Università degli Studi di Roma "La Sapienza"

Dipartimento di Informatica e Sistemistica

Decidability of Class-Based Knowledge Representation Formalisms

Giuseppe De Giacomo

PhD Thesis
1995

Author's Address:
Giuseppe De Giacomo
Dipartimento di Informatica e Sistemistica
Università degli Studi di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italia
E-mail: degiacomo@dis.uniroma1.it

Contents

Preface

This document is the English version of my doctoral thesis [35]. It presents the research work I have done, mostly at the Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", from 1992 to the beginning of 1995, under the supervision of Maurizio Lenzerini.

The thesis is concerned with logic-based knowledge representation formalisms in the tradition of KL-ONE, called Description Logics. The description logics investigated are distinguished by being much more expressive than the usual ones, but still decidable. High expressivity makes it possible to represent all relevant knowledge of complex domains, as those managed by Medical Terminology Servers studied in Medical Informatics. Decidability makes it possible to implement sound and complete reasoning algorithms for such description logics. A great variety of expressive description logics is considered, and for each of them, reasoning procedures are developed and EXPTIME-decidability is proved. This complexity bound is the best one can achieve since the simplest logics considered are already EXPTIME-hard.

The key idea in establishing these results it that, rather then trying to construct a new algorithm for each extended logic, the decision problems for the extended logics are polynomially reduced to decision problems of already known logics in an incremental fashion. Several of these reductions are quite sophisticated, and the proof of correctness is, in many cases, rather involved. Certain reductions are somewhat surprising since the extended logics have logical properties that are quite different from those from which they derive. For instance the basic logics have the finite model property while some of the extended ones don't. Overall this "reduction-based" way of proceeding has proved to be very effective.

The thesis heavily exploits the correspondence that exists between description logics and certain modal logics of programs, called Propositional Dynamic Logics. This correspondence has already allowed other researchers to obtain decidability and complexity characterization for various description logics from well-known results about propositional dynamic logics. However, a significant step further is made in this thesis: instead of just using results for known propositional dynamic logics, new propositional dynamic logics, corresponding to description logics of interest, are introduced, and their decidability and complexity characterization is established.

For the sake of brevity and homogeneity, the work concerning applications to Medical Terminology Servers, which was documented in the original thesis, has been largely sacrificed in the present version, I briefly discuss the main issues in the first chapter.

I have preferred to concentrate on the theoretical results, which are reported in every detail, including full-fledged proofs, in the following chapters. I have also included an appendix concerning a further reduction that I devised just after the original thesis was completed.

Many of the results presented in this thesis were obtained in collaboration with Maurizio Lenzerini, and some of them have already appeared in [34, 36, 41, 37, 38, 40, 39, 43, 45].

It is a pleasure to acknowledge the people that have helped me most during this work. I wish to thank the thesis committee, particularly Paolo Atzeni, Carlo Batini, Paolo Ercoli, and the external reviewers Franz Baader, Piero Torasso, and Angelo Rossi Mori. I am specially indebted to Franz Baader for the numerous suggestions that allowed me to reformulate more rigorously some of the results, and to improve the overall quality of the thesis. I wish to thank all the people of the Dipartimento di Informatica e Sistemistica, in particular the people working in Artificial Intelligence that have always been ready to talk about issues related to the thesis: Luigia Carlucci Aiello, Marco Cadoli, Diego Calvanese, Amedeo Cesta, Francesco Donini, Daniele Nardi, Eugenio Omodeo, Fiora Pirri, Marco Schaerf, Andrea Schaerf. I am especially grateful to Diego Calvanese with whom I discussed in detail many technical points.

I express my gratitude to Fabrizio Consorti (Istituto di IV Clinica Chirurgica, Università di Roma "La Sapienza"), Angelo Rossi Mori and Aldo Gangemi (Istituto Tecnologie Biomediche – CNR), for introducing me to Medical Terminology Servers; to Enrico Franconi, for references to the work done at the German Heart Center, University of Berlin; to Rocco De Nicola, for exposing me to logics for reactive processes, including modal mu-calculus; to Eugenio Omodeo, for a conversation that helped me to finally set right the proof of Lemma 18; to Konstantinos Georgatos, with whom I discussed some aspects of the proof of Lemma 9.

I also would like to thank Yoav Shoham for his hospitality at the Computer Science Department of Stanford University, where part of this research was carried out, and Johan van Benthem for a quite inspiring chat we had in the courtyard of the Center for the Study of Language and Information at Stanford University.

Finally my deepest debt of gratitude goes to Maurizio Lenzerini, who has been a fantastic advisor, co-worker, and friend. He strongly supported and encouraged me throughout the fulfillment of this work.

Special thanks to my loving wife, Claudia, for putting up with many evenings and weekends that I have spent working at this thesis.

Giuseppe De Giacomo

Chapter 1

Introduction

1.1 Background

The research in Artificial Intelligence and Computer Science has always paid special attention to formalisms for the structured representation of information. In Artificial Intelligence, the investigation of such formalisms began with semantic networks and frames, which have been influential for many formalisms proposed in the areas of knowledge representation, databases, and programming languages, and developed towards formal logic-based languages, that will be called here *description logics*[1]. Basically, description logics represent knowledge in terms of objects (individuals) grouped into classes (concepts) and pairs of objects grouped into relations (roles). Classes are denoted by using appropriate constructs. Interdependencies between classes (such as inclusion, disjointness, etc.) are established by means of assertions.

Two main advantages in using structured formalisms for knowledge representation have been advocated, namely, epistemological adequacy, and computational effectiveness. In the last decade, many efforts have been devoted to an analysis of these two aspects. In particular, starting with [14], the research on the computational complexity of the reasoning tasks associated with description logics has shown that in order to ensure decidability and/or efficiency of reasoning in all cases, one must renounce some of the expressive power [76, 81, 83, 82, 47, 48, 49]. These results have led to a debate on the trade-off between expressive power of representation formalisms and worst-case efficiency of the associated reasoning tasks. Recently, this issue has been one of the main themes in the area of description logics, and has led to at least four different approaches to the design of knowledge representation systems.

- In the first approach, the main goal of a description logic is to offer powerful mechanisms for structuring knowledge, as well as sound and complete (but possibly non terminating) reasoning procedures. Little attention is paid to both decidability and computational complexity of the reasoning procedures. Systems like OMEGA [1] can be considered as following this approach.

[1] Terminological logics, and concept languages are other possible names.

- The second approach advocates a careful design of the description logics so as to offer as much expressive power as possible while retaining the possibility of sound, complete, and efficient (often polynomial in the worst case) inference procedures. Much of the research on CLASSIC [15] follows this approach.

- The third approach, similarly to the first one, advocates very expressive languages, but, in order to achieve efficiency, accepts incomplete reasoning procedures. LOOM [78] and KL-ONE [16] are representatives of this approach. No general consensus exists on what kind of incompleteness is acceptable. Perhaps, the most interesting attempts are those which resort to a non-standard semantics for characterizing the form of incompleteness [91, 13, 49].

- Finally, the fourth approach is based on what we can call "the expressiveness and decidability thesis", and aims at defining description logics that are both very expressive and decidable, i.e. designed in such a way that sound, complete, and terminating procedures exist for the associated reasoning tasks. Great attention is given in this approach to the complexity analysis for the various sublogics, so as to devise suitable optimization techniques and to single out tractable subcases. This approach is the one followed in the design of KRIS [4].

The work presented in this thesis adheres to the fourth approach. It aims at both identifying very expressive description logics with decidable associated decision problems, and characterizing the computational complexity of reasoning in such description logics.

1.2 Medical Terminology Servers

In investigating description logics of this type, we have in mind a particular application of Medical Informatics, namely Medical Terminology Servers. It is a common opinion in Medical Informatics that the quality and the effectiveness of automatic information and record keeping in health-care depends, to a large extent, on the efficient processing and interpreting of medical terminologies and concepts [12, 116, 29, 103]. This has led to the proposal of isolating a special subsystem of health-care information systems to which are delegated services involving the representation and reasoning about medical concepts [100, 53, 19, 111]. We call such a subsystem Medical Terminology Server, borrowing this name from GALEN, one of the main research projects in the area.

A Medical Terminology Server is a knowledge representation system in which knowledge about a given medical domain (ranging from very specific to very general) is represented in terms of concepts (classes) and links between concepts. Only part of such knowledge is represented explicitly, and reasoning services are provided to extract implicit knowledge from the explicit one.

Typical reasoning services of a Medical Terminology Server are: *subsumption checking*, i.e. checking, taking in account the knowledge possessed by the server, if a concept is a specialization of another one, or if two concepts are equivalent; and *consistency checking*, i.e. checking if a piece of information (e.g. a concept, or an assertion of inclusion or equivalence between concepts) is consistent with the knowledge in the server.

By exploiting the basic reasoning services above, additional services may be provided. For example a Medical Terminology Server may generate "canonical forms" of concepts wrt certain parameters, or it may translate the inner representation of concepts in various target languages understandable by humans or by other information systems.

Applications of Medical Terminology Servers span a broad spectrum of Medical Informatics. They include decision support tools, literature retrieval systems, outcome research (extracting information from medical records), structured data entry, predictive data entry, intelligent medical records, patient record systems, patient record retrieval, interlingua, expert systems, hospital departmental information systems [101, 85, 53, 57, 25, 111, 24, 20, 6, 79, 116]. Below we show some examples which clarify the role Medical Terminology Servers may have in different application domains. The examples are taken from [53, 85].

Example 1. A hospital has installed *decision-support tools.* The hospital information system will check the new orders entered into the system for conflicts. One of such target alert may be warning to the ordering clinician whenever a nonsteroidal anti-inflammatory agent is ordered for a patient diagnosed as having peptic ulcer disease. The designers must ensure that the system will recognize every existing nonsteroidal anti-inflammatory drug by name, as well as every possible reference to acid peptic disease.

We can delegate to the Medical Terminology Server the task of recognizing if a given kind of drug is a nonsteroidal anti-inflammatory one or not, and recognizing if the disease diagnosed in a patient is an acid-peptic disease or if it induces an acid-peptic condition. Note that the reasoning service involved is subsumption wrt the knowledge in the server.

Example 2. A *literature-retrieval system* is built that will attempt to recognize concepts that are synonymous. A query is entered about calcium-channel blockers and their use in stroke. The system must deal with the fact that writers of articles use many synonymous terms for calcium-channel blockers, such as "calcium blockers", "calcium antagonist", and the individual name of different agents. Likewise, stroke has many synonyms, such as "cerebrovascular accident" or "CVA", and may be referred to generally as "cerebrovascular disease".

We can delegate to the Medical Terminology Server (possibly assisted by a natural language recognition system) the task of recognizing whether different terms denote the same entity (calcium-channel blockers, stroke), and recognizing entities that are specializations of those requested. Note that again the reasoning service involved is subsumption wrt the knowledge in the server.

Example 3. A group is responsible for *outcomes research.* It is desirable to track all patient data, including symptoms, yet not have to process charts manually. A computer program is designed to extract information from patient records, but the effectiveness of the program depends on the its ability to deal with variations in the descriptions of patients' symptoms. For example, what is written as "post-prandial stomach pain" in one chart is described as "abdominal pain after meals" in another.

We can delegate to the Medical Terminology Server the task of recognizing that the concept denoted by "post-prandial stomach pain" is actually the same one denoted by "abdominal pain after meals". In general, the Medical Terminology Server can be used to recognize if two symptoms reported on different patient records are the same or one a specialization of the other, in order to aggregate data correctly for statistics. Note that again we make use of subsumption wrt the knowledge in the server.

Example 4. A renal dialysis center wants to develop a medical record that will support observation and controlled trials as part of routine patient care. Much of the patient information will be collected by nurses and physicians through *structure data entry* to ensure that study parameters are rigorously assessed. The designers require a standard source of possible concepts, symptoms, and corresponding values in order to integrate different trials with overlapping data elements and share their data with other participating centers.

We can use the Medical Terminology Server as the standard source of concepts required above. The server will have to use its reasoning services for accomplishing this task.

Example 5. A clinician is visiting a patient using a *predictive data entry* device. He discovers that the patient has a fracture, and enters "fracture" into the system. The system automatically displays "has location" among others modifiers, possibly suggesting some of the most common alternatives as "humerus", "ulna", "radius", etc. The clinician answers "humerus", the system accordingly displays other modifiers. Meanwhile the system on-line checks for the consistency of the data entered. For example it does not allow one to specify that the patient has a fracture of the eyebrow because eyebrow is not a bone and fractures may only be located in bones.

We can designate the Medical Terminology Server as the provider of a canonical description of the concept given in input ("fracture") so to exploit its structure to ask for more data (the qualification of the modifier "has location"). In producing the canonical description of a concept, the server must reason on the knowledge about the domain it has. In addition the server can be used to check for the consistency of the data entered wrt its knowledge (refuting "fracture that has location in eyebrow").

From what has been said so far it should be apparent that there are strong connections between the notion of Medical Terminology Server and many knowledge representation systems proposed in Artificial Intelligence and in Computer Science (indeed, the formalisms adopted by the various Medical Terminology Server proposals come from these fields.) However it must be stressed that the notion of Medical Terminology Server has deep roots, specific to Medicine, in the so called medical concept classification systems, and systematized medical nomenclatures [102, 104, 103]. Roughly, we may divide these systems into three categories according to their representing and reasoning capabilities.

The first category is that of the so called coding systems. Coding systems (e.g. ICD9-CM [124], CMIT [61], SNOMED-III [30], MeSH [80]) are based on the enumerative classification of medical concepts in a given domain. They are composed by experts systematically enumerating all possible concepts in the domain. Concepts are organized in hierarchies (sometimes a single hierarchy, sometimes multiple hierarchies), which are also represented by explicit enumeration. To each concept is assigned a standard code (usually, an alphanumeric string) according to such hierarchies. For example in ICD9-CM under "chronic obstructive pulmonary disease and allied conditions (490-496)" we find "asthma" coded as 493, "extrinsic asthma" as 493.0, "intrinsic asthma" as 493.1, "asthma, unspecified" as 493.9. Originally coding systems were developed for a paper-based support (indeed, usually they are contained in books) and their primary purpose is to discipline the use of medical terms by providing a controlled vocabulary in which each concept has a precise connotation, its code. Given their nature, coding systems do not provide for reasoning procedures.

Coding systems have recently evolved in concept systems that allow for a structured representation of medical concepts, though they do not provide reasoning procedures. An example is the semantic network developed within UMLS [77]. UMLS (Unified Medical Language System) is a project of the National Library of Medicine that aims at providing an integration of existing controlled medical vocabularies to facilitate access and transformation between computer-based information sources. UMLS organizes the more general concepts ("concept types") by means of a semantic network, while more specific concepts are supplied by the so called source vocabularies which are again hierarchical. Another example is MED (Medical Entity Dictionary) [27]. MED is a structured knowledge representation language developed by the Center for Medical Informatics of Columbia University at the Columbia-Presbyterian Medical Center. Concepts, in MED, are represented by means of a frame-based language similar to those developed in Artificial Intelligence. No reasoning services are provided. There have also been proposals of systems for describing medical concepts based on semantical data models for databases, as the Entity-Relationship Model. The most attractive aspects of such semantical data models are their high descriptive power, and the ease with which they interface actual databases [105, 69, 70].

The third category of concept systems is the one that directly led to the notion of Medical Terminological Server. It includes systems that allow for a structured representation of knowledge and provide reasoning procedures to extract implicit knowledge from the knowledge explicitly represented. Several proposals have been made. Such proposals are based on different formalisms, and aim at somewhat different goals, but they all share the notion of Medical Terminology Server. One of the major proposals is the one developed within the GALEN project [98, 101, 99, 100]. GALEN (Generalized Architecture for Language Encyclopedias and Nomenclature in Medicine) is a project funded by the European Community, having the purpose of developing language-independent concept representation systems as the foundations for the next generation of multilingual coding systems. A specific concept representation formalism, GRAIL (GALEN Representation and Integration Language), has been developed within GALEN. Such a formalism is a logic-based language resembling a description logic. Similar proposals have been formulated within the CANON Group [26, 23, 53, 52, 19, 57]. The CANON Group was founded in United States by Medical Informatics researchers having the goal of establishing a basis for the "canonical" representation of medical concepts. They are analyzing different formalisms to approach the problem, based on frame-based languages, semantic networks, and conceptual graphs. Conceptual graphs have been proposed as a formalism for concept-based representation systems by others researchers as well, e.g. in [9, 97]. Another research direction is that of the project group Medicine-Informatics at the German Hearth Center in Berlin [112, 111, 113]. This group is exploring the possible application of the system BACK [92] (a general-purpose knowledge representation system based on a description logic) for modeling medical and patient-related information, in order to exploit its representing and reasoning capabilities as the core of a Medical Terminology Server.

Observe that though different formalisms have been proposed as basic representing and reasoning paradigms for Medical Terminology Servers – i.e. semantic networks, frame-based languages, conceptual graphs, semantic data models, logic-based languages – all these formalisms belong to the same family, that of *class-based representation formalisms*.

1.3 Goals and main results of the thesis

Description logics offer a clean, formal and effective framework for analyzing several important issues related to class-based representation formalisms, such as expressive power, deduction algorithms, and computational complexity of reasoning. Note, however, that in order to address these issues, description logics should be sufficiently general, but, at the same time, sufficiently simple so as to not fall into undecidability of reasoning.

Currently those description logics that have been studied from a formal point of view suffer from several limitations that prevent them form being able to capture a sufficiently broad family of class-based representation formalisms. In particular they are too weak to meet the requirements imposed by modeling complex domains as those often involved in Medical Terminology Servers. Several papers (e.g. [100, 112, 62, 18, 115, 50, 68]) have pointed out that in real applications, the following features are often called for.

1. The availability of assertions for imposing mutual dependencies between classes. The basic mechanism for this feature is the so-called inclusion assertion, stating that every instance of a class is also an instance of another class. Much of the work done in description logics assumes that all the knowledge on classes is expressed through the use of class descriptions, and rules out the possibility of using this kind of assertion (note that the power of assertions vanishes with the usual assumption of acyclicity of class definitions).

2. The availability of a full range of constructs in order to form concept and role descriptions. Besides the constructs corresponding to the usual boolean connectives (union, intersection, complement), and existential and universal qualifications, two important types of constructs must be mentioned: those for building complex role descriptions, in particular inverse roles (e.g. "has-direct-part" is the inverse of "direct-part-of") and reflexive transitive closure (e.g. "part-of" is the reflexive transitive closure of "direct-part-of"); and those for expressing cardinality constraints ranging from functional restrictions (i.e. that a role is functional for the instances of a given class) to qualified number restrictions (a

generalization of functional restrictions, stating the minimum and the maximum number of links an instance of a class has with instances of another class).

3. The availability of boolean constructs on roles, and the possibility to state assertions on roles, expressing inclusion, disjointness, etc.

4. The possibility of aggregating individuals in tuples, and then of grouping tuples into n-ary relations as opposed to binary relations only (roles).

5. The possibility of asserting properties of individuals. Usually this is done in terms of the so-called membership assertions. Two kinds of membership assertions are taken into account, one for stating that an object is an instance of a given class, and another one for stating that two objects are related to by means of a given role.

6. The possibility of defining classes recursively. In this way it is possible to model, for example, terminating sequences, non-terminating sequences, as well as many other data structures of Computer Science.

The main goal of this thesis is to introduce description logics with the above features, studying their properties, and to devise reasoning procedures for them, investigating decidability and characterizing their computational complexity.

To this end, we resort to the work by Schild [108], which singled out a tight correspondence between description logics and propositional dynamic logics, which are modal logics specifically designed for reasoning about program schemes. The correspondence is based on the similarity between the interpretation structures of the two kinds of logics: at the extensional level, objects in description logics correspond to states in propositional dynamic logics, whereas connections between two objects correspond to state transitions. At the intensional level, classes correspond to propositions, and roles corresponds to programs. This correspondence is extremely useful for two reasons. On the one hand, it makes it clear that reasoning about assertions on classes is equivalent to reasoning about single dynamic logic formula. On the other hand, the large body of research on decision procedures in propositional dynamic logics (see, for example, [74]) can be exploited in the context of description logics, and, inversely, the various works on tractability/intractability of description logics (see for example [48]) can be used in the context of propositional dynamic logics. We sustain that the work on propositional dynamic logics is a good starting point for our investigation, because it provides a general method for reasoning with:

- assertion on classes;

- inverses of roles (indeed, several propositional dynamic logics proposed in the literature include a construct that exactly corresponds to the inverse of roles).

The reasoning tasks we focus on are the usual ones.

- Satisfiability of concepts, i.e. checking if a concept expression C admits a non-empty interpretation (has some instances).

- Satisfiability of TBoxes, i.e. checking if a TBox is consistent, where a TBox is the collection of inclusion assertions that makes up the knowledge the system is provided with. We assume inclusion assertions to have the form $C_1 \sqsubseteq C_2$, where C_1 and C_2 can be any concept expressions (this is the most general form of assertions on classes).

- Logical implication in TBoxes, i.e. checking if a concept C_1 is subsumed by a concept C_2 wrt the knowledge in the TBox $\mathcal{K}$, written $\mathcal{K} \models C_1 \sqsubseteq C_2$.[2]

We also consider assertional reasoning, i.e. reasoning taking into account knowledge about single individuals. However such knowledge will be expressed not only through the usual membership assertions (ABox), but also through inclusion assertions involving special atomic concepts denoting exactly a single individual. This allows us not to include assertional reasoning among the basic reasoning tasks.

The basic reasoning tasks above are not independent. In particular we can easily reformulate both satisfiability of single concepts and satisfiability of TBoxes in terms of logical implications. Indeed logical implication seems to be the most general reasoning task. However we will see that for most of the logics we will introduce, logical implication can be reformulated as satisfiability of a single concept (it is essentially the ability of expressing reflexive transitive closure of roles that allows us to capture the knowledge in the TBox within a single concept).

Next to each description logic we will introduce a corresponding propositional dynamic logic[3]. Most of these propositional dynamic logics have not been studied yet, and decidability and computational characterization (of both satisfiability and logical implication) are established within this thesis.

Figure 1.1 depicts the description logics studied in the thesis. The weaker logics are at the bottom of the figure while the stronger ones are at the top.

A *line* (either thin or thick) between two logics denotes that the logic above is an extension (in the sense that it has more constructs) of the logic below. If the line is a *thick line* then it means that, a (non trivial) reduction, from (the reasoning tasks of) the logic above to (the reasoning tasks of) the logic below, is exhibited in the thesis. The *dashed thick line* between $\mathcal{CI}$ and $\mathcal{C}$ denotes a reduction from $\mathcal{CI}$ to $\mathcal{C}$ which was not contained in the original version of the thesis and it is included here as an appendix.

The logics in the *closed area* have already been studied, and the decidability and computational complexity characterization of the basic reasoning tasks is already known.

Figure 1.2 is the analogue of Figure 1.1 for the corresponding propositional dynamic logics. The meaning of the various lines and of the closed area is the same as before.

Let us briefly introduce the logics in the pictures.

[2]Note that checking if a new piece of information is consistent with the knowledge of the system, is expressible as checking that the TBox does not logically imply the negation of the new piece of information.

[3]We use the term propositional dynamic logic in a slightly more general sense then usual, so as to include the basic multimodal logic K_i, and modal mu-calculus.

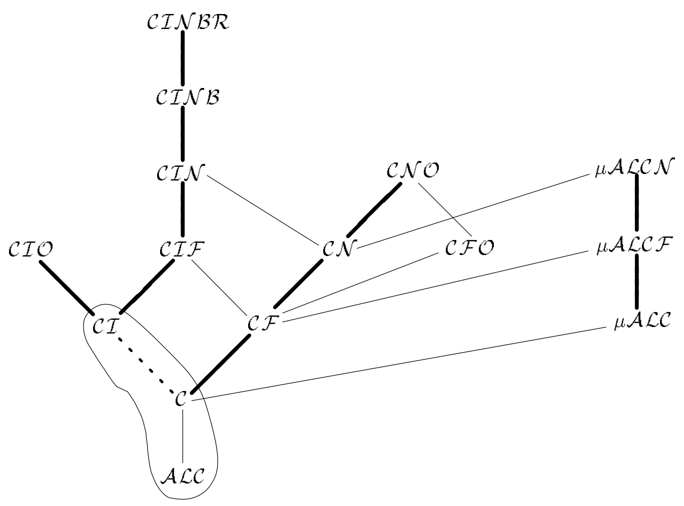

Figure 1.1: Description logics studied in the thesis

$\mathcal{ALC}$ is a very well known description logic [110]. It includes boolean constructs (union, intersection, and complement), existential qualification and universal qualification for building complex concept expressions, while role can only be atomic. $\mathcal{ALC}$ corresponds to the well-known modal logic K_i [108], which is the basic normal multimodal logic [60, 63, 22, 66]. Satisfiability of an $\mathcal{ALC}$ concept (satisfiability of a $\mathcal{K}_i$ formula) is known to be PSPACE-complete while logical implication for $\mathcal{ALC}$ (for K_i) is EXPTIME-complete.

$\mathcal{C}$ is the description logic obtained from $\mathcal{ALC}$ by adding the following role constructs: union, chaining, reflexive transitive closure, and identity role over a concept (see [108, 3]). $\mathcal{C}$ corresponds to the propositional dynamic logic $\mathcal{D}$, which is the original propositional dynamic logic introduced in [56]. All the basic reasoning tasks in $\mathcal{C}$ ($\mathcal{D}$) are known to be EXPTIME-complete.

$\mu\mathcal{ALC}$ is obtained from $\mathcal{ALC}$ by adding two concept constructs denoting the least fixpoint and the greatest fixpoint of concept expressions (see Chapter 8 for details). Notabily, the fixpoint constructs allow for recursive concept definitions within the usual descriptive semantics. Observe that even if no role constructs are present, $\mu\mathcal{ALC}$ is actually an extension of $\mathcal{C}$, since all concept denotable in $\mathcal{C}$ are also denotable in $\mu\mathcal{ALC}$. Indeed using fixpoints we can emulate all role expressions occurring in a $\mathcal{C}$ concept. The correspondent propositional dynamic logic μK_i is the modal mu-calculus [71], which is known to be decidable and EXPTIME-complete. The correspondence was derived independently by both Schild and the author in [109] and [38] respectively.

The description logics (propositional dynamic logics) introduced in this thesis are

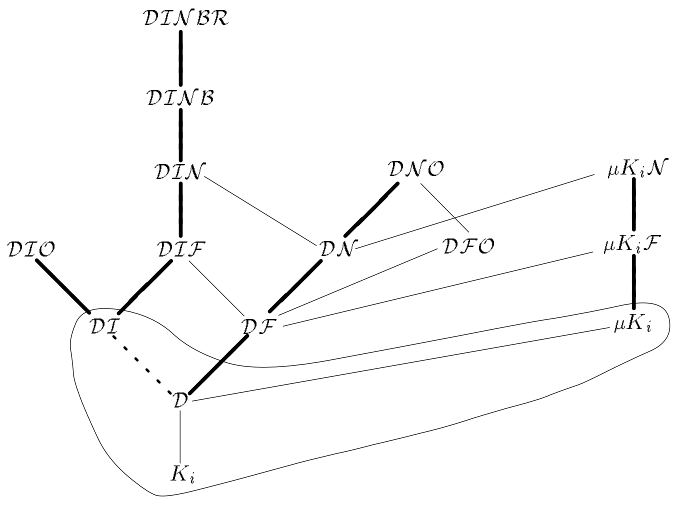

Figure 1.2: Propositional dynamic logics studied in the thesis

obtained from $\mathcal{C}$ and $\mu\mathcal{ALC}$ ($\mathcal{D}$ and μK_i) by adding constructs either on concepts (formulae) or roles (programs). The presence of such constructs is reflected in the name of the logics.

$\mathcal{I}$ in the name of a logic indicated the presence of *inverse roles* (converse programs in propositional dynamic logic). In all description logics introduced that include inverse roles, there is a perfect symmetry between atomic roles and inverse of atomic roles, in the sense that all constructs dealing with atomic roles, deal with inverse of atomic roles as well, similarly for the corresponding propositional dynamic logics.

$\mathcal{F}$ in the name of a logic indicates the presence of *functional restrictions*. In description logics, a functional restriction forces a specified atomic role or its inverse to be functional wrt the individuals that satisfy it. Similarly for the corresponding propositional dynamic logics. Observe the difference between functional restrictions on atomic programs and the assumption that atomic programs are deterministic, characterizing the so called *deterministic propositional dynamic logics*. The first impose the functionality of a given program *locally* (i.e. wrt states that are forced to satisfy the restriction), while the other assumes the functionality of each atomic program once and for all (i.e. for all possible states).

$\mathcal{N}$ in the name of a logic indicates the presence of *qualified number restrictions*. Qualified number restrictions have a correspondent notion in modal logic, the *graded modalities*. Though, to our knowledge we are the first to study full-fledged propositional dynamic logics that include graded modalities.

$\mathcal{B}$ in the name of a logic indicates the presence of *boolean constructs for atomic roles* (programs), together with the ability of stating assertions on boolean combinations of atomic roles (programs). Although negation of a role is allowed, it is defined so as not to introduce, as a side effect, the ability to denote the universal role, by means of a boolean expression of atomic roles (see the discussion in Chapter 5).

$\mathcal{R}$ in the name of a description logic indicates the presence of *n-ary relations* in place of atomic roles, with suitable constructs to build complex relations, roles, and concepts, similarly for the corresponding propositional dynamic logics.

Finally, $\mathcal{O}$ in the name of a logic indicates the presence of special atomic concepts (formulae) called *names* denoting exactly a single individual. Note that by means of names, ABoxes (collections of membership assertions), and constructs involving single individuals as ONE-OF or FILLS can be represented. Names corresponds to the notion of *nominals* in modal logics. Propositional dynamic logics with nominals are often called *combinatory propositional dynamic logics.* The results on names in this thesis close some open problems related to combinatory propositional dynamic logics, by characterizing the computational complexity of *deterministic combinatory propositional dynamic logic* (which is easily reduced to $\mathcal{DFO}$), and establishing the decidability and characterizing the computational complexity of *converse combinatory propositional dynamic logic* (which is easily reduced to $\mathcal{DIO}$).

The main results of the thesis can be summarized as follows:

- *We have defined and studied the new logics shown in Figure 1.1 and Figure 1.2.*
- *We have established the decidability of their reasoning tasks.*
- *We have characterized the computational complexity of their reasoning tasks as EXPTIME-complete, by reducing them to reasoning tasks of known propositional dynamic logics (either $\mathcal{D}$, $\mathcal{DI}$, or μK_i).*

Research on description logics has systematically investigated concept satisfiability and concept subsumption for a wide range of constructs (e.g. [47]). The work reported in this thesis can be seen as the analogue of that study, when TBoxes of the most general form (no restrictions on cycles) are taken into account. Indeed, we systematically investigate satisfiability and concept subsumption wrt to TBoxes (both expressible in terms of logical implication in TBoxes) for a wide variety of constructs. Note, however, that emphasis is put on extending the set of constructs, instead of cutting it down. This is because, as mentioned above, even for the simple description logic $\mathcal{ALC}$, reasoning tasks that take into account TBoxes are EXPTIME-complete.

1.4 Structure of the thesis

The thesis is organized in nine chapters plus an appendix. The contents of each chapter is reported below.

Chapter 1 is the present introduction.

Chapter 2 introduces the relevant background on both description logics and propositional dynamic logics, the correspondence between description logics and propositional dynamic logics [108], and some convenient notions used later.

Chapter 3 introduces the description logic $\mathcal{CIF}$ ($\mathcal{C}$ plus inverse roles and functional restrictions) and the corresponding propositional dynamic logic $\mathcal{DIF}$. The decidability and the computational characterization as EXPTIME-complete of the reasoning tasks in $\mathcal{CIF}$ and $\mathcal{DIF}$, is established by exhibiting a polynomial reduction from satisfiability in $\mathcal{DIF}$ to satisfiability in $\mathcal{DI}$ (i.e. converse propositional dynamic logic). The reduction is based on adding special "constraints" so as to represent functional restrictions within $\mathcal{DI}$. A discussion on the results ends the chapter.

Chapter 4 introduces the description logic $\mathcal{CIN}$ ($\mathcal{C}$ plus inverse roles and qualified number restrictions) and the corresponding propositional dynamic logic $\mathcal{DIN}$. The decidability and computational characterization as EXPTIME-complete of the reasoning tasks in $\mathcal{CIN}$ and $\mathcal{DIN}$ is established by showing a polynomial reduction from satisfiability in $\mathcal{DIN}$ to satisfiability in $\mathcal{DIF}$. The reduction is defined in two steps: first every atomic role is *reified*; then qualified number restrictions are reduced to expressions involving only functional restrictions as cardinality constraints. A much simpler technique to reduce $\mathcal{CN}$ and $\mathcal{DN}$ to deterministic propositional dynamic logic is also presented. A discussion on the results ends the chapter.

Chapter 5 introduces the description logic $\mathcal{CINB}$ ($\mathcal{CIN}$ plus boolean expressions of roles and inclusion assertions on atomic roles) and the corresponding propositional dynamic logic $\mathcal{DINB}$. The decidability and the computational characterization as EXPTIME-complete of the reasoning tasks, is established by exhibiting a polynomial reduction from logical implication in $\mathcal{DINB}$ to logical implication in $\mathcal{DIN}$. Such reduction makes use of the reification technique introduced in Chapter 4. Note that for such logics, logical implication cannot be readily reduced to satisfiability as in the previous cases, because of the presence of inclusion assertions on atomic roles (axioms on atomic programs). A discussion on the results ends the chapter.

Chapter 6 introduces the description logic $\mathcal{CINBR}$ ($\mathcal{CINB}$ plus boolean expression on atomic n-ary relations and inclusion assertions on atomic n-ary relations). The corresponding propositional dynamic logic $\mathcal{DINBR}$ is not explicitly presented in this case. The decidability and the computational characterization as EXPTIME-complete of the reasoning tasks, is established by exhibiting a polynomial reduction from logical implication in $\mathcal{CINBR}$ to logical implication in $\mathcal{DIN}$. Such a reduction is in line with the one shown in Chapter 5.

Chapter 7 deals with individuals. It shows how to reduce reasoning with both ABox and TBox to reasoning with only a TBox for the description logics $\mathcal{CN}$ and $\mathcal{CI}$. This is done by showing a polynomial reduction from satisfiability of $\mathcal{CN}$ ($\mathcal{CI}$) ABoxes and TBoxes to satisfiability of a single $\mathcal{DN}$ ($\mathcal{DI}$) formula. A discussion on the results ends the chapter, showing that the reductions presented are general enough to allow for polynomially reducing reasoning tasks in $\mathcal{CNO}$ and $\mathcal{CIO}$ to reasoning tasks in $\mathcal{CN}$ and $\mathcal{CI}$, respectively. Thus decidability and computational characterization as EXPTIME-complete is established for $\mathcal{CNO}$ and $\mathcal{CIO}$ and the corresponding propositional dynamic logics.

Chapter 8 starts with a discussion on the various semantics for recursive definitions of concepts, arguing for an unifying approach that allows for the various semantics to coexist in the same formalism. To this end, fixpoints of concept expressions are introduced, and a correspondence is devised with modal mu-calculus. Specifically the reasoning tasks in the description logics $\mu\mathcal{ALC}$ are shown to be reducible to satisfiability of modal mu-calculus, thus establishing their decidability and computational characterization as EXPTIME-complete. Next, qualified number restrictions are taken into account, getting the description logic $\mu\mathcal{ALCN}$ and the corresponding extended modal mu-calculus $\mu K_i\mathcal{N}$. Decidability and computational characterization as EXPTIME-complete are established for these logics as well, by showing a reduction from satisfiability in $\mu\mathcal{ALCN}$ to satisfiability in modal mu-calculus interpreted over deterministic structures. Other interesting properties of description logics with fixpoints are discussed in the chapter.

Finally the appendix contains the details of a polynomial reduction from $\mathcal{DI}$ ($\mathcal{CI}$) to $\mathcal{D}$ ($\mathcal{C}$), which was not included in the original version of the thesis.

Chapter 2

Preliminaries

In this chapter we present the basic notions regarding both description logics and propositional dynamic logics. We refer the reader to [82] and [74] for an introduction to the subjects. We also prove some propositions to be used in the following chapters.

2.1 Description logics

Description logics allow one to represent a domain of interest in terms of *concepts* and *roles*. Concepts model classes of individuals, while roles model relationships between classes. Starting with atomic concepts and atomic roles, which are concepts and roles described simply by a name, complex concepts and roles can be built by means of suitable constructs.

In the following, we focus on the description logic $\mathcal{CI}$ which has been studied in [108].[1] The formation rules of $\mathcal{CI}$ are specified by the following abstract syntax:

$$\begin{array}{lcl} C & ::= & \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid C_1 \Rightarrow C_2 \mid \neg C \mid \exists R.C \mid \forall R.C \\ R & ::= & P \mid R_1 \sqcup R_2 \mid R_1 \circ R_2 \mid R^* \mid R^- \mid id(C) \end{array}$$

where A denotes an atomic concept, C (possibly with a subscript) denotes a concept, P denotes an atomic role, and R (possibly with a subscript) denotes a role.

