C^*, \mathcal{T}^*, \mathcal{D}', \mathcal{V}')$. In what follows it will be important to bear in mind that $\mathcal{V}$ is computed by $approx\_holds(Q, \mathcal{KB}, \mathcal{T})$ in a way that $\mathcal{V}(\neg C)$ denotes the set of all those individual names which disagree in $\mathcal{KB}$ with all individual names of $\mathcal{V}(C)$. The claim that $a^* \in ext(\neg C^*, \mathcal{T}^*, \mathcal{D}', \mathcal{V}')$ will be shown by *reductio ad absurdum*. This is to say, the contrary assumption will be shown to lead to a contradiction. Now, $ext(\neg CN^*, \mathcal{T}^*, \mathcal{D}', \mathcal{V}')$ returns $\mathcal{D}' \setminus \mathcal{V}(C)^*$ and, therefore, if a^* was no member of $ext(\neg C^*, \mathcal{T}^*, \mathcal{D}', \mathcal{V}')$, then a^* would obviously be a member of $\mathcal{V}(C)^*$. But then a^* would be identical to some individual name contained in $\mathcal{V}(C)$, say, b. As a is known to disagree in $\mathcal{KB}$ with *all* individual names of $\mathcal{V}(C)$, this would imply that

a must particularly disagree in $\mathcal{KB}$ with $b = a^*$. Therefore, $\mathcal{A}$ would have to contain as its member either $a \neq a^*$ or $a^* \neq a$. This immediately contradicts the assumption that for every assertion, $Q_{\mathcal{A}}$, of $\mathcal{A}$, $Q^*_{\mathcal{A}}$ holds in $\mathcal{KB}'$ with respect to $\mathcal{T}^*$. For, if $Q_{\mathcal{A}}$ is either $a \neq a^*$ or $a^* \neq a$, $Q^*_{\mathcal{A}}$ is in both cases $a^* \neq a^*$. It should be clear that in such a case $Q^*_{\mathcal{A}}$ cannot hold in $\mathcal{KB}'$ with respect to $\mathcal{T}^*$. The remaining cases proceed analogously. □

4.5 The Database Query Power of $\mathcal{U}\mu$

In the last section we have seen that querying vivid knowledge bases by means of the description logic $\mathcal{U}\mu^-$ is tractable in the sense of Vardi's [1982] combined complexity, This is to say, it is computable in deterministic time bounded above by a polynomial in the sum of the sizes of the given knowledge base, the terminology, and the query. We have also demonstrated that an arbitrary vivid knowledge base, say, $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$, determines an interpretation of all concept and role names of $\mathcal{S}ig$, which is unique up to renaming the elements of the interpretation domain. This fact was captured in a formally precise fashion by showing that all models of $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ are $(\mathcal{S}ig, \mathcal{D})$-isomorphic to each other. In particular, all models of $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ are $(\mathcal{S}ig, \mathcal{D})$-isomorphic to an initial interpretation of $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$. That is, they are $(\mathcal{S}ig, \mathcal{D})$-isomorphic to those models of $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ whose domain is not only $\mathcal{D}$, but deal also with the concept and role names of $\mathcal{S}ig$ in exactly the way suggested by the assertions of $\mathcal{A}$. This observation just confirms the fact that vivid knowledge bases actually serve the very purpose they are designed for: a syntactic representation of an interpretation. Of course, such an interpretation can be viewed as a relational database and so can a vivid knowledge base. Take, for instance, the vivid knowledge base $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$, we met before, where $\mathcal{D}$ is $\{\mathsf{a}, \mathsf{b}, \mathsf{table}\}$, $\mathcal{S}ig$ is $\{\mathsf{Block}, \mathsf{on}\}$, and $\mathcal{A}$ denotes the set $\{\mathsf{Block} \doteq \{\mathsf{a}, \mathsf{b}\}, \mathsf{on} \doteq \{\langle \mathsf{a}, \mathsf{b} \rangle, \langle \mathsf{b}, \mathsf{table} \rangle\}, \mathsf{a} \neq \mathsf{b}, \mathsf{a} \neq \mathsf{table}, \mathsf{b} \neq \mathsf{table}\}$. This knowledge base can clearly be thought of as one possible syntactic representation of a relational database of the form $\langle D, R_1, R_2 \rangle$ the domain, D, of which is $\{\mathsf{a}, \mathsf{b}, \mathsf{table}\}$. As one might suspect, R_1 is then a relation of rank 1 comprising a and b as its only elements. On the other hand, R_2 is a relation of rank 2 which comprises the ordered pairs $\langle \mathsf{a}, \mathsf{b} \rangle$ as well as $\langle \mathsf{b}, \mathsf{table} \rangle$ as its only elements. This very database corresponds in a one-to-one fashion to exactly that part which all initial interpretations of $\langle \mathcal{D}, \mathcal{S}ig, \mathcal{A} \rangle$ have in common. In view of this close relationship, it is interesting to note that closed knowledge bases correspond to those databases which incorporate unknown values, known as *null values* in database theory.[4]

[4]Technically speaking, closed knowledge bases correspond to what Reiter [1986] called *extended relational theories* and what Vardi [1986] called *closed-world logical databases*, whereas vivid knowledge bases would be called in Vardi's terminology *physical databases*.

4.5.1 Polynomial-Time Queries to Databases

For the following formal definition of relational databases, we presuppose the existence of a fixed, countable **universal domain**, $\boldsymbol{U}$.

Definition 54. A **type** of a relational database is a tuple of natural numbers greater than or equal to 0. If $\vec{a} = \langle a_1, ..., a_n \rangle$ is such a type, then a **relational database of type $\vec{a}$** (also called **database** for short) is a tuple of the form $\langle D, R_1, ..., R_n \rangle$ such that D is a *finite* subset of the universal domain U and, moreover, for every i with $1 \leq i \leq n$, R_i is a subset of D^{a_i}, i.e., it is an a_i-ary relation over D. The set D is called the **domain** of the relational database. For each i with $1 \leq i \leq n$, a_i is called the **rank** of R_i.

Note if a_i is 0, R_i is a subset of D^0, the set containing the empty tuple $\langle\rangle$ as its only element. Thus R_i takes in this case exactly one of the values $\emptyset$ and $\{\langle\rangle\}$. In this sense relations of rank 0 are to be thought of as Boolean relations.

From the database point of view, a natural question that arises concerns the exact expressive power of $\mathcal{U}\mu^-$ as a query language for relational databases. In database theory the expressive power of query languages is tackled in the following abstract sense: If any database query is treated as a function mapping databases to certain answer sets, then an interesting question is to what extent a given query language is able to capture the set of all such query functions which are computable in principle. The following definition formalizes this interpretation of database queries as functions mapping databases to certain answer sets. The original definition can be found in Section 2 of [Chandra and Harel, 1980], page 158.

Definition 55. Assume $\vec{a}$ is a type of a relational database and b is a natural number greater than or equal to 0. A **computable query function of type $\vec{a} \rightarrow b$** (also called **query function** for short) is a partial function, f_Q, mapping relational databases of type $\vec{a}$ to subsets of U^b such that the following three conditions are met.

1. Applied to a relational database the domain of which is D, f_Q yields a subset of D^b, at least whenever it is actually defined on the given database.
2. f_Q is a partial recursive.
3. Let $DB = \langle D, R_1, ..., R_n \rangle$ and $DB' = \langle D', R'_1, ..., R'_n \rangle$ be isomorphic in the sense that there exists a one-to-one function, h, mapping D onto D' such that for every i with $1 \leq i \leq n$, R'_i is $\{\langle h(d_1), ..., h(d_{a_i}) \rangle : \langle d_1, ..., d_{a_i} \rangle \in R_i\}$. Then $f_Q(DB')$ has to be equal to $\{\langle h(d_1), ..., h(d_b) \rangle : \langle d_1, ..., d_b \rangle \in f_Q(DB)\}$.

The first two conditions impose the restriction on f_Q that it is a partial recursive (i.e., computable) function, mapping relational data bases to relations over the domain

of the given database. The third condition is usually referred to as a *consistency criterion* and states that the result of the query function should not depend on the internal representation of the database it is applied to. Rather, databases should be treated as *sets* of tuples the particular codification of which should play no essential role whatsoever.

As databases are supposed to deal with large amounts of data, it does make sense, of course, to restrict attention to those query functions which are computable in *polynomial time* rather than imposing no upper bound at all on the time needed to compute a query function. The following definition gives a neat characterization of this class of query functions and is taken from [Chandra and Harel, 1982], page 117.

Definition 56. The set of all **polynomial-time query functions**, in symbols, **QP**, is the set of all those query functions, f_Q, which are total and for which the set $\{\langle DB, \vec{d}\rangle : \vec{d} \in f_Q(DB)\}$ is a member of P.

Whether restricted in this way or not, the notion of a computable query function does not take into account the size of the query needed to express the query function. The definition of a polynomial-time query function, for instance, relates the time needed for the computation of a query function just to the sum of the sizes of the given database and the outcome. It is for this reason that it may be the case that a query language is actually able to express a certain polynomial-time query function, but the size of the database query needed to do so increases, say, exponentially with the size of the answer set. At this level of abstraction, however, there seems to be no possibility of including the size of database queries too. This is only possible when particular query languages are stated.

4.5.2 Fixed-Point Queries to Databases

Many query languages for databases are based on the calculus of relations together with those based on predicate calculus. Recall from Chapter 3.2 that the former can be recast in terms of the predicate logic too. We shall therefore concentrate in the following discussion on predicate logic as a query language for databases. We shall pay particular attention to the question whether such query languages are able to express query functions beyond the polynomial-time computable ones and, if not, whether they can express at least *all* polynomial-time query functions. It is well-known from [Aho and Ullman, 1979] that as it stands first-order predicate logic is not able to express such an important query function as the transitive closure of a binary relation. This is why we shall consider the following extension of first-order logic incorporating fixed-point operators.

Definition 57. Let $\boldsymbol{L}$ be first-order logic with equality and no nonlogical symbols other than the predicate symbols P_1, P_2, P_3 and so on. Then $\boldsymbol{L\mu}$ is L augmented by

the following additional formation rule: Assume α is a formula of $L\mu$ such that the predicate symbol P_i, the arity of which is a_i, occurs only positively in α, that is, P_i appears in an even number of distinct subformulae of α which are of the form $\neg\beta$, where β is in each particular case an arbitrary formula of $L\mu$. If the only free variables of α are among $X_1, ..., X_n$, then both $(\boldsymbol{X_1, ..., X_n})\boldsymbol{\mu P_i.\alpha}$ and $(\boldsymbol{X_1, ..., X_n})\boldsymbol{\nu P_i.\alpha}$ are formulae of $L\mu$ as well. $L\mu$ is, of course, closed under Boolean connectives and under universal and existential quantification.

The additional formation rule introduces least and greatest fixed-point operators along much the same line as the corresponding concept and role-structuring primitives of $\mathcal{U}\mu$, though, here they are more general in that they allow least fixed point operators applied to predicate symbols of arbitrary arity. Loosely speaking, a formula of the form $(X_1, ..., X_n)\mu P_i.\alpha$, for instance, is to be thought of as representing the least P_i such that $P_i(X_1, ..., X_n) \leftrightarrow \alpha$ becomes true. An example of a formula of this kind is the following, representing the reflexive-transitive closure of the binary predicate symbol P_1:

$$(X, Y)\mu P_2.\Big(X = Y \vee \exists Z.P_1(X, Z) \wedge P_2(Z, Y)\Big).$$

On the basis of formulae of this kind, one can formulate database queries too:

Definition 58. If α is a formula of $L\mu$ the only free variables of which are among $X_1, ..., X_n$ and the only free predicate symbols of which are among $P_1, ..., P_k$, then $\boldsymbol{(X_1, ..., X_n).(P_1, ..., P_k)\alpha}$ is a **database query of $L\mu$**.