Note that $\mathcal{CI}$ is a very expressive language, comprising all usual concept constructs, and a rich set of role constructs, namely: union of roles $R_1 \sqcup R_2$, chaining of roles $R_1 \circ R_2$, reflexive-transitive closure of roles R^*, inverse roles R^-, and the identity role $id(C)$ projected on C.

Concepts are interpreted as subsets of a domain, while roles are interpreted as binary relations over such a domain. Formally, an *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a *domain of interpretation* $\Delta^\mathcal{I}$, and an *interpretation function* $\cdot^\mathcal{I}$ mapping every

[1]The description logics $\mathcal{C}$ and $\mathcal{ALC}$ are obtained from $\mathcal{CI}$ by dropping inverse roles and all role constructs respectively.

concepts to a subset of $\Delta^\mathcal{I}$ and every role to a subset of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ as follows[2]:

$$\begin{array}{l}
A^\mathcal{I} \subseteq \Delta^\mathcal{I} \\
\top^\mathcal{I} = \Delta^\mathcal{I} \\
\bot^\mathcal{I} = \emptyset \\
(\neg C)^\mathcal{I} = \Delta^\mathcal{I} - C^\mathcal{I} \\
(C_1 \sqcap C_2)^\mathcal{I} = C_1^\mathcal{I} \cap C_2^\mathcal{I} \\
(C_1 \sqcup C_2)^\mathcal{I} = C_1^\mathcal{I} \cup C_2^\mathcal{I} \\
(C_1 \Rightarrow C_2)^\mathcal{I} = (\neg C_1)^\mathcal{I} \cup C_2^\mathcal{I} \\
(\exists R.C)^\mathcal{I} = \{d \in \Delta^\mathcal{I} \mid \exists d'.(d,d') \in R^\mathcal{I} \text{ and } d' \in C^\mathcal{I}\} \\
(\forall R.C)^\mathcal{I} = \{d \in \Delta^\mathcal{I} \mid \forall d'.(d,d') \in R^\mathcal{I} \text{ implies } d' \in C^\mathcal{I}\} \\
\\
P^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I} \\
(R_1 \sqcup R_2)^\mathcal{I} = R_1^\mathcal{I} \cup R_2^\mathcal{I} \\
(R_1 \circ R_2)^\mathcal{I} = R_1^\mathcal{I} \circ R_2^\mathcal{I} \\
(R^*)^\mathcal{I} = (R^\mathcal{I})^* = \bigcup_{i \geq 0} (R^\mathcal{I})^i \\
(R^-)^\mathcal{I} = \{(d_1,d_2) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid (d_2,d_1) \in R^\mathcal{I}\} \\
id(C)^\mathcal{I} = \{(d,d) \in \Delta^\mathcal{I} \times \Delta^\mathcal{I} \mid d \in C^\mathcal{I}\}.
\end{array}$$

A concept is satisfiable if there exists an interpretation $\mathcal{I}$ such that $C^\mathcal{I} \neq \emptyset$, otherwise the concept is unsatisfiable. An interpretation $\mathcal{I}$ is a model of a concept C if $\mathcal{I}$ satisfies C.

A TBox $\mathcal{K}$ (i.e. intensional knowledge base) is a finite set on inclusion assertions of the form $C_1 \sqsubseteq C_2$, where C_1 and C_2 are general concepts. An interpretation $\mathcal{I}$ is a model of an inclusion assertion $C_1 \sqsubseteq C_2$ if $C_1^\mathcal{I} \subseteq C_2^\mathcal{I}$. An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{K}$ if $\mathcal{I}$ is a model of each inclusion assertion in $\mathcal{K}$. A TBox $\mathcal{K}$ is satisfiable if it has a model. A TBox $\mathcal{K}$ logically implies an assertion $C_1 \sqsubseteq C_2$, written $\mathcal{K} \models C_1 \sqsubseteq C_2$, if $C_1 \sqsubseteq C_2$ is satisfied by every model of $\mathcal{K}$. Observe that $\mathcal{K} \models C_1 \sqsubseteq C_2$ expresses that the concept C_1 is subsumed by C_2 wrt the TBox $\mathcal{K}$.[3]

Note that each basic reasoning task can be (linearly) reformulated as logical implication in a TBox. Namely satisfiability of a concept C can be reformulated as $\emptyset \not\models C \sqsubseteq \bot$, satisfiability of a TBox $\mathcal{K}$ as $\mathcal{K} \not\models \top \sqsubseteq \bot$.

2.2 Propositional dynamic logics

We focus on the propositional dynamic logic $\mathcal{DI}$ (Converse PDL [56]) which as it turns out corresponds to $\mathcal{CI}$.[4] The abstract syntax of $\mathcal{DI}$ is as follows:

$$\phi \quad ::= \quad \top \mid \bot \mid A \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \Rightarrow \phi_2 \mid \neg\phi \mid < r > \phi \mid [r]\phi$$

$$r \quad ::= \quad P \mid r_1 \cup r_2 \mid r_1 ; r_2 \mid r^* \mid r^- \mid \phi?$$

[2]The notation $(R^\mathcal{I})^i$ stands for i repetitions of $R^\mathcal{I}$ – i.e., $(R^\mathcal{I})^1 = R^\mathcal{I}$, and $(R^\mathcal{I})^i = R^\mathcal{I} \circ (R^\mathcal{I})^{i-1}$.

[3]Accordingly, a concept C_1 is subsumed by a concept C_2, if $\emptyset \models C_1 \sqsubseteq C_2$, i.e. if for every interpretation $\mathcal{I}$, $C_1^\mathcal{I} \subseteq C_2^\mathcal{I}$.

[4]The propositional dynamic logic $\mathcal{D}$ (PDL [56]) is obtained from $\mathcal{D}$ by dropping converse programs r^-, while K_i is obtained by allowing only atomic programs.

where A denotes a propositional letter, ϕ (possibly with a subscript) denotes a formula, P denotes an atomic program, and r (possibly with a subscript) denotes a program.

The semantics of propositional dynamic logics (see [74]) is based on the notion of (Kripke) structure, which is defined as a triple $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$, where $\mathcal{S}$ denotes a non-empty set of states, $\{\mathcal{R}_P\}$ is a family of binary relations over $\mathcal{S}$ such that each atomic program P is given a meaning through $\mathcal{R}_P$, and Π is a mapping from $\mathcal{S}$ to propositional letters such that $\Pi(s)$ determines the letters that are true in the state s. The basic semantical relation is "a formula ϕ holds at a state s of a structure M", which is written $M, s \models \phi$ and is defined by induction on the formation of ϕ:

$$\begin{array}{l}
M, s \models A \text{ iff } A \in \Pi(s) \\
M, s \models \top \quad \textit{always} \\
M, s \models \bot \quad \textit{never} \\
M, s \models \phi_1 \wedge \phi_2 \text{ iff } M, s \models \phi_1 \text{ and } M, s \models \phi_2 \\
M, s \models \phi_1 \vee \phi_2 \text{ iff } M, s \models \phi_1 \text{ or } M, s \models \phi_2 \\
M, s \models \phi_1 \Rightarrow \phi_2 \text{ iff } M, s \models \phi_1 \text{ implies } M, s \models \phi_2 \\
M, s \models \neg\phi \text{ iff } M, s \not\models \phi \\
M, s \models < r > \phi \text{ iff } \exists s'.(s, s') \in \mathcal{R}_r \text{ and } M, s' \models \phi \\
M, s \models [r]\phi \text{ iff } \forall s'.(s, s') \in \mathcal{R}_r \text{ implies } M, s' \models \phi
\end{array}$$

where the family $\{\mathcal{R}_P\}$ is systematically extended so as to include, for every program r, the corresponding relation $\mathcal{R}_r$ defined by induction on the formation of r:

$$\begin{array}{l}
\mathcal{R}_P \subseteq \mathcal{S} \times \mathcal{S} \\
\mathcal{R}_{r_1 \cup r_2} = \mathcal{R}_{r_1} \cup \mathcal{R}_{r_2} \\
\mathcal{R}_{R_1;R_2} = \mathcal{R}_{r_1} \circ \mathcal{R}_{r_2} \quad \text{(seq. comp. of } \mathcal{R}_{r_1} \text{ and } \mathcal{R}_{r_2}\text{)} \\
\mathcal{R}_{r^*} = (\mathcal{R}_r)^* \quad \text{(refl. trans. closure of } \mathcal{R}_r\text{)} \\
\mathcal{R}_{r^-} = \{(s_1, s_2) \in \mathcal{S} \times \mathcal{S} \mid (s_2, s_1) \in \mathcal{R}_r\} \\
\mathcal{R}_{\phi?} = \{(s, s) \in \mathcal{S} \times \mathcal{S} \mid M, s \models \phi\}.
\end{array}$$

We often denote a structure $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ by $(\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$, where $\{\mathcal{R}_r\}$ includes a binary relation for every program (atomic or non-atomic).

A structure $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ is called a model of a formula ϕ if there exists a state $s \in \mathcal{S}$ such that $M, s \models \phi$. A formula ϕ is satisfiable if there exists a model of ϕ, otherwise the formula is unsatisfiable. A formula ϕ is valid in structure M, if for all $s \in \mathcal{S}$, $M, s \models \phi$. We call *axioms*, formulae that are assumed to be valid. Formally, a structure M is a model of an axiom ϕ, if ϕ is valid in M. An axiom is satisfiable, if it has a model. A structure M is a model of a finite set of axioms Γ, if M is a model of all axioms in Γ. A finite set of axioms is satisfiable if it has a model. We say that a finite set Γ of axioms logically implies a formula ϕ, written $\Gamma \models \phi$, if ϕ is valid in every model of Γ.

Observe that satisfiability of a formula ϕ as well as satisfiability of a finite set of axioms Γ can be reformulated by means of logical implication, as $\emptyset \not\models \neg\phi$ and $\Gamma \not\models \bot$ respectively. In turn logical implication can be reformulated in terms of satisfiability, by making use of the following theorem (see [74]).

Theorem 1 *Let Γ be a finite set of $\mathcal{DI}$ axioms, and ϕ a $\mathcal{DI}$ formula. Then $\Gamma \models \phi$ if and only if the $\mathcal{DI}$ formula*

$$[(P_1 \cup \ldots \cup P_m \cup P_1^- \cup \ldots \cup P_m^-)^*]\Gamma' \wedge \neg\phi$$

is unsatisfiable, where $P_1, \ldots, P_m$ are all atomic programs occurring in $\Gamma \cup \{\phi\}$ and Γ' is the conjunction of all axioms in Γ.

An analogous result holds for most propositional dynamic logics.[5] Observe that such a result exploits the power of program constructs (union, reflexive transitive closure) and the "connected model property"[6] of propositional dynamic logics in order to represent axioms (valid formulae).

Theorem 1 (and its analogues) is one of the main reasons to exploit the correspondence between description logics and propositional dynamic logics.

2.3 The correspondence between DLs and PDLs

The correspondence between $\mathcal{CI}$ and $\mathcal{DI}$, first pointed out by Schild [108], is based on the similarity between the interpretation structures of the two logics: at the extensional level, individuals (members of $\Delta^{\mathcal{I}}$) in description logics correspond to states in propositional dynamic logics, whereas connections between two individuals correspond to state transitions. At the intensional level, classes correspond to propositions, and roles corresponds to programs. The correspondence is realized through a (one-to-one and onto) mapping δ from $\mathcal{CI}$ concepts to $\mathcal{DI}$ formulae, and from $\mathcal{CI}$ roles to $\mathcal{DI}$ programs. The mapping δ is defined inductively as follows (we assume $\sqcup, \Rightarrow$ to be expressed by means of $\sqcap, \neg$):

$$\begin{array}{ll}
\delta(A) = A & \delta(P) = P \\
\delta(C_1 \sqcap C_2) = \delta(C_1) \wedge \delta(C_2) & \delta(\neg C) = \neg\delta(C) \\
\delta(\exists R.C) =< \delta(R) > \delta(C) & \delta(\forall R.C) = [\delta(R)]\delta(C) \\
\delta(R_1 \sqcup R_2) = \delta(R_1) \cup \delta(R_2) & \delta(R_1 \circ R_2) = \delta(R_1); \delta(R_2) \\
\delta(R^*) = \delta(R)^* & \delta(id(C)) = \delta(C)? \\
\delta(R^-) = \delta(R)^-. &
\end{array}$$

The mapping δ can be extended to a mapping δ^+ from $\mathcal{CI}$ TBoxes to $\mathcal{DI}$ formulae. Namely, if $\mathcal{K} = \{k_1, \cdots, k_n\}$ is a TBox in $\mathcal{CI}$, and $P_1, \ldots, P_m$ are all atomic roles appearing in $\mathcal{K}$, then

$$\begin{array}{l}
\delta^+(\mathcal{K}) = [(P_1 \cup \cdots \cup P_m \cup P_1^- \cdots \cup P_m^-)^*]\ \delta^+(\{k_1\}) \wedge \cdots \wedge \delta^+(\{k_n\}), \\
\delta^+(\{C_1 \sqsubseteq C_2\}) = (\delta(C_1) \Rightarrow \delta(C_2)).
\end{array}$$

Making use of Theorem 1, we can state the following: if $\mathcal{K}$ is a TBox, then $\mathcal{K} \models C_1 \sqsubseteq C_2$ (where atomic concepts and roles in C_1, C_2 are also in $\mathcal{K}$) if and only if the $\mathcal{DI}$ formula

$$\delta^+(\mathcal{K}) \wedge \delta(C_1) \wedge \delta(\neg C_2)$$

[5] In the analogue of Theorem 1 for $\mathcal{D}$ the formula to check for unsatisfiability is $[(P_1 \cup \ldots \cup P_m)^*]\Gamma' \wedge \neg\phi$.

[6] That is, if a formula has a model, it has a model which is connected.

is unsatisfiable. Note that the size of the above formula is polynomial with respect to the size of $\mathcal{K}, C_1$ and C_2.

By virtue of δ and δ^+, respectively, both satisfiability of $\mathcal{CI}$ concepts, and logical implication for $\mathcal{CI}$ TBoxes, can be (polynomially) reduced to satisfiability of $\mathcal{DI}$ formulae. Since satisfiability for $\mathcal{DI}$ an EXPTIME-complete problem, so are satisfiability of $\mathcal{CI}$ concepts and logical implication for $\mathcal{CI}$ TBoxes. It is straightforward to extend the correspondence, and hence both δ and δ^+, to other description logics and propositional dynamic logics.

2.4 Other preliminary notions

In this section, we introduce several notions, propositions, and notations that will be used in the chapters which follow. We assume, without loss of generality, $\vee, \Rightarrow, [\cdot]$ to be expressed by means of $\neg, \wedge, < \cdot >$, and the converse operator to be applied to atomic programs only[7].

Fisher-Ladner closure

The *Fisher-Ladner closure* ([56]) of a $\mathcal{DI}$ formula Φ, denoted $CL(\Phi)$, is the least set F such that $\Phi \in F$ and such that:

$$\begin{array}{lcl}
\phi_1 \wedge \phi_2 \in F & \Rightarrow & \phi_1, \phi_2 \in F \\
\neg\phi \in F & \Rightarrow & \phi \in F \\
\phi \in F & \Rightarrow & \neg\phi \in F \text{ (if } \phi \text{ is not of the form } \neg\phi') \\
< r > \phi \in F & \Rightarrow & \phi \in F \\
< r_1; r_2 > \phi \in F & \Rightarrow & < r_1 >< r_2 > \phi \in F \\
< r_1 \cup r_2 > \phi \in F & \Rightarrow & < r_1 > \phi, < r_2 > \phi \in F \\
< r^* > \phi \in F & \Rightarrow & < r >< r^* > \phi \in F \\
< \phi'? > \phi \in F & \Rightarrow & \phi' \in F.
\end{array}$$

The notion of Fisher-Ladner closure of a formula is closely related to the notion of set of subformulae in simpler modal logics: intuitively, given a formula Φ, $CL(\Phi)$ includes all the formulae that play some role in establishing the truth-value of Φ. Both the number and the size of the formulae in $CL(\Phi)$ are linearly bounded by the size of Φ (see [56]). Note that, by definition, if $\phi \in CL(\Phi)$, then $CL(\phi) \subseteq CL(\Phi)$. We remark that the notion of Fisher-Ladner closure can be easily extended to formulae of other propositional dynamic logics.

Let us denote the *empty sequence of programs* by the program ε, and define $< \varepsilon > \phi \doteq \phi$ and $[\varepsilon]\phi \doteq \phi$. We call $Post(r)$ the set of programs defined by induction

[7]We recall that the following equations hold: $(r_1; r_2)^- = r_2^-; r_1^-$, $(r_1 \cup r_2)^- = r_1^- \cup r_2^-$, $(r_1^*)^- = (r_1^-)^*$, $(\phi?)^- = \phi?$.

on the structure of r as follows ($a = P \mid P^-$):

$$\begin{array}{lcl}
Post(a) & = & \{\varepsilon, a\} \\
Post(r_1; r_2) & = & \{r_1'; r_2 \mid r_1' \in Post(r_1)\} \cup Post(r_2) \\
Post(r_1 \cup r_2) & = & Post(r_1) \cup Post(r_2) \\
Post(r_1^*) & = & \{r_1'; r_1^* \mid r_1' \in Post(r_1)\} \\
Post(\phi?) & = & \{\varepsilon, \phi?\}.
\end{array}$$

Similarly, we call $Pre(r)$ the set of programs defined by induction on the structure of r as follows:

$$\begin{array}{lcl}
Pre(a) & = & \{\varepsilon, a\} \\
Pre(r_1; r_2) & = & \{r_1; r_2' \mid r_2' \in Pre(r_2)\} \cup Pre(r_1) \\
Pre(r_1 \cup r_2) & = & Pre(r_1) \cup Pre(r_2) \\
Pre(r_1^*) & = & \{r_1^*; r_1' \mid r_1' \in Pre(r_1)\} \\
Pre(\phi?) & = & \{\varepsilon, \phi?\}.
\end{array}$$

Roughly, $Post(r)$ is the set formed by those programs that are "postfix" of the program r, while $Pre(r)$ is the set formed by those programs that are "prefix" of r. The size of both $Post(r)$ and $Pre(r)$ is polynomial in the size of r. Moreover the programs in $Post(r)$ have the following two properties.

Proposition 2 *Let $< r > \phi$ be a formula. For all $r' \in Post(r)$, $< r' > \phi \in CL(< r > \phi)$.*

Proof By induction on r.

- $r = a$ or $r = \phi'?$. Then $Post(r) = \{\varepsilon, r\}$. By definition, both $\phi \in CL(< r > \phi)$ and $< r > \phi \in CL(< r > \phi)$.
- $r = r_1; r_2$. Then $Post(r_1; r_2) = \{r_1'; r_2 \mid r_1' \in Post(r_1)\} \cup Post(r_2)$.

  Since r_1 is a subprogram of $r_1; r_2$, by induction hypothesis, for all $r_1' \in Post(r_1)$:

  $$< r_1' > (< r_2 > \phi) \in CL(< r_1 >< r_2 > \phi) \subseteq CL(< r_1; r_2 > \phi).$$

  On the other hand, since r_2 is subprogram of $r_1; r_2$, by induction hypothesis, for all $r_2' \in Post(r_2)$:

  $$< r_2' > \phi \in CL(< r_2 > \phi) \subseteq CL(< r_1; r_2 > \phi).$$

- $r = r_1 \cup r_2$. Then $Post(r_1 \cup r_2) = Post(r_1) \cup Post(r_2)$. By induction hypothesis, for $i = 1, 2$, for all $r_i' \in Post(r_i)$:

  $$< r_i' > \phi \in CL(< r_i > \phi) \subseteq CL(< r_1 \cup r_2 > \phi).$$

- $r = r_1^*$. Then $Post(r_1^*) = \{r_1'; r_1^* \mid r_1' \in Post(r_1)\}$. By induction hypothesis, for all $r_1' \in Post(r_1)$:

  $$< r_1' > (< r_1^* > \phi) \in CL(< r_1 >< r_1^* > \phi) \subseteq CL(< r_1^* > \phi).$$

□

Proposition 3 *Let $< r_1 > \ldots < r_l > \phi$ be a formula. For all $r' \in Post(r_1;\ldots;r_l)$, there is a formula $\psi \in CL(< r_1 > \ldots < r_l > \phi)$ such that ψ is equivalent to $< r' > \phi$ (i.e. $\psi \equiv < r' > \phi$ is valid).*

Proof By induction on l. If $l = 1$, the thesis holds trivially, by Proposition 2. If $l > 1$ then $Post(r_1; r_2; \ldots; r_l) = \{r_1'; r_2; \ldots; r_l \mid r_1' \in Post(r_1)\} \cup Post(r_2; \ldots; r_l)$. By Proposition 2, for all $r_1' \in Post(r_1)$: $< r_1' > (< r_2 > \ldots < r_l >)\phi \in CL(< r_1 >< r_2 > \ldots < r_l > \phi)$. While, by induction hypothesis, for all $r' \in Post(r_2;\ldots;r_l)$: $< r' > \phi$ is equivalent to ψ, for some $\psi \in CL(< r_2 > \ldots < r_l > \phi) \subseteq CL(< r_1 >< r_2 > \ldots < r_l > \phi)$. □

Paths

Next we introduce the notion of *path*, which is similar to the notion of *trajectory* used in [7], and to that of *execution sequence* in [119].

A *path* in a structure M is a sequence $(s_0, \ldots, s_q)$ of states of M ($q \geq 0$), such that for each $i = 1, \ldots, q$, $(s_{i-1}, s_i) \in \mathcal{R}_a$ for some $a = P \mid P^-$. The length of $(s_0, \ldots, s_q)$ is q. Intuitively a path describes the sequence of states a given run of a program goes through.

We inductively define the set of paths $Paths_M(r)$ of a program r in a structure M, as follows[8]:

$$\begin{array}{lcl}
Paths_M(a) & = & \mathcal{R}_a \ (a = P \mid P^-) \\
Paths_M(r_1 \cup r_2) & = & Paths_M(r_1) \cup Paths_M(r_2) \\
Paths_M(r_1; r_2) & = & \{(s_0, \ldots, s_u, \ldots, s_q) \mid (s_0, \ldots, s_u) \in Paths_M(r_1) \\
 & & \ \text{and } (s_u, \ldots, s_q) \in Paths_M(r_2)\} \\
Paths_M(r^*) & = & \{(s) \mid s \in \mathcal{S}\} \cup (\bigcup_{i>0} Paths_M(r^i)) \\
Paths_M(\phi'?) & = & \{(s) \mid M, s \models \phi'\}.
\end{array}$$

We say that a path (s_0) in M *satisfies* a formula ϕ which is not of the form $< r > \phi'$ if $M, s_0 \models \phi$. We say that a path $(s_0, \ldots, s_q)$ in M *satisfies* a formula ϕ of the form $< r_1 > \cdots < r_l > \phi'$, where ϕ' is not of the form $< r' > \phi''$, if $(s_0, \ldots s_q) \in Paths_M(r_1; \cdots; r_l)$ and $M, s_q \models \phi'$.

The following two propositions describe the basic properties of paths: they concern paths of length 0 and paths of length greater then 0 respectively.

Proposition 4 *Let M be a structure and $< r > \phi$ a formula such that: $M, s \models < r > \phi$, $(s) \in Paths_M(r)$, and $M, s \models \phi$. Then there exists a formula $< \phi_1?; \ldots; \phi_g? > \phi$, with $g \geq 0$, such that:*

- *all tests $\phi_i?$ occur in r;*
- *$M, s \models < \phi_1?; \ldots; \phi_g? > \phi$;*

[8]The notation r^i stands for i repetitions of r – i.e., $r^1 = r$, and $r^i = r; r^{i-1}$.

- *$< \phi_1?;\ldots;\phi_g? > \phi \Rightarrow < r > \phi$ is valid.*

Proof By induction on r.

1) $r = \phi'?$.
The thesis holds trivially.

2) $r = r_1; r_2$.
$M, s \models < r_1; r_2 > \phi$ and $(s) \in Paths_M(r)$ implies that $M, s \models < r_1 >< r_2 > \phi$ and $(s) \in Paths_M(r_1)$ and $(s) \in Paths_M(r_2)$.
By induction hypothesis, we can assume that:

- there is a formula $< \phi_{1,1}?;\ldots;\phi_{1,g_1}? >< r_2 > \phi$ such that all tests $\phi_{1,j}?$ occur in r_1, $M, s \models < \phi_{1,1}?;\ldots;\phi_{1,g_1}? >< r_2 > \phi$, and $< \phi_{1,1}?;\ldots;\phi_{1,g_1}? >< r_2 > \phi \Rightarrow < r_1 >< r_2 > \phi$ is valid;
- there is a formula $< \phi_{2,1}?;\ldots;\phi_{2,g_2}? > \phi$ such that all tests $\phi_{2,j}?$ occur in r_2, $M, s \models < \phi_{2,1}?;\ldots;\phi_{2,g_1}? > \phi$, and $< \phi_{2,1}?;\ldots;\phi_{2,g_2}? > \phi \Rightarrow < r_2 > \phi$ is valid.

Hence, we can conclude that the formula

$$< \phi_{1,1}?;\ldots;\phi_{1,g_1}?;\phi_{2,1}?;\ldots;\phi_{2,g_2}? > \phi$$

is such that: (1) all tests $\phi_{i,j}?$ occur in r_1 or r_2 and therefore in r; (2) $M, s \models < \phi_{1,1}?;\ldots;\phi_{1,g_1}?;\phi_{2,1}?;\ldots;\phi_{2,g_2}? > \phi$; (3) $< \phi_{1,1}?;\ldots;\phi_{1,g_1}?;\phi_{2,1}?;\ldots;\phi_{2,g_2}? > \phi \Rightarrow < r_1; r_2 > \phi$ is valid, as can be easily verified, considering that $< \phi_{1,1}?;\ldots;\phi_{1,g_1}? >< r_2 > \phi \Rightarrow < r_1 >< r_2 > \phi$ is valid, and any formula of the form $< \psi_1?;\ldots;\psi_g? > \psi$ is equivalent to $\psi_1 \wedge \ldots \wedge \psi_g \wedge \psi$.

3) $r = r_1 \cup r_2$.
$M, s \models < r_1 \cup r_2 > \phi$ implies that, either for $i = 1$ or for $i = 2$, $M, s \models < r_i > \phi$ and $(s) \in Paths_M(r_i)$. By induction hypothesis we can assume that there is a formula $< \phi_{i,1}?;\ldots;\phi_{i,g_i}? > \phi$ such that all tests $\phi_{i,j}?$ occur in r_i, $M, s \models < \phi_{i,1}?;\ldots;\phi_{i,g_i}? > \phi$, and $< \phi_{i,1}?;\ldots;\phi_{i,g_i}? > \phi \Rightarrow < r_i > \phi$ is valid. Therefore, considering that $< r_i > \phi \Rightarrow < r_1 \cup r_2 > \phi$, we get the thesis.

4) $r = r_1^*$.
Since $(s) \in Paths_M(r_1^*)$, $< r_1^* > \phi$ is equivalent $\phi \vee < r_1 >< r_1^* > \phi$, and $M, s \models \phi$, the thesis holds trivially (with $g = 0$). $\Box$

Proposition 5 *Let M be a structure, and $< r > \phi$ a formula such that: $M, s \models < r > \phi$, $(s = s_0, \ldots, s_q) \in Paths_M(r)$ with $q > 0$, $M, s_q \models \phi$. Then there exists a formula $< \phi_1?;\ldots;\phi_g?; a >< r' > \phi$, with $g \geq 0$, such that:*

- *all tests $\phi_i?$ occur in r;*
- *$r' \in Post(r)$ (and hence $< r' > \phi$ is equivalent to ψ – i.e. $< r' > \phi \equiv \psi$ is valid – for some $\psi \in CL(< r > \phi)$);*
- *$(s_0, s_1) \in \mathcal{R}_a$;*

- $M, s_1 \models < r' > \phi$;
- $(s_1, \ldots, s_q) \in Paths_M(r')$;
- $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi \Rightarrow < r > \phi$ *is valid.*

Proof By induction on r.

1) $r = a$.
The thesis holds trivially.

2) $r = r_1; r_2$.
Let $(s_0, \ldots, s_i)$ be the segment of $(s_0, \ldots, s_q)$ such that $(s_0, \ldots, s_i) \in Paths_M(r_1)$ and $(s_i, \ldots, s_q) \in Paths_M(r_2)$. We consider two cases:

- $i > 0$. Consider that: (1) $M, s_0 \models < r_1 > \phi'$ for $\phi' = < r_2 > \phi$; (2) $(s_0, \ldots, s_i) \in Paths_M(r_1)$ with $i > 0$; (3) $M, s_i \models < r_2 > \phi$. By induction hypothesis, there is a formula $< \phi_1?; \ldots; \phi_g?; a >< r'_1 >< r_2 > \phi$ such that:
  - all tests $\phi_i?$ occur in r_1, and hence in r;
  - $r'_1 \in Post(r_1)$, and hence $r'_1; r_2 \in Post(r_1; r_2)$;
  - $(s_0, s_1) \in \mathcal{R}_a$;
  - $M, s_1 \models < r'_1 >< r_2 > \phi$, and hence $M, s_1 \models < r'_1; r_2 > \phi$;
  - $(s_1, \ldots, s_i) \in Paths_M(r'_1)$ with $i \leq q$, and hence $(s_1, \ldots, s_q) \in Paths_M(< r'_1; r_2 > \phi)$;
  - $< \phi_1?; \ldots; \phi_g?; a >< r'_1 >< r_2 > \phi \Rightarrow < r_1 >< r_2 > \phi$ is valid, and hence $< \phi_1?; \ldots; \phi_g?; a >< r'_1; r_2 > \phi \Rightarrow < r_1; r_2 > \phi$ is valid.
- $i = 0$. By Proposition 4, there exists a formula $< \phi_{1,1}?; \ldots; \phi_{1,g_1}? >< r_2 > \phi$ such that
  - all tests $\phi_{1,j}?$ occur in r_1;
  - $M, s_0 \models < \phi_{1,1}?; \ldots; \phi_{1,g_1}? >< r_2 > \phi$;
  - $< \phi_{1,1}?; \ldots; \phi_{1,g_1}? >< r_2 > \phi \Rightarrow < r_1 >< r_2 > \phi$ is valid.

  On the other hand, observe that $< r_2 > \phi$ is such that: (1) $M, s \models < r_2 > \phi$; (2) $(s = s_0, \ldots, s_q) \in Paths_M(r_2)$ with $q > 0$; (3) $M, s_q \models \phi$. Therefore, by induction hypothesis, there is a formula $< \phi_{2,1}?; \ldots; \phi_{2,g_2}?; a >< r'_2 > \phi$ such that
  - all tests $\phi_{2,j}?$ occur in r_2;
  - $r'_2 \in Post(r_2)$ $(\subseteq Post(r_1; r_2))$;
  - $(s_0, s_1) \in \mathcal{R}_a$;
  - $M, s_1 \models < r'_2 > \phi$;
  - $(s_1, \ldots, s_q) \in Paths_M(r'_2)$;
  - $< \phi_{2,1}?; \ldots; \phi_{2,g_2}?; a >< r'_2 > \phi \Rightarrow < r_2 > \phi$ is valid.

Hence the formula $< \phi_{1,1}?; \ldots; \phi_{1,g_1}?; \phi_{2,1}?; \ldots; \phi_{2,g_2}?; a >< r'_2 > \phi$ is such that

- all tests $\phi_{i,j}?$ occur in either in r_1 or in r_2;
- $r'_2 \in Post(r_1; r_2)$;
- $(s_0, s_1) \in \mathcal{R}_a$;
- $M, s_1 \models < r'_2 > \phi$;
- $(s_1, \ldots, s_q) \in Paths_M(r'_2)$;
- $< \phi_{1,1}?; \ldots; \phi_{1,g_1}? >< \phi_{2,1}?; \ldots; \phi_{2,g_2}?; a >< r'_2 > \phi \Rightarrow < r_1 >< r_2 > \phi$ is valid, and hence $< \phi_{1,1}?; \ldots; \phi_{1,g_1}?; \phi_{2,1}?; \ldots; \phi_{2,g_2}?; a >< r'_2 > \phi \Rightarrow < r_1; r_2 > \phi$ is valid (recall that any formula of the form $< \psi_1?; \ldots; \psi_g? > \psi$ is equivalent to $\psi_1 \wedge \ldots \wedge \psi_g \wedge \psi$).

3) $r = r_1 \cup r_2$.
$M, s \models < r_1 \cup r_2 > \phi$ with $(s = s_0, \ldots, s_q) \in Paths_M(r_1 \cup r_2)$ implies that either for $i = 1$ or $i = 2$: (1) $M, s \models < r_i > \phi$; (2) $(s = s_0, \ldots, s_q) \in Paths_M(r_i)$ with $q > 0$; (3) $M, s_q \models \phi$. Thus, by induction hypothesis, there is a formula $< \phi_{i,1}?; \ldots; \phi_{i,g_1}?; a_i >< r'_i > \phi$ such that:

- all tests $\phi_{i,j}?$ occur in r_i, and hence in $r_1 \cup r_2$;
- $r'_i \in Post(r_i) \subseteq Post(r_1 \cup r_2)$;
- $(s_0, s_1) \in \mathcal{R}_a$;
- $M, s_1 \models < r'_i > \phi$;
- $(s_1, \ldots, s_q) \in Paths_M(r'_i)$;
- $< \phi_{i,1}?; \ldots; \phi_{i,g_i}?; a_i >< r'_i > \phi \Rightarrow < r_i > \phi$ is valid, and therefore, considering that, $< r_i > \phi \Rightarrow < r_1 \cup r_2 > \phi$ is valid, we get that $< \phi_{i,1}?; \ldots; \phi_{i,g_i}?; a_i >< r'_i > \phi \Rightarrow < r_1 \cup r_2 > \phi$ is valid.

4) $r = r_1^*$.
Since $q > 0$, we have that $M, s \models < r_1^* > \phi$ implies $M, s \models < r_1 >< r_1^* > \phi$, and furthermore there is a segment $(s_0, \ldots, s_i)$ of $(s_0, \ldots, s_q)$ with $0 < i \leq q$, such that $(s_0, \ldots, s_i) \in Paths_M(r_1)$ and $(s_i, \ldots, s_q) \in Paths_M(r_1^*)$. Thus we have: (1) $M, s_0 \models < r_1 > \phi'$ with $\phi' = < r_1^* > \phi$; (2) $(s_0, \ldots, s_i) \in Paths_M(r_1)$ with $i > 0$; (3) $M, s_i \models < r_1^* > \phi$. By induction hypothesis there exists a formula $< \phi_1?; \ldots; \phi_g?; a >< r'_1 >< r_1^* > \phi$ such that

- all tests $\phi_i?$ occur in r_1, and hence in r_1^*;
- $r'_1 \in Post(r_1)$, and hence $r'_1; r_1^* \in Post(r_1^*)$;
- $(s_0, s_1) \in \mathcal{R}_a$;
- $M, s_1 \models < r'_1 >< r_1^* > \phi$, and hence $M, s_1 \models < r'_1; r_1^* > \phi$;

- $(s_1, \ldots, s_i) \in Paths_M(r_1')$, and hence $(s_1, \ldots, s_q) \in Paths_M(r_1'; r_1^*)$;
- $< \phi_1?; \ldots; \phi_g?; a >< r_1' >< r_1^* > \phi \Rightarrow < r_1 >< r_1^* > \phi$ is valid, hence $< \phi_1?; \ldots; \phi_g?; a >< r_1'; r_1^* > \phi \Rightarrow < r_1; r_1^* > \phi$ is valid. Therefore, considering that $< r_1; r_1^* > \phi \Rightarrow < r_1^* > \phi$, we get that $< \phi_1?; \ldots; \phi_g?; a >< r_1'; r_1^* > \phi \Rightarrow < r_1^* > \phi$ is valid.

□

Finally, if a denotes the atomic program P (resp. the inverse of an atomic program P^-), then we write a^- to denote P^- (resp. P).

Chapter 3

Functional Restrictions

In this chapter we study the description logic $\mathcal{CIF}$ and the propositional dynamic logic $\mathcal{DIF}$ obtained from $\mathcal{CI}$ and $\mathcal{DI}$ by adding the functional restriction construct $(\leq 1\ a)$. The functional restriction $(\leq 1\ a)$ imposes that the role a, where a is either an atomic role (program) or the inverse of an atomic role (program), is functional wrt individuals (states) in which $(\leq 1\ a)$ holds.

3.1 The logics $\mathcal{CIF}$ and $\mathcal{DIF}$

Concepts of $\mathcal{CIF}$ are formed according to the following abstract syntax:

$$\begin{array}{lcl} C & ::= & \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid C_1 \Rightarrow C_2 \mid \neg C \mid \\ & & \exists R.C \mid \forall R.C \mid (\leq 1\ a) \\ \\ a & ::= & P \mid P^- \\ R & ::= & a \mid R_1 \sqcup R_2 \mid R_1 \circ R_2 \mid R^* \mid R^- \mid id(C) \end{array}$$

where A denotes an atomic concept, C (possibly with a subscript) a generic concept, P an atomic role, a a *simple role*, i.e. either an atomic role or the inverse of an atomic role, R (possibly with a subscript) a generic role.