Each database query of $L\mu$ induces a unique query function. If $(X_1, ..., X_n).(P_1, ..., P_k)\alpha$ is a database query of $L\mu$ such that the predicate symbols $P_1, ..., P_k$ are of arity $a_1, ..., a_k$, respectively, then the associated query function, f_Q, is of type $\langle a_1, ..., a_k\rangle \rightarrow n$ and is given as follows. When applied to a database of type $\langle a_1, ..., a_k\rangle$, say $\langle D, R_1, ..., R_k\rangle$, the query function f_Q yields a specific n-ary relation over D: It yields the set of all those tuples $\langle d_1, ..., d_n\rangle$ of D^n such that α evaluates to true in an interpretation with interpretation domain D if X_i is assigned with d_i and P_j is interpreted as R_j, for every i and j with $1 \leq i \leq n$ and $1 \leq j \leq k$. For details, including the precise semantics of $L\mu$, the reader is referred to [Chandra and Harel, 1982], page 110f. The set of all those query functions associated with at least one database query of $L\mu$ is denoted by **FP**.

It can readily be seen that, if an arbitrary but fixed database query of $L\mu$ is given, then the relevant query function can be computed in deterministic time bounded above by a polynomial in the sum of the sizes of the input database and the output relation. Therefore, FP, the set of all those query functions associated with some database query of $L\mu$, is a subset of the set of all polynomial-time computable query functions. A proof of this fact was outlined by Chandra and Harel [1982], page 119, item (i). In the same paper, however, they prove also that FP does not cover *all* polynomial-time computable query functions. A polynomial-time computable query

function not expressible by any database query of $L\mu$ is, for instance, the one which merely checks whether the cardinality of the domain of the given database is even or not, cf. [Chandra and Harel, 1982], page 122f, also item (ii) on page 119. The same applies to such natural query functions as those counting the number of tuples contained in a relation, see page 125 of [Chandra and Harel, 1982].

Expressiveness Theorem 6 (Chandra & Harel). FP *is a proper subset of* QP.

There exist, however, query languages covering exactly the class of all polynomial-time query functions. One query language for which this has been shown to be true is Chandra and Harel's *QL* having much of a programming language computing relations over some finite domain. For details, including a proof that *QL* covers QP, see Section 4 of [Chandra and Harel, 1980].

We aim in this section at relating the expressive power of $\mathcal{U}\mu^-$ to that of existing query languages. In this connection, a specific subset of FP is of particular significance. This subset of FP is restricted to those database queries of $L\mu$ which do not involve any fixed-point operator applied to a predicate symbol whose arity is greater than 2. This is to say, subformulae of the form $(X_1, ..., X_{a_i})\mu P_i.\alpha$ and $(X_1, ..., X_{a_i})\nu P_i.\alpha$ are admissible only if the arity of the predicate symbol P_i is not greater than 2. The corresponding subset of FP will be denoted by $\mathbf{FP^{[2]}}$. This set of query functions is of particular importance in that fixed-point operators as they manifest themselves in $\mathcal{U}\mu$ (and therefore also in $\mathcal{U}\mu^-$) are subject to exactly the same restriction. Recall in this connection that, when viewed in terms of predicate logic, description logics such as $\mathcal{U}\mu$ allow *per se* only predicate symbols of arity 1 or 2, namely those corresponding to concepts and roles. Gaifman investigated the consequences of adopting this very restriction on fixed-point operators of $L\mu$ and came up with the conclusion that it actually excludes some query functions from FP. The following result was reported in [Chandra and Harel, 1982], page 116, item (2), and is ascribed there to personal communication with Gaifman.

Expressiveness Theorem 7 (Gaifman). $\mathrm{FP}^{[2]}$ *is a proper subset of* FP.

4.5.3 Database Queries of $\mathcal{U}\mu$

When viewed in terms of predicate logic, the description logic $\mathcal{U}\mu$ (and therefore also $\mathcal{U}\mu^-$) is a sublanguage of $L\mu$, even with the fixed-point operators of $L\mu$ restricted to be applicable only to predicate symbols the arity of which is not greater than 2. The query functions associated with database queries of $\mathcal{U}\mu^-$ are, therefore, a subset of $\mathrm{FP}^{[2]}$. We have, of course, to state what exactly is meant by such database queries.

Definition 59. Assume $\mathcal{L}$ is a set of concepts and roles. If T is an element of $\mathcal{L}$ such that T does not involve any occurrence of an individual name and such that the

only concept and role names occurring in T are among $TN_1, ..., TN_k$, except for the bounded ones, then $(\boldsymbol{TN_1, ..., TN_k}).\boldsymbol{T}$ is said to be a **database query of $\mathcal{L}$**.

We precluded individual names from being part of a database query of $\mathcal{L}$ because constants usually are not admissible in database queries of $L\mu$ either. However, nothing but some additional complexity in notation prevents us from including individual names.

Again, each database query of $\mathcal{L}$ induces a unique query function. In particular, if $(TN_1, ..., TN_k).T$ is a database query of $\mathcal{L}$, then the associated query function, f_Q, is of type $\langle a_1, ..., a_k\rangle \to b$, where the a_i's and b are either 1 or 2, dependent on whether in each particular case TN_i and T are concepts or roles respectively. When applied to a database of type $\langle a_1, ..., a_k\rangle$, say, $\langle D, R_1, ..., R_k\rangle$, the query function f_Q then yields $T^{\mathcal{I}}$, where $.^{\mathcal{I}}$ is the interpretation function of an arbitrary interpretation of the form $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}\rangle$ such that $\Delta^{\mathcal{I}}$ is D and, moreover, for every i with $1 \leq i \leq k$, $\mathcal{V}(T_i)$ is R_i. For every set of concepts and roles, $\mathcal{L}$, we denote with $\boldsymbol{Q\mathcal{L}}$ the set of all those query functions associated with some database query of $\mathcal{L}$.

A natural question that arises in this connection concerns the relationship between querying databases by database queries of, say, $\mathcal{U}\mu$, and querying corresponding vivid knowledge bases by $\mathcal{U}\mu$-assertions. It should not be surprising that this relationship is actually very close. Take an arbitrary database query of $\mathcal{U}\mu$, say, $(TN_1, ..., TN_k).T$, the associated query function of which will be denoted by f_{Q_T}. We are going to query an arbitrary database, $DB = \langle D, R_1, ..., R_k\rangle$, of the appropriate type, that is, the rank of each R_i $(1 \leq i \leq k)$ is 1 if TN_i is a concept name, and 2 otherwise. If the domain of the database is treated as a set of individual names, then such a database can clearly be represented directly by a vivid knowledge base. In particular, it can be represented by $\mathcal{KB} = \langle \mathcal{D}, \mathcal{S}ig, \mathcal{A}\rangle$, where $\mathcal{D}$ is D, $\mathcal{S}ig$ is $\{TN_1, ..., TN_k\}$ and $\mathcal{A}$ is the union of the set $\{TN_i \doteq R_i : 1 \leq i \leq k\}$ with the set of all uniqueness axioms over $\mathcal{D}$. Then for every element, a, of D, it holds that $a \in f_{Q_T}(DB)$ if and only if $\mathcal{KB} \models_\emptyset a{:}T$. In this way a query function associated with some database query of $\mathcal{U}\mu$ can be simulated.

We have already argued that, when viewed in terms of predicate logic, $\mathcal{U}\mu^-$ is a sublanguage of $L\mu$, even when the latter is restricted such that fixed-point operators are applicable to unary and binary predicate symbols only. But then it is immediately obvious that $Q\mathcal{U}\mu^-$ is a subset of $\mathrm{FP}^{[2]}$. Recall that $\mathrm{FP}^{[2]}$ restricts the arity of those predicate symbols to which fixed-point operators are applied, but all others are not subject to any restriction at all. This is why $\mathrm{FP}^{[2]}$ includes query functions of type $\langle a_1, ..., a_n\rangle \to b$ with at least one a_i $(1 \leq i \leq n)$ being greater than 2. Neither $\mathcal{U}\mu$ nor $\mathcal{U}\mu^-$ does include any such query function. Therefore, neither $Q\mathcal{U}\mu$ nor $Q\mathcal{U}\mu^-$ covers $\mathrm{FP}^{[2]}$.

Expressiveness Theorem 8. *$Q\mathcal{U}\mu^-$ is a proper subset of* $\mathrm{FP}^{[2]}$.

Proof. We have just argued that $Q\mathcal{U}\mu^-$ and FP$^{[2]}$ do not denote the same class of query functions. So we shall concentrate on proving that the first is a subset of the second. We have to show that all query functions associated with at least one database query of $\mathcal{U}\mu^-$ are included in FP$^{[2]}$.

Consider an arbitrary query of $\mathcal{U}\mu^-$, say, $(TN_1, ..., TN_n).T$. In what follows it is presupposed that each TN_i with $1 \leq i \leq n$ has a unique predicate symbol associated with it, say, P_{TN_i}. In addition, each concept and role variable X appearing in T, no matter whether bounded or not, is associated with a unique predicate symbol, say, P_X. Of course, predicate symbols associated with a concept must be unary, whereas those associated with a role must be binary. On the basis of $(TN_1, ..., TN_n).T$, we are going to construct a particular database query of $\mathcal{U}\mu^-$. This query is as follows:

$$\begin{array}{ll} (X).(P_{TN_1}, ..., P_{TN_n})\widetilde{T}(X) & \text{if } T \text{ is a concept;} \\ (X,Y).(P_{TN_1}, ..., P_{TN_n})\widetilde{T}(X,Y) & \text{if } T \text{ is a role.} \end{array}$$

The function $\widetilde{\cdot}$ maps a concept of $\mathcal{U}\mu$ along with a variable to a formula of $L\mu$. Similarly, a role of $\mathcal{U}\mu$ is mapped along with a ordered pair of variables to a formula of $L\mu$. This function is inductively defined as follows:

$$\widetilde{\boldsymbol{C}}(\boldsymbol{X}) \;=\; \begin{cases} P_C(X) & \textit{if } C \textit{ is a concept name;} \\ true & \textit{if } C \textit{ is } \top; \\ \widetilde{C_1}(X) \wedge \widetilde{C_2}(X) & \textit{if } C \textit{ is } C_1 \sqcap C_2; \\ \widetilde{C_1}(X) \vee \widetilde{C_2}(X) & \textit{if } C \textit{ is } C_1 \sqcup C_2; \\ \neg\, \widetilde{D}(X) & \textit{if } C \textit{ is } \neg D; \\ \exists Y_1. ... \exists Y_n.\ \bigwedge_{i=1}^{n} \widetilde{R}(X,Y_i) \wedge \widetilde{D}(Y_i) & \textit{if } C \textit{ is } \exists^{\geq n} R{:}D; \\ \neg\, \exists Y_1. ... \exists Y_{m+1}.\ \bigwedge_{i=1}^{m+1} \widetilde{R}(X,Y_i) \wedge \widetilde{D}(Y_i) & \textit{if } C \textit{ is } \exists^{\leq m} R{:}D; \\ (X)\mu P_{X_i}.\widetilde{D}(X) & \textit{if } C \textit{ is } \mu X_i.D; \\ (X)\nu P_{X_i}.\widetilde{D}(X) & \textit{if } C \textit{ is } \nu X_i.D. \end{cases}$$

$$\widetilde{\boldsymbol{R}}(\boldsymbol{X},\boldsymbol{Y}) \;=\; \begin{cases} R(X,Y) & \textit{if } R \textit{ is a role name;} \\ X = Y & \textit{if } R \textit{ is } \epsilon; \\ \widetilde{R_1}(X,Y) \wedge \widetilde{R_2}(X,Y) & \textit{if } R \textit{ is } R_1 \sqcap R_2; \\ \widetilde{R_1}(X,Y) \vee \widetilde{R_2}(X,Y) & \textit{if } R \textit{ is } R_1 \sqcup R_2; \\ \neg\, \widetilde{S}(X,Y) & \textit{if } R \textit{ is } \neg S; \\ \exists Z.\widetilde{R_1}(X,Z) \wedge \widetilde{R_2}(Z,Y) & \textit{if } R \textit{ is } R_1 \circ R_2; \\ \forall Z.\widetilde{R_1}(X,Z) \vee \widetilde{R_2}(Z,Y) & \textit{if } R \textit{ is } R_1 \oplus R_2; \\ \widetilde{S}(Y,X) & \textit{if } R \textit{ is } S^{-1}; \\ \widetilde{C}(X) \wedge \widetilde{D}(Y) & \textit{if } R \textit{ is } C \times D; \\ (X,Y)\mu P_{X_i}.\widetilde{S}(X,Y) & \textit{if } R \textit{ is } \mu X_i.S; \\ (X,Y)\nu P_{X_i}.\widetilde{S}(X,Y) & \textit{if } R \textit{ is } \nu X_i.S. \end{cases}$$

The variables Z and Y_i must be fresh variables drawn from $L\mu$.