The semantics of $\mathcal{CIF}$ is the same as for $\mathcal{CI}$, except for functional restrictions $(\leq 1\ a)$ whose meaning in an interpretation $\mathcal{I}$ is the following (recall $a = P \mid P^-$):

$$(\leq 1\ a)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \text{ there exists } \textit{at most one } d' \text{ such that } (d, d') \in a^{\mathcal{I}}\}.$$

Note that in $\mathcal{CIF}$ there is a complete symmetry between atomic roles and inverse of atomic roles. This symmetry is often needed in representing complex domains. Furthermore it makes $\mathcal{CIF}$ suitable to explore extensions of the logic based on the reification of the relations as shown later.

The corresponding propositional dynamic logic is called $\mathcal{DIF}$ and its syntax is as follows:

$$\begin{array}{lcl}\phi & ::= & \top \mid \bot \mid A \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \Rightarrow \phi_2 \mid \neg\phi \mid \\ & & <r> \phi \mid [r]\phi \mid (\leq 1\ a) \\ \\ a & ::= & P \mid P^- \\ r & ::= & a \mid r_1 \cup r_2 \mid r_1 ; r_2 \mid r^* \mid r^- \mid \phi? \end{array}$$

where A denotes a propositional letter, ϕ (possibly with a subscript) a formula, P an atomic program, a a *simple program*, i.e. an atomic program or the inverse of an atomic program, and r (possibly with a subscript) a generic program.

Consistently with its interpretation in $\mathcal{CIF}$, the new construct is interpreted as follows: given a structure $M = (\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$ and a state $s \in \mathcal{S}$,

$$M, s \models (\leq 1\ a) \text{ iff there exists } \textit{at most one } t \text{ such that } (s,t) \in \mathcal{R}_a.$$

The rest of the constructs are interpreted as in $\mathcal{DI}$.

Observe that the functional restriction $(\leq 1\ a)$ allows the notion of local determinism for both atomic programs and the converse of atomic programs to be represented in the logic. With this construct, we can denote states in which the running of an atomic program, or the converse of an atomic program, is deterministic, i.e. it leads to at most one state. It is easy to see that this possibility allows one to impose the so-called global determinism too, i.e. that all runs of a given atomic program or the converse of an atomic program, are deterministic. Therefore, $\mathcal{DIF}$ subsumes the logic studied in [131], called Converse Deterministic PDL, where atomic programs, *not their converse*, are (globally) deterministic.

3.2 Reasoning in $\mathcal{CIF}$ and $\mathcal{DIF}$

The decidability and complexity of both satisfiability of $\mathcal{CIF}$ concepts and logical implication in $\mathcal{CIF}$ TBoxes can be derived immediately by exploiting the correspondence between $\mathcal{CIF}$ and $\mathcal{DIF}$. This is realized through the mappings δ and δ^+ described in Chapter 2, suitably extended in order to deal with functional restrictions (given the semantics of these constructs in $\mathcal{CIF}$ and $\mathcal{DIF}$, the extension is trivial). Note however that the decidability and the complexity of satisfiability in $\mathcal{DIF}$ have yet to be established. We establish them below by exhibiting an encoding of $\mathcal{DIF}$-formulae in $\mathcal{DI}$. More precisely we show that, for any $\mathcal{DIF}$-formula Φ, there is a $\mathcal{DI}$-formula, denoted $\gamma(\Phi)$, whose size is polynomial with respect to the size of Φ, and such that Φ is satisfiable if and only if $\gamma(\Phi)$ is satisfiable. Since satisfiability in $\mathcal{DI}$ is EXPTIME-complete, this ensures that satisfiability in $\mathcal{DIF}$ is EXPTIME-complete too.

In what follows, we assume, without loss of generality, that Φ is in negation normal form (i.e. negations are pushed inside as much as possible). It is easy to check that the transformation of any PDL formula in negation normal form can be performed in linear time in the size of the formula.

Definition Let Φ be a $\mathcal{DIF}$ formula in negation normal form. We define the *$\mathcal{DI}$-counterpart* $\gamma(\Phi)$ of Φ as the conjunction of two formulae, $\gamma(\Phi) = \gamma_1(\Phi) \wedge \gamma_2(\Phi)$, where:

- $\gamma_1(\Phi)$ is obtained from the original formula Φ by replacing each $(\leq 1\ a)$ with a new propositional letter $A_{(\leq 1a)}$, and each $\neg(\leq 1\ a)$ with $(< a > H_{(\leq 1a)}) \wedge (< a > \neg H_{(\leq 1a)})$, where $H_{(\leq 1a)}$ is again a new propositional letter.
- $\gamma_2(\Phi) = [(P_1 \cup \ldots \cup P_m \cup P_1^- \cup \ldots \cup P_m^-)^*]\gamma_2^1 \wedge \ldots \wedge \gamma_2^g$, where $P_1, \ldots, P_m$ are all atomic programs appearing in Φ, and with one conjunct γ_2^i of the form
$$(A_{(\leq 1a)} \wedge < a > \phi) \Rightarrow [a]\phi$$
for every $A_{(\leq 1a)}$ occurring in $\gamma_1(\Phi)$ and every $\phi \in CL(\gamma_1(\Phi))$.

□

Lemma 6 *Let Φ be a $\mathcal{DIF}$ formula, and $\gamma(\Phi)$ its $\mathcal{DI}$-counterpart. Then $\gamma(\Phi)$ is a $\mathcal{DI}$ formula, and its size is polynomially related to the size of Φ.*

Proof Straightforward. □

The purpose of $\gamma_1(\Phi)$ is to introduce the new propositional letters $A_{(\leq 1a)}$ and $H_{(\leq 1a)}$ in place of $(\leq 1\ a)$. Positive occurrences of $(\leq 1\ a)$ are represented by the letter $A_{(\leq 1a)}$, while negative occurrences of $(\leq 1\ a)$ are represented by $< a > H_{(\leq 1a)} \wedge < a > \neg H_{(\leq 1a)}$. Note that every state where $< a > H_{(\leq 1a)} \wedge < a > \neg H_{(\leq 1a)}$ holds, has at least two a-successors.

The purpose of $\gamma_2(\Phi)$ is less obvious. Intuitively, it constrains the models M of $\gamma(\Phi)$ so that: for every state s of M, if $A_{(\leq 1a)}$ holds in s, and t_1 and t_2 are two a-successors of s, then t_1 and t_2 are equivalent wrt the formulae in $CL(\gamma_1(\Phi))$. We show that this allows us to actually "collapse" t_1 and t_2 into a single state.

Observe that if, instead of adding $\gamma_2(\Phi)$, we imposed the axiom schema
$$(A_{(\leq 1a)} \wedge < a > \phi) \Rightarrow [a]\phi$$
where ϕ is any formula, then the models of $\gamma_1(\Phi)$ would be models of the original formula as well. However, the problem of whether a $\mathcal{DI}$ formula ψ is deducible from an axiom schema is in general undecidable [74]. So, adding the above axiom schema to $\mathcal{DI}$ is of no use in establishing the decidability of $\mathcal{DIF}$.

Instead, the formula $\gamma_2(\Phi)$ can be thought of as a finite instantiation of the axiom schema $(A_{(\leq 1a)} \wedge < a > \phi) \Rightarrow [a]\phi$ (one instance for each formula in $CL(\gamma_1(\Phi))$).[1] Intuitively, imposing the validity of such finite instantiation guarantees that if $\gamma_1(\Phi)$ has a model then it has a model that is model of the original formula as well.

Next we introduce a function ES that given a state s of a model M identifies the states of M that can be thought of as "replicas" of s. We consider (without loss of generality) "connected" models only.[2]

Definition Let $M = (\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$ be a model of $\gamma(\Phi)$. For any state $s \in \mathcal{S}$, we denote by $ES(s)$ the smallest set $\mathcal{E}(s)$ of states in M such that

[1] Actually, $\gamma_2(\Phi)$ already takes into account the reduction from logical implication to satisfiability.

[2] A connected model of ϕ is a model $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ such that $\mathcal{S} = \{t \mid (s,t) \in (\cup_P(R_P \cup R_P^-)^*\}$ and $M, s \models \phi$.

- $s \in \mathcal{E}(s)$, and
- if $s' \in \mathcal{E}(s)$ and for some simple program a
  - $(s', t') \in \mathcal{R}_a$,
  - $M, t' \models A_{(\leq 1\, a^-)}$,
  - $t'' \in \mathcal{E}(t_1)$,
  - $(t'', s'') \in \mathcal{R}_a^-$,

  then $s'' \in \mathcal{E}(s)$.

□

Note that, as a consequence of $\gamma_2(\Phi)$, we have $M, t_2 \models A_{(\leq 1\, a^-)}$, since $A_{(\leq 1\, a^-)} \in CL(\gamma_1(\Phi))$ (all atomic prepositions occurring in $\gamma_1(\Phi)$ are in $CL(\gamma_1(\Phi))$.

An alternative way to see $ES(s)$ is the following:

$$ES(s) = \{s' \mid (s, s') \in \mathcal{R}_X\}$$

where X is program, which is not denoted by a regular expression, but by the following context-free grammar:

$$X ::= \varepsilon \mid (a; A_{(\leq 1\, a^-)}?; X; a^-); X \text{ (a any simple program).}$$

By definition, $ES(s)$ is obtained starting from s by including recursively the states s' which are linked by

$$\begin{array}{l}
\mathcal{R}_{(a_i; A_{(\leq 1 a_i^-)}?; a_i^-)} \text{ or} \\
\mathcal{R}_{(a_i; A_{(\leq 1 a_i^-)}?; (a_j; A_{(\leq 1 a_j^-)}?; a_j^-); a_i^-)} \text{ or} \\
\mathcal{R}_{(a_i; A_{(\leq 1 a_i^-)}?; (a_j; A_{(\leq 1 a_j^-)}?; (a_k; A_{(\leq 1 a_k^-)}?; a_k^-); a_j^-); a_i^-)} \text{ or} \\
\ldots
\end{array}$$

to states that are already known to be in $ES(s)$. By $\gamma_2(\Phi)$, the states s' satisfy the same formulae of $CL(\gamma_1(\Phi))$ as the state s. Hence all states in $ES(s)$ satisfy the same formulae, wrt those in $CL(\gamma_1(\Phi))$.

Observe, however, that $ES(s)$ does not contain all the states of M satisfying the same formulae of $CL(\gamma_1(\Phi))$ that s satisfies. The discussion above clarifies that the states in $ES(s)$ are only those in the connected component of the relation $\mathcal{R}_X$ containing s.

In the reminder of this section we shall prove that any $\mathcal{DIF}$-formula Φ is satisfiable if and only if its $\mathcal{DI}$-counterpart $\gamma(\Phi)$ is satisfiable. We start by showing that if $\gamma(\Phi)$ is satisfiable then Φ is satisfiable. We proceed as follows:

1. Given a model M of $\gamma(\Phi)$, we build a tree-like model M^t such that $M^t, \mathit{root} \models \gamma(\Phi)$ (*root* is the root of the tree-structure).

2. Then, by suitably modifying M^t, we construct a model M^f, such that all functional restriction requirements are satisfied – i.e., every state s in which $A_{(\leq 1\, a)}$ holds has at most one a-successor.

3. Finally, by eliminating the interpretation on the atomic propositions $A_{(\leq 1\, a)}$ and $H_{(\leq 1\, a)}$, we get a model $M^{\mathcal{F}}$ of Φ.

We construct M^t from M in two steps.

Step 1. Let $< a_1 > \psi_1, \ldots, < a_h > \psi_h$ be all the formulae of the form $< a > \phi'$, with a a simple program and ϕ' any formula, included in $CL(\Phi)$.[3] We consider an infinite h-ary tree $\mathcal{T}$ whose root is *root* and such that every node x has h children $child_i(x)$, one for each formula $< a_i > \psi_i$. We write $father(x)$ to denote the father of a node x in $\mathcal{T}$. We define two partial mappings m and l: m maps nodes of $\mathcal{T}$ to states of M, and l is used to label the arcs of $\mathcal{T}$ by either atomic programs, or the converse of atomic programs. For the definition of m and l, we proceed level by level. Let $s \in \mathcal{S}$ be any state such that $M, s \models \gamma(\Phi)$. We put $m(root) = s$, and, for all arcs $(root, child_i(root))$ corresponding to a formula $< a_i > \psi_i$ such that $M, s \models < a_i > \psi_i$, we put $l((root, child_i(root))) = a_i$. Suppose we have defined m and l up to level k, let x be a node at level $k+1$, and let $l((father(x), x)) = a_j$ – this implies $M, m(father(x)) \models < a_j > \psi_j$. We choose a *minimal* path (i.e. a path with minimal length) in M satisfying $< a_j > \psi_j$, say $(s_0, s_1, \ldots, s_q)$, such that $s_0 \in ES(father(x))$; we put $m(x) = s_1$ and for every $< a_i > \psi_i \in CL(\Phi)$ such that $M, t \models < a_i > \psi_i$ we put $l((x, child_i(x))) = a_i$.

Step 2. For each P, let $\mathcal{R}'_P = \{(x,y) \in \mathcal{T} \mid l((x,y)) = P \text{ or } l((y,x)) = P^-\}$. We define the structure $M^t = (\mathcal{S}^t, \{\mathcal{R}^t_P\}, \Pi^t)$ as follows:

$$\begin{array}{l} \mathcal{S}^t = \{x \in \mathcal{T} \mid (root, x) \in (\bigcup_P(\mathcal{R}'_P \cup \mathcal{R}'^{-}_P))^*\} \\ \mathcal{R}^t_P = \mathcal{R}'_P \cap (\mathcal{S}^t \times \mathcal{S}^t) \\ \Pi^t(x) = \Pi(m(x)) \quad \text{for all } x \in \mathcal{S}^t). \end{array}$$

Observe that the structure M^t is a generally infinite tree. However the set of states $\mathcal{S}^t$ is *countable.*

Next we construct the structure M^f, satisfying all the functional restrictions requirements.

Step 1 Let us enumerate level by level the states $x \in S^t$ such that $M^t, x \models A_{(\leq 1\, a)}$ for some simple program a.[4]

Step 2 From this enumeration, we define a sequence of structures $M^t = M^0, M^1, M^2, \ldots$.l, where each M^k is obtained from $M^{(k-1)}$ by considering the $k-th$ state $x^k \in \mathcal{S}^t$ such that $M^t, x^k \models A_{(\leq 1\, a)}$ for some simple program a and proceeding as follows:

- If $x^k \notin \mathcal{S}^{(k-1)}$, then $M^k = M^{(k-1)}$.

[3] Notice that the formulae ψ_i may be of the form $< r > \phi$, and that $\psi_i \in CL(\Phi)$.

[4] In this way states at level (depth) i are all enumerated before states at level $i+1$.

- If $x^k \in \mathcal{S}^{(k-1)}$, then for each simple program a such that $M^t, x^k \models A_{(\leq 1\, a)}$:
  - if $(x^k, father(x^k)) \notin \mathcal{R}_a^{(k-1)}$, we define
  $$\mathcal{R}_a^{(k-1)'} = \mathcal{R}_a^{(k-1)} - \{(x, child_i(x)) \in \mathcal{R}_a^{(k-1)} \text{ except one}\};$$
  - if $(x^k, father(x^k)) \in \mathcal{R}_a^{(k-1)}$, we define
  $$\mathcal{R}_a^{(k-1)'} = \mathcal{R}_a^{(k-1)} - \{(x, child_i(x)) \in \mathcal{R}_a^{(k-1)}\}.$$

We define $M^k = (\mathcal{S}^k, \{\mathcal{R}_P^k\}, \Pi^k)$ as follows:

$$\begin{array}{l}
\mathcal{S}^k = \{x \in \mathcal{S}^t \mid (root, x) \in (\bigcup_P (\mathcal{R}_P^{k\,\prime} \cup (\mathcal{R}_P^{k\,\prime})^-))^*\} \\
\mathcal{R}_P^k = \mathcal{R}_P^{k\,\prime} \cap (\mathcal{S}^k \times \mathcal{S}^k) \\
\Pi^k(x) = \Pi^t(x) \quad \text{for all } x \in \mathcal{S}^k.
\end{array}$$

Observe that M^k satisfies the functional restriction requirements for the first k states. Observe also that $root \in \mathcal{S}^k$ for all k, and that

$$\begin{array}{l}
\mathcal{S}^t = \mathcal{S}^0 \supseteq \mathcal{S}^1 \supseteq \mathcal{S}^2 \supseteq \ldots \\
\mathcal{R}_P^t = \mathcal{R}_P^0 \supseteq \mathcal{R}_P^1 \supseteq \mathcal{R}_P^2 \supseteq \ldots.
\end{array}$$

Step 3 We define $M^f = (\mathcal{S}^f, \{\mathcal{R}_P^f\}, \Pi^f)$ as follows:

$$\begin{array}{l}
\mathcal{S}^f = \bigcap_{k \geq 0} \mathcal{S}^k \\
\mathcal{R}_P^f = \bigcap_{k \geq 0} \mathcal{R}_P^k \\
\Pi^f(x) = \Pi^t(x) \quad \text{for all } x \in \mathcal{S}^f.
\end{array}$$

Intuitively, the model M^f is a (generally infinite) tree, obtained by "visiting" level by level M^t and eliminating, for each state x, all the states in $ES(x)$ except one (which must be connected to the root). Observe that, in general, M^f contains many states satisfying the same formulae, wrt those in $CL(\gamma_1(\Phi))$. Hence, M^f is not a filtration[5] of M by $CL(\gamma_1(\Phi))$. We will come back to this point at the end of the chapter.

Finally, we define $M^{\mathcal{F}} = (\mathcal{S}^{\mathcal{F}}, \{\mathcal{R}_P^{\mathcal{F}}\}, \Pi^{\mathcal{F}})$ as follows:

$$\begin{array}{l}
\mathcal{S}^{\mathcal{F}} = S^f \\
\mathcal{R}_P^{\mathcal{F}} = \mathcal{R}_P^f \\
\Pi^{\mathcal{F}}(x) = \Pi^f(x) - \{A_{(\leq 1 a)}, H_{(\leq 1 a)} \mid A_{(\leq 1 a)}, H_{(\leq 1 a)} \in \Pi^t(x)\} \;\; \text{for all } x \in \mathcal{S}^{\mathcal{F}}.
\end{array}$$

The following three lemmas state the basic properties of M^t, M^f, and $M^{\mathcal{F}}$.

Lemma 7 *Let M be a model of $\gamma(\Phi)$. Then, for every formula $\phi \in CL(\gamma(\Phi))$ and every $x \in \mathcal{S}^t$, $M^t, x \models \phi$ iff $M, m(x) \models \phi$.*

[5] See for example [60] for the definition of filtration in Modal Logic.

Proof We prove the lemma by induction on the formation of ϕ (called formula induction in the following). We assume, without loss of generality, $\vee, [\cdot]$ to be expressed by means of $\neg, \wedge, < \cdot >$, and that the converse operator is applied only to atomic programs.

- $\phi = A$.

  $M, m(x) \models A$ iff $A \in \Pi(m(x))$ iff (by construction of M^t) $A \in \Pi^t(x)$ iff $M^t, x \models A$.

- $\phi = \phi_1 \wedge \phi_2$.

  $M, m(x) \models \phi_1 \wedge \phi_2$ iff $M, m(x) \models \phi_1$ and $M, m(x) \models \phi_2$ iff (by formula induction hypothesis) $M^t, x \models \phi_1$ and $M^t, x \models \phi_2$ iff $M^t, x \models \phi_1 \wedge \phi_2$.

- $\phi = \neg\phi'$.

  $M, m(x) \models \neg\phi'$ iff $M, m(x) \not\models \phi'$ iff (by formula induction hypothesis) $M^t, x \not\models \phi'$ iff $M^t, x \models \neg\phi'$.

- $\phi =< r_1 > \ldots < r_l > \phi'$ with ϕ' not of the form $< r'' > \phi''$.

  Let r be $r_1; \ldots ; r_l$. We recall that $< r > \phi'$ is equivalent to $< r_1 > \ldots < r_l > \phi'$, and that, by Proposition 3, for all $r' \in Post(r)$, $< r' > \phi'$ is equivalent to some $\psi \in CL(< r_1 > \ldots < r_l > \phi')$.

  Let $M, m(x) \models < r > \phi'$, we prove that $M^t, x \models < r > \phi'$. We proceed by induction on the length of the path in M satisfying $< r > \phi$ (called path induction in the following)

  If $(m(x)) \in Paths_M(r)$ and $M, m(x) \models \phi'$, then, by Proposition 4, there exists a formula $< \phi_1?; \ldots ; \phi_g? > \phi'$, with $g \geq 0$, such that:

  - all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;
  - $M, m(x) \models < \phi_1?; \ldots ; \phi_g? > \phi'$;
  - $< \phi_1?; \ldots ; \phi_g? > \phi' \Rightarrow < r > \phi'$ is valid.

  By formula induction hypothesis, for every $\psi \in \{\phi_1, \ldots \phi_g, \phi'\}$, $M, m(x) \models \psi$ iff $M^t, x \models \psi$, and hence $M^t, x \models < r > \phi'$.

  Otherwise, let $(m(x) = s_0, s_1, \ldots , s_q)$ be a path in M satisfying $< r > \phi'$. By Proposition 5, there exists a formula $< \phi_1?; \ldots ; \phi_g?; a >< r' > \phi'$, with $g \geq 0$, such that:

  - all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;
  - $r' \in Post(r)$, and hence by Proposition 3, the formula $< r' > \phi'$ is equivalent to ψ for some $\psi \in CL(< r_1 > \ldots < r_l > \phi') \subseteq CL(\gamma(\Phi))$;
  - $(s_0, s_1) \in \mathcal{R}_a$;
  - $(s_1, \ldots , s_q) \in Paths_M(r')$;

– $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi' \Rightarrow < r > \phi'$ is valid.

By formula induction hypothesis, for every $\phi_i \in \{\phi_1, \ldots \phi_g\}$, $M, m(x) \models \phi_i$ iff $M^t, x \models \phi_i$.

By construction of M^t, there is a minimal path $(s'_0, s'_1, \ldots, s'_{q'})$ satisfying $< a >< r' > \phi'$ such that $s'_0 \in ES(m(x))$ and $s'_1 = m(child_i(x))$, which is shorter or of the same length as $(s_0, s_1, \ldots, s_q)$. Therefore, by path induction hypothesis, $M, m(child_i(x)) \models < r' > \phi'$ implies $M^t, child_i(x) \models < r' > \phi'$ and so $M^t, x \models < a >< r' > \phi'$. Hence we can conclude that $M^t, x \models < r > \phi'$.

Let $M^t, x \models < r > \phi'$, we prove $M, m(x) \models < r > \phi'$.

If $(x) \in Paths_{M^t}(r)$ and $M^t, x \models \phi$, then, by Proposition 4, there exists a formula $< \phi_1?; \ldots; \phi_g? > \phi'$, with $g \geq 0$, such that:

– all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;

– $M^t, x \models < \phi_1?; \ldots; \phi_g? > \phi'$;

– $< \phi_1?; \ldots; \phi_g? > \phi' \Rightarrow < r > \phi'$ is valid.

By formula induction hypothesis, for every $\psi \in \{\phi_1, \ldots \phi_g, \phi'\}$, $M^t, x \models \psi$ iff $M, m(x) \models \psi$, and hence $M, m(x) \models < r > \phi'$.

Otherwise, let $(x = x_0, x_1, \ldots, x_q)$ be a path in M^t satisfying $< r > \phi'$. By Proposition 5, there exists a formula $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi'$, with $g \geq 0$, such that:

– all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;

– $r' \in Post(r)$, and hence by Proposition 3, the formula $< r' > \phi'$ is equivalent to ψ for some $\psi \in CL(< r_1 > \ldots < r_l > \phi') \subseteq CL(\gamma(\Phi))$;

– $(x_0, x_1) \in \mathcal{R}_a$;

– $(x_1, \ldots, x_q) \in Paths_{M^t}(r')$;

– $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi' \Rightarrow < r > \phi'$ is valid.

By formula induction hypothesis, for every $\phi_i \in \{\phi_1, \ldots \phi_g\}$, $M^t, x \models \phi_i$ iff $M, m(x) \models \phi_i$.

By path induction hypothesis $M^t, x_1 \models < r' > \phi'$ implies $M, m(x_1) \models < r' > \phi'$.

If $x_1 = child_i(x)$ then, by construction of M^t, there is an $s \in ES(m(x))$ such that $(s, m(child_i(x))) \in \mathcal{R}_a$. Hence we have $M, s \models < a >< r' > \phi'$, and in turn $M, m(x) \models < a >< r' > \phi'$, since $< a >< r' > \phi' \in CL(< r > \phi') \subseteq CL(\gamma_1(\Phi))$, and $s \in ES(m(x))$.

If $x_1 = father(x)$ then, by construction of M^t, there is an $s \in ES(m(x_1))$ such that $(s, m(x)) \in \mathcal{R}_a^-$ and $M, s \models < r' > \phi'$, since $< r' > \phi' \in CL(< r > \phi') \subseteq CL(\gamma_1(\Phi))$ and $m(x_1) \in ES(s)$. Hence we have $M, m(x) \models < a >< r' > \phi'$.

Hence in both cases we can conclude that $M, m(x) \models < r > \phi'$.

$\square$

Observe that, by inspecting the proof above, it is easy to verify that for all x of M^t and for all $< r > \phi' \in CL(\gamma_1(\Phi))$, if $M^t, x \models < r > \phi'$ then there exists a path of the form $(x, child_{i_1}(x) = x_1, \ldots, child_{i_q}(\ldots child_{i_1}(x)\ldots) = x_q) \in Paths_{M^t}(r)$ with $q \geq 0$ such that $M^t, x_q \models \phi'$.

Lemma 8 *Let $M = (\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$ be a model of $\gamma_1(\Phi)$, and $M^t = (\mathcal{S}^t, \{R_r\}, \Pi)$ be the structure defined as above. Let $< r > \phi$ be a formula such that*

- *ϕ is not of the form $< r' > \phi'$;*
- *$< r > \phi$ is equivalent to some $\psi \in CL(\gamma_1(\phi))$.*

Then, for all $x \in \mathcal{S}^t$, if there is a path $(x = x_0, \ldots, x_q)$ on M^t satisfying $< r > \phi$ then there is a path $(x = x'_0, \ldots, x'_{q'})$ in M^t satisfying $< r > \phi$ which is shorter or of the same length as $(x = x_0, \ldots, x_q)$, and such that: for all $x'_i \in \{x'_2, \ldots, x'_{q'}\}$, for all simple programs a:

$$(x'_{i-2}, x'_{i-1}, x'_i) \in Paths_{M^t}(a; A_{(\leq 1\, a^-)}?; a^-) \textit{ implies } x'_{i-2} = x'_i.$$

Proof By induction on the length of the path $(x_0, \ldots, x_q)$.

If such a length is less than 3, then the thesis holds vacuously.

Let the length of $(x_0, \ldots, x_q)$ be greater or equal to 3.

By applying Proposition 5, we can conclude that, there exists a formula $< (\phi_0?; \ldots; \phi_g?); a; r' > \phi$, with $g \geq 0$, such that:

- all tests $\phi_i?$ occur in r;
- $r' \in Post(r)$, and hence by Proposition 2, the formula $< r' > \phi$ is equivalent to $\psi' \in CL(\gamma_1(\Phi))$;
- $(x_0, x_1) \in \mathcal{R}^t_a$;
- $(x_1, \ldots, x_q) \in Paths_{M^t}(r')$;
- $< (\phi_0?; \ldots; \phi_g?); a; r' > \phi \Rightarrow < r > \phi$ is valid.

1. If $(x_0, x_1, x_2) \notin Paths_{M^t}(a; A_{(\leq 1\, a^-)}?; a^-)$ or $x_0 = x_2$, then by induction hypothesis there exists a path $(x_1 = x'_1, \ldots, x'_{q'})$ satisfying $< r' > \phi$ such that: for all $x'_i \in \{x'_3, \ldots, x'_{q'}\}$, for all simple programs a

$$(x'_{i-2}, x'_{i-1}, x'_i) \in Paths_{M^t}(a; A_{(\leq 1\, a^-)}?; a^-) \text{ implies } x'_{i-2} = x'_i.$$

Hence the path $(x_0 = x'_0, x'_1, \ldots, x'_{q'})$ satisfies $< r > \phi$ and is such that: for all $x'_i \in \{x'_2, \ldots, x'_{q'}\}$, for all simple programs a

$$(x'_{i-2}, x'_{i-1}, x'_i) \in Paths_{M^t}(a; A_{(\leq 1\, a^-)}?; a^-) \text{ implies } x'_{i-2} = x'_i.$$

2. If $(x_0, x_1, x_2) \in Paths_{M^t}(a; A_{(\leq 1\, a^-)}?; a^-)$ and $x_0 \neq x_2$, then by Proposition 5, there exists a formula $< (\phi_0?; \ldots; \phi_g?); a^-; r'' > \phi$, with $g \geq 0$, such that:

   - all tests $\phi_i?$ occur in r;
   - $r'' \in Post(r')$, and hence by Proposition 2, the formula $< r'' > \phi$ is equivalent to $\psi' \in CL(\gamma_1(\Phi))$;
   - $(x_1, x_2) \in \mathcal{R}^t_{a^-}$;
   - $(x_2, \ldots, x_q) \in Paths_{M^t}(r'')$;
   - $< (\phi_0?; \ldots; \phi_g?); a^-; r'' > \phi \Rightarrow < r' > \phi$ is valid.

   Since $(x_0, x_2) \in \mathcal{R}^t_{a; A_{(\leq 1\, a^-)}?; a^-}$ and $< r'' > \phi$ is equivalent to $\psi' \in CL(\gamma_1(\Phi))$, we have that $M^t, x_0 \models < r'' > \phi$ iff $M^t, x_2 \models < r'' > \phi$.

   Moreover, by construction of M^t, we have that $m(x_2) \in ES(m(x_0))$, and this implies that there exists a path $(x_0 = x''_2, \ldots, x''_{q''})$ satisfying $< r'' > \phi$ which is shorter or of the same length as $(x_2, \ldots, x_q)$. Now consider the path $(x_0 = x''_0, x_1 = x''_1, x_0 = x''_2, \ldots, x''_{q''})$: it satisfies $< r > \phi$, it is shorter or of the same length as $(x_0, \ldots, x_q)$, $(x_0, x_1, x_2) \in Paths_{M^t}(a; A_{(\leq 1\, a^-)}?; a^-)$, and $x''_0 = x''_2$. Hence applying the reasoning at item 1. we get the thesis.

□

Lemma 9 *For every formula $\phi \in CL(\gamma_1(\Phi))$ and every $x \in \mathcal{S}'$, $M^t, x \models \phi$ iff $M^f, x \models \phi$.*

Proof Consider that M^f is the limit of the (infinite) sequence of models $M^t = M^0, M^1, \ldots$. We prove that for each $h \geq 0$, $M^t, x \models \phi$ iff $M^h, x \models \phi$, for all $x \in S^h$ and all $\phi \in CL(\gamma_1(\Phi))$. We proceed by induction on h (called state induction in the following).

• $h = 0$. Since $M^0 = M^t$, the thesis holds trivially.

• $h = k+1$. It suffices to prove that, for all $x \in \mathcal{S}^{k+1}$ and all $\phi \in CL(\gamma_1(\Phi))$, $M^k, x \models \phi$ iff $M^{k+1}, x \models \phi$. Indeed, by state induction hypothesis, for all $\phi \in CL(\gamma_1(\Phi))$ and all $x \in \mathcal{S}^k (\supseteq \mathcal{S}^{k+1})$, $M^k, x \models \phi$ iff $M^t, x \models \phi$.

We proceed by induction on the formation of ϕ (called formula induction in the following). We assume, without loss of generality, $\vee, [\cdot]$ to be expressed by means of $\neg, \wedge, < \cdot >$, and that the converse operator is applied only to atomic programs.

- $\phi = A$.

  $M^k, x \models A$ iff (by construction of M^{k+1}) $M^{k+1}, x \models A$.

- $\phi = \phi_1 \wedge \phi_2$.

  $M^k, x \models \phi_1 \wedge \phi_2$ iff $M^k, x \models \phi_1$ and $M^k, x \models \phi_2$ iff (by formula induction hypothesis) $M^{k+1}, x \models \phi_1$ and $M^{k+1}, x \models \phi_2$ iff $M^{k+1}, x \models \phi_1 \wedge \phi_2$.

- $\phi = \neg\phi'$.

  $M^k, x \models \neg\phi'$ iff $M^k, x \not\models \phi'$ iff (by formula induction hypothesis) $M^{k+1}, x \not\models \phi'$ iff $M^{k+1}, x \models \neg\phi'$.

- $\phi =< r > \phi'$.

  Let $M^{k+1}, x \models< r > \phi'$, we prove that $M^k, x \models< r > \phi'$.

  $M^{k+1}, x \models< r > \phi'$ iff for there is a path $(x = x_0, \ldots, x_q) \in Paths_{M^{k+1}}(r)$ such that $M^{k+1}, x_q \models \phi'$. By construction of M^{k+1}, $(x = x_0, \ldots, x_q) \in Paths_{M^k}(r)$, while, by formula induction hypothesis $M^k, x_q \models \phi'$. Hence, we have $M^k, x \models< r > \phi'$.

  Let $M^k, x \models< r > \phi'$, we prove that $M^{k+1}, x \models< r > \phi'$.

  If $(x) \in Paths_{M^k}(r)$ and $M^k, x \models \phi'$, then, by Proposition 4, there exists a formula $< \phi_1?; \ldots; \phi_g? > \phi'$, with $g \geq 0$, such that:

  – all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;

  – $M^k, x \models< \phi_1?; \ldots; \phi_g? > \phi'$, and hence $M^t, x \models< \phi_1?; \ldots; \phi_g? > \phi'$;

  – $< \phi_1?; \ldots; \phi_g? > \phi \Rightarrow< r > \phi'$ is valid.

  By formula induction hypothesis, for every $\psi \in \{\phi_1, \ldots \phi_g, \phi'\}$, $M^k, x \models \psi$ iff $M^{k+1}, x \models \psi$, Therefore $M^{k+1}, x \models< r > \phi'$.

  Otherwise, there is a path $(x = x_0, \ldots, x_q) \in Paths_{M^k}(r)$ such that $M^k, x_q \models \phi'$. By applying Proposition 5 q times and Proposition 4 once, we can conclude that there exists a formula

$$\begin{aligned} < & (\phi_{01}?; \ldots; \phi_{0g_0}?); a_1; \ldots; \\ & (\phi_{(q-1)1}?; \ldots; \phi_{(q-1)g_{(q-1)}}?); a_q; \\ & (\phi_{q1}?; \ldots; \phi_{qg_q}?) > \phi' \end{aligned}$$

  with $g_i \geq 0$, such that:

  – all tests $\phi_{ij}?$ occur in r, and hence all ϕ_{ij} are subformulae of $< r > \phi'$;

  – $(x_{i-1}, x_i) \in \mathcal{R}^k_{a_i}$, for $i = 1, \ldots, q$;

  – The following formula is valid:

$$\begin{aligned} < & (\phi_{01}?; \ldots; \phi_{0g_0}?); a_1; \ldots; \\ & (\phi_{(q-1)1}?; \ldots; \phi_{(q-1)g_{(q-1)}}?); a_q; \\ & (\phi_{q1}?; \ldots; \phi_{qg_q}?) > \phi' \Rightarrow< r > \phi'. \end{aligned}$$

  If, for $x_i = x_0, \ldots, x_q$, $x_i \in \mathcal{S}^{k+1}$, then, by construction of M^{k+1}, $(x_{i-1}, x_i) \in \mathcal{R}^k_{a_i}$ implies $(x_{i-1}, x_i) \in \mathcal{R}^{k+1}_{a_i}$. By formula induction hypothesis, for every $\psi \in \{\phi_{0g_0}, \ldots \phi_{qg_q}, \phi'\}$, $M^k, x_i \models \psi$ iff $M^{k+1}, x_i \models \psi$. Therefore we can conclude $M^{k+1}, x \models< r > \phi'$.

Otherwise, let x_{m+1}, with $(m \geq 0)$, be the first state in $(x_0, \ldots, x_q)$ such that $x_{m+1} \notin \mathcal{S}^{k+1}$ – i.e., x_m is the state of M^k whose successors have been modified in order to get M^{k+1}. By applying Proposition 5 m times only, we can conclude that, there exists a formula $< (\phi_{01}?; \ldots; \phi_{0g_0}?); a_1; \ldots; a_m; (\phi_{(m-1)1}?; \ldots; \phi_{mg_{(m-1)}}?); a_{m+1}; r' > \phi'$, with $g_i \geq 0$, such that:

- all tests $\phi_{ij}?$ occur in r, and hence all ϕ_{ij} are subformulae of $< r > \phi'$;
- $r' \in Post(r)$, and hence by Proposition 2, the formula $< r' > \phi'$ is equivalent to ψ for some $\psi \in CL(< r > \phi') \subseteq CL(\gamma_1(\Phi))$;
- $(x_{i-1}, x_i) \in \mathcal{R}^k_{a_i}$, for $i = 1, \ldots m+1$;
- $(x_{m+1}, \ldots, x_q) \in Paths_{M^k}(r')$;
- $< (\phi_{01}?; \ldots; \phi_{0g_0}?); a_1; \ldots; a_m; (\phi_{(m-1)1}?; \ldots; \phi_{mg_{(m-1)}}?); a_{m+1}; r' > \phi' \Rightarrow < r > \phi'$ is valid.