It is easily seen that the resulting formulae are, in fact, formulae of $L\mu$: The syntactic restriction imposed on $L\mu$'s fixed-point operators, is met because a corresponding restriction is invoked on fixed-point operators of $\mathcal{U}\mu$ as well. In this connection, it is important to note that the notion of formal monotonicity in $\mathcal{U}\mu$ is defined in such a way that any concept or role name is formally monotonic in $\exists^{\leq m} R{:}C$ just in case it occurs both in C as well as in R solely negatively (see page 31). This situation is reflected by the negation sign occurring in the resulting formulae of $L\mu$. Moreover, it should be obvious that the resulting formula of $L\mu$ involve only those fixed-point operators which are applied to unary and binary predicate symbols.

The fact that the query functions associated with the database queries $(TN_1, ..., TN_n).C$ and $(TN_1, ..., TN_n).R$ of $\mathcal{U}\mu$ agree with those associated with the corresponding database queries of $L\mu$ can be shown by induction on the structure of C and R. The proof, however, is as straightforward as tedious. We therefore omit the full proof. □

By the transitivity of the *proper subset* relation, we are now in a position to conclude that $Q\mathcal{U}\mu^-$ is not only a proper subset of $\mathrm{FP}^{[2]}$, but also one of FP and QP. Among others, this implies that $Q\mathcal{U}\mu^-$ does not cover such simple polynomial-time query functions as those counting the number of tuples of a relation because these query functions are not included in FP either.

We should not close this section without emphasizing that $\mathcal{U}\mu^-$ is the only query language mentioned in this section which is tractable in the sense of Vardi's [1982] combined complexity. For $L\mu$ is known to be EXPTIME-complete in the sense of the combined complexity. This remains true even with fixed-point operators restricted in such a way that they are applicable to predicate symbols whose arity is not greater than 2 [Vardi, 1982].

4.6 Discussion

In this chapter singled out a subclass of terminological reasoning in $\mathcal{U}\mu^-$ which is decidable, i.e., can in principle be mechanized. The main result was that terminological reasoning in this powerful description logic becomes not only decidable, but also tractable just by replacing a customary knowledge base by a vivid (also called physical) knowledge base. The characteristic of a vivid knowledge base is that it denotes, in effect, a single model rather than a number of models. We were concerned with the problem of checking in each particular case whether a query, Q, holds in a knowledge base, $\mathcal{KB}$, with respect to a terminology $\mathcal{T}$. We proved this problem to be decidable in deterministic time bounded above by a fixed polynomial in the sum of the sizes of $\mathcal{KB}$, Q, and $\mathcal{T}$ if the following conditions are met:

(a) The knowledge base $\mathcal{KB}$ fixes an interpretation domain, say, $\mathcal{D}$, consisting of a finite number of individual names with a general unique-name assumption imposed on.

(b) At least all those concept and role names are specified extensionally in $\mathcal{KB}$ which occur in Q or $\mathcal{T}$, except for those defined in $\mathcal{T}$. In addition, $\mathcal{D}$ has to contain at least all those individual names which occur in $\mathcal{T}$ or Q.

(c) The terminology $\mathcal{T}$ is acyclic.

This result is of interest both from Halpern and Vardi's [1991] model checking point of view as well as from the viewpoint of description logics. From Halpern and Vardi's point of view, our result singles out a useful fragment of first-order logic which gives rise to tractable model checking, even when this fragment is enriched by fixed-point operators. As opposed to the corresponding tractability results for fixed-point languages based on full first-order logic [Vardi, 1982], ours is to be understood in terms of combined complexity rather than the far weaker notion of data complexity.

The description logic's point of view suggests the following interpretation: Whenever we are able to abandon any kind of incompleteness from the knowledge base, we gain not only tractability, but also a great deal of expressive power that covers every concept and role-structuring primitive one could ask for.[5] In this connection it should be emphasized that the description logic $\mathcal{U}\mu^-$ under consideration is based on Patel-Schneider's universal description logic $\mathcal{U}$. The description logic $\mathcal{U}$ is *universal* in that it incorporates every traditional concept and role structuring primitive. In particular, it encompasses all those primitives attributed to the very system which has attracted most attention in the realm of terminological knowledge representation, the KL-ONE-system. Notably, Patel-Schneider [1989b] as well as Schmidt-Schauß [1989] proved that even in very small sublanguages of KL-ONE's description logic, subsumption is undecidable.

In addition to the concept and role-structuring primitives of $\mathcal{U}$, $\mathcal{U}\mu^-$ includes additional primitives which provide a general framework for recursive concept and role definitions. This extension of $\mathcal{U}$ allows for an explicit representation of recursion by least and greatest fixed-point operators in a way that condition (c), requiring the given terminology to be acyclic, is never violated. We deal not only with fixed-point operators on concepts, but also with those on *roles*. Despite the fact that in $\mathcal{U}\mu^-$ only those fixed-point operators are admissible which have an *alternation depth* of at most 1, as Emerson and Lei [1986] would call them, this extension of $\mathcal{U}$ is yet able to capture *mutual* recursion.

[5] The observation that tractability can be gained in principle when abandoning any kind of incomplete knowledge has been made previously by Donini *et al.* [1992]. However, their observation concerns solely a description logic which is very pure in expressive power. Moreover, Donini *et al.*'s result does not take any nonempty terminology into account.

Our tractability result is the *only* in existence for terminological reasoning, at least when the fundamental assumption that terminologies are *nonempty* is put into effect. Although all other tractability results retreat either to description logics which are pure in expressive power or to nonstandard semantics, all these results presuppose an *expansion* process applied to terminologies in advance.[6] This expansion process replaces all occurrences of those concept and role names which are defined in the given terminology by the right-hand side of the relevant concept or role introduction, confer Chapter 3.2.5 of [Nebel, 1990a] for details. In the course of such an expansion process, however, a terminology may increase by an exponential. The overall upper complexity bound obtained by all but our tractability result is therefore still *exponential* in nature. So far as standard terminological reasoning is concerned, Nebel [1990b] proved that in the presence of nonempty terminologies, no tractability result can be achieved, unless P is NP. This holds even with the expressive power limited to that of the very weakest description logic.

Not only is the subclass of terminological inferences that we singled out tractable, but it is also *useful*. Many examples show this. In the previous chapter we saw that such standard objects as directed acyclic graphs, trees, and binary trees can easily be defined by means of an acyclic terminology of $\mathcal{U}$. With the help of the additional fixed-point operators of $\mathcal{U}\mu^-$, even objects such as balanced binary trees as well as an AND-OR graph for which there exists at least one well-founded solution can be captured. Moreover, it should be clear that any graph or any finite collection of graphs can be directly represented by a single vivid knowledge base. But then the tractable subclass of terminological inferences we singled out includes the retrieval of exactly those graphs which are an instance of any of the objects just mentioned. This is certainly also interesting from the database point of view. In fact, the restriction imposed on knowledge bases so as to obtain tractability, results in what is usually called a relational database. Our result thus shows that $\mathcal{U}\mu^-$ can serve as a powerful but tractable query language for relational databases of this kind. From this point of view, it is important to note that we know from the previous chapter that even the first-order fragment of $\mathcal{U}$ is *strictly* stronger in expressive power than the well-known calculus of binary relations, the basis for many traditional database query languages [Codd, 1971]. But then the question arises how powerful $\mathcal{U}\mu^-$ is compared to other more traditional database query languages. This issue was also explored in this chapter. It turned out that $\mathcal{U}\mu$ (and therefore also $\mathcal{U}\mu^-$) is strictly weaker than a well-known fixed-point language based on full first-order logic. However, this is not surprising at all because the latter is (in contrast to $\mathcal{U}\mu^-$) intractable in the sense of the combined complexity. Of course, tractability cannot be achieved for free.

Our tractability result suggests that the main source of computational complexity of terminological reasoning is the ability to express incomplete knowledge. This finding

[6] This applies to the work on restricted languages carried out by Levesque and Brachman [1987], Lenzerini and Schaerf [1991], Donini *et al.* [1991], and Buchheit *et al.* [1994], but also to Patel-Schneider's [1989a] results for nonstandard semantics.

is confirmed by another outcome of this chapter. It states that even a quite limited form of incomplete knowledge by means of what is called in databases *null value* causes co-NP-completeness. The co-NP-completeness remains valid even with terminologies restricted to be empty and with queries additionally restricted to be some *fixed* query of the standard description logic $\mathcal{ALC}$. But then the co-NP-completeness holds not only in the sense of combined complexity, but also in the sense of data complexity. Based on Vardi's [1986] work on null values in databases, we were yet able to give an algorithm capable of dealing with null values soundly though approximately. The algorithm runs not only in polynomial time, but is also complete whenever no null value is present.

However, this is not to say that model checking could *replace* ordinary terminological reasoning. On the contrary, abandoning any kind of incomplete knowledge and allowing for an unlimited use of incomplete knowledge are just two extreme points on a wide-ranged spectrum. One should aim for a happy mean. An auspicious compromise was already indicated in this this chapter: We presented a result concerning terminological reasoning in a more traditional sense. The result was that in $\mathcal{U}\mu^-$, entailment with respect to a finite set of arbitrary assertions and axioms becomes a member of co-NP just by adding a domain-closure axiom. Such a single domain-closure axiom restricts, in effect, the interpretation domain to a finite number of individual names. Notably, in this case terminological reasoning in $\mathcal{U}\mu^-$ has essentially the same worst-case complexity as the very weakest description logic's setting with acyclic terminologies only. From a theoretical point of view, the membership of co-NP means that terminological reasoning shares with propositional logic the desirable property that counterexamples are always not only short, that is, polynomial length-bounded, but also verifiable in deterministic polynomial time. From a practical point of view, especially from the viewpoint of deterministic computations, this means that in the presence of a domain-closure axiom, terminological reasoning in $\mathcal{U}\mu^-$ is yet decidable, in exponential time. This is because a nondeterministic guess of an appropriate counterexample can be simulated by exhaustive search. This proves that decidability can be gained just in return for a simple domain-closure axiom. Requiring the existence of a domain-closure axiom is to most applications no essential restriction at all. On the other hand, terminological reasoning seems not to become applicable until an expressive power such as that of $\mathcal{U}\mu^-$ has been attained. Practical algorithms obeying an acceptable average case complexity, however, still await development.