By definition of M^{k+1}, we have:

- $M^k, x_m \models A_{(\leq 1 a)}$, with $a = a_{m+1}$
- $x_{m-1} = father(x_m)$
- $x_{m+1} = child_i(x_m)$.

Therefore, one of the following two cases holds:

- $(x_m, father(x_m)) \notin \mathcal{R}^k_a$ – i.e., $a_m \neq a^-$. Then, M^{k+1} is obtained from M^k by removing all $(x_m, child_l(x_m))$ from $\mathcal{R}^k_a$, except one, say $child_j(x_m)$. By $\gamma_2(\Phi)$, for every $\psi \in CL(\gamma_1(\Phi))$ we have that $M^k, child_i(x_m) \models \psi$ iff $M^k, child_j(x_m) \models \psi$. Furthermore, by Lemma 8, we can conclude that there is a path $(child_j(x_m) = x'_1, \ldots, x'_{q'}) \in Paths_{M^k}(r')$, with $M^k, x'_{q'} \models \phi'$, such that $child_j(x_m)$ is the only child of x_m occurring in it. Hence, as shown above, we have that $M^{k+1}, x'_1 \models < r' > \phi'$. Therefore it is easy to see that $M^{k+1}, x \models < r > \phi'$.
- $(x_m, father(x_m)) \in \mathcal{R}^k_a$ – i.e., $a_m = a^-$. Then, M^{k+1} is obtained from M^k by removing all $(x_m, child_l(x_m))$ from $\mathcal{R}^k_a$. By $\gamma_2(\Phi)$, for every $\psi \in CL(\gamma_1(\Phi))$, we have that $M^k, father(x_m) \models \psi$ iff $M^k, child_i(x_m) \models \psi$. Furthermore, by Lemma 8, can conclude that there is a path $(father(x_m) = x'_1, \ldots, x'_{q'}) \in Paths_{M^k}(r)$, with $M^k, x'_{q'} \models \phi'$, that does not include any $child_l(x_m)$. Hence, as shown above, we have that $M^{k+1}, x'_1 \models < r' > \phi'$. Therefore it is easy to see that $M^{k+1}, x \models < r > \phi'$.

We have proved that, for each $h \geq 0$, $M^h, x \models \phi$ iff $M^t, x \models \phi$, for all $x \in S^h$ and all $\phi \in CL(\gamma_1(\Phi))$. By considering the definition of M^f it is now easy to conclude that $M^f, x \models \phi$ iff $M^t, x \models \phi$, for all $x \in S^h$ and all $\phi \in CL(\gamma_1(\Phi))$. □

Note that M^f is a model of $\gamma(\Phi)$, since, on the one hand, by Lemma 9, $M^f, root \models \gamma_1(\Phi)$, and on the other hand $M^f, root \models \gamma_2(\Phi)$, because whenever $M^f, x \models A_{(\leq 1 a)}$, there exists at most one x' such that $(x, x') \in \mathcal{R}^f_a$.

Lemma 10 *$M^f, root \models \gamma_1(\Phi)$ implies $M^{\mathcal{F}}, root \models \Phi$.*

Proof Observe that, if $M^f, x \models A_{(\leq 1 a)}$ then, by construction of M^f, there exists at most one x' such that $(x, x') \in \mathcal{R}^f_a$, implying that $M^{\mathcal{F}}, x \models (\leq 1\ a)$. On the other hand, if $M^f, x \models (< a > H_{(\leq 1 a)}) \wedge (< a > \neg H_{(\leq 1 a)})$, then there are at least two states x_1, x_2 such that $(x, x_1) \in \mathcal{R}^f_a$ and $(x, x_2) \in \mathcal{R}^f_a$, implying that $M^{\mathcal{F}}, x \models \neg(\leq 1\ a)$. The proof is easily completed by induction on the structure of Φ. $\Box$

By Lemma 7, Lemma 9, and Lemma 10, we can state the following result.

Theorem 11 *A $\mathcal{DIF}$-formula Φ is satisfiable only if its $\mathcal{DI}$-counterpart $\gamma(\Phi)$ is satisfiable.*

Next we turn to the converse of Theorem 11. We remark that transforming a model of Φ into a model of $\gamma(\Phi)$ is not always possible, since conflicts may arise in assigning the extensions of the atomic propositions $H_{(\leq 1\, a)}$. For example suppose that $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ is a model of a given $\mathcal{DIF}$-formula Φ such that:

$$\{s_1, s_2, s_3, t_1, t_2, t_3\} \subseteq \mathcal{S}$$
$$\{(s_1, t_1), (s_1, t_2), (s_1, t_3), (s_2, t_1), (s_2, t_3), (s_3, t_2), (s_3, t_3)\} \subseteq \mathcal{R}_P.$$

The states s_1, s_2, s_3, s_4 satisfy $\neg(\leq 1\, P)$. Nevertheless it is impossible to assign suitably the atomic proposition $H_{(\leq 1\, P)}$ to t_1, t_2, t_3.

One way to overcome this problem is to prove the tree model property for $\mathcal{DIF}$, i.e. that any model can be transformed into a tree-like model. Indeed for tree-like models the above conflicts cannot arise. We can construct tree-like models, following the construction of M^t shown above as a blueprint.

In fact, in proving the converse of Theorem 11, we exploit a weaker property.

Theorem 12 *A $\mathcal{DIF}$-formula Φ is satisfiable if its $\mathcal{DI}$-counterpart $\gamma(\Phi)$ is satisfiable.*

Proof Let $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ be a model of Φ. We assume, without loss of generality, that $H_{(\leq 1\, a)} \notin \Pi(s)$ and $A_{(\leq 1\, a)} \notin \Pi(s)$, for all $s \in \mathcal{S}$, and for all $(\leq 1\, a)$ occurring in Φ. We also assume the converse operator applied only to atomic programs in Φ.

Starting from M, we build a model $M' = (\mathcal{S}', \{\mathcal{R}'_P\}, \Pi')$ of $\gamma(\Phi)$ in two steps: first we transform M into M'', and then we transform M'' into M'.

Step 1 We transform M into M'' so as to assign the propositions $H_{(\leq 1\, a)}$ to states (or equivalently states to the propositions $H_{(\leq 1\, a)}$) in a suitable way. Let $(\leq 1\, a_1), \ldots, (\leq 1\, a_l)$ be all the functional restrictions occurring in Φ. M'' is obtained inductively from M, by applying the transformation below l times.

$l = 0$. No functional restrictions occur in Φ, hence $M'' = M$.

$l > 0$. Suppose that M has been transformed into $M_i = (\mathcal{S}_i, \{\mathcal{R}_{iP}\}, \Pi_i)$ by applying the transformation i times, so that states are suitably assigned to propositions $H_{(\leq 1\, a)}$, for $a = a_1, \ldots, a_i$. We show how to get M_{i+1} from M_i, and we prove that M_{i+1} is still a model of Φ.

Let $M_i^1 = (\mathcal{S}_i^1, \{\mathcal{R}_{iP}^1\}, \Pi_i^1)$ and $M_i^2 = (\mathcal{S}_i^2, \{\mathcal{R}_{iP}^2\}, \Pi_i^2)$ be two disjoint copies of M_i, i.e. $\mathcal{S}_i^1 \cap \mathcal{S}_i^2 = \emptyset$. Given a state $s \in S_i$, we denote by $s^1 \in S_i^1$ and $s^2 \in S_i^2$ the copies of s in M_i^1 and M_i^2 respectively.

Now, let $M_i^{1\uplus 2}$ be the disjoint union of M_i^1 and M_i^2, defined as:

$$\begin{array}{l}
\mathcal{S}_i^{1\uplus 2} = \mathcal{S}_i^1 \cup \mathcal{S}_i^2 \\
\mathcal{R}_{iP}^{1\uplus 2} = \mathcal{R}_{iP}^1 \cup \mathcal{R}_{iP}^2 \\
\Pi_i^{1\uplus 2}(s) = \left\{ \begin{array}{ll} \Pi_i^1(s) & \text{if } s \in \mathcal{S}_i^1 \\ \Pi_i^2(s) & \text{if } s \in \mathcal{S}_i^2 \end{array} \right.
\end{array}$$

Observe that $M_i^{1\uplus 2}$ is a model of Φ, since $M_i, s \models \Phi$ if and only if $M_i^{1\uplus 2}, s^1 \models \Phi$, if and only if $M_i^{1\uplus 2}, s^2 \models \Phi$.

From $M_i^{1\uplus 2}$ we get $M_{i+1} = (\mathcal{S}_{i+1}, \{\mathcal{R}_{i+1P}\}, \Pi_{i+1})$ by defining $\mathcal{S}_{i+1}$ and Π_{i+1} as:

$$\begin{array}{l}
\mathcal{S}_{i+1} = \mathcal{S}_i^{1\uplus 2} \\
\Pi_{i+1}(s) = \left\{ \begin{array}{ll} \Pi_i^1(s) \cup \{H_{(\leq 1\, a_{i+1})}\} & \text{if } s \in \mathcal{S}_i^1 \\ \Pi_i^2(s) & \text{if } s \in \mathcal{S}_i^2 \end{array} \right.
\end{array}$$

and by defining $\mathcal{R}_{i+1P}$ as follows:

- if $a_{i+1} \neq P$, then $\mathcal{R}_{i+1P} = \mathcal{R}_{iP}^{1\uplus 2}$;
- if $a_{i+1} = P$, then $\mathcal{R}_{i+1P}$ is obtained from $\mathcal{R}_{iP}^{1\uplus 2}$ as follows: for all $s \in S_i$ such that $M_i, s \models \neg(\leq 1\, P)$ we choose one of its P-successors, say t, and we replace (s^1, t^1) with (s^1, t^2) and (s^2, t^2) with (s^2, t^1) in $\mathcal{R}_{iP}^{1\uplus 2}$. Note that, for every simple program a, the number of a-successors of all states in M_{i+1}, and in particular of s^1, s^2, t^1, t^2, remains unchanged wrt $M_i^{1\uplus 2}$.
- if $a_{i+1} = P^-$ then $\mathcal{R}_{i+1P}$ is obtained from $\mathcal{R}_i^{1\uplus 2}{}_P$ as follows: for all $s \in S_i$ such that $M_i, s \models \neg(\leq 1\, P^-)$ we choose one of its P^--successors, say t, and we replace (t^1, s^1) with (t^2, s^1) and (t^2, s^2) with (t^1, s^2) in $\mathcal{R}_{iP}^{1\uplus 2}$. Note that, for every simple program a, the number of a-successors of all states in M_{i+1}, and in particular of s^1, s^2, t^1, t^2, remains unchanged wrt $M_i^{1\uplus 2}$.

By construction, $M_{i+1}, s \models \neg(\leq 1\, a_j)$ implies $M_{i+1}, s \models < a_j > H_{(\leq 1\, a_j)} \wedge < a_j > \neg H_{(\leq 1\, a_j)}$ for $a_j = a_1 \ldots a_{i+1}$.

Next we verify that M_{i+1} is a model of Φ. Specifically, we prove that, for any $s \in \mathcal{S}_i$, any $\phi \in CL(\Phi)$, and $h = 1, 2$, $M_i, s \models \phi$ if and only if $M_{i+1}, s^h \models \phi$. We proceed by induction on the formula ϕ (called formula induction in the following).

- $\phi = A$.

  $M_i, s \models A$ iff $A \in \Pi_i(s)$ iff $A \in \Pi_{i+1}(s^h)$ iff $M_{i+1}, s^h \models A$.
- $\phi = \phi_1 \wedge \phi_2$.

  $M_i, s \models \phi_1 \wedge \phi_2$ iff $M_i, s \models \phi_1$ and $M_i, s \models \phi_2$ iff (by formula induction hypothesis) $M_{i+1}, s^h \models \phi_1$ and $M_{i+1}, s^h \models \phi_2$ iff $M_{i+1}, s^h \models \phi_1 \wedge \phi_2$.

- $\phi = \neg\phi'$.

  $M_i, s \models \neg\phi'$ iff $M_i, s \not\models \phi'$ iff (by formula induction hypothesis) $M_{i+1}, s^h \not\models \phi'$ iff $M_{i+1}, s^h \models \neg\phi'$.

- $\phi = (\leq 1\, a)$.

  $M_i, s \models (\leq 1\, a)$ iff $M_{i+1}, s^h \models (\leq 1\, a)$ by construction of M_{i+1}.

- $\phi =< r > \phi'$.

  Let $M_i, s \models< r > \phi'$, then there exists a path $(s = s_0, \ldots, s_q) \in Paths_{M_i}(r)$, with $q \geq 0$, such that $M_i, s_q \models \phi'$. We prove $M_{i+1}, s^h \models< r > \phi'$ by induction on the length q of the path (called path induction).

  If $q = 0$, then $(s) \in Paths_{M_i}(r)$, and, by Proposition 4, there exists a formula $< \phi_1?; \ldots; \phi_g? > \phi'$, with $g \geq 0$, such that:

  - all tests $\phi_j?$ occur in r, and hence all ϕ_j are subformulae of $< r > \phi'$;
  - $M_i, s \models< \phi_1?; \ldots; \phi_g? > \phi'$;
  - $< \phi_1?; \ldots; \phi_g? > \phi' \Rightarrow< r > \phi'$ is valid.

  By formula induction hypothesis, for every $\psi \in \{\phi_1, \ldots \phi_g, \phi'\}$, $M_i, s \models \psi$ iff $M_{i+1}, s^h \models \psi$, and hence $M_{i+1}, s^h \models< r > \phi'$.

  If, $q > 0$, then, by Proposition 5, there exists a formula $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi'$, with $g \geq 0$, such that:

  - all tests $\phi_j?$ occur in r, and hence all ϕ_j are subformulae of $< r > \phi'$;
  - $r' \in Post(r)$, and hence by Proposition 3, the formula $< r' > \phi'$ is equivalent to ψ for some $\psi \in CL(< r > \phi') \subseteq CL(\Phi)$;
  - $(s_0, s_1) \in \mathcal{R}_{i\,a}$;
  - $(s_1, \ldots, s_q) \in Paths_{M_i}(r')$;
  - $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi' \Rightarrow< r > \phi'$ is valid.

  By formula induction hypothesis, for every $\phi_x \in \{\phi_1, \ldots \phi_g\}$, $M_i, s \models \phi_x$ iff $M_{i+1}, s^h \models \phi_x$.

  By path induction hypothesis, $M_i, s_1 \models< r' > \phi'$ implies $M_{i+1}, s_1^1 \models< r' > \phi'$ and $M_{i+1}, s_1^2 \models< r' > \phi'$, since $(s_1, \ldots, s_q) \in Paths_{M_i}(r')$ is shorter than $(s_0, \ldots, s_q)$.

  While, by definition, $(s_0, s_1) \in \mathcal{R}_{i\,a}$ implies that:

  - if $a \neq a_{i+1}$, then $(s_0^h, s_1^h) \in \mathcal{R}_{i+1\,a}$;
  - if $a = a_{i+1}$, then $(s_0^h, s_1^k) \in \mathcal{R}_{i+1\,a}$, with $k = 2$ if $h = 1$ and $k = 1$ if $h = 2$.

Hence we can conclude that $M_{i+1}, s^h \models < r > \phi'$.

Let $M_{i+1}, s^h \models < r > \phi'$, then there exists a path $(s^h = s_0^{h_0}, \ldots, s_q^{h_q}) \in Paths_{M_{i+1}}(r)$ such that $M_{i+1}, s_q^{h_q} \models \phi'$. We prove $M_i, s \models < r > \phi'$ by induction on the length q of the path.

If $q = 0$, then $(s^h) \in Paths_{M_{i+1}}(r)$, and, by Proposition 4, there exists a formula $< \phi_1?; \ldots; \phi_g? > \phi'$, with $g \geq 0$, such that:

- all tests $\phi_j?$ occur in r, and hence all ϕ_j are subformulae of $< r > \phi'$;
- $M_{i+1}, s^h \models < \phi_1?; \ldots; \phi_g? > \phi'$;
- $< \phi_1?; \ldots; \phi_g? > \phi' \Rightarrow < r > \phi'$ is valid.

By formula induction hypothesis, for every $\psi \in \{\phi_1, \ldots \phi_g, \phi'\}$, $M_{i+1}, s^h \models \psi$ iff $M_i, s \models \psi$, and hence $M_i, s \models < r > \phi'$.

If, $q > 0$, then, by Proposition 5, there exists a formula $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi'$, with $g \geq 0$, such that:

- all tests $\phi_j?$ occur in r, and hence all ϕ_j are subformulae of $< r > \phi'$;
- $r' \in Post(r)$, and hence by Proposition 3, the formula $< r' > \phi'$ is equivalent to ψ for some $\psi \in CL(< r > \phi') \subseteq CL(\Phi)$;
- $(s_0^{h_0}, s_1^{h_1}) \in \mathcal{R}_{i+1_a}$;
- $(s_1^{h_1}, \ldots, s_q^{h_q}) \in Paths_{M_{i+1}}(r')$;
- $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi' \Rightarrow < r > \phi'$ is valid.

By formula induction hypothesis, for every $\phi_x \in \{\phi_1, \ldots \phi_g\}$, $M_{i+1}, s^h \models \phi_x$ iff $M_i, s \models \phi_x$.

By path induction hypothesis, $M_{i+1}, s_1^{h_1} \models < r' > \phi'$ implies $M_i, s_1 \models < r' > \phi'$, since $(s_1^{h_1}, \ldots, s_q^{h_q}) \in Paths_{M_{i+1}}(r')$ is shorter than $(s_0^{h_0}, \ldots, s_q^{h_q})$.

While, by definition, $(s_0^{h_0}, s_1^{h_1}) \in \mathcal{R}_{i+1_a}$ implies that $(s_0, s_1) \in \mathcal{R}_{i_a}$, either in the case $h_0 = h_1$ or $h_0 \neq h_1$ $(h_0 = h)$.

Hence we can conclude that $M_i, s \models < r > \phi'$.

Finally, as $M'' = M_l$, we have that all states of M'' are suitably assigned to the propositions $H_{(\leq 1\, a)}$, for $a = a_1 \ldots a_l$, and M'' is a model of Φ.

Step 2 We transform the model $M'' = (\mathcal{S}'', \{R''_P\}, \Pi'')$ of Φ into a model $M' = (\mathcal{S}', \{R'_P\}, \Pi')$ of $\gamma(\Phi)$. Considering a state $s \in S''$ such that $M'', s \models \Phi$, we define M' as follows:

$$\mathcal{S}' = \{t \mid (s,t) \in (\bigcup_P (\mathcal{R}''_P \cup \mathcal{R}''_{P^-}))^*\}$$
$$\mathcal{R}'_P = \mathcal{R}''_P \cap \mathcal{S}' \times \mathcal{S}'$$
$$\Pi'(t) = \begin{cases} \Pi''(t) \cup \{A_{(\leq 1\, a)}\} & \text{if } M'', t \models (\leq 1\, a) \\ \Pi''(t) & \text{otherwise.} \end{cases}$$

It is easy to verify that $M', s \models \gamma_1(\Phi)$, and (trivially) $M', s \models \gamma_2(\Phi)$. Therefore, M' is a model of $\gamma(\Phi)$. $\square$

We can now formulate the main result of this chapter.

Theorem 13 *Satisfiability in $\mathcal{DIF}$ is an EXPTIME-complete problem.*

Proof The satisfiability problem for $\mathcal{DI}$ is EXPTIME-complete, and by Lemma 6 the size of the $\mathcal{DI}$-counterpart $\gamma(\Phi)$ of a $\mathcal{DIF}$-formula Φ is polynomially related to the size of Φ. $\square$

As an immediate consequence we can characterize the computational complexity of reasoning in $\mathcal{CIF}$.

Theorem 14 *Satisfiability of $\mathcal{CIF}$ concepts, satisfiability of $\mathcal{CIF}$ TBoxes, and logical implication in $\mathcal{CIF}$ TBoxes, are EXPTIME-complete problems.*

3.3 Discussion

We did not use a standard filtration argument to prove our result. In fact, the standard filtration argument does not work in proving Theorem 11. Here is an example: Let the $\mathcal{DIF}$ formula Φ be $A \wedge [P^*]((\leq 1\ P^-) \wedge < P > \neg A)$, where A is an atomic proposition and P an atomic program (Φ is already in negation normal form).

The $\mathcal{DI}$ formula $\gamma_1(\Phi)$ is $A \wedge [P^*](A_{(\leq 1 P^-)} \wedge < P > \neg A)$ and its Fisher-Ladner Closure, $CL(\gamma_1(\Phi))$, is formed by

$$\begin{array}{l}
A \wedge [P^*](A_{(\leq 1 P^-)} \wedge < P > \neg A) \\
A \\
[P^*](A_{(\leq 1 P^-)} \wedge < P > \neg A) \\
[P][P^*](A_{(\leq 1 P^-)} \wedge < P > \neg A) \\
A_{(\leq 1 P^-)} \wedge < P > \neg A \\
A_{(\leq 1 P^-)} \\
< P^- > \neg A
\end{array}$$

and their negations.

The $\mathcal{DI}$ counterpart of Φ is $\gamma(\Phi) = \gamma_1(\Phi) \wedge \gamma_2(\Phi)$ where $\gamma_2(\Phi)$ assures that in every model of $\gamma(\Phi)$ if a state satisfies $A_{(\leq 1 P^-)}$ then all its P-successors satisfy the same formulae, wrt those that are members of $CL(\gamma_1(\Phi))$.

Now consider the structure $M = (S, R_P, \Pi)$:

$$\begin{array}{l}
S = \{d_1, d_2, d_3\} \\
R_P = \{(d_1, d_2)(d_2, d_3), (d_3, d_3)\} \\
\Pi(A) = \{d_1\}, \quad \Pi(A_{(\leq 1 P^-)}) = \{d_1, d_2, d_3\}
\end{array}$$

It is easy to verify that M is a model of $\gamma(\Phi)$, but not of Φ since d_3, has two (P^-)-successors, and therefore does not satisfy $(\leq 1\ P^-)$.

The states d_2 and d_3 satisfy the same formulae, wrt those that are members of $CL(\gamma_1(\Phi))$. Hence a filtration technique would allow us to merge them into a single state d_2', getting the new structure M'. In M', the state d_2' has two (P^-)-successors so it does not satisfy $(\leq 1\ P^-)$ and, as a consequence, again M' is not a model of Φ. Moreover M' is not even a model of $\gamma(\Phi)$, since d_2' has one (P^-)-successor, d_1, satisfying $A \in CL(\gamma_1(\Phi))$ and one, d_2, satisfying $\neg A \in CL(\gamma_1(\Phi))$, therefore $\gamma_2(\Phi)$ is not satisfied anymore.

In general, the ability to get a filtration of a model by a finite set of formulae (as $CL(\gamma_1(\Phi))$) leads to a finite model property. But $\mathcal{DIF}$ does not have the finite model property. Indeed, the above Φ is an example of a formula having only infinite models[6]. So we can conclude that filtration techniques are not suitable to prove the decidability of $\mathcal{DIF}$.

The construction we have described in this chapter, builds, from a given model of $\gamma(\Phi)$, a model of Φ that can be an infinite tree. In the example above, our construction gives as a result a new structure $M^{\mathcal{F}}$ which is an infinite chain of P such that all the states along the chain satisfy $(\leq 1\ P^-)$ while only the first state satisfies A. It is easy to verify that $M^{\mathcal{F}}$ is indeed a model of Φ. We find it quite surprising and interesting that satisfiability in $\mathcal{DIF}$, which is a logic that does not have the finite model property, can be reduced, in a natural way, to satisfiability in $\mathcal{DI}$, a logic that does have it.

We have already mentioned that the formula $\gamma_2(\Phi)$ can be thought of as a finite instantiation of the axiom schema $(A_{(\leq 1 a)} \wedge < a > \phi) \Rightarrow [a]\phi$, which is sufficient to guaranty that the $\mathcal{DI}$ formula $\gamma(\Phi)$ is satisfiable if and only if $\mathcal{DIF}$ formula Φ is satisfiable. The methodology of reducing satisfiability in a given logic to satisfiability in a target logic by constraining structures of the target logic, through a finite (polynomial) number of instances of an axiom schema, can be exploited to establish decidability (and complexity) in many situations. In fact, it is the central element behind many of the results in this thesis.

It is worth noting that, since $\mathcal{DIF}$ subsumes Converse Deterministic PDL, also formulae of that logic can be encoded in $\mathcal{DI}$. This fact gives us an optimal procedure[7] to decide the satisfiability of Converse Deterministic PDL formulae that does not rely on techniques based on automata on infinite structures as those in [129, 131].

Observe also that the mapping γ can be easily restricted to encode Deterministic PDL formulae in PDL. Though, in this simpler case there is no need of a sophisticated technique, as the one above, to build a model of a Deterministic PDL formula from its PDL counterpart, a standard filtration argument being sufficient. Indeed, the method adopted in [7] to study satisfiability of Deterministic PDL, can be rephrased making use of a mapping similar to γ.

[6] Such a formula is a variant of the Converse Deterministic PDL formula $A \wedge [(P^-)^*] < P^- > \neg A$ (see for example [131]).

[7] Note that satisfiability of Converse Deterministic PDL is an EXPTIME-complete problem.

Chapter 4

Qualified Number Restrictions

In this chapter we study the description logic $\mathcal{CIN}$ and the propositional dynamic logic $\mathcal{DIN}$ obtained from $\mathcal{CI}$ and $\mathcal{DI}$ by adding the qualified number restriction constructs $(\leq n\, a.C)$ and $(\geq n\, a.C)$ with $n \geq 1$. In the setting of description logics, qualified number restrictions where first considered (without inverse and reflexive-transitive closure of roles) in [65]. The qualified number restriction $(\leq n\, a.C)$ denotes the set of objects that have links with at most n objects in C, and $(\geq n\, a.C)$ the set of objects that have links with at least n objects in C, where a is either an atomic role or the inverse of an atomic role.

4.1 The logics $\mathcal{CIN}$ and $\mathcal{DIN}$

Concepts of $\mathcal{CIN}$ are formed according to the following abstract syntax:

$$\begin{array}{rcl} C & ::= & \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid C_1 \Rightarrow C_2 \mid \neg C \mid \\ & & \exists R.C \mid \forall R.C \mid (\leq n\, a.C) \mid (\geq n\, a.C) \\ \\ a & ::= & P \mid P^- \\ R & ::= & a \mid R_1 \sqcup R_2 \mid R_1 \circ R_2 \mid R^* \mid R^- \mid id(C) \end{array}$$

where A denotes an atomic concept, C (possibly with a subscript) a generic concept, P an atomic role, a a simple role, i.e. either an atomic role or the inverse of an atomic role, R (possibly with a subscript) a generic role.

The semantics of $\mathcal{CIN}$ is the same as for $\mathcal{CI}$, except for qualified number restrictions $(\leq n\, a.C)$ and $(\geq n\, a, C)$ with $n \geq 1$, whose meaning in an interpretation $\mathcal{I}$ is the following (recall $a = P \mid P^-$):

$$\begin{array}{rcl} (\leq n\, a.C)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \text{ there exists } \textit{at most } n\ d' \text{ such that} \\ & & (d, d') \in a^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\} \\ (\geq n\, a.C)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \text{ there exists } \textit{at least } n\ d' \text{ such that} \end{array}$$

$$(d, d') \in a^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\}$$

Observe that the two kinds of qualified number restrictions are interdefinable since $(\geq 1\, a.C)$ is equivalent to $\exists a.C$ and $(\geq n\, a.C)$ with $n \geq 2$ is equivalent to $\neg(\leq n-1\, a.C)$. A functional restriction $(\leq 1\, a)$ is expressible by $(\leq 1\, a.\top)$, hence $\mathcal{CIN}$ is a generalization of $\mathcal{CIF}$. Similarly the well-known constructs called *number restrictions* $(\leq n\, a)$ and $(\geq n\, a)$ are expressed in $\mathcal{CIN}$ by $(\leq n\, a.\top)$ and $(\geq n\, a.\top)$ respectively. Indeed qualified number restrictions are the most general kind of cardinality constraints, while functional restrictions can be considered the simplest ones. Note that in $\mathcal{CIN}$, as in $\mathcal{CIF}$, there is complete symmetry between atomic roles and inverse of atomic roles.

The corresponding propositional dynamic logic is called $\mathcal{DIN}$ and its syntax is as follows:

$$\begin{array}{lcl} \phi & ::= & \top \mid \bot \mid A \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \Rightarrow \phi_2 \mid \neg\phi \mid \\ & & < r > \phi \mid [r]\phi \mid (\leq n\, a.\phi) \mid (\geq n\, a.\phi) \\ \\ a & ::= & P \mid P^- \\ r & ::= & a \mid r_1 \cup r_2 \mid r_1; r_2 \mid r^* \mid r^- \mid \phi? \end{array}$$

where A denotes a propositional letter, ϕ (possibly with a subscript) a formula, P an atomic program, a a *simple program*, i.e. an atomic program or the converse of an atomic program, and r (possibly with a subscript) a generic program.

Consistently with its interpretation in $\mathcal{CIN}$ the new construct is interpreted as follows: given a structure $M = (\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$ and a state $s \in \mathcal{S}$,

$$\begin{array}{lll} M, s \models (\leq n\ a.\phi) & \text{iff} & \text{there are } \textit{at most } n \text{ states } t \text{ such that} \\ & & (s,t) \in \mathcal{R}_a \text{ and } M, t \models \phi \\ M, s \models (\geq n\ a.\phi) & \text{iff} & \text{there are } \textit{at least } n \text{ states } t \text{ such that} \\ & & (s,t) \in \mathcal{R}_a \text{ and } M, t \models \phi. \end{array}$$

The rest of the constructs are interpreted as in $\mathcal{DI}$.

Intuitively, if s is a state satisfying $(\leq n\ a.\phi)$ (respectively $(\geq n\ a.\phi)$), then there are at most (at least) n a-successors of s satisfying ϕ. By means of qualified number restrictions we can control the nondeterminism of simple programs in a quite sophisticated way. Local determinism of a simple program a can be imposed by $(\leq 1\ a.\top)$.

Qualified number restrictions are sometimes called graded nondeterminism constructs. Indeed, as such a name suggests, they are strongly related to *graded modalities* in modal logic (see the final section of this chapter).

4.2 Reasoning in $\mathcal{CN}$ and $\mathcal{DN}$

Before discussing reasoning in $\mathcal{CIN}$ and $\mathcal{DIN}$, we discuss some of the issues involved in the simpler logics $\mathcal{CN}$ and $\mathcal{DN}$ obtained by dropping the constructs for inverse roles and converse programs respectively. This will allow us to gain some intuition about results for $\mathcal{CIN}$ and $\mathcal{DIN}$.

The decidability and complexity of both satisfiability of $\mathcal{CN}$ concepts and logical implication in $\mathcal{CN}$ TBoxes can be derived by exploiting the correspondence between $\mathcal{CN}$ and $\mathcal{DN}$[1]. Hence it suffices to establish decidability and complexity of satisfiability for $\mathcal{DN}$. We do so by translating $\mathcal{DN}$ formulae in Deterministic Propositional Dynamic Logic formulae whose satisfiability is known to be decidable and EXPTIME-complete [7].

Let us ignore for a moment the qualified number restriction constructs. Formulae of $\mathcal{DN}$ without qualified number restrictions are, in fact, formulae of the basic PDL. It is well-known (see [87]) that such formulae can be reduced to Deterministic PDL formulae: we replace each atomic program P in a formula Φ by $F_P;(F'_P)^*$ where F_P and F'_P are new atomic programs that are (globally) deterministic. Let us call the resulting formula Φ', we have that Φ is satisfiable if and only if Φ' is so.[2]

We briefly sketch the reasoning behind the proof of this statement. The if direction is straightforward. The only if direction is as follows. We recall that both PDL and Deterministic PDL have the tree model property: if a formula has a model it has a tree model, i.e. a model having the form of a *tree*.[3] So we can restrict our attention to tree models only without loss of generality. Now there is a one-to-one transformation from tree models M^T of Φ to (tree) models M^B of Φ'. Indeed, we put $\mathcal{S}^B = \mathcal{S}^T$, $\Pi^B = \Pi^T$, and given a state x of M^T having as P-successors $z_1, \ldots, z_l$,[4] we put $(x, z_1) \in \mathcal{R}^B_{F_P}$, and $(z_i, z_{i+1}) \in \mathcal{R}^B_{F'_P}$, for $i = 1, \ldots, l-1$. In this way we have $(x, z_i) \in \mathcal{R}^T_P$ if and only if $(x, z_i) \in \mathcal{R}^B_{F_P;(F'_P)^*}$.[5]

We remark that M^T is required to be a tree because once we get M^B we need to recover the "original" P-predecessor x of a state z_i, namely we need $(F_P;(F'_P)^*)^-$ to be *deterministic*, otherwise, given a state z_i, we would not know which of the various $(F_P;(F'_P)^*)^-$-successors is its original P-predecessor x, and therefore we would not be able to reconstruct M^T from M^B.

Representing atomic programs P as $F_P;(F'_P)^*$, where F_P and F'_P are deterministic, makes it easy to express qualified number restrictions as constraints on the chain of $F_P;(F'_P)^*$-successors of a state. For example, let us denote the transitive closure of r as r^+ – i.e. $r^+ \doteq r;r^*$:

$(\leq 3\ P.\phi)$ can be expressed by

$$[F_P;(F'_P)^*;\phi?;(F'_P)^+;\phi?;(F'_P)^+;\phi?;(F'_P)^+]\neg\phi$$

[1]The correspondence is realized by modifying straightforwardly the mappings δ and δ^+ described in Chapter 2 so to consider the absence of inverse roles and the presence of qualified number restrictions.

[2]Note that while it is necessary to introduce one F_P for each P, we could introduce just one F_U, instead of all F'_{P_i}. Here we have preferred to be slightly redundant, for the sake of clarity.

[3]Given a model of Φ we get a tree model simply by "unfolding" the original one.

[4]We implicitly assume that M^T is a finite branching tree model. This can be done without loss of generality since *PDL* has the finite model property, and hence unfolding a finite model we get a finite branching tree model. Note however that it would suffice to assume M^T to be a countable branching tree model.

[5]Note that this construction is similar to the one often used in programming to reduce n-ary trees to binary trees by coding children of a node as the combination of one child and its siblings.

that is equivalent to

$$[F_P;(F'_P)^*](\phi \Rightarrow [(F'_P)^+](\phi \Rightarrow [(F'_P)^+](\phi \Rightarrow [(F'_P)^+]\neg\phi)))$$

and can be read as "everywhere along the chain $F_P;(F'_P)^*$ there are at most three states in which ϕ holds", that corresponds exactly to the intended meaning.

$(\geq 3\ P.\phi)$ can be expressed by

$$< F_P;(F'_P)^*;\phi?;(F'_P)^+;\phi?;(F'_P)^+ > \phi$$

that is equivalent to

$$< F_P;(F'_P)^* > (\phi \wedge < (F'_P)^+ > (\phi \wedge < (F'_P)^+ > \phi))$$

and can be read as "somewhere along the chain $F_P;(F'_P)^*$ there are at least three states in which ϕ holds", that again corresponds exactly to the intended meaning.

The above discussion leads to the following results. Let Φ be a $\mathcal{DN}$ formula. We call the Deterministic PDL counterpart $v(\Phi)$ of Φ the formula obtained as follows:

1. We replace every atomic program P by $F_P;(F'_P)^*$, where F_P and F'_P are new deterministic atomic programs.
2. We replace every qualified number restriction

$$\begin{array}{l} (\leq\ n\ (F_P;(F'_P)^*).\phi) \text{ by } [(F_P;(F'_P)^*;(\phi?;(F'_P)^+)^n]\neg\phi, \\ (\geq\ n\ (F_P;(F'_P)^*).\phi) \text{ by } < F_P;(F'_P)^*;(\phi?;(F'_P)^+)^{n-1} > \phi, \end{array}$$

   where the notation r^n stands for n repetitions of r.[6]

Theorem 15 *A $\mathcal{DN}$ formula Φ is satisfiable if and only if the Deterministic PDL formula $v(\Phi)$ is satisfiable.*

Theorem 16 *Satisfiability in $\mathcal{DN}$ is a EXPTIME-complete problem.*

Observe that we are translating $\mathcal{DN}$ to Deterministic PDL. As a special case we can translate $\mathcal{DF}$ to Deterministic PDL by expressing $(\leq 1\ P)$ as $[F_P][F'_P]\bot$.

4.3 Reasoning in $\mathcal{CIN}$ and $\mathcal{DIN}$

Let us go back to $\mathcal{CIN}$ and $\mathcal{DIN}$. The decidability and computational complexity of both satisfiability of $\mathcal{CIN}$ concepts and logical implication in $\mathcal{CIN}$ TBoxes can be derived by exploiting the correspondence between $\mathcal{CIN}$ and $\mathcal{DIN}$[7]. Hence it suffices to establish decidability and computational complexity of satisfiability in $\mathcal{DIN}$.

[6]Note that, in accordance with $(\geq n\ P.\phi) = \neg(\leq n-1\ P.\phi)$, we have

$$\neg[F_P;(F'_P)^*;(\phi?;(F'_P)^+)^{n-1}]\neg\phi = < F_P;(F'_P)^*;(\phi?;(F'_P)^+)^{n-1} > \phi.$$

[7]The correspondence is realized by extending straightforwardly the mappings δ and δ^+ described in Chapter 2 in order to deal with qualified number restrictions.

Let us ignore once again qualified number restrictions for the moment, i.e. we cut $\mathcal{DIN}$ down to $\mathcal{DI}$. The presence of converse programs in $\mathcal{DI}$ makes its structures no longer reducible to tree structures as above[8], making the technique shown in the previous section inapplicable. Nonetheless we are able to obtain essentially the same results, by developing a more involved reduction.

Indeed, we are going to prove that for any $\mathcal{DIN}$ formula Φ there exists a $\mathcal{DIF}$ formula Φ', whose size is polynomial wrt the size of Φ, that is satisfiable if and only if Φ' is so. Since we have proved in Chapter 3 that satisfiability in $\mathcal{DIF}$ is EXPTIME-complete, this guarantees that satisfiability in $\mathcal{DIN}$ is EXPTIME-complete too. In order to carry out this reduction we first need to *reify* the relations associated with atomic programs, then we can exploit a technique similar to the one used in the previous section.

4.3.1 Reification of binary relations

Atomic programs are interpreted as binary relations. Reifying a binary relation means creating an object for each tuple in the relation. The set of such objects represents the set of tuples forming the relation. However the following problem arises: in general, there may be two or more objects referring to the same tuple. Obviously in order to have a faithful representation of a relation such a situation must be avoided.

Given an atomic program P, we call its *reified form* the following program

$$f_1^-; A_P?; f_2$$

where A_P is a new proposition denoting objects representing the tuples of the relation associated with P, and f_1 and f_2 denote two functions that, given an object in A_P, return the first and the second component respectively of the tuple represented by the object. In other words, given a running of an atomic program starting in s and ending in t, we replace it by adding an intermediate state z and two deterministic atomic programs f_1 and f_2 starting from z and ending in s and t respectively. We call states such as z *pseudo states* since they denote the materialization of tuples such as (s,t) and not real states.

Note that there is a clear symmetry between the program $f_1^-;A_P?;f_2$ and its converse $f_2^-;A_P?;f_1$.

After the reification of P, formulae of the form $(\cdot\ n\ P.\varphi)/(\cdot\ n\ P^-.\varphi)$ assume the form $(\cdot\ n\ f_1^-;A_P?;f_2.\phi)/(\cdot\ n\ f_2^-;A_P?;f_1.\phi)$, thus denoting qualified number restrictions of complex programs. Yet, since the programs f_1 and f_2 denote partial functions, the following equivalences hold:

$$\begin{array}{lcl}
(\leq n\ (f_1^-;A_P?;f_2).\phi) & \equiv & (\leq n\ f_1^-.(<A_P?;f_2>\phi)),\\
(\geq n\ (f_1^-;A_P?;f_2).\phi) & \equiv & (\geq n\ f_1^-.(<A_P?;f_2>\phi)),\\
(\leq n\ (f_1^-;A_P?;f_2)^-.\phi) & \equiv & (\leq n\ f_2^-.(<A_P?;f_1>\phi)),\\
(\geq n\ (f_1^-;A_P?;f_2)^-.\phi) & \equiv & (\geq n\ f_2^-.(<A_P?;f_1>\phi)).
\end{array}$$

Definition Let Φ be a $\mathcal{DIN}$ formula. We define the *reified-counterpart* $\upsilon_1(\Phi)$ of Φ as the conjunction of two formulae, $\upsilon_1(\Phi)=\upsilon_0(\Phi)\wedge\Theta_1$, where:

[8] Indeed the presence of converse programs makes the structures reducible to "two-way" tree structures, as opposed to the "one-way" tree structures needed here.

- $\upsilon_0(\Phi)$ is obtained from the original formula Φ by replacing
  - every atomic program P_i, $i = 1 \ldots m$, by the complex program $f_1^-; A_{P_i}?; f_2$, where f_1, f_2 are new atomic programs (the only ones present after the transformation) and A_{P_i} is a new atomic proposition;
  - and then every qualified number restriction

$$\begin{array}{lcl} (\leq n\ (f_1^-; A_P?; f_2).\phi) & \text{by} & (\leq n\ f_1^-.(< A_P?; f_2 > \phi)), \\ (\geq n\ (f_1^-; A_P?; f_2).\phi) & \text{by} & (\geq n\ f_1^-.(< A_P?; f_2 > \phi)), \\ (\leq n\ (f_1^-; A_P?; f_2)^-.\phi) & \text{by} & (\leq n\ f_2^-.(< A_P?; f_1 > \phi)), \\ (\geq n\ (f_1^-; A_P?; f_2)^-.\phi) & \text{by} & (\geq n\ f_2^-.(< A_P?; f_1 > \phi)). \end{array}$$

- $\Theta_1 = [(f_1 \cup f_2 \cup f_1^- \cup f_2^-)^*]((\leq\ 1\ f_1) \wedge (\leq\ 1\ f_2))$.[9]

□

Lemma 17 *Let Φ be a $\mathcal{DIN}$ formula, and $\upsilon_1(\Phi)$ its reified-counterpart. Then $\upsilon_1(\Phi)$ is a $\mathcal{DIN}$ formula, and its size is polynomially related to the size of Φ.*

Proof Straightforward. □

Observe that Θ_1 imposes the global determinism of both f_1 and f_2, that is, in every model $M = (\mathcal{S}, \{\mathcal{R}_{f_1}, \mathcal{R}_{f_2}\}, \Pi)$ of $\upsilon_0(\Phi)$, the relations $\mathcal{R}_{f_1}$ and $\mathcal{R}_{f_2}$ are partial functions.

The next lemma guarantees us that, without loss of generality, we can restrict our attention to models of $\upsilon_1(\Phi)$ that faithfully represent relations associated with atomic programs, i.e. models in which each tuple of such relations is represented by a single (pseudo) state. This is an essential property and guarantees the soundness of our reified representation of relations associated with atomic programs.

Lemma 18 *If the formula $\upsilon_1(\Phi)$ has a model $M = (\mathcal{S}, \{\mathcal{R}_{f_1}, \mathcal{R}_{f_2}\}, \Pi)$ then it has a model $M' = (\mathcal{S}', \{\mathcal{R}'_{f_1}, \mathcal{R}'_{f_2}\}, \Pi')$ such that for each $(x, y) \in \mathcal{R}'_{f_1^-; A_{P_i}?; f_2}$ there is exactly one z_{xy} such that $(z_{xy}, x) \in \mathcal{R}'_{f_1}$ and $(z_{xy}, y) \in \mathcal{R}'_{f_2}$. That is, for all $z_1, z_2, x, y \in \mathcal{S}'$ such that $z_1 \neq z_2$ and $x \neq y$, the following condition holds:*

$$\begin{array}{l} (A_{P_i} \in \Pi'(z_1) \wedge A_{P_i} \in \Pi'(z_2)) \Rightarrow \\ \qquad \neg((z_1, x) \in \mathcal{R}'_{f_1} \wedge (z_2, x) \in \mathcal{R}'_{f_1} \wedge (z_1, y) \in \mathcal{R}'_{f_2} \wedge (z_2, y) \in \mathcal{R}'_{f_2}). \end{array}$$

Proof Suppose that the condition is not already satisfied by the model M, we show how to build a model M' in which the condition is satisfied.

Let us introduce some notation. Given a pseudo state z denoting a tuple $(x, y) \in \mathcal{R}_{f_1^-; A_P?; f_2}$ we denote x by $f_1(z)$ and y by $f_2(z)$, this is in agreement with $\mathcal{R}_{f_1}$ and $\mathcal{R}_{f_2}$ being functional. We call *conflict* the presence of more pseudo states referring to the same tuple. Let $(x, y) \in \mathcal{R}_{f_1^-; A_P?; f_2}$, if there is more that one pseudo state z referring to (x, y) – that is, if there is more that one (pseudo) state z in M such that

[9]Observe that f_1, f_2 are the only atomic programs occurring in $\upsilon_0(\Phi)$.

$A_P \in \Pi(z)$ and $(z,x) \in \mathcal{R}_{f_1}$ and $(z,y) \in \mathcal{R}_{f_2}$ – then we randomly choose one such pseudo state to represent (x,y) and we say that the others *induce a conflict*. We call $Conf$ the set of all pseudo states inducing a conflict.[10]

We start our construction, by defining a structure $M_{2^{Conf}}$ as the disjoint union of $|2^{Conf}|$ copies of M, one copy, denoted by $M^{\mathcal{E}}$, for every set $\mathcal{E} \in 2^{Conf}$. We denote by $s^{\mathcal{E}}$ the copy in $M^{\mathcal{E}}$ of the state s in M. The structure $M_{2^{Conf}}$ is trivially a model of $\upsilon_1(\Phi)$ as M is.

Let $M^{\mathcal{E}}$ and $M^{\mathcal{E}'}$ be two copies of M in $M_{2^{Conf}}$, we call "exchanging $f_2(t^{\mathcal{E}})$ with $f_2(t^{\mathcal{E}'})$" the operation on $M_{2^{Conf}}$ consisting of removing the tuple $(t^{\mathcal{E}}, f_2(t^{\mathcal{E}}))$ from $\mathcal{R}^{\mathcal{E}}_{f_2}$ replacing it with $(t^{\mathcal{E}}, f_2(t^{\mathcal{E}'}))$ and, at the same time, removing $(t^{\mathcal{E}'}, f_2(t^{\mathcal{E}'}))$ from $\mathcal{R}^{\mathcal{E}'}_{f_2}$ replacing it with $(t^{\mathcal{E}'}, f_2(t^{\mathcal{E}}))$ [11]. By exchanging $f_2(t^{\mathcal{E}})$ with $f_2(t^{\mathcal{E}'})$, we resolve t for both $M^{\mathcal{E}}$ and $M^{\mathcal{E}'}$, in the sense that $t^{\mathcal{E}}$ and $t^{\mathcal{E}'}$ no longer induce conflicts.

Note that given a $t \in Conf$, we can univocally associate to a set $\mathcal{E} \in 2^{Conf}$ such that $t \in \mathcal{E}$, the set $\mathcal{E} - \{t\}$. The set of all $\mathcal{E}$ and $\mathcal{E} - \{t\}$ such that $\mathcal{E} \in 2^{Conf}$ and $t \in \mathcal{E}$ for some $t \in Conf$, is equal to 2^{Conf}.

Now we can complete our construction. We get a model M' with the desired property by modifying $M_{2^{Conf}}$ as follows: For each state $t \in Conf$, for each $\mathcal{E} \in 2^{Conf}$ such that $t \in \mathcal{E}$, we exchange $f_2(t^{\mathcal{E}})$ with $f_2(t^{\mathcal{E}-\{t\}})$.[12]

Indeed proceeding in this way, on the one hand all conflicts present in the original model M are eliminated from all its copies in $M_{2^{Conf}}$. On the other hand no new conflicts are created as shown in the following. New conflicts could be created only by resolving two $t, t' \in Conf$ in the same $\mathcal{E} \in 2^{Conf}$, since otherwise we are guaranteed by definition of $M_{2^{Conf}}$ that $f_1(t^{\mathcal{E}}) \neq f_1(t^{\mathcal{E}'})$ if $\mathcal{E} \neq \mathcal{E}'$. However, given two pseudo states $t, t' \in Conf$ and a set $\mathcal{E} \in 2^{Conf}$, following the construction proposed we have that:

- if $t, t' \in \mathcal{E}$, we exchange $f_2(t^{\mathcal{E}})$ with $f_2(t^{\mathcal{E}-\{t\}})$ to resolve t and $f_2((t')^{\mathcal{E}})$ with $f_2((t')^{\mathcal{E}-\{t'\}})$ to resolve t';
- if $t, t' \notin \mathcal{E}$. we exchange $f_2(t^{\mathcal{E}})$ with $f_2(t^{\mathcal{E}\cup\{t\}})$ to resolve t and $f_2((t')^{\mathcal{E}})$ with $f_2((t')^{\mathcal{E}\cup\{t'\}})$ to resolve t';
- if $t \in \mathcal{E}$ and $t' \notin \mathcal{E}$, we exchange $f_2(t^{\mathcal{E}})$ with $f_2(t^{\mathcal{E}-\{t\}})$ to resolve t and $f_2((t')^{\mathcal{E}})$ with $f_2((t')^{\mathcal{E}\cup\{t'\}})$ to resolve t'.

Observe that in all cases we resolve t, t' by acting on different copies on the original model M, so no conflicts can be introduced.

Finally, M' is indeed a model of $\upsilon_1(\Phi)$, since by construction Θ_1 is satisfied everywhere in M', and it is straightforward to check by induction on $\upsilon_0(\Phi)$, that, for all $\mathcal{E} \in 2^{Conf}$, $M, s \models \upsilon_0(\Phi)$ if and only if $M', s^{\mathcal{E}} \models \upsilon_0(\Phi)$. □

By using Lemma 18 we can prove the result below, which constitutes the preliminary step of our reduction form $\mathcal{DIN}$ to $\mathcal{DIF}$.

[10]Note that $Conf$ can be uncountable.

[11]Obviously the same thing can be done acting on $f_1(t^{\mathcal{E}})$ and $f_1(t^{\mathcal{E}'})$.

[12]Note that the transformation leading from $M_{2^{Conf}}$ to M' does not change the number of tuples in which a state occurs.

Lemma 19 *A $\mathcal{DIN}$ formula Φ is satisfiable if and only if its reified-counterpart $\upsilon_1(\Phi)$ is satisfiable.*

Proof $\Rightarrow$ Let $M = \{\mathcal{S}, \{\mathcal{R}_P\}, \Pi\}$ be a model of Φ. We define a model of $M' = \{\mathcal{S}', \{\mathcal{R}'_{f_1}, \mathcal{R}'_{f_2}\}, \Pi'\}$ of $\upsilon_1(\Phi)$ as follows:

- $\mathcal{S}' = \mathcal{S} \cup \{z_{xy} \mid (x,y) \in \mathcal{R}_{P_i} \text{ for some } P_i\}$,
- $\mathcal{R}'_{f_1} = \{(z_{xy}, x) \mid (x,y) \in \mathcal{R}_{P_i}\}$, $\mathcal{R}'_{f_2} = \{(z_{xy}, y) \mid (x,y) \in \mathcal{R}_{P_i}\}$,
- $\Pi'(t) = \begin{cases} \Pi(t) & \text{for } t \in \mathcal{S} \\ \{A_{P_i} \mid (x,y) \in \mathcal{R}_{P_i}\} & \text{for } t = z_{xy}. \end{cases}$

The construction above implies $(x,y) \in \mathcal{R}_{P_i}$ iff $(x,y) \in \mathcal{R}'_{f_1^-;A_{P_i}?;f_2}$.

Since $\mathcal{R}'_{f_1}, \mathcal{R}'_{f_2}$ are partial functions, it follows that Θ_1 is satisfied all over M', and by induction on Φ, it is easy to verify that $M, s \models \Phi$ if and only if $M', s \models \upsilon_0(\Phi)$.

$\Leftarrow$ Let $M' = \{\mathcal{S}', \{\mathcal{R}'_{f_1}, \mathcal{R}'_{f_2}\}, \Pi'\}$ be a model of $\upsilon_1(\Phi)$, By Lemma 18 we can assume that for each $(x,y) \in \mathcal{R}'_{f_1^-;A_{P_i}?;f_2}$ there is exactly one z_{xy} such that $(z_{xy}, x) \in \mathcal{R}'_{f_1}$ and $(z_{xy}, y) \in \mathcal{R}'_{f_2}$. This guarantees that qualified number restrictions holding in the states x and y, restrict correctly the number of $(f_1^-;A_{P_i}?;f_2)$-successors and $(f_2^-;A_{P_i}?;f_1)$-successors, respectively.[13]

We define a model $M = \{\mathcal{S}, \{\mathcal{R}_P\}, \Pi\}$ of Φ as follows. First we define $\overline{\mathcal{R}}_{P_i} = \mathcal{R}'_{f_1^-;A_{P_i}?;f_2}$ Then, let $s \in \mathcal{S}'$ be a state such that $M', s \models \upsilon_1(\Phi)$, we define

$$\begin{array}{l} \mathcal{S} = \{t \mid (s,t) \in (\bigcup_i(\overline{\mathcal{R}}_{P_i} \cup \overline{\mathcal{R}}_{P_i}^-))^*\}, \\ \mathcal{R}_{P_i} = \overline{\mathcal{R}}_{P_i} \cap (\mathcal{S} \times \mathcal{S}), \\ \Pi(t) = \Pi'(t) - \{A_{P_i} \text{ for any } P_i\}, \quad \text{for all } t \in \mathcal{S}. \end{array}$$

Finally it is easy to verify by induction of $\upsilon_0(\Phi)$ that $M', s \models \upsilon_0(\Phi)$ if and only if $M, s \models \Phi$. $\square$

4.3.2 Reducing $\mathcal{DIN}$ to $\mathcal{DIF}$

By Lemma 19, we can concentrate on the reified-counterparts of $\mathcal{DIN}$ formulae. Note that these are $\mathcal{DIN}$ formulae themselves, but their special form allows us to convert them into $\mathcal{DIF}$ formulae. We adopt a technique resembling the one exploited for reducing $\mathcal{DN}$ to Deterministic PDL, in the previous section. Intuitively the technique works as follows. We represent a reified program $f_1^-;A_P?;f_2$ by the program $F_1;A_P?;(F_1';A_P?)^*;(F_2;A_P?;(F_2';A_P?)^*)^-$, where F_j, F_j' (with $j = 1,2$) are new deterministic atomic programs. In this way the program $f_j^-;A_P?$ $(j = 1,2)$ which is not deterministic in general, is expressed by chain $F_j;A_P?;(F_j';A_P?)^*$ of deterministic programs, and qualified number restrictions can be encoded as constraints on such a chain. The only cardinality constraints that are present in the resulting formulae

[13] Otherwise we could get something like: $(z_{xy},x),(z'_{xy},x) \in \mathcal{R}'_{f_1}$, $(z_{xy},y),(z'_{xy},y) \in \mathcal{R}'_{f_2}$, $A_P \in \Pi'(z_{xy})$ and $A_P \in \Pi'(z'_{xy})$. In this case $(\geq 2\,(f_1^-;A_{P_i}?;f_2).\top)$ holds in x, but actually there is only one tuple $(x,y) \in \mathcal{R}'_{f_1^-;A_{P_i}?;f_2}$.

are functional restrictions. Hence by first transforming a $\mathcal{DIN}$ formulae into their reified-counterpart and then applying the technique sketched above we reduced $\mathcal{DIN}$ to $\mathcal{DIF}$ which has been studied in Chapter 3.

Formally we define the $\mathcal{DIF}$-counterpart of a $\mathcal{DIN}$ formula as follows.

Definition Let Φ be a $\mathcal{DIN}$ formula and $\upsilon_1(\Phi) = \upsilon_0(\Phi) \wedge \Theta_1$ its reified-counterpart. We define the *$\mathcal{DIF}$-counterpart* $\upsilon_2(\Phi)$ of Φ as the conjunction of two formulae, $\upsilon_2(\Phi) = \upsilon_0'(\Phi) \wedge \Theta_2$, where:

- $\upsilon_0'(\Phi)$ is obtained from $\upsilon_0(\Phi)$ by replacing
  - every occurrence of program $f_1^-; A_{P_i}?; f_2$ by
  $$F_1; A_{P_i}?; (F_1'; A_{P_i}?)^*; (F_2; A_{P_i}?; (F_2'; A_{P_i}?)^*)^-,$$
  where F_j, F_j' $(j = 1, 2)$ are new atomic programs;
  - every $(\leq n\ f_1^- . < A_{P_i}?; f_2 > \phi)$ by
  $$[F_1; A_{P_i}?; (F_1'; A_{P_i}?)^*; (\phi'?; (F_1'; A_{P_i}?)^+)^n]\neg\phi',$$
  and every $(\geq n\ f_1^- . < A_{P_i}?; f_2 > \phi)$ by
  $$< F_1; A_{P_i}?; (F_1'; A_{P_i}?)^*; (\phi'?; (F_1'; A_{P_i}?)^+)^{n-1} > \phi',$$
  where $\phi' =< (F_2; A_{P_i}?; (F_2'; A_{P_i}?)^*)^- > \phi$;[14]
  - every $(\leq n\ f_2^- . < A_{P_i}?; f_1 > \phi)$ by
  $$[F_2; A_{P_i}?; (F_2'; A_{P_i}?)^*; (\phi''?; (F_2'; A_{P_i}?)^+)^n]\neg\phi'',$$
  and every $(\geq n\ f_2^- . < A_{P_i}?; f_1 > \phi)$ by
  $$< F_2; A_{P_i}?; (F_2'; A_{P_i}?)^*; (\phi''?; (F_2'; A_{P_i}?)^+)^{n-1} > \phi'',$$
  where $\phi'' =< (F_1; A_{P_i}?; (F_1'; A_{P_i}?)^*)^- > \phi$;
- $\Theta_2 = [(\bigcup_{j=1,2}(F_j \cup F_j' \cup F_j^- \cup F_j'^-))^*]\theta_1 \wedge \theta_2$, with each conjunct θ_j of the form:
$$\begin{array}{l}(\leq 1\ F_j) \wedge (\leq 1\ F_j') \wedge (\leq 1\ F_j^-) \wedge (\leq 1\ F_j'^-) \wedge \\ \neg(< F_j^- > \top \wedge < (F_j')^- > \top).\end{array}$$

□

Lemma 20 *Let Φ be a $\mathcal{DIN}$ formula, and $\upsilon_2(\Phi)$ its $\mathcal{DIF}$-counterpart. Then $\upsilon_2(\Phi)$ is a $\mathcal{DIF}$ formula, and its size is polynomially related to the size of Φ.*

[14] As before the notation r^+ stands for $r; r*$, and the notation r^n stands for n repetitions of r.

Proof Straightforward. □

Observe that Θ_2 constrains the models $M = (S, \{R_F\}, \Pi)$ of $\upsilon_2(\Phi)$ so that the relations $\mathcal{R}_{F_j}, \mathcal{R}_{F_j^-}, \mathcal{R}_{F'_j}, \mathcal{R}_{F_j'^-}$ are partial functions, and each state cannot be linked to other states by both $\mathcal{R}_{F_j^-}$ and $\mathcal{R}_{(F'_j)^-}$. As a consequence we get that $\mathcal{R}_{(F_j;A_{P_i}?;(F'_j;A_{P_i}?)^*)^-}$ is a partial function. This condition is required to prove the lemma below.

Lemma 21 *Let Φ be a $\mathcal{DIN}$ formula and $\upsilon_1(\Phi)$ its reified-counterpart. $\upsilon_1(\Phi)$ is satisfiable if and only if $\upsilon_2(\Phi)$ is satisfiable.*

Proof $\Rightarrow$ Let $M = \{\mathcal{S}, \{\mathcal{R}_{f_1}, \mathcal{R}_{f_2}\}, \Pi\}$ be a model of $\upsilon_1(\Phi)$. Then we build a model $M' = \{\mathcal{S}', \{\mathcal{R}'_F\}, \Pi'\}$ of $\upsilon_2(\Phi)$ as follows. First, we define $\{\overline{\mathcal{R}}'_F\}$. For each state $x \in \mathcal{S}$ such that $M, x \models < f_1^-; A_{P_i}?; f_2 > \top$, let $z_1, z_2 \ldots$ be the states such that $(x, z_k) \in \mathcal{R}_{f_1^-}$ and $M, z_k \models < A_{P_i}?; f_2 > \top$.[15] We put $(x, z_1) \in \overline{\mathcal{R}}'_{F_1}$, and for all $k = 1, 2, \ldots$ we put $(z_k, z_{k+1}) \in \overline{\mathcal{R}}'_{F'_1}$. Similarly, for each $x \in S$ such that $M, x \models < f_2^-; A_{P_i}?; f_1 > \top$, let $z_1, z_2, \ldots$ be the states such that $(x, z_k) \in \mathcal{R}_{f_2^-}$ and $M, z_k \models < A_{P_i}?; f_1 > \top$. We put $(x, z_1) \in \overline{\mathcal{R}}'_{F_2}$, and for all $k = 1, 2, \ldots$ we put $(z_k, z_{k+1}) \in \overline{\mathcal{R}}'_{F'_2}$. Then, let $s \in \mathcal{S}$ be such that $M, s \models \upsilon_1(\Phi)$, we define

$$\begin{array}{l} \mathcal{S}' = \{t \mid (s,t) \in (\bigcup_{j=1,2}(\overline{\mathcal{R}}'_{F_j} \cup \overline{\mathcal{R}}'_{F'_j} \cup \overline{\mathcal{R}}'^-_{F_j} \cup \overline{\mathcal{R}}'^-_{F'_j}))^*\}, \\ \mathcal{R}_F = \overline{\mathcal{R}}'_F \cap (\mathcal{S}' \times \mathcal{S}'), \\ \Pi'(t) = \Pi(t) \quad \text{for all } t \in \mathcal{S}'. \end{array}$$

Note that since $\mathcal{R}_{f_j}$ is a partial function, $\mathcal{R}'_{(F_j;A_{P_i}?;(F'_j;A_{P_i}?)^*)^-}$ is a partial function as well. By this construction we have that

$$(x,y) \in \mathcal{R}_{f_1^-;A_{P_i}?;f_2} \ \text{ iff } \ (x,y) \in \mathcal{R}'_{F_1;A_{P_i}?;(F'_1;A_{P_i}?)^*;(F_2;A_{P_i}?;(F'_2;A_{P_i}?)^*)^-}.$$

Moreover, Θ_2 is satisfied all over M'.

Considering that $\mathcal{R}'_{(F_j;A_{P_i}?;(F'_j;A_{P_i}?)^*)^-}$ is a partial function, and that

$$\begin{array}{l} [F_j; A_{P_i}?; (F'_j; A_{P_i}?)^*; (\phi?; (F'_j; A_{P_i}?)^+)^n]\neg\phi \\ < F_j; A_{P_i}?; (F'_j; A_{P_i}?)^*; (\phi?; (F'_j; A_{P_i}?)^+)^{n-1} > \phi \end{array}$$

specify that there are at most, at least respectively, n states satisfying ϕ, along the chain $F_j; A_{P_i}?; (F'_j; A_{P_i}?)^*$, it is easy to verify by induction on $\upsilon_0(\Phi)$ that $M, s \models \upsilon_0(\Phi)$ if and only if $M', s \models \upsilon'_0(\Phi)$.

$\Leftarrow$ Let $M' = \{\mathcal{S}', \{\mathcal{R}'_F\}, \Pi'\}$ be a model of $\upsilon_2(\Phi)$. We can define a model $M = \{\mathcal{S}, \{\mathcal{R}_{f_1}, \mathcal{R}_{f_2}\}, \Pi\}$ of $\upsilon_1(\Phi)$ as follows. First we define $\overline{\mathcal{R}}_{f_j} = \mathcal{R}'_{(F_j;A_{P_i}?;(F'_j;A_{P_i}?)^*)^-}$

[15] Without loss of generality, we implicitly assume that each state $x \in \mathcal{S}$ has a countable number of f_i^--successors, $i = 1, 2$.

$(j = 1, 2)$. Then let $s \in S'$ be such that $M', s \models \upsilon_2(\Phi)$, we define

$$\begin{array}{l} \mathcal{S} = \{t \mid (s,t) \in (\overline{\mathcal{R}}_{f_1} \cup \overline{\mathcal{R}}_{f_2} \cup \overline{\mathcal{R}}_{f_1}^{-} \cup \overline{\mathcal{R}}_{f_2}^{-})^*\}, \\ \mathcal{R}_{f_j} = \overline{\mathcal{R}}_{f_j} \cap (\mathcal{S} \times \mathcal{S}), \\ \Pi(t) = \Pi'(t) \quad \text{for all } t \in S. \end{array}$$

Note that, by Θ_2, $\mathcal{R}'_{(F_j;A_{P_i}?;(F'_j;A_{P_i}?)^*)^-}$ is a partial function, and hence $\mathcal{R}_{f_j}$ is a partial function as well, thus Θ_1 is satisfied all over M.

Considering again the meaning of

$$\begin{array}{l} [F_j; A_{P_i}?; (F'_j; A_{P_i}?)^*; (\phi?; (F'_j; A_{P_i}?)^+)^n]\neg\phi \\ < F_j; A_{P_i}?; (F'_j; A_{P_i}?)^*; (\phi?; (F'_j; A_{P_i}?)^+)^{n-1} > \phi \end{array}$$

it is easy to verify by induction on $\upsilon'_0(\Phi)$ that $M', s \models \upsilon'_0(\Phi)$ if and only if $M, s \models \upsilon_0(\Phi)$. □

Now we are ready to state the main results of this section.

Theorem 22 *A formula Φ of $\mathcal{DIN}$ is satisfiable if and only if the formula $\upsilon_2(\Phi)$ of $\mathcal{DIF}$ is satisfiable.*

Proof By Lemma 19 and Lemma 21. □

Theorem 23 *Satisfiability in $\mathcal{DIN}$ is an EXPTIME-complete problem.*

Proof The satisfiability problem for $\mathcal{DIF}$ is EXPTIME-complete as shown in Chapter 3, and, by Lemma 20 the size of the $\mathcal{DIF}$-counterpart $\upsilon_2(\Phi)$ of a $\mathcal{DIN}$ formula Φ is polynomially related to the size of Φ. □

As an immediate consequence we can characterize the computational complexity of reasoning in $\mathcal{CIN}$.

Theorem 24 *Satisfiability of $\mathcal{CIN}$ concepts, satisfiability of $\mathcal{CIN}$ TBoxes, and logical implication in $\mathcal{CIN}$ TBoxes, are EXPTIME-complete problems.*

4.4 Discussion

Let us illustrate with an example the basic relationships between models of $\mathcal{DIN}$ formulae and those of their reified-counterparts and $\mathcal{DIF}$-counterparts.

Consider the following $\mathcal{DIN}$ formula:

$$\Phi = < P > (= 2\ P^-.(= 2\ P.\top))$$

where the notation $(= n\ a.\phi)$ stands for $(\leq n\, a.\phi) \wedge (\geq n\, a.\phi)$.

Figure 4.1 shows a model M of Φ such that $M, a \models \Phi$.

In Figure 4.2 the model M of Φ is transformed in a model M' of its reified-counterpart $\upsilon_1(\Phi)$ as done in the proof of Lemma 19.

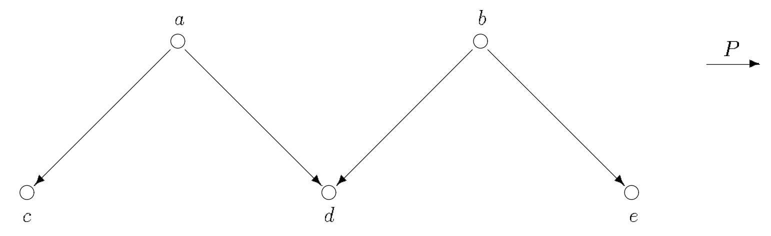

Figure 4.1: A model of a $\mathcal{DIF}$ formula Φ

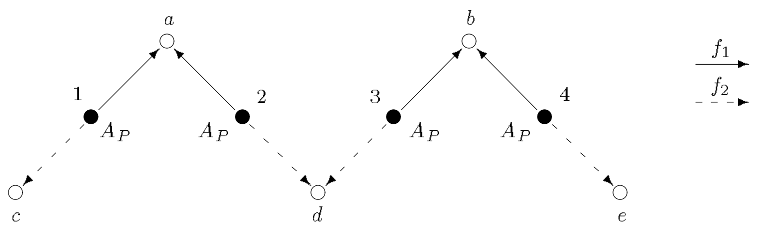

Figure 4.2: A model of the reified-counterpart $\upsilon_1(\Phi)$ of Φ

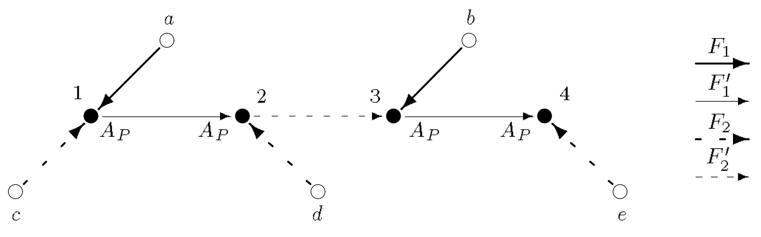

Figure 4.3: A model of the $\mathcal{DIF}$-counterpart $\upsilon_2(\Phi)$ of Φ

Finally, in Figure 4.3 the model M' of $\upsilon_1(\Phi)$ is transformed into a model M'' of the $\mathcal{DIF}$-counterpart $\upsilon_2(\Phi)$ of Φ as in the proof of Lemma 21. Notice that from M'' we can easily reconstruct M', and from it the model M of the original formula.

The other direction, transforming models of $\upsilon_2(\Phi)$ first into models of $\upsilon_1(\Phi)$ and then into models of Φ is slightly more involved in general, for two reasons. First we are not imposing any explicit distinction between states and pseudo states in both models of $\upsilon_1(\Phi)$ and $\upsilon_2(\Phi)$. Indeed nothing prevents a state, which is not intended to represent a relation, from satisfying $< f_j > \top$ in models of $\upsilon_1(\Phi)$ and similarly $< (F_j; A_P?; (F'_j; A_P?)^*)^- > \top$ in models of $\upsilon_2(\Phi)$. However starting from a state satisfying $\upsilon_1(\Phi)$ (respectively $\upsilon_2(\Phi)$) we can isolate the component, connected by means of programs $f_1^-; A_P?; f_2$ (respectively $F_1; A_P?; (F'_1; A_P?)^*; (F_2; A_P?; (F'_2; A_P?)^*)^-$) or their converse. In such a connected component, formulae $< f_j > \top$ (respectively $< F_j; A_P?; (F'_j; A_P?)^* > \top$) are satisfied only by states that are intended to represent a relation. The second difficulty is that in general models of $\upsilon_1(\Phi)$ may contain more pseudo states referring to the same tuple of a relation, however by Lemma 18 we can restrict our attention to models in which this difficulty does not arise, without loss of generality.

We remark that the only condition required by the proof of Lemma 18 is that, given a model of a formula, the disjoint union of copies of this model is still a model. This condition is very general, and most modal logics satisfy it. Note, however, that in the following we will introduce a family of propositional dynamic logics in which it is possible to denote a property satisfied by exactly one state. Because of this, such logics violate the condition above.

We end the chapter with a few words about the tight relation between qualified number restriction and graded modalities in modal logic [128, 127, 54, 55]. The graded modal operator $< a >_n \phi$ (with $n \geq 0$) is equivalent to the qualified number restriction, $(\geq n+1\, a.\phi)$, and its dual $[a]_n\phi = \neg < a >_n \neg\phi$ is equivalent to $(\leq n\, a.\phi)$ for $n \geq 1$ and to $[a]\phi$ for $n = 0$. The decidability and computational complexity of a propositional dynamic logic comprising graded modal operators on atomic programs and converse of atomic programs, were not known. Since such logic is straightforwardly polynomially reducible to $\mathcal{DIN}$, as an immediate consequence of the results on $\mathcal{DIN}$, we can state its decidability and characterize its computational complexity as *EXPTIME*-complete.[16]

[16] Note that even for the much simpler basic PDL augmented with graded modalities on atomic actions, decidability and computational complexity were not known. Such logic is essentially $\mathcal{DN}$, whose decidability and computational complexity has been discussed in the section on reasoning in $\mathcal{CN}$ and $\mathcal{DN}$.

Chapter 5

Boolean Properties and Assertions on Atomic Roles

In this chapter we add to $\mathcal{CIN}$ the possibility of expressing boolean combinations of atomic roles, in particular the intersection of atomic roles $P_1 \sqcap P_2$, and the negation of atomic roles $\neg P$ interpreted as "any role but P". We also allow for stating inclusion assertions on atomic roles, thus expressing hierarchies of roles, disjointness of roles, etc. The corresponding propositional dynamic logic is introduced and studied at the same time.

5.1 The logics $\mathcal{CINB}$ and $\mathcal{DINB}$

The abstract syntax of the description logic $\mathcal{CINB}$ is as follows:

$$\begin{array}{lll} C & ::= & \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid C_1 \Rightarrow C_2 \mid \neg C \mid \\ & & \exists R.C \mid \forall R.C \mid (\leq n\, a.C) \mid (\geq n\, a.C) \\ \\ \rho & ::= & \mathbf{any} \mid P \mid \rho_1 \sqcap \rho_2 \mid \rho_1 \sqcup \rho_2 \mid \rho_1 \Rightarrow \rho_2 \mid \neg \rho \\ a & ::= & \rho \mid \rho^- \\ R & ::= & a \mid R_1 \sqcup R_2 \mid R_1 \circ R_2 \mid R^* \mid R^- \mid id(C) \end{array}$$

where A denotes an atomic concept, C (possibly with a subscript) a generic concept, **any** "the most general" atomic role, P an atomic role, ρ a *basic role*, i.e. a boolean combination of atomic roles, a a simple role, i.e. either a basic role or the inverse of a basic role, R (possibly with a subscript) a generic role. Note that wrt $\mathcal{CIN}$ qualified number restrictions are extended from atomic roles and their inverse to basic roles and their inverse.