Another promising direction for future research is a generalization of our results to the nonstandard semantics which Patel-Schneider [1987] developed for $\mathcal{U}$ on the basis of Belnap's [1977] four-valued semantics. Interpretation functions of this kind of semantics map, for instance, a concept to a pair of subsets of the domain rather than to a single subset. One subset is to be interpreted in exactly the same way as in the traditional case, that is, it is the set of those objects belonging to the given concept. In contrast to this, the second subset is to be thought of as set of those objects belonging to the *complementation* of the concept. Patel-Schneider's semantics deviates from

standard semantics in that these two subsets are neither required to cover the whole domain nor are they required to be disjoint. In other words, an arbitrary object of the domain may belong to a concept, to its complementation, to both, or to neither. It is in this sense that this semantics is *four-valued* in nature. This makes it possible to handle not only *indifferent*, but also *inconsistent* information. Formally, this semantic variant is given in terms of what can be called *extended interpretations*. Such an extended interpretation, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}, .^{\overline{\mathcal{I}}}, \overline{\mathcal{V}} \rangle$, consists of two interpretations having the same interpretation domain. The interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V} \rangle$ is to be understood in the standard sense, whereas $\langle \Delta^{\mathcal{I}}, .^{\overline{\mathcal{I}}}, \overline{\mathcal{V}} \rangle$ is a so-called *complementary* interpretation. Complementary interpretations are defined exactly as an ordinary interpretations, except that concept and role-structuring primitives are interpreted in the same way as their *duals* in standard interpretations. For instance, $(C \sqcap D)^{\overline{\mathcal{I}}}$ is defined to be $C^{\overline{\mathcal{I}}} \cup D^{\overline{\mathcal{I}}}$. This reflects the situation that an object belongs to the complementation of $C \sqcap D$ just in case it belongs either to the complementation of C or to the complementation of D. Both $.^{\mathcal{I}}$ and $.^{\overline{\mathcal{I}}}$ are also nonstandard in their interpretation of negation. In particular, $\neg T^{\mathcal{I}}$ is defined to be $T^{\overline{\mathcal{I}}}$, while $\neg T^{\overline{\mathcal{I}}}$ is $T^{\mathcal{I}}$. But then $\neg T^{\mathcal{I}}$ and $T^{\mathcal{I}}$ are neither necessarily disjoint nor do they always cover the whole interpretation domain $\Delta^{\mathcal{I}}$ (or $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ if T is a role). For details, the reader is referred to Chapter 5.3.1 of [Patel-Schneider, 1987].

Although this four-valued semantics was originally developed for $\mathcal{U}$, it can be extended to deal with the additional fixed-point operators of $\mathcal{U}\mu$ as well.[7] This nonstandard semantics makes it possible to relax the notion of a vivid knowledge base considerably by allowing for indifferent as well as inconsistent knowledge. Suppose, for instance, it is known that Hillary is a first lady, but it is not clear at all whether she is President or not. Suppose also that, as far as Bill is concerned, it is known that he is definitely no first lady (because of his sex), but there are contradictory assertions whether he is President or not. This very situation is captured by a nonstandard vivid knowledge base comprising the following assertions:

$$\begin{aligned} \mathsf{FirstLady} &\doteq \{\mathsf{Hillary}\}, \\ \neg\mathsf{FirstLady} &\doteq \{\mathsf{Bill}\}, \\ \mathsf{President} &\doteq \{\mathsf{Bill}\}, \\ \neg\mathsf{President} &\doteq \{\mathsf{Bill}\}. \end{aligned}$$

[7] Just for the readers familiar with it, a four-valued semantics for $\mu X.T$, for instance, as follows. Take an arbitrary extended interpretation, say, $\langle \Delta^{\mathcal{I}}, .^{\mathcal{I}}, \mathcal{V}, .^{\overline{\mathcal{I}}}, \overline{\mathcal{V}} \rangle$. Let f_T be the so-called *function on X induced by T*, $\mathcal{V}$, and $\overline{\mathcal{V}}$. If Γ is either $\Delta^{\mathcal{I}}$ or $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, depending on whether $\mu X.T$ is a concept or a role, then this function maps every subset, S, of Γ to $T^{\mathcal{J}}$. Here, $.^{\mathcal{J}}$ is the interpretation function of an arbitrary extended interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{J}}, \mathcal{V}_{\langle X,S \rangle}, .^{\overline{\mathcal{J}}}, \overline{\mathcal{V}} \rangle$. Let $\overline{f}_T$ be the corresponding *complementary* function mapping $S \subseteq \Gamma$ to $T^{\overline{\mathcal{J}}}$, for an arbitrary extended interpretation $\langle \Delta^{\mathcal{I}}, .^{\mathcal{J}}, \mathcal{V}, .^{\overline{\mathcal{J}}}, \overline{\mathcal{V}}_{\langle X,S \rangle} \rangle$. Then $(\mu X.T)^{\mathcal{I}}$ is defined to be $\bigcap\{S \subseteq \Gamma : S \subseteq f_T(S)\}$, whereas $(\mu X.T)^{\overline{\mathcal{I}}}$ is $\bigcup\{S \subseteq \Gamma : S \supseteq \overline{f}_T(S)\}$. This means that $(\mu X.T)^{\mathcal{I}}$ is handled just as in the standard semantics, but $(\mu X.T)^{\overline{\mathcal{I}}}$ is dealt with as if it were a greatest fixed-point operator.

Observe that this knowledge base is four-valued in that its knowledge about Bill's status as President is contradictory, while Hillary's presidential status is indefinite. If we pose the following two queries to this knowledge base, we obtain in the former case a definite negative answer, but in the latter case still an indefinite answer:

$$\begin{aligned} \mathcal{KB} &\models \mathsf{Bill}{:}(\mathsf{FirstLady} \sqcap \neg \mathsf{President}), \\ \mathcal{KB} &\models \mathsf{Hillary}{:}(\mathsf{FirstLady} \sqcap \neg \mathsf{President}). \end{aligned}$$

The explanation is that the positive extension of $\mathsf{FirstLady} \sqcap \neg \mathsf{President}$ is $\{\mathsf{Hillary}\} \cap \{\mathsf{Bill}\} = \emptyset$, whereas its negative extension is $\{\mathsf{Bill}\} \cup \{\mathsf{Bill}\}$. Hillary, therefore, belongs neither to $\mathsf{FirstLady} \sqcap \neg \mathsf{President}$ nor to the complementation of this concept, but Bill is included in the complementation.

It can easily be seen that it does not matter whether queries are evaluated with respect to the four-valued or the standard semantics whenever four-valued knowledge bases of this kind bear neither indifferent nor inconsistent knowledge. A polynomial-time algorithm which mimics the four-valued semantics can be developed entirely analogously to that for the standard two-valued semantics. But then our tractability result holds also within framework of the four-valued semantics, while additionally allowing for inconsistency as well as indefiniteness.

Chapter 5

Summary & Conclusion

In the introduction several one-to-one correspondences between description logics and propositional modal and dynamic logics were presented. We argued that all these correspondences become quite obvious once description logics are viewed as propositional logics rather than sublanguages of first-order logic. As De Giacomo and Lenzerini [1994a] put it, particularly the correspondence with propositional dynamic logic "provides an invaluable tool for devising decision procedures for very expressive D[escription] L[ogic]s." One can add that it is also an invaluable source for results on the model theory, expressive power, and axiomatics of description logics.

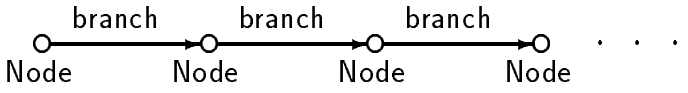
The most powerful correspondence is the one relying on combinatory PDL, presented in Chapter 2. This correspondence is powerful enough to incorporate all standard features of a description logic, including arbitrary concept introductions and axioms. This works with a rather straightforward translation, even in case of assertions. The description logic's framework thus covered is quite general. Thanks to the correspondence with combinatory PDL, this framework is at the same time well-understood in terms of computational complexity, model theory, as well as axiomatics. As a matter of fact, the framework thus obtained is one of the most powerful for which all kinds of terminological and assertional reasoning are known to be decidable.

One fundamental facility is missing though: recursion. In the setting of description logics recursion refers to terminologies containing concepts or roles the definition of which depend on each other. Of course, such cyclic dependencies can be direct or indirect. The lack of recursion contrasts with the fact that especially many concepts are most naturally defined by recursion. For instance, it is rather natural to define trees in the following recursive manner:

$$\mathsf{Tree} \doteq \mathsf{Leaf} \sqcup (\mathsf{Node} \sqcap \forall \mathsf{branch}{:}\mathsf{Tree}).$$

The translation into combinatory PDL does not impose any restriction on the syntactic shape of axioms of this kind. This seems to imply that the framework obtained

through the correspondence with combinatory PDL as it stands is already capable of dealing with recursion. However, in this framework recursive concept introductions are given the same semantics as acyclic concept introductions, an approach which is by no means adequate. The problem with customary semantics is that it might give rise to ambiguities of the following type if recursion is present. To see this, consider again the recursive definition above. Suppose, the interpretation of Node, Leaf, as well as branch has already been fixed, but the extension of Tree is to be determined by the given recursive concept introduction. We would then expect something functioning as a definition that it uniquely determines what it is supposed to define. In this case, we would expect that the interpretations of Node, Leaf, and branch result in a unique interpretation of Tree. In general, this means that an interpretation of all primitive concepts and roles (i.e., all those concept and role names occurring in the concept introduction, except for the concept to be defined) should always determine a unique model. In contrast to acyclic concept introductions, those being recursive may violate this very characteristic of a definition. Such a situation, for instance, is encountered with the following infinite structure.



As far as this structure is concerned, the recursive concept introduction above does not uniquely determine an interpretation of Tree. In particular, there are two different ways of interpreting Tree, both in accordance with this concept introduction. One possibility is to interpret Tree as the set of all nodes, in which case it denotes the same set as Node. On the other hand, Tree can also be interpreted just as the empty set. This means that there are two rather different ways of satisfying this recursive definition, notwithstanding that all primitive parts of the definition have been fixed properly.

This is why the traditional notion of a model is no longer reasonable as soon as recursion enters the picture. In view of this problem, Nebel [1990a, 1991] refined the common idea of a model. Inspired by Lloyd's [1984] work on the semantic foundations of logic programming, Nebel put forward two alternative semantics. In analogy to Lloyd's terminology, he baptized these alternatives *least* and *greatest fixed-point semantics*. In a nutshell, fixed-point semantics does not consider all models as admissible, but only those which are the least or the greatest with respect to the interpretation of the concept to be defined. Of course, in order to give this minimization and maximization process reasonable limits, the terms *least* and *greatest* refer only to those models which have the same interpretation domain and agree in the interpretation of all primitive concepts and roles. These two different fixed-point semantics can be thought of as giving rise to *inductive* or *co-inductive* definitions. This is to say, the two alternative semantics capture different definitions along the line of the

dichotomy "the least set such that..." versus "the greatest set such that....."

Fixed-point semantics turned out to be more suited for recursion than customary semantics. However, it was disputed for a while which of the two different fixed-point semantics should be taken, where the most substantial contribution to this debate is due to Baader [1990a]. He argued that careful inspection of recursive definitions reveals that recursion is often used to express the reflexive-transitive closure of a role. Consequently, he then argued, any semantics of recursion should take into account this fact. The question then arises what kind of fixed-point semantics captures this reading of recursion. Baader was able to prove that, as far as concept introductions fitting the pattern $CN \doteq C \sqcap \forall R{:}CN$ (without any occurrence of CN in C) are concerned, only the greatest fixed-point semantics supports this reading. Only the greatest fixed-point semantics aptures in this case exactly the meaning of $CN \doteq \forall R^*{:}C$. One can add that the recursive concept introduction for Tree fits into this scheme, too, at least when taking into account an appropriate concept introduction for Leaf. For in this case Leaf $\sqcup$ (Node $\sqcap$ $\forall$branch:Tree) is actually equivalent to Node $\sqcap$ $\forall$branch:Tree.

Taking for granted that recursion is commonly used to express regular role expressions such as the reflexive-transitive closure, Baader finally concluded that the greatest fixed-point semantics comes off best. As regards the very weakest description logic, this is, in fact, the only semantics supporting this reading of recursion.

Baader's result, however, is somewhat misleading in that its validity is strictly limited to the very weakest description logic. As a matter of fact, his result is just a consequence of the fact that he considered only concept conjunction as well as universal role quantification, but left out the dual concept-structuring primitives. In chapter 2, we pointed out that for the dual concept-structuring primitives the situation is just the opposite. In particular, in order to express regular role expressions occurring within *existential* role quantification, we have to resort to *least* fixed-point semantics rather than to greatest fixed points. Take, for instance, the recursive concept introduction $CN \doteq C \sqcup \exists R{:}CN$. If this concept introduction is to capture the meaning of $CN \doteq \exists R^*{:}C$, the only way to enforce this reading is by least fixed-point semantics, which shows that both kinds of fixed-point semantics are needed.

The reader might object that in stronger languages such as $\mathcal{ALC}$, one could employ the duality between $\exists R{:}C$ and $\neg\forall R{:}\neg C$ anyway. Instead of expressing, say, $\exists R^*{:}C$, one could first capture $\forall R^*{:}\neg C$ with the help of the of $CN \doteq \neg C \sqcap \forall R{:}CN$. Of course, in order to achieve this, greatest fixed-point semantics must be invoked. If one would simply add the concept introduction $\overline{CN} \doteq \neg CN$, then $\overline{CN}$ should finally capture $\exists R^*{:}C$. Such a representation does not work because terminologies of the form $\{CN \doteq \neg C \sqcap \forall R{:}CN, \overline{CN} \doteq \neg CN\}$ do not have any greatest fixed-point model at all, nor have they any least fixed-point model. To see this, observe that it is impossible to maximize (or to minimize) the interpretation of CN and its complement at the same time. This proves not only that both kinds of fixed-point semantics are needed,

but that they are even needed in *coexistence*.

This is, however, not the only argument in support of the coexistence of the two different fixed-point semantics. Another argument attacks the claim that recursion is mostly used to express regular role expressions. This claim cannot be put into effect without a thorough investigation on what the least and greatest fixed-point semantics really express. Such an investigation was carried out in Chapter 2. This was done with the help of a previously unknown, but nevertheless extremely useful concept-structuring primitive. In particular, we introduced new concepts of the form $\exists R^{\omega}$, where R can be an arbitrary role. Such a concept denotes all those objects, d, such that there is at least one infinite R-chain emanating from d. The significance of a concept-structuring primitive capable of stipulating or forbidding (when negated) such infinite role chains should be evident. The addition of this new concept-structuring primitive to the regular extension of $\mathcal{ALC}$ results in a description logic which we refer to as the *ω-regular extension of $\mathcal{ALC}$*. With the help of this new description logic, the consequences of enforcing the different kinds of fixed-point semantics can be exactly stated. The analysis, however, should be restricted to standard recursion following the patterns $CN \doteq C \sqcap \forall R{:}CN$ and $CN \doteq C \sqcup \exists R{:}CN$. The following table sums up the analysis of Chapter 2:

	least fixed-point semantics	*greatest fixed-point semantics*
$CN \doteq C \sqcap \forall R{:}CN$	$CN \doteq \forall R^{*}{:}C \sqcap \neg\exists R^{\omega}$	$CN \doteq \forall R^{*}{:}C$
$CN \doteq C \sqcup \exists R{:}CN$	$CN \doteq \exists R^{*}{:}C$	$CN \doteq \exists R^{*}{:}C \sqcup \exists R^{\omega}$

The table shows that the situation in least and greatest-fixed point semantics is completely symmetric. What this table also shows is that the question which of the two different fixed-point semantics should be preferred depends on what one intends to express in that particular case. For example, in the case of the concept introduction of a tree given above, it is just a question of whether or not trees of infinite depth are excluded. According to the table just given, the least fixed-point semantics does exclude them, whereas the greatest fixed-point semantics does not. Of course, this raises the question whether such an analysis is limited to recursion fitting into the standard patterns included in the table above.

In Chapter 2 it was pointed out that questions like the latter can be tackled perfectly well in terms of the explicit fixed-point operators known from program logics, and so can recursion in $\mathcal{ALC}$ as a whole. This is particularly true in view of the need for coexistence of both kinds of fixed-point semantics. Explicit least and greatest fixed-point operators are used in logics of programs to state specific correctness properties not expressible by ordinary dynamic logics. They have been employed successfully to state deadlock freedom and starvation, see [Flon and Suzuki, 1978]. In the context of first-order logic, explicit fixed-point operators were investigated by Park [1970],

Hitchcock and Park [1973], as well as de Bakker and de Roever [1973]. Regarding description logics, the propositional case is more interesting though. The propositional case was first investigated by Pratt [1981] and, more influentially, by Kozen [1983]. In both cases, least and greatest fixed-point operators are treated just as a new kind of formulae. This treatment makes it possible to express nested fixed-points operators easily. In the propositional case, the following notation is common. If x is an arbitrary propositional variable and α is an arbitrary formula, then $\mu x.\alpha$ is a least fixed-point operator, while $\nu x.\alpha$ is a greatest fixed-point operator. Fixed-point formulae of the form $\mu x.\alpha$ and $\nu x.\alpha$ are to be read as "the least x such that α" and "the greatest x such that α."

From a semantic point of view, fixed-point formulae represent the least and the greatest fixed point of a certain function, hence the name *fixed-point* operator. This function can be best described with the following notation. If $\mathcal{M}$ is an arbitrary Kripke structure and α is a formula, then let $\alpha^{\mathcal{M}}$ denote the set of all those states, w, such that $\mathcal{M} \models_w \alpha$. This means that $\alpha^{\mathcal{M}}$ denotes exactly those states in which the proposition α holds. The function we have in mind then maps every set, S, of states to $\alpha^{\mathcal{M}_{x/S}}$. Here $\mathcal{M}_{x/S}$ is supposed to denote the Kripke structure which agrees with $\mathcal{M}$ except for the fact that $x^{\mathcal{M}}$ is S. This function is, of course, interesting only if α involves at least one occurrence of x; otherwise it would denote a constant function that always yields $\alpha^{\mathcal{M}}$.

Of course, the meaning of fixed-point operators is then given in terms of the least and greatest fixed points of the function just introduced. This would not be possible if the uniqueness of these least and greatest fixed points was not guaranteed. This is usually ensured by imposing a simple restriction on the possible syntactic shape of fixed-point formulae, referred to as *formal monotonicity*. This restriction requires every occurrence of x in $\mu x.\alpha$ and $\nu x.\alpha$ to be *positive*. This is to say, every occurrence of x must lie under an even number of negations. The function described above is then guaranteed to be monotonic. According to the well-known Knaster-Tarski Theorem, monotonicity of a function in turn ensures the existence and the uniqueness of its least as well as its greatest fixed point [Tarski, 1955].

It was Kozen [1983] who enriched the Hennessy-Milner Logic with exactly this kind of fixed-point formulae. The resulting logic is called *propositional μ-calculus*. Kozen already observed that this logic is at least as strong in expressive power as the propositional dynamic logic. The PDL formula $\langle \textbf{while } \alpha \textbf{ do } a \rangle \beta$, for instance, can be expressed in a recursive fashion with the help of the least fixed-point formula $\mu x.\big((\neg\alpha \wedge \beta) \vee (\alpha \wedge \langle a \rangle x)\big)$. Kozen additionally noted that there are even formulae of the propositional μ-calculus not expressible in propositional dynamic logic.

Having in mind that $\mathcal{ALC}$ is a notational variant of the Hennessy-Milner logic, we should expect that recursion can be captured in $\mathcal{ALC}$ in terms of explicit fixed-point operators because so can recursion in the Hennessy-Milner logic. In the context of $\mathcal{ALC}$, fixed-point operators are to be treated as concept-structuring primitives. In

analogy to the μ-calculus, in Chapter 2 we chose the following syntax for the new concept-structuring primitives to be introduced: If C is an admissible concept, then so are $\mu X.C$ and $\nu X.C$. The variable X is treated as a special kind of a concept name, referred to as *concept variable*. Of course, formal monotonicity has to be imposed on C in this case too. But then an arbitrary recursive concept introduction, $CN \doteq C$, can directly be recast by one of the following two acyclic concept introductions:

$$\begin{aligned} CN &\doteq \mu X.C_{CN/X}, \\ CN &\doteq \nu X.C_{CN/X}. \end{aligned}$$

The only precondition that $CN \doteq C$ has to meet is that of formal monotonicity. This is to say, all occurrences of CN in C must be positive. In any case, the expression $CN_{CN/X}$ is supposed to denote the concept obtained from C by simultaneously replacing each occurrence of CN with X throughout C. Observe that this kind of representation enables us to capture recursion with the help of acyclic concept introductions rather than cyclic ones. Of course, the choice between the two alternatives given above depends on whether least or greatest fixed-point semantics is preferred.

If there is an indirect recursion leading through more than one concept introduction, we have to resort either to nested fixed-point operators or to fixed-point operators dealing with mutual recursion. Fixed-point operators of the latter type were investigated by Vardi and Wolper [1984] in the framework of the Hennessy-Milner logic. In Chapter 2 we enriched $\mathcal{ALC}$ with explicit fixed-point operators in the style of Vardi and Wolper. We thereby obtained a new kind of description logic which is actually a notational variant of Vardi and Wolper's version of the propositional μ-calculus. We chose the name $\mathcal{ALC}\mu$ for the extended standard description logic $\mathcal{ALC}$.

Thanks to the one-to-one correspondence with the propositional μ-calculus, we can take advantage of a number of results established for the μ-calculus. Most importantly, this includes several complete decision procedures. The first decision procedure for full propositional μ-calculus is due to Kozen and Parikh [1983]. However, the upper time bound thus obtained was non-elementary. The first decision elementary procedure was given by Street and Emerson [1984, 1989]. The employed algorithm, however, still has a triply exponential worst-case complexity. Exponential-time procedures were devised by Vardi and Wolper [1984], Emerson and Jutla [1988], as well as Safra [1988]. Vardi and Wolper's algorithm is capable of dealing with mutual recursion, but it works only with the possible occurrences of nested alternating fixed-point operators restricted. This restriction, however, does not affect those nested fixed-point operators which are of practical use. In contrast to this, the Emerson and Jutla's as well as Safra's algorithms capture the propositional μ-calculus to its full extent, but without mutual recursion. Axiomatics of full propositional μ-calculus has been a longstanding open problem. This problem was solved recently by Walukiewicz [1993].

Other interesting results concern the expressive power of the μ-calculus. One result

states that the propositional μ-calculus is strictly stronger than PDL, even when there are no nested fixed-point operators available [Kozen, 1983]. One interpretation of this result is that regular role expressions are by no means an alternative for recursion in $\mathcal{ALC}$ because they are strictly weaker in expressive power. Another result states that even when PDL is augmented by so-called *repeat* formulae, full propositional μ-calculus is still strictly stronger in expressive power than this extension of PDL. This result is due to Niwinsky and was reported in [Streett, 1985], page 363. The interesting point about Niwinsky's result is that repeat formulae correspond in a one-to-one fashion to concepts of the form $\exists R^{\omega}$ which we have met before. But then the question whether the exact meaning of recursion in $\mathcal{ALC}$ can always be characterized in terms of the ω-regular role expressions must be answered in the negative.

However, even when enhanced by recursion, $\mathcal{ALC}$ is doubtlessly too weak to be considered as a general purpose representation language. It is not even clear at all what the representational merits of this description logic are. The definition of a tree, for instance, can be represented in this setting only partially, not to mention other definitions of practical significance. The integration of a few additional concept-structuring primitives as successfully carried out in [de Giacomo and Lenzerini, 1994b] does not improve the situation in principle. On the other hand, this limited framework is computationally feasible only in a very weak sense. From the correspondence with the propositional μ-calculus, we know that in this setting subsumption is computable in deterministic exponential time [Emerson and Jutla, 1988, Safra, 1988]. In this sense, recursion in $\mathcal{ALC}$ can be computed at least in principle. From Chapter 2 it is also known that computing subsumption in this setting requires for an infinite number of inputs an amount of time at best bounded by a fixed exponential in the size of the input. But this means that even in this limited framework the basic inference is provably intractable. Even worse, according to [Nebel, 1990b], subsumption in the very weakest description logic is tractable only if P is NP, which is commonly assumed to be false. In technical terms, subsumption is in this minimal setting co-NP-complete. Notably, for this co-NP-result to be valid, only acyclic terminologies must be taken into account.

Therefore, Chapter 2 left us with a dilemma. On the one hand, the definitional power of recursion in $\mathcal{ALC}$ is by no means strong enough to capture concepts of practical significance. On the other hand, even in its restrictedness, this framework can be considered as computationally feasible only with an interpretation of *feasible* which is certainly too liberal for most applications.