The semantics of $\mathcal{CINB}$ is similar to that of $\mathcal{CIN}$ except for the basic roles which are not present in $\mathcal{CIN}$. To such roles an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ assigns meaning as follows:

$$\begin{array}{l}
\mathbf{any}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \\
P^{\mathcal{I}} \subseteq \mathbf{any}^{\mathcal{I}} \\
(\rho_1 \sqcap \rho_2)^{\mathcal{I}} = \rho_1^{\mathcal{I}} \cap \rho_2^{\mathcal{I}} \\
(\rho_1 \sqcup \rho_2)^{\mathcal{I}} = \rho_1^{\mathcal{I}} \cup \rho_2^{\mathcal{I}} \\
(\rho_1 \Rightarrow \rho_2)^{\mathcal{I}} = \neg\rho_1^{\mathcal{I}} \cup \rho_2^{\mathcal{I}} \\
\neg\rho^{\mathcal{I}} = \mathbf{any}^{\mathcal{I}} - \rho^{\mathcal{I}}.
\end{array}$$

Observe that $\mathbf{any}^{\mathcal{I}} \neq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, in general. As a consequence $\neg P$ is to be interpreted as "the set of pairs of individuals that are *linked (by* **any***) but not by* P" ($\neg P^{\mathcal{I}} = \mathbf{any}^{\mathcal{I}} - P^{\mathcal{I}}$) as opposed to "the set of pairs of individuals that are *not linked by* P" ($\neg P^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} - P^{\mathcal{I}}$). The following example should further clarify the difference between the two interpretations. The concept $\forall\neg P.C$, wrt the first interpretation, means: " the class of individuals such that all their successors, that are not P-successors, are in C"; wrt the second interpretation, it means: "the class of individuals such that all individuals, that are not their P-successors, are in C". We shall return to this point at the end of the chapter.

The basic role $P_1 \sqcap P_2$ denotes the intersection of P_1 and P_2, i.e. the pairs of individuals that are both in P_1 and in P_2. So, for example, $\exists P_1 \sqcap P_2.C$ denotes individuals which have a P_1-successor in C that is also a P_2 successor. Similarly $(\leq n\, P_1 \sqcap P_2.C)$ denotes individuals which have at most n P_1-successors in C that are also P_2 successors. While $\forall P_1 \sqcap P_2.C$ denotes individuals of whose P_1-successors that are also P_2-successors are in C.

Differently from what is usually assumed, we allow for directly specifying interdependencies between basic roles. In other words, besides of inclusion assertions on concepts, $\mathcal{CINB}$ TBoxes allow for inclusion assertions on basic roles. Analogously we are also interested in checking for subsumption between basic roles wrt a TBox. Simple examples of inclusion assertions on basic roles are:

$$\begin{array}{l}
father \sqsubseteq parent \\
mother \sqsubseteq parent \\
parent \sqsubseteq mother \sqcup father \\
father \sqsubseteq \neg mother
\end{array}$$

specifying that both the roles *father* and *mother* are specializations of the role *parent*, that *parent* is in turn a specialization of $mother \sqcup father$, and that *father* and *mother* are disjoint, i.e. their intersection is empty.[1] Obviously inclusion assertions on roles must be taken into account in logical inference: for example, from $father \sqsubseteq parent$ and $human \sqsubseteq \forall parent.human$ we infer $human \sqsubseteq \forall father.human$.

Formally we define a $\mathcal{CINB}$ TBox to be a set of inclusion assertions both on concepts ($C_1 \sqsubseteq C_2$) and on basic roles ($\rho_1 \sqsubseteq \rho_2$). As usual we say that an interpretation $\mathcal{I}$ is a model of an inclusion assertion on concepts $C_1 \sqsubseteq C_2$, if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. Similarly we say that an interpretation $\mathcal{I}$ is a model of an inclusion assertion on basic roles $\rho_1 \sqsubseteq \rho_2$, if $\rho_1^{\mathcal{I}} \sqsubseteq \rho_2^{\mathcal{I}}$. We say that an interpretation $\mathcal{I}$ is a model of a TBox, if it is a model of all inclusion assertions in it (both on concepts and basic roles). We say

[1]Note that these inclusion assertions affirm that the role *parent* is partitioned into the roles *father* and *mother*.

that a TBox $\mathcal{K}$ logically implies an inclusion assertion on concepts or on basic roles, written as $\mathcal{K} \models C_1 \sqsubseteq C_2$, $\mathcal{K} \models \rho_1 \sqsubseteq \rho_2$ respectively, if all models of $\mathcal{K}$ are models of the inclusion assertion.

We also introduce the notion of satisfiability for basic roles, besides the usual satisfiability of concepts. A basic role ρ is satisfiable, if there exists an interpretation $\mathcal{I}$ such that $\rho^{\mathcal{I}} \neq \emptyset$.

The corresponding propositional dynamic logic is called $\mathcal{DINB}$. Formulae of $\mathcal{DINB}$ are of two sorts: *program formulae* and *state formulae*.

Program formulae are boolean combinations of atomic programs and their syntax is as follows:

$$\rho ::= \mathbf{any} \mid P \mid \rho_1 \cap \rho_2 \mid \rho_1 \cup \rho_2 \mid \rho_1 \Rightarrow \rho_2 \mid \neg\rho$$

where **any** is "the most general" atomic program, P an atomic program, and ρ a generic program formula also called *basic program.*

State formulae (the usual sort of propositional dynamic logic formulae), describing property of states, have the following abstract syntax:

$$\begin{array}{lcl} \phi & ::= & \top \mid \bot \mid A \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \Rightarrow \phi_2 \mid \neg\phi \mid \\ & & < r > \phi \mid [r]\phi \mid (\leq n\, a.\phi) \mid (\geq n\, a.\phi) \\ & & \\ a & ::= & \rho \mid \rho^- \\ r & ::= & a \mid r_1 \cup r_2 \mid r_1; r_2 \mid r^* \mid r^- \mid \phi? \end{array}$$

where A denotes a propositional letter, ϕ (possibly with a subscript) a state formula, ρ a basic program, i.e. a boolean combination of atomic programs, a a *simple program*, i.e. a basic program or the converse of an basic program, and r (possibly with a subscript) a generic program.

Consistently with the interpretation of basic roles in $\mathcal{CINB}$, basic programs are interpreted as follows: for all structures $M = (\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$,

$$\begin{array}{l} \mathcal{R}_{\mathbf{any}} \subseteq \mathcal{S} \times \mathcal{S} \\ \mathcal{R}_P \subseteq \mathcal{R}_{\mathbf{any}} \\ \mathcal{R}_{\rho_1 \cap \rho_2} = \mathcal{R}_{\rho_1} \cap \mathcal{R}_{\rho_1} \\ \mathcal{R}_{\rho_1 \cup \rho_2} = \mathcal{R}_{\rho_1} \cup \mathcal{R}_{\rho_2} \\ \mathcal{R}_{\rho_1 \Rightarrow \rho_2} = \mathcal{R}_{\neg\rho_1} \cup \mathcal{R}_{\rho_2} \\ \mathcal{R}_{\neg\rho} = \mathcal{R}_{\mathbf{any}} - \mathcal{R}_{\rho}. \end{array}$$

The rest of the constructs are interpreted as in $\mathcal{DIN}$.

Intuitively a program $P_1 \cap P_2$ denotes the *concurrent execution* of P_1 and P_2, while $\neg P$ denotes the *non-execution* of P. In general program formulae (basic programs) denote a set of atomic programs executed concurrently and a set of atomic programs not executed at all. Note that nothing is said about atomic programs that are not contained in one of these sets, they could be executed or not, i.e. we are adopting an open semantics for program formulae.

By forcing validity of program formulae (which correspond to state inclusion assertions on roles in $\mathcal{CINB}$) we can represent hierarchies of basic programs, for example by forcing validity of $resize\_icon \Rightarrow resize\_picture$ we can represent that $resize\_icon$ is a specialization of $resize\_picture$. In the same way we can represent *mutual exclusion*, for example by forcing the validity of $\neg(open\_window \wedge close\_window)$ we represent that the program $open\_window$ and $close\_window$ cannot be executed together.

Formally, we say that a program formula ρ is valid in a structure $M = (\mathcal{S}, \{R_r\}, \Pi)$, if $\mathcal{R}_\rho = \mathcal{R}_{\mathbf{any}}$, while a state formula ϕ is valid in M, if for all $s \in S$, $M, s \models \phi$. We call *axioms* formulae (either program or a state formulae) that are assumed to be valid. Formally, we say that a structure M is a model of an axiom ψ, if ψ is valid in M. We say that an axiom is satisfiable, if it has a model. We say that a structure M is a model of a finite set of axioms Γ, if M is a model of all axioms in Γ. We say that a finite set of axioms is satisfiable, if it has a model. We say that a finite set Γ of axioms logically implies a formula ψ (either program or state formula), written $\Gamma \models \psi$,if ψ is valid in every model of Γ.

Note that $\mathcal{CINB}$ inclusion assertions are analogue to $\mathcal{DINB}$ valid formulae, and $\mathcal{CINB}$ TBoxes are analogue to sets of $\mathcal{DINB}$ axioms. Hence satisfiability of $\mathcal{CINB}$ TBoxes correspond to satisfiability of finite sets of $\mathcal{DINB}$ axioms,and logical implication in $\mathcal{CINB}$ corresponds to logical implication in $\mathcal{DINB}$.

We also introduce the notion of satisfiability for program formula (the analogue of the notion of satisfiability for a basic role). A program formula ρ is satisfiable, if there exists a structure $M = (\mathcal{S}, \{\mathcal{R}_r\}, \Pi)$ such that $\mathcal{R}_\rho \neq \emptyset$.

5.2 Reasoning in $\mathcal{CINB}$ and $\mathcal{DINB}$

The $\mathcal{CINB}$ reasoning services we are interested in are satisfiability of concepts, satisfiability of basic roles, satisfiability of TBoxes, and logical implication in TBoxes.

It is easy to check that satisfiability of basic roles is reducible to satisfiability in propositional logic, hence it is computational characterized as NP-complete.[2] The other reasoning services are EXPTIME-hard since $\mathcal{CINB}$ contains $\mathcal{CIN}$, and their decidability and computational complexity is to be established yet.

We can derive such results for $\mathcal{CINB}$ by exploiting the correspondence between $\mathcal{CINB}$ and $\mathcal{DINB}$. The correspondence is realized by suitably extending the mapping δ in Chapter 2 to deal with basic roles and qualified number restrictions. Note that the mapping δ^+, reducing logical implication in the description logic to satisfiability in the correspondent propositional dynamic logic, cannot be extended because of the presence of inclusion assertions on basic roles. Hence logical implication in $\mathcal{CINB}$ is directly mapped, by the extension of δ, to logical implication in $\mathcal{DINB}$ and vice versa.

In the following we concentrate on logical implication in $\mathcal{DINB}$ [3], proving that it can be polynomially reduced to logical implication in $\mathcal{DIN}$ (which in turn can be reduced to satisfiability of a single formula as usual). As corollaries of this result we establish decidability of both $\mathcal{CINB}$ and $\mathcal{DINB}$ and characterize the computational

[2] Similarly, satisfiability of $\mathcal{DINB}$ program formulae is NP-complete.

[3] Observe that satisfiability of a state formula ϕ as well as satisfiability of a finite set of axioms Γ can be reformulated by means of logical implication as $\emptyset \not\models \neg\phi$ and $\Gamma \not\models \bot$, respectively.

complexity of satisfiability and logical implication in both the logics as EXPTIME-complete.

The reduction from logical implication in $\mathcal{DINB}$ to logical implication in $\mathcal{DIN}$ is based on reifying relations associated with basic programs, similarly to what was done in Chapter 4. Intuitively, the key idea underlying the reduction is to represent each pair of states (x, y) associated with the program **any** by a (pseudo) state z_{xy}, introducing two deterministic programs f_1 and f_2 linking each pseudo state z_{xy} to the first component x and the second component y of the corresponding pair (x, y). In this way, we can translate program formulae that hold for a pair (x, y) into state formulae that hold in the corresponding pseudo state z_{xy}.

We start presenting the reduction, by defining two mappings: τ_p from $\mathcal{DINB}$ program formulae to $\mathcal{DIN}$ (state) formulae, and τ_s from $\mathcal{DINB}$ state formulae to $\mathcal{DIN}$ formulae.

Definition We define a mapping τ_p from $\mathcal{DINB}$ program formulae ρ to $\mathcal{DIN}$ (state) formulae $\tau_p(\rho)$ as follows:

$$\begin{aligned}
&\tau_p(\mathbf{any}) = \top_{\mathbf{any}}\\
&\tau_p(P) = \top_{\mathbf{any}} \wedge A_P\\
&\tau_p(\rho_1 \cap \rho_1) = \tau_p(\rho_1) \wedge \tau_p(\rho_2)\\
&\tau_p(\rho_1 \cup \rho_1) = \tau_p(\rho_1) \vee \tau_p(\rho_2)\\
&\tau_p(\rho_1 \Rightarrow \rho_1) = \tau_p(\rho_1) \Rightarrow \tau_p(\rho_2)\\
&\tau_p(\neg\rho) = \top_{\mathbf{any}} \wedge \neg\tau_p(\rho)
\end{aligned}$$

where $\top_{\mathbf{any}}$ and A_P are new propositional letters. □

Note that $\tau_p(P) \Rightarrow \tau_p(\mathbf{any})$ is equivalent to $(\top_{\mathbf{any}} \wedge A_P) \Rightarrow \top_{\mathbf{any}}$ which is an instance of a propositional tautology.

Definition We define a mapping τ_s from $\mathcal{DINB}$ state formulae ϕ to $\mathcal{DIN}$ formulae $\tau_s(\phi)$ as follows:

$$\begin{aligned}
&\tau_s(\top) = \top_s\\
&\tau_s(\bot) = \neg\top_s\\
&\tau_s(A) = A\\
&\tau_s(\phi_1 \wedge \phi_2) = \tau_s(\phi_1) \wedge \tau_s(\phi_2)\\
&\tau_s(\phi_1 \vee \phi_2) = \tau_s(\phi_1) \vee \tau_s(\phi_2)\\
&\tau_s(\neg\phi) = \neg\tau_s(\phi)\\
&\tau_s([r]\phi) = [\tau_s'(r)]\tau_s(\phi)\\
&\tau_s(< r > \phi) =< \tau_s'(r) > \tau_s(\phi)\\
&\tau_s((\cdot\ n\ \rho.\phi)) = (\cdot\ n\ f_1^-. < \tau_p(\rho?); f_2 > \tau_s(\phi))\\
&\tau_s((\cdot\ n\ \rho^-.\phi)) = (\cdot\ n\ f_2^-. < \tau_p(\rho?); f_1 > \tau_s(\phi))
\end{aligned}$$

where f_1 and f_2 are two new atomic programs, and τ_s' is a mapping from $\mathcal{DINB}$

programs r to $\mathcal{DIN}$ programs $\tau'_s(r)$ defined as follows:

$$\begin{array}{l}
\tau'_s(\rho) = f_1^-; \tau_p(\rho); f_2 \\
\tau'_s(\rho^-) = f_2^-; \tau_p(\rho); f_1 \\
\tau'_s(r_1; r_2) = \tau'_s(r_1); \tau'_s(r_2) \\
\tau'_s(r_1 \cup r_2) = \tau'_s(r_1) \cup \tau'_s(r_2) \\
\tau'_s(r^*) = \tau'_s(r)^* \\
\tau'_s(\phi?) = \tau'_s(\phi)? \\
\tau'_s(r^-) = \tau'_s(r)^-.
\end{array}$$

□

Note that f_1 and f_2 are the only atomic programs occurring in $\tau_s(\phi)$. Note also that $\mathcal{DINB}$ basic programs are transformed into their *reified form*, similarly to what was shown in Chapter 4.

Making use of the above mappings, we define a mapping τ from $\mathcal{DINB}$ formulae to $\mathcal{DIN}$ formulae, and a mapping T from finite sets of $\mathcal{DINB}$ axioms to finite sets of $\mathcal{DIN}$ axioms.

Definition Let ψ be a $\mathcal{DINB}$ formula. We define $\tau(\psi)$ as the following $\mathcal{DIN}$ formula:

- if ψ is a state formula, then $\tau(\psi) = \top_s \Rightarrow \tau_s(\psi)$
- if ψ is a program formula, then $\tau(\psi) = \top_{\mathbf{any}} \Rightarrow \tau_p(\psi)$.

□

Lemma 25 *Let ψ be a $\mathcal{DINB}$ formula, and τ the mapping defined above. Then $\tau(\psi)$ is a $\mathcal{DIN}$ formula, and its size is polynomially related to the size of ψ.*

Proof Straightforward. □

Definition Let Γ be a finite set of $\mathcal{DINB}$ axioms. We define $T(\Gamma)$ as the set $T_1(\Gamma) \cup T_2$ of $\mathcal{DIN}$ axioms, where:

- $T_1(\Gamma) = \{\tau(\psi) \mid \psi \in \Gamma\}$
- T_2 is the set composed by the following three axioms:

$$\begin{array}{l}
\top_{\mathbf{any}} \equiv \neg\top_s \wedge (< f_1 > \top_s) \wedge (< f_2 > \top_s) \\
(\leq 1\, f_1.\top) \\
(\leq 1\, f_2.\top).
\end{array}$$

□

Lemma 26 *Let Γ be a finite set of $\mathcal{DINB}$ axioms, and T the mapping defined above. Then $T(\Gamma)$ is a finite set of $\mathcal{DIN}$ axioms, and its size is polynomially related to the size of Γ.*

Proof Straightforward. $\square$

Intuitively, in the models of $T(\Gamma)$ we distinguish states satisfying $\top_s$, which represent states in the models of Γ, and (pseudo) states satisfying $\top_{\mathbf{any}}$, which represent pairs of states of **any** in the models of Γ. Such pseudo states have exactly one f_1-successor and one f_2-successor, both satisfying $\top_s$.

Next, by showing that models of a finite set Γ of $\mathcal{DINB}$ axioms can be transformed into models of $T(\Gamma)$, we prove the following lemma.

Lemma 27 *Let Γ be a finite set of $\mathcal{DINB}$ axioms, ψ a $\mathcal{DINB}$ formula, and T and τ the mappings above. Then $\Gamma \models \psi$ if $T(\Gamma) \models \tau(\psi)$.*

Proof By contradiction: suppose that $T(\Gamma) \models \tau(\psi)$ and there exists a model $M = (\mathcal{S}, \{R_r\}, \Pi)$ of Γ, in which ψ is not valid.

From M we define the structure $M' = (\mathcal{S}', \{\mathcal{R}'_{f_1}, \mathcal{R}'_{f_2}\}, \Pi')$ as follows:

- $\mathcal{S}' = \mathcal{S} \cup \{z_{xy} \mid (x,y) \in \mathcal{R}_{\mathbf{any}}\}$
- for each $(x,y) \in \mathcal{R}_{\mathbf{any}}$, we put $\top_{\mathbf{any}} \in \Pi'(z_{xy})$, $(z_{xy}, x) \in \mathcal{R}'_{f_1}$, and $(z_{xy}, y) \in \mathcal{R}'_{f_2}$
- for each P, for each $(x,y) \in \mathcal{R}_P$, we put $A_P \in \Pi'(z_{xy})$
- for each state $x \in \mathcal{S}$, we put $\Pi'(x) = \Pi(x) \cup \{\top_s\}$.

Observe that for each pair $(x,y) \in \mathcal{R}_{\mathbf{any}}$ there is exactly one element $z_{xy} \in \mathcal{S}'$. Moreover, by construction, M' is a model of T_2.

Next we prove that for all program formulae ρ, $(x,y) \in \mathcal{R}_\rho$ if and only if $(x,y) \in \mathcal{R}'_{f_1^-;\tau_p(\rho)?;f_2}$. We proceed by induction on the formation of ρ (without loss of generality we skip the cases $\rho = \rho_1 \cup \rho_2$ and $\rho = \rho_1 \Rightarrow \rho_2$).

- $(x,y) \in \mathcal{R}_{\mathbf{any}}$ iff $\top_{\mathbf{any}} \in \Pi'(z_{xy})$, $(z_{xy}, x) \in \mathcal{R}'_{f_1}$, $(z_{xy}, y) \in \mathcal{R}'_{f_2}$, by construction.
- $(x,y) \in \mathcal{R}_P$ iff $\top_{\mathbf{any}}, A_P \in \Pi'(z_{xy})$, $(z_{xy}, x) \in \mathcal{R}'_{f_1}$, $(z_{xy}, y) \in \mathcal{R}'_{f_2}$, by construction.
- $(x,y) \in \mathcal{R}_{\rho_1 \cap \rho_2}$ iff $(x,y) \in \mathcal{R}_{\rho_1}$ and $(x,y) \in \mathcal{R}_{\rho_2}$, iff (by induction hypothesis) $\tau_p(\rho_1) \in \Pi'(z_{xy})$, $(z_{xy}, x) \in \mathcal{R}'_{f_1}$, $(z_{xy}, y) \in \mathcal{R}'_{f_2}$, and $\tau_p(\rho_2) \in \Pi'(z_{xy})$, $(z_{xy}, x) \in \mathcal{R}'_{f_1}$, $(z_{xy}, y) \in \mathcal{R}'_{f_2}$, i.e. $\tau_p(\rho_1 \cap \rho_2) \in \Pi'(z_{xy})$, $(z_{xy}, x) \in \mathcal{R}'_{f_1}$, $(z_{xy}, y) \in \mathcal{R}'_{f_2}$.
- $(x,y) \in \mathcal{R}_{\neg\rho}$ iff $(x,y) \in \mathcal{R}_{\mathbf{any}}$ and $(x,y) \notin \mathcal{R}_\rho$, iff (by induction hypothesis) $\top_{\mathbf{any}} \in \Pi'(z_{xy})$, $(z_{xy}, x) \in \mathcal{R}'_{f_1}$, $(z_{xy}, y) \in \mathcal{R}'_{f_2}$, and $\tau_p(\rho) \notin \Pi'(z_{xy})$, $(z_{xy}, x) \in \mathcal{R}'_{f_1}$, $(z_{xy}, y) \in \mathcal{R}'_{f_2}$, i.e. $\tau_p(\neg\rho) \in \Pi'(z_{xy})$, $(z_{xy}, x) \in \mathcal{R}'_{f_1}$, $(z_{xy}, y) \in \mathcal{R}'_{f_2}$.

Observe that by definition of M', $(x,y) \in \mathcal{R}'_{f_1^-;\tau_p(\rho)?;f_2}$ if and only if $M', z_{xy} \models \tau_p(\rho)$.

Proceeding again by induction, it is easy to prove that for all state formulae ϕ and all $x \in \mathcal{S}$, $M, x \models \phi$ if and only if $M', x \models \tau_s(\phi)$. For example, let us consider a state formula of the form $(\leq n\, \rho.\phi)$. By definition, $M, x \models (\leq n\, \rho.\phi)$ if there are at most n states y such that $(x,y) \in \mathcal{R}_\rho$ and $M, y \models \phi$. We have already proved that $(x,y) \in \mathcal{R}_\rho$ if and only if $(x,y) \in \mathcal{R}'_{f_1^-;\tau_p(\rho)?;f_2}$, and by inductive hypothesis we can assume $M, y \models \phi$ iff $M', y \models \tau_s(\phi)$. Considering that f_2 is functional we have, $\{y \mid (x,y) \in \mathcal{R}'_{f_1^-;\tau_p(\rho)?;f_2}$ and $M', y \models \tau_s(\phi)\}$ is equal to $\{y \mid (x, z_{xy}) \in \mathcal{R}'_{f_1^-}, (z_{xy}, y) \in \mathcal{R}'_{f_2}$ and $M', z_{xy} \models < \tau_p(\rho)?; f_2 > \tau_s(\phi)\}$. Thus we conclude that $M, x \models (\leq n\, \rho.\phi)$ if and only if $M', x \models (\leq n\, f_1^-. < \tau_p(\rho)?; f_2 > \tau_s(\phi))$.

Let ϕ be a $\mathcal{DINB}$ state formula, ϕ is valid in M if and only if for all $s \in \mathcal{S}$, $M, s \models \phi$. Considering the definition of M' this holds, if and only if, for all $s \in \mathcal{S}'$ such that $M', s \models \top_s$, we have $M', s \models \tau_s(\phi)$, i.e. if and only if $\tau(\phi)$ is valid in M'.

Similarly, let ρ be a $\mathcal{DINB}$ program formula, ρ is valid in M if and only if for all $x, y \in \mathcal{S}$ such that $(x,y) \in \mathcal{R}_{\mathbf{any}}$, we have $(x,y) \in \mathcal{R}_\rho$. Considering again the definition of M', this holds if and only if, for all $z \in \mathcal{S}'$ such that $z \in \top_{\mathbf{any}}$, we have $z \in \tau_p(\rho)$, i.e. if and only if $\tau(\rho)$ is valid in M'.

Hence M' is a model of $T(\Gamma)$ and yet $\tau(\psi)$ is not valid in M', contradicting the hypothesis. $\Box$

In order to prove the converse of Lemma 27, we show that, given a model $M = (\mathcal{S}, \{\mathcal{R}_{f_1}, \mathcal{R}_{f_2}\}, \Pi)$ of $T(\Gamma)$, we can construct a model of Γ on the basis of M. However such construction can be carried out only starting from models of $T(\Gamma)$ satisfying the following condition: for each pair $(x,y) \in \mathcal{R}_{f_1^-;\tau_p(\rho)?;f_2^-}$, there is a single pseudo state z_{xy} such that $(z_{xy}, x) \in \mathcal{R}_{f_1}$ and $(z_{xy}, y) \in \mathcal{R}_{f_2}$. The next lemma guarantees that we can assume such condition to be satisfied in the models of $T(\Gamma)$, without loss of generality.

Lemma 28 *Let Γ be a finite set of $\mathcal{DINB}$ axioms, and T the mapping defined above. If $T(\Gamma)$ has a model $M = (\mathcal{S}, \{\mathcal{R}_{f_1}, \mathcal{R}_{f_2}\}, \Pi)$, then it has a model $M' = (\mathcal{S}', \{\mathcal{R}'_{f_1}, \mathcal{R}'_{f_2}\}, \Pi)$ such that, for each $(x,y) \in \mathcal{R}'_{f_1^-;\tau_p(\rho)?;f_2}$ there is exactly one z_{xy} such that $(z_{xy}, x) \in \mathcal{R}'_{f_1}$ and $(z_{xy}, y) \in \mathcal{R}'_{f_2}$. That is, for all $z_1, z_2, x, y \in \mathcal{S}'$ such that $z_1 \neq z_2$ and $x \neq y$, the following condition holds:*

$$M', z_1 \models \tau_p(\rho) \text{ and } M', z_2 \models \tau_p(\rho) \text{ implies}$$
$$\neg((z_1, x) \in \mathcal{R}'_{f_1} \wedge (z_2, x) \in \mathcal{R}'_{f_1} \wedge (z_1, y) \in \mathcal{R}'_{f_2} \wedge (z_2, y) \in \mathcal{R}'_{f_2}).$$

Proof The proof is almost identical to that of Lemma 18 in Chapter 4. We sketch it here for completeness. Suppose that the condition is not already satisfied by M. We show how to build a model M' in which the condition above is satisfied. Given an pseudo state z referring to a pair $(x,y) \in \mathcal{R}_{f_1^-;\tau_p(\rho)?;f_2}$ we denote x by $f_1(z)$ and y by $f_2(z)$ (this is in agreement with $\mathcal{R}_{f_1}$ and $\mathcal{R}_{f_2}$ being functional). We call the presence of more pseudo states referring to the same pair, *conflict*. Let $(x,y) \in \mathcal{R}_{f_1^-;\tau_p(\rho)?;f_2}$. If there is more than one pseudo state z referring to (x,y), then we randomly choose one such pseudo state to represent (x,y) and we say that the others *induce a conflict*. We call $Conf$ the set of all pseudo states inducing a conflict. Note that $Conf$ can be uncountable.

We define a structure $M_{2^{Conf}}$ as the disjoint union of $|2^{Conf}|$ copies of M, one copy, denoted by $M^{\mathcal{E}}$, for every set $\mathcal{E} \in 2^{Conf}$. We denote by $s^{\mathcal{E}}$ the copy in $M^{\mathcal{E}}$ of the state s in M. Trivially, $M_{2^{Conf}}$ is a model of $T(\Gamma)$ as is M.

Let $M^{\mathcal{E}}$ and $M^{\mathcal{E}'}$ be two copies of M in $M_{2^{Conf}}$, we call "exchanging $f_2(t^{\mathcal{E}})$ with $f_2(t^{\mathcal{E}'})$" the operation on $M_{2^{Conf}}$ consisting of removing the pair $(t^{\mathcal{E}}, f_2(t^{\mathcal{E}}))$ from $\mathcal{R}^{\mathcal{E}}_{f_2}$ replacing it with $(t^{\mathcal{E}}, f_2(t^{\mathcal{E}'}))$ and, at the same time, removing $(t^{\mathcal{E}'}, f_2(t^{\mathcal{E}'}))$ from $\mathcal{R}^{\mathcal{E}'}_{f_2}$ replacing it with $(t^{\mathcal{E}'}, f_2(t^{\mathcal{E}}))$. By exchanging $f_2(t^{\mathcal{E}})$ with $f_2(t^{\mathcal{E}'})$, we resolve t for both $M^{\mathcal{E}}$ and $M^{\mathcal{E}'}$, in the sense that $t^{\mathcal{E}}$ and $t^{\mathcal{E}'}$ no longer induce conflicts.

We get a model M' with the desired property by modifying $M_{2^{Conf}}$ as follows: for each state $t \in Conf$, for each $\mathcal{E} \in 2^{Conf}$ such that $t \in \mathcal{E}$, we exchange $f_2(t^{\mathcal{E}})$ with $f_2(t^{\mathcal{E}-\{t\}})$.

Indeed, proceeding in this way, on the one hand all conflicts present in the original model M are eliminated from all its copies in $M_{2^{Conf}}$. On the other hand no new conflicts are created.

Finally, M' is a model of $T(\Gamma)$, since by construction T_2 is valid in M', and it is straightforward to check by induction that for every $\phi \in T_1(\Gamma)$, for all $\mathcal{E} \in 2^{Conf}$, $M, s \models \phi$ if and only if $M', s^{\mathcal{E}} \models \phi$. $\Box$

Lemma 29 *Let Γ be a finite set of $\mathcal{DINB}$ axioms, ψ a $\mathcal{DINB}$ formula, and T and τ the mappings defined above. Then $T(\Gamma) \models \tau(\psi)$ if $\Gamma \models \psi$.*

Proof By contradiction: suppose that $\Gamma \models \psi$ and there exists a model $M' = (\mathcal{S}', \{\mathcal{R}'_{f_1}, \mathcal{R}'_{f_2}\}, \Pi')$ of $T(\Gamma)$ such that $\tau(\psi)$ is not valid. As a consequence of Lemma 28, we can assume that, in M', for each pair $(x, y) \in \mathcal{R}'_{f_1^-;\top_{\mathbf{any}}?;f_2}$ there exists a single pseudo state z_{xy} such that $(z_{xy}, x) \in \mathcal{R}'_{f_1}$ and $(z_{xy}, y) \in \mathcal{R}'_{f_2}$.

From M' we define a structure $M = (\mathcal{S}, \{R_r\}, \Pi)$ as follows:

- $\mathcal{S} = \{s \in \mathcal{S}' \mid M', s \models \top_s\}$
- $(x, y) \in \mathcal{R}_{\mathbf{any}}$ if and only if $(x, y) \in \mathcal{R}'_{f_1^-;\top_{\mathbf{any}}?;f_2}$, and similarly for all atomic programs P, $(x, y) \in \mathcal{R}_P$ if and only if $(x, y) \in \mathcal{R}'_{f_1^-;(\top_{\mathbf{any}} \wedge A_P)?;f_2}$ (note that $x, y \in \top_s$ by definition of $T(\Gamma)$)
- $\Pi(x) = \Pi'(x)$, for all $x \in \mathcal{S}$.

We prove that for all program formulae ρ, $(x, y) \in \mathcal{R}'_{f_1^-;\tau_p(\rho)?;f_2}$ if and only if $(x, y) \in \mathcal{R}_\rho$. We proceed by induction on the formation of ρ (without loss of generality we skip the cases $\rho = \rho_1 \cup \rho_2$ and $\rho = \rho_1 \Rightarrow \rho_2$).

- $(x, y) \in \mathcal{R}_{\mathbf{any}}$ if and only if $(x, y) \in \mathcal{R}'_{f_1^-;\top_{\mathbf{any}}?;f_2}$, by construction of M.
- $(x, y) \in \mathcal{R}_P$ if and only if $(x, y) \in \mathcal{R}'_{f_1^-;(\top_{\mathbf{any}} \wedge A_P)?;f_2}$, by construction of M.
- $(x, y) \in \mathcal{R}'_{f_1^-;\tau_p(\rho_1 \cap \rho_2)?;f_2}$ iff $(x, y) \in \mathcal{R}'_{f_1^-;\tau_p(\rho_1)?;f_2}$ and $(x, y) \in \mathcal{R}'_{f_1^-;\tau_p(\rho_2)?;f_2}$ iff by inductive hypothesis $(x, y) \in \mathcal{R}_{\rho_1}$ and $(x, y) \in \mathcal{R}_{\rho_2}$, i.e. $(x, y) \in \mathcal{R}_{\rho_1 \cap \rho_2}$.
- $(x, y) \in \mathcal{R}'_{f_1^-;\tau_p(\neg\rho)?;f_2}$, i.e. $(x, y) \in \mathcal{R}'_{f_1^-;\top_{\mathbf{any}}?;f_2}$ and $(x, y) \notin \mathcal{R}'_{f_1^-;\tau_p(\rho)?;f_2}$ iff by inductive hypothesis $(x, y) \in \mathcal{R}_{\mathbf{any}}$ and $(x, y) \notin \mathcal{R}_\rho$, i.e. $(x, y) \in \mathcal{R}_{\neg\rho}$.

Observe that $(x,y) \in \mathcal{R}'_{f_1^-;\tau_p(\rho)?;f_2}$ if and only if the single (pseudo) state z_{xy}, such that $(z_{xy},x) \in \mathcal{R}'_{f_1}$ and $(z_{xy},y) \in \mathcal{R}'_{f_2}$, satisfies $\tau_p(\rho)$.

It is easy to prove by induction that, for all state formulae ϕ, for all states $x \in \mathcal{S}$ $M,x \models \phi$ if and only if $M',x \models \tau_s(\phi)$.

Let ϕ be a $\mathcal{DINB}$ state formula. Then $\tau(\phi)$ is valid in M' if and only if for all $s \in \mathcal{S}'$, $M',s \models \top_s \Rightarrow \tau_s(\phi)$. Considering the definition of M this holds if and only if, for all $s \in \mathcal{S}$, $M,s \models \phi$, i.e. if and only if ϕ is valid in M.

Similarly, let ρ be a program formula. Then $\tau(\rho)$ is valid in M' if and only if for all $s \in \mathcal{S}'$ $M',s \models \top_{\mathbf{any}} \Rightarrow \tau_p(\rho)$, i.e. if and only if for all $(x,y) \in \mathcal{R}'_{f_1^-;\top_{\mathbf{any}}?;f_2}$, we have $(x,y) \in \mathcal{R}'_{f_1^-;\tau_p(\rho)?;f_2}$. Considering again the definition of M this holds if and only if, for all $(x,y) \in \mathcal{R}_{\mathbf{any}}$ we have $(x,y) \in \mathcal{R}_\rho$, i.e. if and only if ρ is valid in M.

Hence M is a model of Γ in which ψ is not valid, contradicting the hypothesis. $\Box$

Putting together the Lemma 27 and Lemma 29 we can state the following theorem.

Theorem 30 *Let Γ be a set of $\mathcal{DINB}$ axioms, ψ a $\mathcal{DINB}$ formula, and T and τ the mappings above. Then $\Gamma \models \psi$ if and only if $T(\Gamma) \models \tau(\psi)$.*

Thus we can reduce logical implication in $\mathcal{DINB}$ to logical implication in $\mathcal{DIN}$ which in turn is reducible to satisfiability in $\mathcal{DIN}$ and hence is decidable and computational characterized as EXPTIME-complete, as established in Chapter 4. As a consequence, we can assert the following complexity results for $\mathcal{DINB}$.

Theorem 31 *Logical implication in $\mathcal{DINB}$ is an EXPTIME-complete problem.*

Proof By Theorem 30, considering Lemma 25, and Lemma 26, and considering that logical implication in $\mathcal{DIN}$ is EXPTIME-complete. $\Box$

Theorem 32 *Satisfiability of $\mathcal{DINB}$ state formulae is an EXPTIME-complete problem.*

Proof Considering that satisfiability of $\mathcal{DINB}$ state formulae is EXPTIME-hard, being $\mathcal{DINB}$ an extension of $\mathcal{DIN}$, by Theorem 31, the thesis follows. $\Box$

As an immediate consequence we can characterize the computational complexity of reasoning in $\mathcal{CINB}$.

Theorem 33 *Satisfiability of $\mathcal{CINB}$ concepts, satisfiability of $\mathcal{CINB}$ TBoxes, and logical implication for $\mathcal{CINB}$ TBoxes, are EXPTIME-complete problems.*

5.3 Discussion

Let us comment on the semantics of negation on basic roles. Given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, there are two ways to assign semantics to such a construct; the one adopted by $\mathcal{CINB}$, namely to interpret a basic role $\neg\rho$ as $\neg\rho^{\mathcal{I}} = \mathbf{any}^{\mathcal{I}} - \rho^{\mathcal{I}}$ where

$\mathbf{any}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ with $\mathbf{any}^{\mathcal{I}} \neq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ in general; and a stronger one, namely to interpret it as $\neg\rho^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} - \rho^{\mathcal{I}}$.[4]

In order to see when the extra power of the stronger interpretation comes in, let us present an example: "A problem is EXPTIME-complete if and only if it is in EXPTIME and every problem in EXPTIME is polynomially reducible to it". In first order logic we can express this sentence as:

$$\begin{aligned} &\forall p(EXPTIME\_comp(p) \equiv \\ &\qquad EXPTIME(p) \wedge \forall p'(EXPTIME(p') \Rightarrow P\_red\_to(p,p'))). \end{aligned}$$

Adopting the stronger interpretation of negation on basic roles we can express the same sentence in a description logic as

$$EXPTIME\_comp \equiv EXPTIME \sqcap \forall \neg P\_red\_to . \neg EXPTIME$$

i.e. "A problem is EXPTIME-complete if it is in EXPTIME and all problems, that are not linked by the role *P_red_to* to it, are not EXPTIME".

Observe that to get the desired meaning we need to refer implicitly to the universal role (the one interpreted as $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$). In fact the extra power of the stronger interpretation of negation on basic roles is tightly connected with the ability to refer to a basic role that is universal, an ability that $\mathcal{CINB}$ does not have[5].

It is our opinion that, interpreting negation on basic roles as in $\mathcal{CINB}$ suffices for most uses. Indeed, it captures the (set-theoretic) *difference* of atomic roles (actually, of conjunction and disjunction of atomic roles). It allows for expressing implications between role expressions involving conjunction, disjunction and difference. It suffices for expressing inclusion assertions on basic roles. We believe that the only strong limitation it has is that it cannot be used to express the Cartesian product of the domain – i.e. the universal role – (or of concepts).

The possibility of denoting a basic role which is universal within $\mathcal{CINB}$ would allow us to express constraints on the cardinality of the domain of interpretation, by means of qualified number restrictions. For example by adopting the stronger interpretation of negation on basic role the concept $(\leq 5\, P \sqcup \neg P . \top)$ would express that there are at most 5 individuals in the domain.

Now, we have seen that a fundamental step in devising decidability of $\mathcal{CINB}$ is Lemma 28. To carry out the proof of this lemma it is only required that the logic fulfills very general conditions (see the discussion on the similar Lemma 18 at the end of Chapter 4). Such conditions are actually violated exactly when constraints on the cardinality of the domain of interpretation can be expressed.

We can conclude that the technique developed in this chapter to establish the decidability of $\mathcal{CINB}$ cannot be applied, if the stronger interpretation of negation on basic roles is adopted. Indeed, to the best of our knowledge, decidability in this case remains an open problem.

[4] This interpretation is stronger, in the sense that it can represent $\neg\rho$ as interpreted in $\mathcal{CINB}$ (as $\mathbf{any} \sqcap \neg\rho$) while the other way round is not true.

[5] Note that $\mathcal{CINB}$ does have the ability to denote the universal role (through the role $(\mathbf{any} \sqcup \mathbf{any}^-)^*$, essentially), it lacks the ability to denote it by means of a basic role.

We close the chapter mentioning the possibility of using $\mathcal{DINB}$ for representing and reasoning about situations evolving as a result of performing actions, in line with the literature on Situation Calculus in Artificial Intelligence. In such respects $\mathcal{DINB}$ offers a formal framework with a clean semantics and a precise computational characterization that in our opinion makes it a kind of Principled Monotonic Propositional Situation Calculus. By exploiting the features of $\mathcal{DINB}$, many advanced issues can be investigated, including complex actions, concurrent actions, hierarchies of actions, etc.[6] In [42] and [44] a preliminary account of this line of research is reported.

[6] In particular, a basic program $\neg\rho$ expresses the performance of an action which is different from ρ. This way of interpreting negations of programs is well suited for reasoning on actions.

Chapter 6

N-ary Relations

In this chapter we study the description logic $\mathcal{CINBR}$ obtained from $\mathcal{CINB}$ by means of suitable mechanisms to aggregate individuals into *tuples*. Each tuple has an associated *arity* which is the number of individuals constituting the tuple. Tuples of the same arity n can be grouped into sets forming *n-ary relations*.[1] The corresponding propositional dynamic logic, $\mathcal{DINBR}$, can easily be defined, however we will not explicitly introduce it here.

6.1 The logic $\mathcal{CINBR}$

An n-ary relation is described by a *name* and n relation roles (r-roles in the following). Each r-role names a component of the relation, i.e. a component of each of its tuples. For each relation $\mathbf{R}$ the set of its r-roles is denoted by $rol(\mathbf{R})$. The cardinality of this set is greater than or equal to 2, and implicitly determines the arity of $\mathbf{R}$. We call "U-component" the component of $\mathbf{R}$ corresponding to the r-role, $U \in rol(\mathbf{R})$.

In $\mathcal{CINBR}$, relations having the same set of r-roles $U_1, \ldots, U_n$ can be composed by means of boolean constructs according to the following abstract syntax:

$$\mathbf{R} ::= \mathbf{Any}_{U_1,\ldots,U_n} \mid \mathbf{P} \mid \mathbf{R}_1 \sqcap \mathbf{R}_2 \mid \mathbf{R}_1 \sqcup \mathbf{R}_2 \mid \mathbf{R}_1 \Rightarrow \mathbf{R}_2 \mid \neg\mathbf{R}$$

where $\mathbf{Any}_{U_1,\ldots,U_n}$ denotes the most general relation having as set of r-roles $U_1, \ldots, U_n$, $\mathbf{P}$ an atomic relation, and $\mathbf{R}$ (possibly with a subscript) a generic relation. We remark that $\mathcal{CINBR}$ allows for composing relations $\mathbf{R}_1$ and $\mathbf{R}_2$ only in case $rol(\mathbf{R}_1) = rol(\mathbf{R}_2)$.

A relation $\mathbf{R}$ can be projected onto two of its components $U, U' \in rol(\mathbf{R})$ getting a binary relation denoted by $\mathbf{R}[U, U']$. In $\mathcal{CINBR}$ such projections play the part that basic roles play in $\mathcal{CINB}$. Projections can be composed into *navigation paths* by means of nondeterministic choice, chaining, reflexive transitive closure, identity binary relation projected on concepts, namely:

$$R ::= \mathbf{R}[U, U'] \mid R_1 \sqcup R_2 \mid R_1 \circ R_2 \mid R^* \mid R^- \mid id(C).$$

[1] We remark that extending description logics with n-ary relations, has already been proposed in [114, 21].

Note that the application of the inverse construct can be restricted to only $\mathbf{R}[U,U']$ projections, without loss of generality. Furthermore, since $(\mathbf{R}[U,U'])^-$ is equivalent to $\mathbf{R}[U',U]$, we could actually do without the inverse construct at all in $\mathcal{CINBR}$.

Next we introduce the constructs to build $\mathcal{CINBR}$ concepts. Concepts in $\mathcal{CINBR}$ have the following abstract syntax:

$$\begin{array}{lll} C & ::= & \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid C_1 \Rightarrow C_2 \mid \neg C \mid \forall R.C \mid \exists R.C \mid \\ & & \forall \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m \mid \exists \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m \\ & & (\leq l\, \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m) \mid (\geq l\, \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m) \end{array}$$

where A denotes an atomic concept, C (possibly with a subscript) a concept, R a navigation path, $\mathbf{R}$ a relation such that $U, T_1, \ldots, T_m \in rol(\mathbf{R})$ and $m \leq |rol(\mathbf{R})|$.

The intuitive meaning of the new concept constructs is explained below (the other constructs have the usual meaning).

- $\forall \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m$ represents the set of individuals x such that for each tuple r in $\mathbf{R}$ with x as U-component, the T_i-component of r belongs to the extension of C_i $(i = 1, \ldots, m)$.
- $\exists \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m$ represents the set of individuals x such that there is a tuple r in $\mathbf{R}$ with x as U-component and x_i $(i = 1, \ldots, m)$ as T_i-component, such that x_i belongs to the extension of C_i.
- $(\leq l\, \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m)$ represents the set of individuals x such that there are *at most* l tuples r in $\mathbf{R}$ with x as U-component and x_i $(i = 1, \ldots, m)$ as T_i-component, such that x_i belongs to the extension of C_i.
- $(\geq l\, \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m)$ represents the set of individuals x such that there are *at least* l tuples r in $\mathbf{R}$ with x as U-component and x_i $(i = 1, \ldots, m)$ as T_i-component, such that x_i belongs to the extension of C_i.

The semantics of $\mathcal{CINBR}$ is given, as usual, through an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, now extended to interpret relations and the new constructs. In particular, if $\mathbf{R}$ is a relation whose set of r-roles is $rol(\mathbf{R}) = \{U_1, \ldots, U_n\}$, then $\mathbf{R}^{\mathcal{I}}$ is a set of labeled tuples of the form $< U_1 : d_1, \ldots, U_n : d_n >$ where $d_1, \ldots, d_n \in \Delta^{\mathcal{I}}$. We write $r[U]$ to denote the value associated with the U-component of the tuple r.

Relations $\mathbf{R}$ with $rol(\mathbf{R}) = \{U_1, \ldots, U_n\}$ are interpreted by $\mathcal{I}$ as:

$$\begin{array}{l} (\mathbf{Any}_{U_1,\ldots,U_n})^{\mathcal{I}} \subseteq \{< U_1 : d_1, \ldots, U_n : d_n > \mid d_1, \ldots, d_n \in \Delta^{\mathcal{I}}\} \\ \mathbf{P}^{\mathcal{I}} \subseteq (\mathbf{Any}_{U_1,\ldots,U_n})^{\mathcal{I}} \\ (\mathbf{R}_1 \sqcap \mathbf{R}_2)^{\mathcal{I}} = \mathbf{R}_1^{\mathcal{I}} \cap \mathbf{R}_2^{\mathcal{I}} \\ (\mathbf{R}_1 \sqcup \mathbf{R}_2)^{\mathcal{I}} = \mathbf{R}_1^{\mathcal{I}} \cup \mathbf{R}_2^{\mathcal{I}} \\ (\mathbf{R}_1 \Rightarrow \mathbf{R}_2)^{\mathcal{I}} = \neg \mathbf{R}_1^{\mathcal{I}} \cup \mathbf{R}_2^{\mathcal{I}} \\ \neg \mathbf{R}^{\mathcal{I}} = (\mathbf{Any}_{U_1,\ldots,U_n})^{\mathcal{I}} - \mathbf{R}^{\mathcal{I}}. \end{array}$$

Projections $\mathbf{R}[U,U']$ of $\mathbf{R}$ onto its U-component and U'-component $(U, U' \in rol(\mathbf{R}))$ are interpreted by $\mathcal{I}$ as follows:

$$\mathbf{R}[U,U']^{\mathcal{I}} = \{(d,d') \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \exists r \in \mathbf{R}^{\mathcal{I}}.d = r[U] \wedge d' = r[U']\}.$$

The other constructs for navigating paths have the usual meaning in $\mathcal{I}$.

Finally the new constructs for concepts are interpreted by $\mathcal{I}$ as follows:

$$\begin{array}{l}
(\forall \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \\
\quad \forall r \in \mathbf{R}^{\mathcal{I}}.r[U] = d \Rightarrow (r[T_1] \in C_1^{\mathcal{I}} \wedge \cdots \wedge r[T_m] \in C_m^{\mathcal{I}})\} \\
(\exists \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \\
\quad \exists r \in \mathbf{R}^{\mathcal{I}}.r[U] = d \wedge r[T_1] \in C_1^{\mathcal{I}} \wedge \cdots \wedge r[T_m] \in C_m^{\mathcal{I}}\} \\
(\leq l\, \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \\
\qquad \text{there are at most } l \text{ tuples } r \in \mathbf{R}^{\mathcal{I}} \text{ such that} \\
\qquad\quad r[U] = d \wedge r[T_1] \in C_1^{\mathcal{I}} \wedge \cdots \wedge r[T_m] \in C_m^{\mathcal{I}}\} \\
(\geq l\, \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid \\
\qquad \text{there are at least } l \text{ tuples } r \in \mathbf{R}^{\mathcal{I}} \text{ such that} \\
\qquad\quad r[U] = d \wedge r[T_1] \in C_1^{\mathcal{I}} \wedge \cdots \wedge r[T_m] \in C_m^{\mathcal{I}}\}
\end{array}$$

An interpretation $\mathcal{I}$ is a model of a $\mathcal{CINBR}$ concept C, if $C^{\mathcal{I}} \neq \emptyset$. Similarly, $\mathcal{I}$ is a model of a $\mathcal{CINBR}$ relation $\mathbf{R}$, if $\mathbf{R}^{\mathcal{I}} \neq \emptyset$. A $\mathcal{CINBR}$ concept is satisfiable, if it has a model. Similarly a $\mathcal{CINBR}$ relation is satisfiable, if it has a model. In $\mathcal{CINBR}$, we allow for inclusion assertions on both concepts, $C_1 \sqsubseteq C_2$, and relations, $\mathbf{R}_1 \sqsubseteq \mathbf{R}_2$ with $rol(\mathbf{R}_1) = rol(\mathbf{R}_2)$. An interpretation $\mathcal{I}$ is a model of an inclusion assertions on concepts $C_1 \sqsubseteq C_2$, if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. Similarly, $\mathcal{I}$ is a model of an inclusion assertion on relations $\mathbf{R}_1 \sqsubseteq \mathbf{R}_2$, if $\mathbf{R}_1^{\mathcal{I}} \subseteq \mathbf{R}_2^{\mathcal{I}}$.

$\mathcal{CINBR}$ TBoxes are defined as a finite set of inclusion assertions on both concepts and relations. An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{K}$ if it is a model of all the inclusion assertions in it. A TBox is satisfiable if it has a model. A TBox $\mathcal{K}$ logically implies an inclusion assertion k (either on concepts or on relations), if every model of $\mathcal{K}$ is a model of k.

Let us show some examples of the use of $\mathcal{CINBR}$. Consider the relation **Parents**, with $rol(\mathbf{Parents}) = \{child, father, mother\}$, denoting the set of tuples, child and his/her (natural) parents (both father and mother). We may force the following inclusion assertion:

$$Human \sqsubseteq \forall \mathbf{Parents}[child].father : Human, mother : Human$$

stating that both the father and the mother of a child, who is human, must be human as well (more precisely, every individual who is *Human* is such that, if (s)he participates, as *child*-component, in a tuple r of the relation **Parents**, then both the *father*-component of r and the *mother*-component of r are *Human*). Note that, in order to represent the (natural) parents of a child, the relation **Parent** must be so that any individual has at most one father and one mother in the relation **Parents** – that is, individuals may occur as *child*-component in at most one tuple of the relation. This fact can be represented in $\mathcal{CINBR}$ by asserting that:

$$\top \sqsubseteq (\leq 1\, \mathbf{Parents}[child].child : \top).$$

6.2 Reasoning in $\mathcal{CINBR}$

We investigate the decidability and the complexity of the reasoning tasks for $\mathcal{CINBR}$. Satisfiability of $\mathcal{CINBR}$ relations is easily reducible to propositional logic and hence is characterized as NP-complete. Satisfiability of concepts, satisfiability of TBoxes, and logical implication in $\mathcal{CINBR}$ TBoxes, are all EXPTIME-hard, being $\mathcal{CINBR}$ a superset of $\mathcal{CIN}$. However their decidability and computational complexity characterization are yet to be established. In the following we concentrate on logical implication[2] showing that logical implication in $\mathcal{CINBR}$ is polynomially reducible to logical implication in $\mathcal{CIN}$, which is decidable and EXPTIME-complete. In fact the argument by which we prove the result follows quite closely the one adopted in Chapter 5 to reduce logical implication in $\mathcal{DINB}$ to logical implication in $\mathcal{DIN}$.

For simplicity of exposition we will implicitly assume that in a logical implication $\mathcal{K} \models k$ the atomic relations (including $\mathbf{Any}_{U_1,\ldots,U_n}$) occurring in k also occur in $\mathcal{K}$.[3]

We start the reduction by defining two mappings: ϱ_r from $\mathcal{CINBR}$ relations to $\mathcal{CIN}$ concepts, and ϱ_c from $\mathcal{CINBR}$ concepts to $\mathcal{CIN}$ concepts.

Definition We define a mapping ϱ_r form $\mathcal{CINBR}$ relations $\mathbf{R}$ having $rol(\mathbf{R}) = \{U_1, \ldots, U_n\}$ to $\mathcal{CIN}$ concepts $\varrho_r(\mathbf{R})$ as follows:

$$\begin{aligned}
&\varrho_r(\mathbf{Any}_{U_1,\ldots,U_n}) = \top_{\mathbf{Any}_{U_1,\ldots,U_n}}\\
&\varrho_r(\mathbf{P}) = \top_{\mathbf{Any}_{U_1,\ldots,U_n}} \sqcap A_{\mathbf{P}}\\
&\varrho_r(\mathbf{R}_1 \sqcap \mathbf{R}_2) = \varrho_r(\mathbf{R}_1) \sqcap \varrho_r(\mathbf{R}_2)\\
&\varrho_r(\mathbf{R}_1 \sqcup \mathbf{R}_2) = \varrho_r(\mathbf{R}_1) \sqcup \varrho_r(\mathbf{R}_2)\\
&\varrho_r(\mathbf{R}_1 \Rightarrow \mathbf{R}_2) = \varrho_r(\mathbf{R}_1) \Rightarrow \varrho_r(\mathbf{R}_2)\\
&\varrho_r(\neg\mathbf{R}) = \top_{\mathbf{Any}_{U_1,\ldots,U_n}} \sqcap \neg\varrho_r(\mathbf{R})
\end{aligned}$$

where $\top_{\mathbf{Any}_{U_1,\ldots,U_n}}$ and $A_{\mathbf{P}}$ are new atomic concepts. $\Box$

Definition We define a mapping ϱ_c from $\mathcal{CINBR}$ concepts C to $\mathcal{CIN}$ concepts $\varrho_c(C)$

[2]We recall that satisfiability of concepts and satisfiability of TBoxes can be reformulated as logical implications, namely, a concept C is satisfiable iff $\emptyset \not\models C \sqsubseteq \bot$, and a TBox $\mathcal{K}$ is satisfiable iff $\mathcal{K} \not\models \top \sqsubseteq \bot$.

[3]Observe that this will not limit the generality of the result, since if an atomic relation $\mathbf{P}$ occurs in k but not in $\mathcal{K}$, we may add the inclusion assertion $\mathbf{P} \sqsubseteq \mathbf{P}$ to $\mathcal{K}$ without changing its model.

as follows:

$$
\begin{array}{l}
\varrho_c(\top) = \top_c \\
\varrho_c(\bot) = \neg\top_c \\
\varrho_c(A) = A \\
\varrho_c(C_1 \sqcap C_2) = \varrho_c(C_1) \sqcap \varrho_c(C_2) \\
\varrho_c(C_1 \sqcup C_2) = \varrho_c(C_1) \sqcup \varrho_c(C_2) \\
\varrho_c(\neg C) = \neg\varrho_c(C) \\
\varrho_c(\forall R.C) = \forall\varrho'_c(R).\varrho_c(C) \\
\varrho_c(\exists R.C) = \exists\varrho'_c(R).\varrho_c(C) \\
\varrho_c(\forall \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m) = \\
\qquad \forall f_U^-.(\varrho_r(\mathbf{R}) \sqcap \exists f_{T_1}.\varrho_c(C_1) \sqcap \ldots \sqcap \exists f_{T_m}.\varrho(C_m)) \\
\varrho_c(\exists \mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m) = \\
\qquad \exists f_U^-.(\varrho_r(\mathbf{R}) \sqcap \exists f_{T_1}.\varrho_c(C_1) \sqcap \ldots \sqcap \exists f_{T_m}.\varrho(C_m)) \\
\varrho_c((\leq l\,\mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m)) = \\
\qquad (\leq l\, f_U^-.\varrho_r(\mathbf{R}) \sqcap \exists f_{T_1}.\varrho_c(C_1) \sqcap \ldots \sqcap \exists f_{T_m}.\varrho(C_m)) \\
\varrho_c((\geq l\,\mathbf{R}[U].T_1 : C_1, \ldots, T_m : C_m)) = \\
\qquad (\geq l\, f_U^-.\varrho_r(\mathbf{R}) \sqcap \exists f_{T_1}.\varrho_c(C_1) \sqcap \ldots \sqcap \exists f_{T_m}.\varrho(C_m))
\end{array}
$$

where f_U are new atomic roles, and ϱ'_c is a mapping from $\mathcal{CINBR}$ navigation paths R to $\mathcal{CIN}$ roles $\varrho'_c(R)$ defined as follows:

$$
\begin{array}{l}
\varrho'_c(\mathbf{R}[U, U']) = f_U^- \circ \varrho_r(\mathbf{R}) \circ f_{U'} \\
\varrho'_c(R_1 \sqcup R_2) = \varrho'_c(R_1) \sqcup \varrho_c(R_2) \\
\varrho'_c(R_1 \circ R_2) = \varrho'_c(R_1) \circ \varrho'_c(R_2) \\
\varrho'_c(R^*) = \varrho'_c(R)^* \\
\varrho'_c(id(C)) = id(\varrho_c(C)) \\
\varrho'_c(R^-) = \varrho'_c(R)^-.
\end{array}
$$

□

Making use of the above mappings we define a mapping ϱ from $\mathcal{CINBR}$ inclusion assertions to $\mathcal{CIN}$ inclusion assertions, and a mapping P from $\mathcal{CINBR}$ TBoxes to $\mathcal{CIN}$ TBoxes.

Definition Let k be a $\mathcal{CINBR}$ inclusion assertions, we define $\varrho(k)$ as follows:

- if $k = C_1 \sqsubseteq C_2$, then $\varrho(k) = \top_c \sqcap \varrho_c(C_1) \sqsubseteq \varrho_c(C_2)$
- if $k = \mathbf{R}_1 \sqsubseteq \mathbf{R}_2$, then $\varrho(k) = \varrho_r(\mathbf{R}_1) \sqsubseteq \varrho_r(\mathbf{R}_2)$.

□

Lemma 34 *Let k be a $\mathcal{CINBR}$ inclusion assertion, and ϱ the mapping defined above. Then $\varrho(k)$ is a $\mathcal{CIN}$ inclusion assertion, and its size is polynomially related to the size of k.*

Proof Straightforward. □

Definition Let $\mathcal{K}$ be a $\mathcal{CINBR}$ TBox. We define a $\mathcal{CIN}$ TBox $P(\mathcal{K})$ as $P(\mathcal{K}) = P_1(\mathcal{K}) \cup P_2(\mathcal{K})$, where

- $P_1(\mathcal{K}) = \{\varrho(k) \mid k \in \mathcal{K}\}$
- $P_2(\mathcal{K})$ is the set constructed by one

  $$\top_{\mathbf{Any}_{U_1,\ldots,U_n}} \equiv \neg\top_c \sqcap \exists f_{U_1}.\top_c \sqcap \ldots \sqcap \exists f_{U_n}.\top_c$$

  for each $\top_{\mathbf{Any}_{U_1,\ldots,U_n}}$ occurring in $P_1(\mathcal{K})$, and one

  $$\top \sqsubseteq (\leq 1\, f_U.\top)$$

  for each f_U occurring in $P_1(\mathcal{K})$.

□

Lemma 35 *Let $\mathcal{K}$ be $\mathcal{CINBR}$ TBox, and P the mapping defined above. Then $P(\mathcal{K})$ is a $\mathcal{CIN}$ TBox, and its size is polynomially related to the size of $\mathcal{K}$.*

Proof Straightforward. □

Intuitively, in the models of $P(\mathcal{K})$, we distinguish individuals in $\top_c$, which represent instances of concepts in models of $\mathcal{K}$, and those in $\top_{\mathbf{Any}_{U_1,\ldots,U_n}}$, which represent instances of the relation $\mathbf{Any}_{U_1,\ldots,U_n}$ in models of $\mathcal{K}$. Individuals in $\top_{\mathbf{Any}_{U_1,\ldots,U_n}}$ have exactly one link for each $f_{U_1},\ldots,f_{U_n}$, and these links connect them to individuals in $\top_c$. In general, a relation $\mathbf{R}$, with $rol(\mathbf{R}) = \{U_1 \ldots, U_n\}$, occurring in $\mathcal{K}$, is represented in $P(\mathcal{K})$ by the concept $\varrho_r(\mathbf{R})$, i.e. the tuples of $\mathbf{R}$ are represented by instances of $\varrho_r(\mathbf{R})$. Observe that this representation is accurate only in the models $\mathcal{I}$ of $P(\mathcal{K})$ where each tuple of $\mathbf{R}$ corresponds to a single individual, otherwise, in $\mathcal{I}$ there would be two individuals representing the same tuple. However, we can show (by using the same technique applied in proving Lemma 18 in Chapter 4 and Lemma 28 in Chapter 5) that if $P(\mathcal{K})$ admits a model, then it admits a model satisfying the above condition. Formally, the following lemma holds.

Lemma 36 *The $\mathcal{CIN}$ TBox $P(\mathcal{K})$ obtained by the above construction has a model $\mathcal{I}$ if and only if it has a model $\mathcal{I}'$ satisfying the condition:*

$$\begin{array}{rcl} d, d' & \in & \varrho_r(\mathbf{R})^{\mathcal{I}'} \Rightarrow \\ & & \neg((d,d_1) \in f_{U_1}^{\mathcal{I}'} \wedge (d',d_1) \in f_{U_1}^{\mathcal{I}'} \wedge \\ & & \;\ldots \wedge (d',d_n) \in f_{U_n}^{\mathcal{I}'} \wedge (d',d_n) \in f_{U_n}^{\mathcal{I}'}) \end{array}$$

for every relation $\mathbf{R}$, with $rol(\mathbf{R}) = \{U_1,\ldots,U_n\}$, occurring in $P(\mathcal{K})$.

Proof The proof is almost identical to that of Lemma 18 in Chapter 4. We sketch it here for completeness. Suppose that the condition is not already satisfied in the model $\mathcal{I}$, we show how to build a model $\mathcal{I}'$ in which the condition above is satisfied. Given an individual $t \in \varrho_r(\mathbf{R})^{\mathcal{I}}$, we denote by $f_U(t)$ $(U = U_1,\ldots,U_n)$ the individual u such that $(t,u) \in f_U^{\mathcal{I}}$, this is in agreement with $f_U^{\mathcal{I}}$ being functional.

We call *conflict* the existence of a non-singleton set $S_{<U_1:d_1,\ldots,U_n:d_n>}$ of individuals t, such that $t \in \varrho_r(\mathbf{R})^{\mathcal{I}}$, and $f_{U_1}(t) = d_1, \ldots, f_{U_n} = d_n$, for some fixed $d_1, \ldots, d_n$. From $S_{<U_1:d_1,\ldots,U_n:d_n>}$ we randomly choose one individual z, and we say that the others *induce the conflict*. We call $Conf$ the set of all individuals inducing a conflict. Note that $Conf$ can be uncountable.

We define a interpretation $\mathcal{I}_{2^{Conf}}$ as the disjoint union of $|2^{Conf}|$ copies of $\mathcal{I}$, one copy, denoted by $\mathcal{I}^{\mathcal{E}}$, for every set $\mathcal{E} \in 2^{Conf}$. We denote by $d^{\mathcal{E}}$ the copy in $\mathcal{I}^{\mathcal{E}}$ of the individual d in $\mathcal{I}$. Trivially, $\mathcal{I}_{2^{Conf}}$ is a model of $P(\mathcal{K})$ as $\mathcal{I}$ is.

Let $\mathcal{I}^{\mathcal{E}}$ and $\mathcal{I}^{\mathcal{E}'}$ be two copies of $\mathcal{I}$ in $\mathcal{I}_{2^{Conf}}$, we call "exchanging $f_{U_n}(t^{\mathcal{E}})$ with $f_{U_n}(t^{\mathcal{E}'})$" the operation on $\mathcal{I}_{2^{Conf}}$ consisting of removing the pair $(t^{\mathcal{E}}, f_{U_n}(t^{\mathcal{E}}))$ from $f_{U_n}^{\mathcal{I}^{\mathcal{E}}}$ replacing it with $(t^{\mathcal{E}}, f_{U_n}(t^{\mathcal{E}'}))$ and, at the same time, removing $(t^{\mathcal{E}'}, f_{U_n}(t^{\mathcal{E}'}))$ from $f_{U_n}^{\mathcal{I}^{\mathcal{E}'}}$ replacing it with $(t^{\mathcal{E}'}, f_{U_n}(t^{\mathcal{E}}))$. By exchanging $f_{U_n}(t^{\mathcal{E}})$ with $f_{U_n}(t^{\mathcal{E}'})$, we resolve t for both $\mathcal{I}^{\mathcal{E}}$ and $\mathcal{I}^{\mathcal{E}'}$, in the sense that $t^{\mathcal{E}}$ and $t^{\mathcal{E}'}$ no longer induce conflicts.

We get a model $\mathcal{I}'$ with the desired property by modifying $\mathcal{I}_{2^{Conf}}$ as follows: for each state $t \in Conf$, for each $\mathcal{E} \in 2^{Conf}$ such that $t \in \mathcal{E}$, we exchange $f_{U_n}(t^{\mathcal{E}})$ with $f_{U_n}(t^{\mathcal{E}-\{t\}})$.

Indeed proceeding in this way, on the one hand all conflicts present in the original model $\mathcal{I}$ are eliminated from all its copies in $\mathcal{I}_{2^{Conf}}$. On the other hand no new conflicts are created.

Finally, $\mathcal{I}'$ is a model of $P(\mathcal{K})$, since by construction inclusion assertions in $P_2(\mathcal{K})$ are satisfied in $\mathcal{I}'$, and it is straightforward to check by induction that for every $C \in P(\mathcal{K})$, for all $\mathcal{E} \in 2^{Conf}$, $d \in C^{\mathcal{I}}$ if and only if $d^{\mathcal{E}} \in C^{\mathcal{I}'}$. $\Box$

Now, we are ready to state the desired result.

Theorem 37 *Let $\mathcal{K}$ be a $\mathcal{CINBR}$ TBox, k a $\mathcal{CINBR}$ inclusion assertions, and P and ϱ the mappings defined above. Then $\mathcal{K} \models k$ if and only if $P(\mathcal{K}) \models \varrho(k)$.*

Proof $\Leftarrow$ By contradiction: suppose there exists a model $\mathcal{I}$ of $\mathcal{K}$ which is not a model of k.

From $\mathcal{I}$ we can define an interpretation $\mathcal{I}'$ as follows:

- $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}} \cup \Delta_r^{\mathcal{I}'}$, where $\Delta_r^{\mathcal{I}'}$ contains one element $z_{<U_1:d_1,\ldots,U_n:d_n>}$ for each $< U_1 : d_1, \ldots, U_n : d_n > \in (\mathbf{Any}_{U_1,\ldots,U_n})^{\mathcal{I}}$, for some $\mathbf{Any}_{U_1,\ldots,U_n}$
- for all $\mathbf{Any}_{U_1,\ldots,U_n}$, for all $< U_1 : d_1, \ldots, U_n : d_n > \in (\mathbf{Any}_{U_1,\ldots,U_n})^{\mathcal{I}}$, we put

$$\begin{array}{l} z_{<U_1:d_1,\ldots,U_n:d_n>} \in \top^{\mathcal{I}'}_{\mathbf{Any}_{U_1,\ldots,U_n}} \\ (z_{<U_1:d_1,\ldots,U_n:d_n>}, d_1) \in f_{U_1}^{\mathcal{I}'} \\ \ldots \\ (z_{<U_1:d_1,\ldots,U_n:d_n>}, d_n) \in f_{U_n}^{\mathcal{I}'} \end{array}$$

- for all $\mathbf{P}$, for all $< U_1 : d_1, \ldots, U_n : d_n > \in \mathbf{P}^{\mathcal{I}}$, we put $z_{<U_1:d_1,\ldots,U_n:d_n>} \in A_{\mathbf{P}}^{\mathcal{I}'}$
- $\top_c^{\mathcal{I}'} = \Delta^{\mathcal{I}}$, and for all atomic concepts A, we put $A^{\mathcal{I}'} = A^{\mathcal{I}}$.

First, note that by construction, $\mathcal{I}'$ is a model of $P_2(\mathcal{K})$.

Next it is easy to verify by induction that, given a $\mathcal{CINBR}$ relation $\mathbf{R}$ with $rol(\mathbf{R}) = \{U_1, \ldots, U_n\}$, for all $d_1, \ldots, d_n \in \Delta^{\mathcal{I}}$,

$$< U_1 : d_1, \ldots, U_n : d_n > \in \mathbf{R}^{\mathcal{I}} \text{ iff } z_{<U_1:d_1,\ldots,U_n:d_n>} \in \varrho_r(\mathbf{R})^{\mathcal{I}'}$$

and that, given a $\mathcal{CINBR}$ concept C, for all $d \in \Delta^{\mathcal{I}}$,

$$d \in C^{\mathcal{I}} \text{ iff } d \in \varrho_c(C)^{\mathcal{I}'}.$$

Let $C_1 \sqsubseteq C_2$ be a $\mathcal{CINBR}$ inclusion assertion on concepts. The interpretation $\mathcal{I}$ is a model of $C_1 \sqsubseteq C_2$ iff for all $d \in \Delta^{\mathcal{I}}$, $d \in C_1^{\mathcal{I}} \Rightarrow d \in C_2^{\mathcal{I}}$. Considering the definition of $\mathcal{I}'$ this holds if and only if: for all $d \in \top_c$, $d \in \varrho_c(C_1)^{\mathcal{I}'} \Rightarrow d \in \varrho_c(C_2)^{\mathcal{I}'}$, i.e. $\mathcal{I}'$ is a model of $\top_c \sqcap \varrho_c(C_1) \sqsubseteq \varrho_c(C_2)$.

Similarly, let $\mathbf{R}_1 \sqsubseteq \mathbf{R}_2$ be a $\mathcal{CINBR}$ inclusion assertion on relations . The interpretation $\mathcal{I}$ is a model of $\mathbf{R}_1 \sqsubseteq \mathbf{R}_2$ iff for all $< U_1 : d_1, \ldots, U_n : d_n > \in (\mathbf{Any}_{U_1,\ldots,U_n})^{\mathcal{I}}$, $< U_1 : d_1, \ldots, U_n : d_n > \in \mathbf{R}_1^{\mathcal{I}} \Rightarrow < U_1 : d_1, \ldots, U_n : d_n > \in \mathbf{R}_2^{\mathcal{I}}$. Considering the definition of $\mathcal{I}'$ this holds if and only if: for all $z \in \top^{\mathcal{I}'}_{\mathbf{Any}_{U_1,\ldots,U_n}}$, $z \in \varrho_r(\mathbf{R}_1)^{\mathcal{I}'} \Rightarrow z \in \varrho_r(\mathbf{R}_2)^{\mathcal{I}'}$, which is equivalent to: for all $z \in \Delta^{\mathcal{I}'}$, $z \in \varrho_r(\mathbf{R}_1)^{\mathcal{I}'} \Rightarrow z \in \varrho_r(\mathbf{R}_2)^{\mathcal{I}'}$, i.e. $\mathcal{I}'$ is a model of $\varrho_r(\mathbf{R}_1) \sqsubseteq \varrho_r(\mathbf{R}_2)$.

Hence $\mathcal{I}'$ is a model of $P(\mathcal{K})$ and yet is not a model of $\varrho(k)$, contradicting the hypothesis.

$\Rightarrow$ Again by contradiction: suppose there exists a model $\mathcal{I}'$ of $P(\mathcal{K})$ which is not a model of $\varrho(k)$. Without loss of generality we assume that $\mathcal{I}'$ satisfies the condition in Lemma 36.