In Chapter 4 we aimed at squaring this circle. We were encouraged to do so by Halpern and Vardi's [Halpern and Vardi, 1991] recent manifesto on model checking versus theorem proving. In this manifesto they put forward that intractability in knowledge representation is often caused by the way logical languages are used rather than by the logics themselves. The traditional way of using logics in knowledge representation goes back to McCarthy [1968]. Traditional knowledge representation

uses logics not only to phrase queries, but first and foremost to represent the world to be modeled itself. Halpern and Vardi pointed out that the need to represent all facts about the world necessitates the use of very expressive logics. One can add that particularly the need to represent all those facts which do *not* hold would not possible without very expressive logics. But even in very small logics, deciding logical consequence is mostly intractable, not only in the realm of description logics. At this very point Halpern and Vardi argued that the intractability inherent in this traditional approach might be circumvented whenever it is possible to represent the given world by a single semantic structure rather than by a collection of formulae denoting a number of such models. In many cases, a representation by a single semantic structure is not only possible, but also the most natural representation. If queries are still phrased within some logic, then in this case the answer depends on whether or not the *given* semantic structure is a *model* of the query. This is to say, it suffices to check whether or not in the given semantic structure the query evaluates to true. In many cases model checking has tremendous advantages in terms of computational complexity. An example in support of this is that checking the truth of arbitrary first-order formulae in a finite semantic structure is decidable though still intractable in a strict sense [Chandra and Merlin, 1977]. In contrast to this, deciding logical consequence in full first-order logic is not only intractable but undecidable [Church, 1936a, 1936b]. Notably, this undecidability result remains valid even with only finite interpretation domains taken into account [Trahtenbrot, 1963].

The lesson of Halpern and Vardi's manifesto is that even in undecidable logics, a type of reasoning can take place which is not only computable in principle, but also useful. Reasoning of this type resorts to world descriptions consisting of single semantic structures, not of collections of formulae. Of course, what we thereby lose is the ability to state only particular aspects of an application domain without giving a complete description of it.

Although Halpern and Vardi's manifesto was aimed at knowledge representation as a whole, it has immediate consequences for description logics. In order to see this, we have to return to propositional logics of programs. Though not explicitly stated, one of the motivations of Halpern and Vardi to advocate model checking for knowledge representation purposes was certainly that in other areas model checking already proved to be useful. This is particularly true for propositional logics of programs, a field in which Vardi has been active. Traditionally, logics of programs are also used in the theorem-proving style to deduce correctness and termination properties from a formal specification of a program. Recently, it has been observed that these logics can also be employed to verify properties of specific transition systems such as finite automatons. A transition system consists of a finite set of states along with a binary transition relation among states. Of course, transition systems are given by single semantic structures rather than by formal logical specifications. Especially the propositional μ-calculus proved to be a powerful tool for verifying properties of transition systems. In this case properties of transition systems are expressed by formulae

of the μ-calculus. Then a check is made whether or not a given transition system satisfies this property. As far as a slightly restricted fragment of the propositional μ-calculus is concerned, this kind of model checking can be performed even in linear time. Linear-time upper bounds for model checking in fragments of the propositional μ-calculus were achieved by Emerson and Lei [1986], Arnold and Crubille [1988], as well as Cleaveland and Steffen [1991]. In each case only the possible occurrences of nested alternating fixed-point operators have to be restricted so as to achieve a linear upper bound. But the fact that recursion in $\mathcal{ALC}$ can be perfectly captured with the help of the propositional μ-calculus already indicates how tractable reasoning can take place in this setting too. Incidentally, those fragments of the propositional μ-calculus for which model checking was shown to be tractable are sufficient to capture practically relevant recursion in $\mathcal{ALC}$. From the description logic's point of view, such a type of reasoning presupposes ordinary knowledge bases to be replaced by single semantic structures. Because semantics structures comprise in Levesque's [1986] sense only *vivid* knowledge, we refer to them in the context of description logics as *vivid knowledge bases*. Thus any incomplete knowledge must be abandoned from knowledge bases if tractability is to be obtained. Although this very restriction is tolerable for many application scenarios, $\mathcal{ALC}$'s limitations in expressive power are not tolerable, even when enhanced by recursion. An important question is then whether this kind of tractable reasoning under complete knowledge can take place in richer languages as well. Therefore, we considered in Chapter 3 Patel-Schneider's [1987] universal description logic $\mathcal{U}$. This description logic is called *universal* with full right because it encompasses all concept and role-structuring primitives ever considered in the literature, except for nonstandard ones such as defaults. $\mathcal{U}$ was then enriched by least and greatest fixed-point operators similar to those introduced in Chapter 2. The main complication was to accommodate the notion of formal monotonicity to deal with the additional concept and role-structuring primitives of $\mathcal{U}$. In particular, negations implicitly residing in number restrictions of the form $\exists^{\leq m}$, role-value maps, and structural descriptions had to be taken into account. In analogy to the corresponding extension of $\mathcal{ALC}$, the resulting description logic is called $\mathcal{U}\mu$. However, in contrast to $\mathcal{ALC}\mu$, $\mathcal{U}\mu$ comprises not only fixed-point operators on concepts, but also those on roles. Notably, fixed-point operators on roles were never considered before.

In Chapter 4 we then investigated the computational complexity of model checking in a certain fragment $\mathcal{U}\mu$. Like $\mathcal{ALC}\mu^-$ this fragment restricts only the possible occurrences of those nested alternating fixed-point operators which are of no practical significance. It turned out that model checking can be performed even in this extremely powerful setting in polynomial time. In particular, it can be performed in deterministic time bounded above by a fixed polynomial in the sum of the sizes of the given vivid knowledge base and the query. Interestingly, this tractability result remains valid even with arbitrary acyclic terminologies taken into account. In this case it must be guaranteed that the extension of each *primitive* concept or role is actually specified by the given vivid knowledge base. This is to say, each concept and

role name occurring somewhere in the query or in the terminology must be specified if it is not defined by the terminology.

Our result is significant and new both from the viewpoint of Halpern and Vardi's [1991] work on model checking as well as from the description logic's point of view. From Halpern and Vardi's point of view, our result singles out a useful fragment of first-order logic which gives rise to tractable model checking, even when this fragment is enriched by a general means of recursion. As opposed to the corresponding tractability results for fixed-point languages based on full first-order logic (see e.g. [Vardi, 1982]), ours is to be understood in terms of Vardi's [1982] notion of *combined complexity* rather than the far weaker notion of *data complexity*. The crucial difference between these two different complexity measures is that data complexity presupposes an arbitrary but *fixed* query and, in the present case, also an arbitrary but *fixed* terminology, whereas combined complexity does not do so. The difference in meaningfulness of these two different complexity measures is perhaps best explained by means of an example. Suppose that we came up with, say, a deterministic lower and upper time bound of $|\mathcal{KB}|^{O(|\mathcal{T}|+|Q|)}$, which is not only exponential, but even *superexponential* in terms of combined complexity. In the sense of data complexity this would yet be a *polynomial* upper bound because for *fixed* $\mathcal{T}$'s and Q's, $|\mathcal{KB}|^{O(|\mathcal{T}|+|Q|)}$ actually constitutes a polynomial time bound. It should be obvious that the notion of data complexity is, therefore, not tailored for terminological reasoning. For it seems far fetched to presuppose that terminological reasoning takes place with some fixed terminology or even with some fixed query.

From the description logic's point of view, our result states that, whenever we are able to abandon any kind of incomplete knowledge from the knowledge base, we gain not only tractability, but also a great deal of expressive power that covers every concept and role-structuring primitive one could ask for.[1] Another interpretation of our tractability result is that the main source of computational complexity of terminological reasoning is the ability to express incomplete knowledge. This finding is confirmed by another outcome of Chapter 4 stating that even a quite limited form of incomplete knowledge by means of what is called a null value in databases causes co-NP-hardness. This co-NP-hardness result remains valid even without any nonempty terminology taken into account and additionally with queries restricted to be some *fixed* query of the standard description logic $\mathcal{ALC}$. This hardness result holds therefore even in the sense of data complexity. We were yet able to devise an algorithm capable of dealing with null values soundly though approximately. The resulting algorithm runs not only in polynomial time, but is also complete whenever no null value is present.

[1] The observation that tractability can be gained in principle when abandoning any kind of incomplete knowledge has been made previously by Donini *et al.* [1992]. However, their observation concerns solely a description logic which is very pure in expressive power and does not take any terminology into account.

Our tractability result is the *only* one in existence for terminological reasoning, at least when nonempty terminologies are are to be taken into account. Despite the fact that all other tractability results retreat either to description logics which are extremely pure in expressive power or to nonstandard semantics, all these results presuppose an *expansion* process applied to terminologies in advance.[2] This process replaces all occurrences of those concept and role names defined in the given terminology by the right-hand side of the relevant concept or role introduction; see Chapter 3.2.5 of [Nebel, 1990a] for details. In the course of such an expansion process, however, a terminology may increase by an exponential, so that the overall upper time complexity bounds obtained by all but our result are still *exponential* in nature. In fact, as far as standard terminological reasoning is concerned, Nebel [1990b] proved that in the presence of terminologies, no tractability result can be achieved, unless P is NP. This holds even with the expressive power limited to that of the very smallest description logic.

Not only is the subclass of terminological inferences that we singled out tractable, but it is also *useful*. Many examples can be put forward to show this. We have seen in Chapter 3 that such standard objects as directed acyclic graphs, trees, as well as binary trees can easily be defined within the setting that we have considered. In addition, even such involved objects as balanced binary trees and those vertices of an AND-OR graph for which there exists at least one well-founded solution were captured as well. It should be clear that any graph or even any finite collection of graphs can straightforwardly be represented by a single vivid knowledge base. But then the tractable subclass of terminological inferences that we singled out includes the retrieval of exactly those graphs being an instance of any of the objects just mentioned. As vivid knowledge bases are nothing but relational databases, this means that $\mathcal{U}\mu^-$ can serve as a powerful but tractable query language for databases. But then the question arises how powerful the description logic $\mathcal{U}\mu^-$ is compared to other more traditional database query languages. This issue was also explored in Chapter 3. It turned out that $\mathcal{U}\mu$ (and therefore also $\mathcal{U}\mu^-$) is strictly weaker than a well-known fixed-point query language based on full first-order logic. However, this is not surprising at all because the latter is in contrast to $\mathcal{U}\mu^-$ still intractable in the sense of the combined complexity. Of course, tractability cannot be achieved for free.

Of course, abandoning any kind of incomplete knowledge and allowing for an unlimited use of incomplete knowledge are just two extreme points on a wide-ranged spectrum. An auspicious compromise was already indicated by another outcome of Chapter 4 concerning terminological reasoning in a more traditional sense. It states that ordinary terminological reasoning in $\mathcal{U}\mu^-$ becomes a member of co-NP just by adding a domain-closure axiom, restricting the interpretation domain to a finite number of individual names. In this case terminological reasoning in $\mathcal{U}\mu^-$ has essentially

[2] This applies to the work on restricted languages carried out by Levesque and Brachman [1987], Lenzerini and Schaerf [1991], Donini *et al.* [1991], and Buchheit *et al.* [1994], but also to Patel-Schneider's [1989a] results for nonstandard semantics.

the computational worst-case complexity which is inherent in the very weakest description logic. That is, domain-closed reasoning in $\mathcal{U}\mu^-$ is co-NP-complete. From a practical point of view, especially from the viewpoint of deterministic computations, this means that in the presence of a domain closure axiom, terminological reasoning in $\mathcal{U}\mu^-$ is yet decidable, namely in exponential time.

This proves that enormous expressive power can be gained in return for a simple domain-closure axiom, while still retaining a worst-case complexity which is essentially not greater than that caused by terminological reasoning in the very weakest setting. As a matter of fact, requiring the existence of a domain-closure axiom is to most applications no essential restriction at all. On the other hand, terminological reasoning seems not to become applicable until an expressive power such as that of $\mathcal{U}\mu^-$ has been attained. Practical algorithms obeying an acceptable average case complexity, however, still await development for terminological reasoning under closed domains.

Bibliography

[Aho and Ullman, 1979] Alfred V. Aho and Jeffrey D. Ullman. Universalty of data retrieval languages. In *Proceedings of the 6th ACM Symposium on Principles of Programming Languages*, pages 110–117, San Antonio, TX, 1979.

[Arnold and Crubille, 1988] Andre Arnold and Paul Crubille. A linear algorithm to solve fixed-point equations on transition systems. *Information Processing Letters*, 29(2):57–66, 1988.

[Baader and Hollunder, 1991] Franz Baader and Bernhard Hollunder. $\mathcal{KRIS}$: $\mathcal{K}$nowledge $\mathcal{R}$epresentation and $\mathcal{I}$nference $\mathcal{S}$ystem. *SIGART Bulletin*, 2(3):8–14, 1991.

[Baader *et al.*, 1994] Franz Baader, Martin Buchheit, and Bernhard Hollunder. Cardinality restrictions on concepts. In *Proceedings of the KI-94*, Saarbrücken, FRG, 1994.

[Baader, 1990a] Franz Baader. A formal definition for expressive power of knowledge representation languages. DFKI Research Report RR-90-05, German Research Center for Artificial Intelligence–DFKI GmbH, Kaiserslautern, FRG, 1990.

[Baader, 1990b] Franz Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, pages 621–626, Boston, MA, 1990.

[Baader, 1991] Franz Baader. Augmenting concept languages by the transitive closure: An alternative to terminological cycles. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 446–451, Sydney, Australia, 1991.

[Balcázar *et al.*, 1988] José Luis Balcázar, Josep Diaz, and Joaquim Gabarró. *Structural Complexity*, volume I. Springer-Verlag, Berlin, FRG, 1988.

[Belnap, 1977] Nuel D. Belnap. A useful four-valued logic. In G. Epstein and J. M. Dunn, editors, *Modern Uses of Multiple-Valued Logics*, pages 30–56. Reidel, Dordrecht, The Netherlands, 1977.

[Ben-Ari *et al.*, 1982] Mordechai Ben-Ari, Joseph Y. Halpern, and Amir Pnueli. Deterministic propositional dynamic logic: Finite models, complexity, and completeness. *Journal of Computer and System Science*, 25:402–417, 1982.

[Borgida, 1993] Alexander Borgida. On the relationship between description logic and predicate logic queries. Technical Report DCS-TR-295A, Department of Computer Science, New Brunswick, NJ, 1993.

[Brachman and Levesque, 1984] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the 4th National Conference of the American Association for Artificial Intelligence*, pages 34–37, Austin, TX, 1984.

[Brachman and Schmolze, 1985] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[Brachman, 1977] Ronald J. Brachman. *A Structural Paradigm for Representing Knowledge*. PhD thesis, Havard University, 1977.

[Brachman, 1979] Ronald J. Brachman. On the epistemological status of semantic networks. In N. V. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 3–50. Academic Press, New York, NY, 1979.

[Brachman, 1983] Ronald J. Brachman. What IS-A is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer*, 16(10):30–36, 1983.

[Buchheit *et al.*, 1994] Martin Buchheit, Manfred A. Jeusfeld, Werner Nutt, and Martin Staudt. Subsumption between queries to object-oriented databases. *Information Systems*, 19(1):33–54, 1994.

[Bull and Segerberg, 1984] Robert Bull and Krister Segerberg. Basic modal logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic*, volume 2, pages 1–88. Reidel, Dordrecht, The Netherlands, 1984.

[Chandra and Harel, 1980] Ashok K. Chandra and David Harel. Computable queries for relational data bases. *Journal of Computer and System Science*, 21:156–178, 1980.

[Chandra and Harel, 1982] Ashok K. Chandra and David Harel. Structure and complexity of relational queries. *Journal of Computer and System Science*, 25:99–128, 1982.

[Chandra and Merlin, 1977] Ashok K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proceedings of the 9th ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[Chellas, 1980] Brian F. Chellas. *Modal Logic: An Introduction.* Chicago University Press, Chicago, IL, 1980.

[Church, 1936a] Alonzo Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1:40–41, 1936.

[Church, 1936b] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

[Cleaveland and Steffen, 1991] Rance Cleaveland and Bernhard Steffen. A linear-time model-checking algorithm for the alternation-free mu-calculus. In *Proceedings of the 3rd International Workshop on Computer Aided Verification*, pages 48–58, Aalborg, Denmark, 1991.

[Codd, 1971] Edgar F. Codd. A data base sublanguage founded on the relational calculus. In *ACM-SIGFIDET Workshop on Data Description, Access, and Control*, San Diego, CA, 1971.

[Danecki, 1984] Ryszard Danecki. Nondeterministic propositional dynamic logic with intersection is decidable. In *Proceedings of the 5th Symposium on Computation Theory*, pages 34–53, 1984.

[de Bakker and de Roever, 1973] Jaco de Bakker and Willem P. de Roever. A calculus for recursive program schemes. In *Proceedings of 1st International Colloquium on Automata, Languages and Programming*, pages 167–196, 1973.

[de Bakker, 1980] Jaco de Bakker. *Mathematical Theory of Program Correctness.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[de Giacomo and Lenzerini, 1994a] Giuseppe de Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the 12th National Conference of the American Association for Artificial Intelligence*, pages 205–212, 1994.

[de Giacomo and Lenzerini, 1994b] Giuseppe de Giacomo and Maurizio Lenzerini. Concept language with number restriction and fixpoints and its relationship with mu-calculus. In *Proceedings of the 14th European Conference on Artificial Intelligence*, 1994.

[de Giacomo and Lenzerini, 1994c] Giuseppe de Giacomo and Maurizio Lenzerini. Description logics with inverse roles, functional restrictions, and n-ary relations. In *Proceedings of the 4th European Workshop on Logics in Artificial Intelligence*, pages 332–346, York, UK, 1994.

[Dionne *et al.*, 1992] Robert Dionne, Eric Mays, and Frank J. Oles. A non-well-founded approach to terminological cycles. In *Proceedings of the 10th National*

Conference of the American Association for Artificial Intelligence, pages 761–766, San Jose, CA, 1992.

[Dionne *et al.*, 1993] Robert Dionne, Eric Mays, and Frank J. Oles. The equivalence of model-theoretic and structural subsumption in description logics. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 710–716, Chambery, France, 1993.

[Donini *et al.*, 1991] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. Tractable concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 458–463, Sydney, Australia, 1991.

[Donini *et al.*, 1992] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Andrea Schaerf, and Werner Nutt. Adding epistemic operators to concept languages. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 342–353, Cambridge, MA, 1992.

[Doyle and Patil, 1991] Jon Doyle and Ramesh S. Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48:261–297, 1991.

[Emerson and Jutla, 1988] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 328–337, 1988.

[Emerson and Lei, 1986] E. Allen Emerson and Chin-Laung Lei. Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In *Proceedings of the 1st Annual IEEE Symposium on Logic in Computer Science*, pages 267–278, Boston, MA, 1986.

[Fischer and Ladner, 1979] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Science*, 18:194–211, 1979.

[Flon and Suzuki, 1978] Lawrence Flon and Norihisa Suzuki. Nondeterminism and the correctness of parallel programs. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, pages 589–608, 1978.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability—A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA, 1979.

[Garey *et al.*, 1976] Michael R. Garey, David S. Johnson, and Larry Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[Gargov and Passy, 1988] George Gargov and Solomon Passy. Determinism and looping in combinatory PDL. *Theoretical Computer Science*, 61, 1988.

[Gargov, 1985] George Gargov. Decidability of basic combinatory propositional dynamic logic. In A. Ershov and D. Skordev, editors, *Mathematical Theory of Programming*. Computer Center of the Siberian Division of Soviet Academy of Sciences, Novosibirsk, USSR, 1985.

[Goldblatt, 1987] Robert Goldblatt. *Logics of Time and Computation*, volume 7 of *CSLI Lecture Notes*. Chicago University Press, Chicago, IL, 1987.

[Halmos, 1974] Paul R. Halmos. *Naive Set Theory*. Springer-Verlag, New York, NY, 1974.

[Halpern and Moses, 1985] Joseph Y. Halpern and Yoram Moses. A guide to the modal logics of knowledge and belief. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 480–490, Los Angeles, CA, 1985.

[Halpern and Reif, 1983] Joseph Y. Halpern and John Reif. The propositional dynamic logic of deterministic, well-structured programs. *Theoretical Computer Science*, 27:127–165, 1983.

[Halpern and Vardi, 1991] Joseph Y. Halpern and Moshe Y. Vardi. Model checking vs. theorem proving: A manifesto. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 325–334, Cambridge, MA, 1991.

[Harel *et al.*, 1977] David Harel, Albert R. Meyer, and Vaughan R. Computability and completeness in logics of programs: Preliminary report. In *Proceedings of the 9th ACM Symposium on Theory of Computing*, pages 261–268, 1977.

[Harel, 1984] David Harel. Dynamic logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic*, volume 2, pages 497–604. Reidel, Dordrecht, The Netherlands, 1984.

[Hayes, 1980] Patrick J. Hayes. The logic of frames. In D. Metzing, editor, *Frame Conceptions and Text Understanding*, pages 46–61. deGruyter, Berlin, Germany, 1980.

[Hendrix, 1979] Gary G. Hendrix. Encoding knowledge in partitioned networks. In N. V. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 51–92. Academic Press, New York, NY, 1979.

[Hennessy and Milner, 1985] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[Hintikka, 1969] Jaakko Hintikka. Semantics for propositional attitudes. In *Models for Modalities*. Reidel, Dordrecht, The Netherlands, 1969.

[Hitchcock and Park, 1973] Peter Hitchcock and David Park. Induction rules and termination proofs. In *Proceedings of 1st International Colloquium on Automata, Languages and Programming*, pages 225–251, 1973.

[Hoare, 1969] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580 and 583, 1969.

[Hughes and Cresswell, 1968] George E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic*. Methuen, London, 1968.

[Hughes and Cresswell, 1984] George E. Hughes and M. J. Cresswell. *A Companion to Modal Logic*. Methuen, London, 1984.

[Kanellakis, 1990] Paris C. Kanellakis. Elements of relational database theory. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 1074–1156, Amsterdam, The Netherlands, 1990. Elsevier.

[Kozen and Parikh, 1981] Dexter Kozen and Rohit Parikh. An elementary proof of the completeness of PDL. *Theoretical Computer Science*, 14(1):113–118, 1981.

[Kozen and Parikh, 1983] Dexter Kozen and Rohit Parikh. A decision procedure for the propositional μ-calculus. In *Proceedings of the Workshop on Logics of Programs*, pages 313–325, Carnegie Mellon University, Pittsburgh, PA, 1983.

[Kozen and Tiuryn, 1990] Dexter Kozen and Jerzy Tiuryn. Logics of programs. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 789–840. Elsevier, Amsterdam, The Netherlands, 1990.

[Kozen, 1983] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[Kripke, 1963] Saul A. Kripke. Semantical analysis of modal logic I: Normal propositional calculi. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, pages 67–96, 1963.

[Ladner, 1975] Richard E. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7(1):18–20, 1975.

[Ladner, 1977] Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computing*, 6(3):467–480, 1977.

[Lemmon, 1966] E. J. Lemmon. Algebraic semantics for modal logic I. *Journal of Symbolic Logic*, 31(1):46–65, 1966.

[Lenzerini and Schaerf, 1991] Maurizio Lenzerini and Andrea Schaerf. Concept languages as query languages. In *Proceedings of the 9th National Conference of the American Association for Artificial Intelligence*, pages 471–476, Anaheim, CA, 1991.

[Levesque and Brachman, 1987] Hector J. Levesque and Ronald J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[Levesque, 1986] Hector J. Levesque. Making believers out of computers. *Artificial Intelligence*, 30(1):81–108, 1986.

[Levesque, 1988] Hector J. Levesque. Logic and the complexity of reasoning. *Journal of Philosophical Logic*, 17:355–389, 1988.

[Lipkis, 1982] Thomas Lipkis. A KL-ONE classifier. In J. G. Schmolze and R. J. Brachman, editors, *Proceedings of the 1981 KL-ONE Workshop*, pages 128–145, Cambridge, MA, 1982. The proceedings appeared as BBN Report 4842, Bolt, Baranek, and Newman Inc., Cambridge, MA.

[Lloyd, 1984] John W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, Berlin, FRG, 1984.

[Löwenheim, 1915] Leopold Löwenheim. Über Möglichkeiten im Relativkalkül. *Mathematische Annalen*, 76(4):447–470, 1915.

[Mark, 1982] William Mark. Realization. In J. G. Schmolze and R. J. Brachman, editors, *Proceedings of the 1981 KL-ONE Workshop*, pages 78–89, Cambridge, Mass., 1982. The proceedings appeared as BBN Report 4842, Bolt, Baranek, and Newman Inc., Cambridge, MA.

[McCarthy, 1968] John McCarthy. Programs with common sense. In M. Minsky, editor, *Semantic Information Processing*, pages 403–418. MIT Press, Cambridge, MA, 1968.

[McDermott, 1978] Drew V. McDermott. Tarskian semantics, or no notation without denotation! *Cognitive Science*, 2(3):277–282, 1978.

[McDermott, 1987] Drew V. McDermott. A critique of pure reason. *Computational Intelligence*, 3(3):151–160, 1987.

[Meyer *et al.*, 1979] Albert R. Meyer, Robert S. Streett, and Grazyna Mirkowska. The deducibility problem in propositional dynamic logic. In *Proceedings of the Workshop on Logics of Programs*, pages 12–22, Zürich, Switzerland, 1979.

[Moschovakis, 1974] Yiannis N. Moschovakis. *Elementary Induction on Abstract Structures*, volume 77 of *Studies in Logic and Foundations in Mathematics*. North-Holland, Amsterdam, The Netherlands, 1974.

[Nebel, 1990a] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes in Computer Science. Springer-Verlag, Berlin, FRG, 1990.

[Nebel, 1990b] Bernhard Nebel. Terminological Reasoning is Inherently Intractable. *Artificial Intelligence*, 43:235–249, 1990.

[Nebel, 1991] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In J. Sowa, editor, *Formal Aspects of Semantic Networks*, pages 331–361. Morgan Kaufmann, San Mateo, CA, 1991.

[Parikh, 1979] Rohit Parikh. Propositional dynamic logics of programs: A survey. In *Proceedings of the Workshop on Logics of Programs*, pages 102–144, Zürich, Switzerland, 1979.

[Park, 1970] David Park. Fixpoint induction and proofs of program properties. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 5, pages 59–78. Edinburgh University Press, Edinburgh, Scotland, 1970.

[Passy and Tinchev, 1985] Solomon Passy and Tinko Tinchev. PDL with data constants. *Information Processing Letters*, 20(2):35–41, 1985.

[Passy and Tinchev, 1991] Solomon Passy and Tinko Tinchev. An essay in combinatory logic. *Information & Computation*, 93(2):263–332, 1991.

[Patel-Schneider, 1987] Peter F. Patel-Schneider. *Decidable, Logic-Based Knowledge Representation.* PhD thesis, University of Toronto, Toronto, Ontario, 1987. Computer Science Department, Technical Report 201/87.

[Patel-Schneider, 1989a] Peter F. Patel-Schneider. A four-valued semantics for terminological logics. *Artificial Intelligence*, 38(3):319–351, 1989.

[Patel-Schneider, 1989b] Peter F. Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39(2):263–272, 1989.

[Pratt, 1976] Vaughan R. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.

[Pratt, 1979] Vaughan R. Pratt. Models of program logics. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 115–122, San Juan, Puerto Rico, 1979.

[Pratt, 1980] Vaughan R. Pratt. A near-optimal method for reasoning about action. *Journal of Computer and System Science*, 20:231–254, 1980.

[Pratt, 1981] Vaughan R. Pratt. A decidable mu-calculus: Preliminary report. In *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*, pages 421–427, Nashville, TN, 1981.

[Quantz *et al.*, 1994] J. Joachim Quantz, Guido Dunker, and Veronique Royer. Flexible inference strategies for DL systems. In *Proceedings of the International Workshop on Description Logics*, pages 27–35, Bonn, FRG, 1994. The proceedings appeared as Document D-94-10, German Research Center for Artificial Intelligence-DFKI GmbH, Saarbrücken, FRG.

[Quillian, 1966] M. Ross Quillian. *Semantic Memory.* PhD thesis, Carnegie Institute of Technology, Pittsburgh, PA, 1966. BBN Report AFCRL-66-189, Bolt, Beranek, and Newman Inc., Cambridge, MA.

[Quillian, 1967] M. Ross Quillian. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12:410–430, 1967.

[Quillian, 1969] M. Ross Quillian. The teachable language comprehender: A simulation program and theory of language. *Communications of the ACM*, 12(8):459–476, 1969.

[Ralston and Reilly, 1993] Anthony Ralston and Edwin D. Reilly, editors. *Encyclopedia of Computer Science.* van Nostrand Reinhold, New York, NY, third edition, 1993.

[Reiter, 1984] Raymond Reiter. Towards a logical reconstruction of relational database theory. In M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, editors, *On Conceptual Modeling*, pages 191–233. Springer-Verlag, Berlin, FRG, 1984.

[Reiter, 1986] Raymond Reiter. A sound and sometimes complete evaluation algorithm for relational databases with null values. *Journal of the Association for Computing Machinery*, 33(2):349–370, 1986.

[Safra, 1988] Shamuel Safra. On the complexity of ω-automata. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 319–327, 1988.

[Schank, 1973] Roger C. Schank. Identification of conceptualization underlying natural language. In R. C. Schank and K. M. Colby, editors, *Computer Models of Thought and Language*, pages 187–247. Freeman, San Francisco, CA, 1973.

[Schild, 1988] Klaus Schild. Undecidability of subsumption in $\mathcal{U}$. KIT Report 67, Department of Computer Science, Technische Universität Berlin, Berlin, FRG, 1988.

[Schild, 1989] Klaus Schild. Towards a theory of frames and rules. KIT Report 76, Department of Computer Science, Technische Universität Berlin, Berlin, FRG, 1989.

[Schild, 1991a] Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 466–471, Sydney, Australia, 1991.

[Schild, 1991b] Klaus Schild. From terminological logics to modal logics. In *Proceedings of the International Workshop on Terminological Logics*, pages 101–104, Dagstuhl, FRG, 1991. The proceedings ppeared as KIT-Report 89, Department of Computer Science, Technische Universität Berliin, Berlin, FRG.

[Schild, 1991c] Klaus Schild. A tense-logical extension of terminological logics. KIT Report 92, Department of Computer Science, Technische Universität Berlin, Berlin, FRG, 1991.

[Schild, 1993a] Klaus Schild. Combining terminological logics with tense logic. In *Proceedings of the 6th Portuguese Conference on Artificial Intelligence*, pages 105–120, Porto, Portugal, 1993.

[Schild, 1993b] Klaus Schild. Terminological cycles and the propositional μ-calculus. DFKI Research Report RR-93-18, German Research Center for Artificial Intelligence–DFKI GmbH, Saarbrücken, FRG, 1993.

[Schild, 1993c] Klaus Schild. Terminological cycles and the propositional μ-calculus: Extended abstract. In *Proceedings of the AKI Workshop Wissensrepräsentation*, Bonn, FRG, 1993.

[Schild, 1994a] Klaus Schild. Terminological cycles and the propositional μ-calculus. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 509–520, Bonn, FRG, 1994.

[Schild, 1994b] Klaus Schild. Tractable reasoning in a universal description logic: Extended abstract. In *Proceedings of the International Workshop on Description Logics*, pages 9–13, Bonn, FRG, 1994. The proceedings appeared as Document D-94-10, German Research Center for Artificial Intelligence -DFKI GmbH, Saarbrücken, FRG.

[Schild, 1994c] Klaus Schild. A universal description logic as a tractable query language for databases. In *Proceedings of the KI-94 Workshops*, pages 295–296, Saarbrücken, FRG, 1994.

[Schild, 1995] Klaus Schild. The use of description logics as database query languages. In *Proceedings of the KI-95 Workshops*, pages 23–24, Bielefeld, FRG, 1995.

[Schmidt-Schauß and Smolka, 1991] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Schmidt-Schauß, 1989] Manfred Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 421–431, Toronto, Ontario, 1989.

[Schmidt, 1991] Renate A. Schmidt. Algebraic terminological representation. Master's thesis, Department of Mathematics, University of Cape Town, South Africa, 1991.

[Schmolze and Israel, 1983] James G. Schmolze and David J. Israel. KL-ONE: Semantics and classification. In *Research in Knowledge Representation and Natural Language Understanding, BBN Technical Report, 5421*, pages 27–39. Bolt, Beranek, and Newman Inc., Cambridge, MA, 1983.

[Schmolze and Lipkis, 1983] James G. Schmolze and Thomas Lipkis. Classification in the KL-ONE knowledge representation system. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 330–332, Karlsruhe, FRG, 1983.

[Shapiro, 1979] Stuart C. Shapiro. The SNePS semantics network processing system. In N. V. Findler, editor, *Associative Networks: Representation and Use of Knowledge by Computers*, pages 179–203. Academic Press, New York, NY, 1979.

[Smith, 1982] Brian C. Smith. *Reflection and Semantics in a Procedural Language.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1982. Report MIT/LCS/TR-272.

[Sowa, 1991] John F. Sowa, editor. *Formal Aspects of Semantic Networks.* Morgan Kaufmann, San Mateo, CA, 1991.

[Streett and Emerson, 1984] Robert S. Streett and E. Allen Emerson. The propositional mu-calculus is elementary. In *Proceedings of 11th International Colloquium on Automata, Languages and Programming*, pages 465–472, Antwerp, Belgium, 1984.

[Streett and Emerson, 1989] Robert S. Streett and E. Allen Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information & Computation*, 81:249–264, 1989.

[Streett, 1985] Robert S. Streett. Fixpoints and program looping: Reductions from the propositional mu-calculus into propositional dynamic logics of looping. In *Proceedings of the Workshop on Logics of Programs*, pages 359–372, Brooklyn, 1985.

[Tarski and Givant, 1987] Alfred Tarski and Steven Givant. *A Formalization of Set Theory without Variables*, volume 41 of *Colloquium Publications.* American Mathematical Society, Providence, Rhode Island, 1987.

[Tarski, 1941] Alfred Tarski. On the calculus of relations. *Journal of Symbolic Logic*, 6:73–89, 1941.

[Tarski, 1955] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[Trahtenbrot, 1963] B. A. Trahtenbrot. Impossibility of an algorithm for the decision problem in finite classes. *American Mathematical Society Translation Series*, 23(2):1–5, 1963.

[Vardi and Wolper, 1984] Moshe Y. Vardi and Pierre Wolper. Automata theoretic techniques for modal logics of programs (extended abstract). In *Proceedings of the 16th ACM Annual Symposium on Theory of Computing*, pages 446–456, Washington, D.C., 1984.

[Vardi and Wolper, 1986] Moshe Y. Vardi and Pierre Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Science*, 32:183–221, 1986.

[Vardi, 1982] Moshe Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing*, pages 137–146, San Francisco, CA, 1982.

[Vardi, 1985] Moshe Y. Vardi. The taming of converse: Reasoning about two-way computations. In *Proceedings of the Workshop on Logics of Programs*, pages 413–424, Brooklyn, 1985.

[Vardi, 1986] Moshe Y. Vardi. Querying logical databases. *Journal of Computer and System Science*, 33:142–160, 1986.

[Walukiewicz, 1993] Igor Walukiewicz. On completeness of the μ-calculus. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*, pages 136–146, Montreal, Canada, 1993.

[Woods and Schmolze, 1992] William A. Woods and James G. Schmolze. The KL-ONE family. In F.W. Lehmann, editor, *Semantic Networks in Artificial Intelligence*, pages 133–178. Pergamon Press, 1992.

[Woods, 1975] William A. Woods. What's in a link: Foundations for semantic networks. In D. G. Bobrow and A. M. Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 35–82. Academic Press, New York, NY, 1975.

[Wright *et al.*, 1993] J. R. Wright, E. S. Weixelbaum, G. T. Vesonder, K. Brown, S. R. Palmer, J. I. Berman, and H. H. Moore. A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T network systems. *AI-Magazine*, 14(3):69–80, 1993.