From $\mathcal{I}'$ we can define an interpretation $\mathcal{I}$ as follows:

- $\Delta^{\mathcal{I}} = \top_c^{\mathcal{I}'}$
- for all $\mathbf{Any}_{U_1,\ldots,U_n}$, for all $z \in \top^{\mathcal{I}'}_{\mathbf{Any}_{U_1,\ldots,U_n}}$ with $(z, d_1) \in f^{\mathcal{I}'}_{U_1}, \ldots, (z, d_n) \in f^{\mathcal{I}'}_{U_n}$, we put $< U_1 : d_1, \ldots, U_n : d_n > \in (\mathbf{Any}_{U_1,\ldots,U_n})^{\mathcal{I}}$
- for all atomic relations $\mathbf{P}$ with $rol(\mathbf{P}) = \{U_1, \ldots, U_n\}$, for all $z \in (\top_{\mathbf{Any}_{U_1,\ldots,U_n}} \sqcap A_{\mathbf{P}})^{\mathcal{I}'}$ with $(z, d_1) \in f^{\mathcal{I}'}_{U_1}, \ldots, (z, d_n) \in f^{\mathcal{I}'}_{U_n}$, we put $< U_1 : d_1, \ldots, U_n : d_n > \in \mathbf{P}^{\mathcal{I}}$
- for all atomic concepts A, we put $A^{\mathcal{I}} = A^{\mathcal{I}'} \cap \Delta^{\mathcal{I}}$.

It is easy to verify by induction that given a $\mathcal{CINBR}$ relation $\mathbf{R}$ with $rol(\mathbf{R}) = \{U_1, \ldots, U_n\}$,

$$< U_1 : d_1, \ldots, U_n : d_n > \in \mathbf{R}^{\mathcal{I}} \text{ iff } z \in \varrho_r(\mathbf{R})^{\mathcal{I}'}$$

where $d_1, \ldots, d_n$ are the individuals such that $(z, d_1) \in f^{\mathcal{I}'}_{U_1}, \ldots, (z, d_n) \in f^{\mathcal{I}'}_{U_n}$ (observe that by Lemma 36 $d_1, \ldots, d_n$ univocally determine z). Similarly we can verify that given $\mathcal{CINBR}$ concept C, for all $d \in \top_c^{\mathcal{I}'}$

$$d \in C^{\mathcal{I}} \text{ iff } d \in \varrho_c(C)^{\mathcal{I}'}.$$

Let $C_1 \sqsubseteq C_2$ be a $\mathcal{CINBR}$ inclusion assertion on concepts. The interpretation $\mathcal{I}'$ is a model of $\top_c \sqcap \varrho_c(C_1) \sqsubseteq \varrho_c(C_2)$ iff for all $d \in \Delta^{\mathcal{I}'}$, $d \in (\top_c \sqcap \varrho_c(C_1))^{\mathcal{I}'} \Rightarrow d \in \varrho_c(C_2)^{\mathcal{I}'}$, that is equivalent to: for all $d \in \top_c^{\mathcal{I}'}$, $d \in \varrho_c(C_1)^{\mathcal{I}'} \Rightarrow d \in \varrho_c(C_2)^{\mathcal{I}'}$. Considering the definition of $\mathcal{I}$ this holds if and only if: for all $d \in \Delta^{\mathcal{I}}$, $d \in C_1^{\mathcal{I}} \Rightarrow d \in C_2^{\mathcal{I}}$, i.e. $\mathcal{I}$ is a model of $C_1 \sqsubseteq C_2$.

Similarly, let $\mathbf{R}_1 \sqsubseteq \mathbf{R}_2$ be a $\mathcal{CINBR}$ inclusion assertion on relations. The interpretation $\mathcal{I}'$ is a model of $\varrho_r(\mathbf{R}_1) \sqsubseteq \varrho_r(\mathbf{R}_2)$ iff for all $z \in \Delta^{\mathcal{I}'}$, $z \in \varrho_r(\mathbf{R}_1)^{\mathcal{I}'} \Rightarrow z \in \varrho_r(\mathbf{R}_2)^{\mathcal{I}'}$, that is equivalent to for all $z \in \top^{\mathcal{I}'}_{\mathbf{Any}_{U_1,\ldots,U_n}}$, $z \in \varrho_r(\mathbf{R}_1)^{\mathcal{I}'} \Rightarrow z \in \varrho_r(\mathbf{R}_2)^{\mathcal{I}'}$. Considering the definition of $\mathcal{I}$ this holds if and only if: for all $< U_1 : d_1, \ldots, U_n : d_n > \in (\mathbf{Any}_{U_1,\ldots,U_n})^{\mathcal{I}}$, $< U_1 : d_1, \ldots, U_n : d_n > \in \mathbf{R}_1 \Rightarrow < U_1 : d_1, \ldots, U_n : d_n > \in \mathbf{R}_2$, i.e. $\mathcal{I}$ is a model of $\mathbf{R}_1 \sqsubseteq \mathbf{R}_2$.

Hence $\mathcal{I}$ is a model of $\mathcal{K}$ and yet is not a model of k, contradicting the hypothesis. □

Theorem 38 *Satisfiability of $\mathcal{CINBR}$ concepts, satisfiability of $\mathcal{CINBR}$ TBoxes, and logical implication in $\mathcal{CINBR}$ TBoxes, are EXPTIME-complete problems.*

Proof Considering that, by Lemma 35 and Lemma 34, $P(\mathcal{K})$ and $\varrho(k)$ are polynomially bounded to $\mathcal{K}$ and k, the decidability and the complexity of reasoning in $\mathcal{CINBR}$ are an immediate consequence of the results on $\mathcal{CIN}$ in Chapter 4. □

Chapter 7

Individuals

In this chapter, we study reasoning involving knowledge on individuals expressed in terms of membership assertions. Given an alphabet $\mathcal{O}$ of symbols for individuals, a membership assertion is of one of the following forms:

$$C(\alpha_1), \quad R(\alpha_1, \alpha_2)$$

where C is a concept, R is a role, and α_1, α_2 belong to $\mathcal{O}$. The semantics of such assertions is stated as follows. An interpretation $\mathcal{I}$ is extended so as to assign to each $\alpha \in \mathcal{O}$ an element $\alpha^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ in such a way that different elements are assigned to different symbols in $\mathcal{O}$. Then, $\mathcal{I}$ satisfies $C(\alpha)$ if $\alpha^{\mathcal{I}} \in C^{\mathcal{I}}$, and $\mathcal{I}$ satisfies $R(\alpha_1, \alpha_2)$ if $(\alpha_1^{\mathcal{I}}, \alpha_2^{\mathcal{I}}) \in R^{\mathcal{I}}$. An extensional knowledge base (ABox) $\mathcal{M}$ is a finite set of membership assertions, and an interpretation $\mathcal{I}$ is called a model of $\mathcal{M}$ if $\mathcal{I}$ satisfies every assertion in $\mathcal{M}$.

A knowledge base is a pair $\mathcal{B} = (\mathcal{K}, \mathcal{M})$, where $\mathcal{K}$ is a TBox, and $\mathcal{M}$ is an ABox. An interpretation $\mathcal{I}$ is called a model of $\mathcal{B}$ if it is a model of both $\mathcal{K}$ and $\mathcal{M}$. $\mathcal{B}$ is satisfiable if it has a model, and $\mathcal{B}$ logically implies an assertion β ($\mathcal{B} \models \beta$), where β is either an inclusion or a membership assertion, if every model of $\mathcal{B}$ satisfies β. Since logical implication can be reformulated in terms of unsatisfiability (e.g. if $\beta = C(\alpha)$, then $\mathcal{B} \models \beta$ iff $\mathcal{B} \cup \{\neg C(\alpha)\}$ is unsatisfiable, similarly if $\beta = C_1 \sqsubseteq C_2$, then $\mathcal{B} \models \beta$ iff $\mathcal{B} \cup \{C_1 \sqcap \neg C_2(\alpha')\}$ is unsatisfiable, where α' does not occur in $\mathcal{B}$), we only need a procedure for checking satisfiability of a knowledge base.

We study the satisfiability problem for knowledge bases expressed in two description logics $\mathcal{CN}$ (Section 7.1) and $\mathcal{CI}$ (Section 7.2).

7.1 Knowledge bases in $\mathcal{CN}$

The description logic $\mathcal{CN}$ is obtained from $\mathcal{CI}$ by dropping inverse roles and adding qualified number restrictions (see Chapter 4). We show that satisfiability of a $\mathcal{CN}$ knowledge base $\mathcal{B}$ can be polynomially reduced to satisfiability of a $\mathcal{DN}$ formula $\varphi(\mathcal{B})$, where $\mathcal{DN}$ is the PDL obtained from $\mathcal{DI}$ by dropping converse programs and including qualified number restrictions.

We start the reduction by defining a mapping φ_0 form $\mathcal{CN}$ knowledge bases to $\mathcal{DN}$ formulae.

Definition Let $\mathcal{B}$ be a $\mathcal{CN}$ knowledge base. We define the $\mathcal{DN}$ formula $\varphi_0(\mathcal{B})$ as the conjunction of the following formulae (there is a new letter A_i in $\varphi_0(\mathcal{B})$ for each individual α_i in $\mathcal{B}$):

- for every individual α_i,
$$A_i \Rightarrow \wedge_{j \neq i} \neg A_j$$
- for every membership assertion of the form $C(\alpha_i)$ (δ is the mapping introduced in Chapter 2),
$$A_i \Rightarrow \delta(C)$$
- for every membership assertion of the form $R(\alpha_i, \alpha_j)$,
$$A_i \Rightarrow < R > A_j$$
- for every inclusion assertion $C_1 \sqsubseteq C_2$ in $\mathcal{K}$,
$$\delta(C_1) \Rightarrow \delta(C_2).$$

□

We call $r_{\neg int}$ the program obtained from r by chaining the test $(\wedge_i \neg A_i)?$ after each atomic program occurring in r, i.e. the program defined inductively as:

$$\begin{array}{lcl} P_{\neg int} & = & P; (\wedge_i \neg A_i)? \\ (r_1; r2)_{\neg int} & = & (r_1)_{\neg int}; (r_2)_{\neg int} \\ (r_1 \cup r_2)_{\neg int} & = & (r_1)_{\neg int} \cup (r_2)_{\neg int} \\ (r_1^*)_{\neg int} & = & (r_1)^*_{\neg int} \\ (\phi?)_{\neg int} & = & \phi?. \end{array}$$

The size of both $Post(r)$ and $Pre(r)$ is polynomial in the size of r.

Next we define the $\mathcal{DN}$-counterpart of a $\mathcal{CN}$ knowledge base.

Definition Let $\mathcal{B}$ be a $\mathcal{CN}$ knowledge base, φ_0 the mapping from above, *create* a new atomic program, and $\mathbf{u}$ an abbreviation for $(P_1 \cup \ldots \cup P_m)^*$, where $P_1, \ldots, P_m$ are all the atomic roles in $\mathcal{B}$. We define the *$\mathcal{DN}$-counterpart* of $\mathcal{B}$ as $\varphi(\mathcal{B}) = \varphi_1(\mathcal{B}) \wedge \varphi_2(\mathcal{B})$, where:

- $\varphi_1(\mathcal{B}) = \varphi_1^1(\mathcal{B}) \wedge \cdots \wedge \varphi_1^n(\mathcal{B}) \wedge [create]([\mathbf{u}]\varphi_0(\mathcal{B}))$, with one $\varphi_1^i(\mathcal{B}) =< create > A_i$ for each individual α_i in $\mathcal{B}$.
- $\varphi_2(\mathcal{B})$ is the conjunction of the following formulae:
  - for all A_i and for all P occurring in $\varphi_0(\mathcal{B})$:
$$[create][\mathbf{u}](\leq 1\, P.A_i) \tag{7.1}$$

– for all A_i in $\varphi_0(\mathcal{B})$, for all $\phi \in CL([\mathbf{u}]\varphi_0(\mathcal{B}))$:

$$[create](<\mathbf{u}> (A_i \wedge \phi) \Rightarrow [\mathbf{u}](A_i \Rightarrow \phi)) \tag{7.2}$$

– for all A_i in $\varphi_0(\mathcal{B})$, for all $< r > \phi \in CL([\mathbf{u}]\varphi_0(\mathcal{B}))$:

$$\begin{array}{r}[create](<\mathbf{u}> (A_i \wedge < r_{\neg ind} > \phi) \Rightarrow \\ [\mathbf{u}](A_i \Rightarrow < r_{\neg ind} > \phi))\end{array} \tag{7.3}$$

– for all A_i, A_j and for all P in $\varphi_0(\mathcal{B})$, for all programs $r' \in Pre(r)$, with r occurring in $CL([\mathbf{u}]\varphi_0(\mathcal{B}))$:

$$\begin{array}{r}[create](<\mathbf{u}> (A_i \wedge < r'_{\neg ind}; P > A_j) \Rightarrow \\ [\mathbf{u}](A_i \Rightarrow < r'_{\neg ind}; P > A_j)).\end{array} \tag{7.4}$$

□

Lemma 39 *Let $\mathcal{B}$ be a $\mathcal{CN}$ knowledge base, and $\varphi(\mathcal{B})$ its $\mathcal{DN}$-counterpart. Then $\varphi(\mathcal{B})$ is a $\mathcal{DN}$ formula, and its size is polynomially related to the size of $\mathcal{B}$.*

Proof Straightforward. □

The role of (7.1),(7.2),(7.3) and (7.4) is to allow us to collapse all the states where a certain A_i holds, so as to be able to transform them into a single state corresponding to the individual α_i.

In the following, without loss of generality, we will implicitly restrict our attention to models $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ of $\varphi(\mathcal{B})$ such that $\mathcal{S} = \{s\} \cup \{s' \mid (s, s') \in \mathcal{R}_{create} \circ (\bigcup_P R_P)^*\}$ and $M, s \models \varphi(\mathcal{B})$.

We call states t of a model M of $\varphi(\mathcal{B})$, *individual-aliases* of an individual α_i iff $M, t \models A_i$. The formulae (7.3) and (7.4) allow us to prove the technical lemma below.

Lemma 40 *Let M be a model of $\varphi(\mathcal{B})$, let t be an individual-alias of α_i, and let $< r > \phi \in CL([\mathbf{u}]\varphi_0(\mathcal{B}))$. If there is a path from t that satisfies $< r > \phi$ and contains N individual-aliases $t = t_1, \ldots, t_N$, of $\alpha_i = \alpha_{i_1}, \ldots, \alpha_{i_N}$ respectively, then from every individual-alias t' of α_i in M, there is a path that satisfies $< r > \phi$ and contains N individual-aliases $t' = t'_1, \ldots, t'_N$ of $\alpha_{i_1}, \ldots, \alpha_{i_N}$, in the same order as $t_1, \ldots, t_N$.*

Proof By induction on the number N of individual-aliases.

Base Case: $N = 1$, i.e. the only individual-alias is t. Then, by (7.3), we have

$$[create](<\mathbf{u}> (A_i \wedge < r_{\neg ind} > \phi) \Rightarrow [\mathbf{u}](A_i \Rightarrow < r_{\neg ind} > \phi)).$$

So from every individual-alias t' of α_i there is a path satisfying $< r > \phi$ in which no individual-aliases, other than the initial t', occur.

Inductive Case: $N > 1$. Assume that from t there is a path satisfying $< r > \phi$ in which $k+1$ individual-aliases of $\alpha_{i_1}, \ldots, \alpha_{i_k+1}$ occur. Let such a path be $(t = s_0, \ldots, s_w, \ldots, s_q)$, where $M, s_0 \models A_{i_1}$, $M, s_w \models A_{i_2}$, and no individual-aliases occur in $(s_1, \ldots, s_{w-1})$. This implies that there exists a program $r'; P \in Pre(r)$ and a program $r'' \in Post(r)$ such that $(s_0, \ldots, s_w) \in Paths_M(r'; P)$, and $(s_w, \ldots, s_q) \in Paths_M(r'')$, and $< r'; P >< r'' > \phi \Rightarrow < r > \phi$.

Note that $< r'' > \phi \in CL([\mathbf{u}]\varphi_0(\mathcal{B}))$ thus, since the path $(s_w, \ldots, s_q)$, satisfying $< r'' > \phi$ contains k individual-aliases of $\alpha_{i_2}, \ldots, \alpha_{i_k}$, by inductive hypothesis we can conclude that from each individual-alias of α_{i_2}, there is a path satisfying $< r'' > \phi$ which goes through one individual-alias for each one of the individuals $\alpha_{i_2}, \ldots, \alpha_{i_k}$, in the same order as in $(s_w, \ldots, s_q)$.

On the other hand, by (7.4), we have:

$$[create](< \mathbf{u} > (A_{i_1} \wedge < r'_{\neg ind}; P > A_{i_2}) \Rightarrow [\mathbf{u}](A_{i_1} \Rightarrow < r'_{\neg ind}; P > A_{i_2})).$$

So for any individual-alias t' of α_{i_1}, there is a path satisfying $< r'; P > A_{i_2}$ in which no other individual-aliases occur. Thus combining these two arguments we get the thesis. $\Box$

Given a model $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ of $\varphi(\mathcal{B})$, we can obtain a new model $M' = (\mathcal{S}', \{\mathcal{R}'_P\}, \Phi')$ of $\varphi(\mathcal{B})$ in which there is exactly one individual-alias, for each individual in $\mathcal{B}$. Let $s \in \mathcal{S}$ be such that $M, s \models \varphi(\mathcal{B})$. For every individual α_i, we randomly choose, among its individual-aliases x such that $(s, x) \in \mathcal{R}_{create}$, a distinguished one denoted by s_{α_i}. We define the relations $\mathcal{R}''_{create}$ and $\mathcal{R}''_P$, for every atomic program P in $\varphi_0(\mathcal{B})$, as follows:

- $\mathcal{R}''_{create} = \{(s, s_{\alpha_i}) \in \mathcal{R}_{create} \mid \alpha_i \text{ is an individual}\}$;
- $\mathcal{R}''_P = (\mathcal{R}_P - \{(x, y) \in \mathcal{R}_P \mid M, y \models A_j \text{ for some } A_j\}) \cup \{(x, s_{\alpha_j}) \mid (x, y) \in \mathcal{R}_P \text{ and } M, y \models A_j \text{ for some } A_j\}$.

Now, we define M' as:

- $\mathcal{S}' = \{s\} \cup \{x \in \mathcal{S} \mid (s, x) \in \mathcal{R}''_{create} \circ (\bigcup_P \mathcal{R}''_P)^*\}$
- $\mathcal{R}'_{create} = \mathcal{R}''_{create} \cap (S' \times S')$
  $\mathcal{R}'_P = \mathcal{R}''_P \cap (\mathcal{S}' \times \mathcal{S}')$
- $\Pi'(x) = \Pi(x)$, for each state $x \in \mathcal{S}'$.

Observe that, for every atomic program P, the number of P-successors of all states in M', remains unchanged wrt M. The following two lemmas concern M'.

Lemma 41 *Let M be a model of $\varphi(\mathcal{B})$, and M' be defined as above. Then for every formula $\phi \in CL(\varphi_1(\mathcal{B}))$, for every state x of M': $M, x \models \phi$ iff $M', x \models \phi$.*

Proof By induction on the formation of ϕ (called formula induction in the following). We assume, without loss of generality, $\vee, \Rightarrow, [\cdot], (\leq n \cdot)$ to be expressed by means of $\neg, \wedge, < \cdot >, (\geq n \cdot)$.

- $\phi = A$ (atomic formula). $M, x \models A$ iff $M', x \models A$, by construction of M'.
- $\phi = \phi_1 \wedge \phi_2$. $M, x \models \phi_1 \wedge \phi_2$ iff $M, x \models \phi_1 \wedge M, x \models \phi_2$ iff $M', x \models \phi_1 \wedge M', x \models \phi_2$ (by formula induction hypothesis) iff $M', x \models \phi_1 \wedge \phi_2$.

- $\phi = \neg\phi'$. $M, x \models \neg\phi'$ iff $M, x \not\models \phi'$ iff $M', x \not\models \phi'$ (by formula induction hypothesis) iff $M', x \models \neg\phi'$.
- $\phi = (\geq n\, P.\phi')$.

  $\Rightarrow$. $M, x \models (\geq n\, P.\phi')$ if there are at least n states $x_1, \ldots, x_n$ such that $M, x_i \models \phi'$. We distinguish two cases.

  - x_i is not an individual-alias. Then $(x, x_i) \in \mathcal{R}_P$ implies $(x, x_i) \in \mathcal{R}'_P$, whereas by formula induction hypothesis $M', x_i \models \phi'$.
  - x_i is an individual-alias for α_j. Then, by (7.1), either $x_i = s_{\alpha_j}$ or $(x, s_{\alpha_j}) \notin R_P$. In both cases, by construction of M', $(x, x_i) \in \mathcal{R}_P$ implies $(x, s_{\alpha_j}) \in \mathcal{R}'_P$. Now, $M, x_i \models \phi'$ implies $M, s_{\alpha_j} \models \phi'$ by (7.2), and thus, by formula induction hypothesis $M', s_{\alpha_j} \models \phi'$.

  Hence we can conclude that $M', x \models (\geq n\, P.\phi')$.

  $\Leftarrow$. $M', x \models (\geq n\, P.\phi')$ if there are at least n states $x_1, \ldots, x_n$ such that $M', x_i \models \phi'$. We distinguish two cases.

  - x_i is not an individual-alias. Then $(x, x_i) \in \mathcal{R}'_P$ implies $(x, x_i) \in \mathcal{R}'_P$, whereas by formula induction hypothesis $M, x_i \models \phi'$.
  - $x_i = s_{\alpha_j}$. Then, by construction of M' and by (7.1), $(x, s_{\alpha_j}) \in \mathcal{R}'_P$ implies that there exists (exactly) one t in M such that $(x, t) \in \mathcal{R}_P$ (possibly $t = s_{\alpha_j}$). Now, $M', s_{\alpha_j} \models \phi'$ implies $M, s_{\alpha_j} \models \phi'$, by formula induction hypothesis, and thus, by (7.2), $M', t \models \phi'$.

  Hence we can conclude that $M, x \models (\geq n\, P.\phi')$.
- $\phi =< r > \phi'$.

  $\Rightarrow$. Let $M, x \models< r > \phi'$ then there is a path $(x = x_0, \ldots, x_q) \in Paths_M(r)$ such that $M, x_q \models \phi'$. We prove $M', x \models< r > \phi'$, by induction on the number k of individual-aliases along the path $(x_0, \ldots, x_q)$, starting the count from the first non-chosen individual-alias (we call this induction, path induction).

  $k = 0$, this means that for all the states x_i along the path, $x_i \in \mathcal{S}'$. By applying Proposition 5 q times and Proposition 4 once, we can conclude that there exists a formula

  $$< (\phi_{0,1}?; \ldots; \phi_{0,g_0}?); P_1; (\phi_{1,1}?; \ldots; \phi_{1,g_1}?); \ldots; P_q; (\phi_{q,1}?; \ldots; \phi_{q,g_q}?) > \phi'$$

  with $g_i \geq 0$, such that:

  - all tests $\phi_{i,j}?$ occur in r, and hence all ϕ_{ij} are subformulae of $< r > \phi'$;
  - $(x_{i-1}, x_i) \in \mathcal{R}_{P_i}$, for $i = 1, \ldots, q$;
  - the formula
    $$\begin{array}{l} < (\phi_{0,1}?; \ldots; \phi_{0,g_0}?); P_1; \\ \quad (\phi_{1,1}?; \ldots; \phi_{1,g_1}?); \ldots; P_q; \\ \quad (\phi_{q,1}?; \ldots; \phi_{q,g_q}?) > \phi' \Rightarrow< r > \phi' \end{array}$$

    is valid.

By formula induction hypothesis we have that, for all $\phi_{i,j}$, $M, x_i \models \phi_{i,j}$ iff $M', x_i \models \phi_{i,j}$, and $M, x_q \models \phi'$ iff $M', x_q \models \phi'$. While by construction of M', $(x_{i-1}, x_i) \in \mathcal{R}_{P_i}$ implies $(x_{i-1}, x_i) \in \mathcal{R}'_{P_i}$. Hence $M', x \models < r > \phi'$.

$k > 0$. Let $(x_0, \ldots, x_q) = (x_0, \ldots, x_u, \ldots x_q)$ where x_u, such that $M, x_u \models A_j$, is the first non-chosen individual-alias along the path $(x_0, \ldots, x_q)$. By applying Proposition 5 m times only, we can conclude that, there exists a formula

$$< (\phi_{0,1}?; \ldots; \phi_{0,g_0}?); P_1; (\phi_{1,1}?; \ldots; \phi_{1,g_1}?); \ldots; P_u >< r' > \phi'$$

with $g_1 \geq 0$, such that:

- all tests $\phi_{i,j}$? occur in r, and hence all ϕ_{ij} are subformulae of $< r > \phi'$;
- $r' \in Post(r')$, and hence by Proposition 2, the formula $< r' > \phi'$ is equivalent to ψ for some $\psi \in CL(< r > \phi') \subseteq CL(\varphi_1(\mathcal{B}))$;
- $(x_{i-1}, x_i) \in \mathcal{R}_{P_i}$, for $i = 1, \ldots u$;
- $(x_u, \ldots, x_q) \in Paths_M(r')$;
- $< (\phi_{0,1}?; \ldots; \phi_{0,g_0}?); P_1; (\phi_{1,1}?; \ldots; \phi_{1,g_1}?); \ldots; P_u >< r' > \phi' \Rightarrow < r > \phi'$ is valid.

The path $(x_u, \ldots, x_q)$ contains k individual-aliases thus, by Lemma 40, from each individual-alias on α_j there is a path satisfying $< r' > \phi'$ which goes through exactly the "same" k individual-aliases in the same order. Let $(s_{\alpha_i} = x'_u, \ldots, x'_{q'})$ be such a path from s_{α_j}. This path contains strictly less than k individual-aliases, excluding x'_u, thus by path induction hypothesis, $M', s_{\alpha_i} \models < r' > \phi'$.

Now, by construction of M', $(x_{u-1}, x_u) \in \mathcal{R}_{P_u}$ implies $(x_{u-1}, s_{\alpha_i}) \in \mathcal{R}'_{P_u}$ thus $M', x_{u-1} \models < P_u >< r' > \phi'$. Whereas, by formula induction hypothesis, for all $\phi_{i,j}$, $M, x_i \models \phi_{i,j}$ iff $M', x_i \models \phi_{i,j}$. Hence considering that for $i = 1, \ldots, u-1$, $(x_{i-1}, x_i) \in \mathcal{R}_{P_i}$ implies $(x_{i-1}, x_i) \in \mathcal{R}'_{P_i}$, we get $M', x \models < r > \phi'$.

$\Leftarrow$. Let $M', x \models < r > \phi'$, then there is a path $(x = y_0, \ldots, y_q) \in Paths_{M'}(r)$ such that $M', y_q \models \phi'$. We prove $M, x \models < r > \phi'$, by induction on the number k of individual-aliases along the path $(y_0, \ldots, y_q)$ excluding x, if x is an individual-alias (we call this induction, path induction).

$k = 0$, this means that for all the states y_i along the path, $y_i \in \mathcal{S}$. By applying Proposition 5 q times and Proposition 4 once, we can conclude that there exists a formula

$$< (\phi_{0,1}?; \ldots; \phi_{0,g_0}?); P_1; (\phi_{1,1}?; \ldots; \phi_{1,g_1}?); \ldots; P_q; (\phi_{q,1}?; \ldots; \phi_{q,g_q}?) > \phi'$$

with $g_i \geq 0$, such that:

- all tests $\phi_{i,j}$? occur in r, and hence all ϕ_{ij} are subformulae of $< r > \phi'$;
- $(y_{i-1}, y_i) \in \mathcal{R}'_{P_i}$, for $i = 1, \ldots, q$;

– the formula

$$\begin{aligned}&< (\phi_{0,1}?; \ldots; \phi_{0,g_0}?); P_1;\\ &\quad (\phi_{1,1}?; \ldots; \phi_{1,g_1}?); \ldots; P_q;\\ &\quad (\phi_{q,1}?; \ldots; \phi_{q,g_q}?) > \phi' \Rightarrow < r > \phi'\end{aligned}$$

is valid.

By formula induction hypothesis, for all tests $\phi_{i,j}$ $M', y_i \models \phi_{i,j}$ iff $M, y_i \models \phi_{i,j}$, and $M', y_q \models \phi'$ iff $M, y_q \models \phi'$. While by construction of M', $(y_{i-1}, y_i) \in \mathcal{R}'_{P_i}$ implies $(y_{i-1}, y_i) \in \mathcal{R}_{P_i}$. Hence $M', x \models < r > \phi'$.

$k > 0$. Let $(y_0, \ldots, y_q) = (y_0, \ldots, y_u, \ldots y_q)$ where $y_u = s_{\alpha_i}$ is the first individual-alias along the path. By applying Proposition 5 m times only, we can conclude that, there exists a formula

$$< (\phi_{0,1}?; \ldots; \phi_{0,g_0}?); P_1; (\phi_{1,1}?; \ldots; \phi_{1,g_1}?); \ldots; P_u >< r' > \phi'$$

with $g_1 \geq 0$, such that:

– all tests $\phi_{i,j}?$ occur in r, and hence all ϕ_{ij} are subformulae of $< r > \phi'$;

– $r' \in Post(r')$, and hence by Proposition 2, the formula $< r' > \phi'$ is equivalent to ψ for some $\psi \in CL(< r > \phi') \subseteq CL(\varphi_1(\mathcal{B}))$;

– $(y_{i-1}, y_i) \in \mathcal{R}'_{P_i}$, for $i = 1, \ldots u$;

– $(y_u, \ldots, y_q) \in Paths_{M'}(r')$;

– $< (\phi_{0,1}?; \ldots; \phi_{0,g_0}?); P_1; (\phi_{1,1}?; \ldots; \phi_{1,g_1}?); \ldots; P_u >< r' > \phi' \Rightarrow < r > \phi'$ is valid.

Notice that $M', s_{\alpha_i} \models < r' > \phi'$, and along $y_u, \ldots, y_q$ there are $k-1$ individual-aliases, excluding y_u. Thus, by path induction hypothesis $M, s_{\alpha_i} \models < r' > \phi'$, and by (7.2) the same is true for all the individual-aliases of α_i appearing in M.

Now, by construction of M', $(y_{u-1}, y_u) \in \mathcal{R}'_{P_u}$ implies that there exists an individual-alias t of α_i such that $(y_{u-1}, t) \in \mathcal{R}_{P_u}$. Thus $M, y_{u-1} \models < P_u >< r' > \phi'$. Whereas, by formula induction hypothesis, for all $\phi_{i,j}$, $M', x_i \models \phi_{i,j}$ iff $M, x_i \models \phi_{i,j}$. Hence considering that, for $i = 1, \ldots, u-1$, $(y_{i-1}, y_i) \in \mathcal{R}'_{P_i}$ implies $(y_{i-1}, y_i) \in \mathcal{R}_{P_i}$, we get $M, x \models < r > \phi'$.

□

Lemma 42 *Let M be a model of $\varphi(\mathcal{B})$ such that $M, s \models \varphi(\mathcal{B})$, and let M' be a structure derived from M as specified above. Then $M', s \models \varphi(\mathcal{B})$.*

Proof By Lemma 41 $M, s \models \varphi_1(\mathcal{B})$ implies $M', s \models \varphi_1(\mathcal{B})$. On the other hand, we trivially have $M', s \models \varphi_2(\mathcal{B})$, since it has one individual-alias of each individual. Hence the thesis holds. □

We can now state the main theorem on reasoning in $\mathcal{CN}$ knowledge bases.

Theorem 43 *A $\mathcal{CN}$ knowledge base $\mathcal{B}$ is satisfiable iff its $\mathcal{DN}$-counterpart $\varphi(\mathcal{B})$ is satisfiable.*

Proof $\Rightarrow$ Let $\mathcal{I}$ be an interpretation satisfying the knowledge base $\mathcal{B}$. Then we can define a model $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ of $\varphi(\mathcal{B})$ as follows: $\mathcal{S} = \Delta^{\mathcal{I}} \cup \{s_{new}\}$, $\mathcal{R}_{\delta(P)} = P^{\mathcal{I}}$, $\mathcal{R}_{create} = \{(s_{new}, \alpha_i^{\mathcal{I}}) \mid \alpha_i \in \mathcal{O}\}$, $\Pi(s) = \{\delta(A) \mid s \in A^{\mathcal{I}}\} \cup \{A_i \mid s = \alpha_i^{\mathcal{I}}\}$. It is easy to see that $M, s_{new} \models \varphi_1(\mathcal{B})$. Furthermore since, by construction, in M there is exactly one individual-alias of each individual, we have trivially $M, s_{new} \models \varphi_2(\mathcal{B})$. Hence $M, s_{new} \models \varphi(\mathcal{B})$.

$\Leftarrow$ If there exists a model M of $\varphi(\mathcal{B})$ then by Lemma 42 we can construct a model M' such that for each individual there exists exactly one individual-alias. Let $M', s \models \varphi(\mathcal{B})$, we can define an interpretation $\mathcal{I}$ as follows: $\Delta^{\mathcal{I}} = \{s' \mid (s, s') \in \mathcal{R}'_{create} \circ (\bigcup_P \mathcal{R}'_P)^*\}$, $R^{\mathcal{I}} = \mathcal{R}'_{\delta(R)}$, $C^{\mathcal{I}} = \{s' \mid M', s' \models \delta(C)\}$, and for each individual α_i, $\alpha_i^{\mathcal{I}} = \{s_{\alpha_i} \mid M', s_{\alpha_i} \models A_i\}$ (please notice that this set is a singleton).

Now for each inclusion assertion $C_1 \sqsubseteq C_2$ in $\mathcal{B}$, we have that $\delta(C_1) \Rightarrow \delta(C_2)$ holds in every state of M', thus $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. For each membership assertion $\alpha_i : C$ in $\mathcal{B}$, we have $M', s_{\alpha_i} \models \delta(C)$. Finally for each membership assertion $\alpha_i R \alpha_j$, we have that $M', s_{\alpha_i} \models < \delta(R) > A_j$ and there is only one state in M' in which A_j holds, thus $(\alpha_i, \alpha_j) \in R_{\delta(R)}$. Hence $\mathcal{I}$ satisfies $\mathcal{B}$. $\Box$

Theorem 44 *Satisfiability and logical implication for $\mathcal{CN}$ knowledge bases (TBox and ABox) are EXPTIME-complete problems.*

Proof By Theorem 43, satisfiability for $\mathcal{CN}$ knowledge bases is polynomially related to satisfiability in $\mathcal{DN}$, which is EXPTIME-complete, by Theorem 16. $\Box$

7.2 Knowledge bases in $\mathcal{CI}$

Analogously to the case of $\mathcal{CN}$, satisfiability of a $\mathcal{CI}$ knowledge bases can be polynomially reduced to satisfiability of $\mathcal{DI}$-formulae.

We define a mapping $\eta_0(\mathcal{B})$ from $\mathcal{CI}$ knowledge bases to $\mathcal{DI}$ formulae, as identical to φ_0 introduced in the previous section. Then we define the $\mathcal{DI}$-counterpart of a $\mathcal{CI}$ knowledge base as follows.

Definition Let $\mathcal{B}$ be a $\mathcal{CI}$ knowledge base, η_0 the mapping defined above, *create* a new atomic program, and $\mathbf{u}$ an abbreviation for $(P_1 \cup \ldots \cup P_m \cup P_1^- \cup \ldots \cup P_m^-)^*$, where $P_1, \ldots, P_m$ are all the atomic roles in $\mathcal{B}$. We define the *$\mathcal{DI}$-counterpart* of $\mathcal{B}$ as $\eta(\mathcal{B}) = \eta_1(\mathcal{B}) \wedge \eta_2(\mathcal{B})$, where:

- $\eta_1(\mathcal{B}) = \eta_1^1(\mathcal{B}) \wedge \cdots \wedge \eta_1^n(\mathcal{B}) \wedge [create]([\mathbf{u}]\eta_0(\mathcal{B}))$, with each $\eta_1^i(\mathcal{B}) =< create > A_i$ for each individual α_i in $\mathcal{B}$.

- $\eta_2(\mathcal{B}) = \eta_2^1(\mathcal{B}) \wedge \cdots \wedge \eta_2^p(\mathcal{B})$, where we have one $\eta_2^i(\mathcal{B})$ of the form

$$[create](< \mathbf{u} > (A_i \wedge \phi) \Rightarrow [\mathbf{u}](A_i \Rightarrow \phi)), \tag{7.5}$$

for each A_i, and for each $\phi \in CL([\mathbf{u}]\eta_0(\mathcal{B}))$.

□

Lemma 45 *Let $\mathcal{B}$ be a $\mathcal{CI}$ knowledge base, and $\eta(\mathcal{B})$ its $\mathcal{DI}$ counterpart. Then $\eta(\mathcal{B})$ is a $\mathcal{DI}$ formula, and its size is polynomially related to the size of $\mathcal{B}$.*

Proof Straightforward. □

Again, the role of (7.5) is to make all the states where a certain A_i holds, equivalent, so as to be able to collapse them into a single state corresponding to the individual α_i. By reasoning similarly to the case of $\mathcal{CN}$, we are going to show that $\mathcal{B}$ is satisfiable iff $\eta(\mathcal{B})$ is satisfiable.[1]

We again use the notion of individual-aliases of an individual in the models of $\eta(\mathcal{B})$.

Let $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ a model of $\eta(\mathcal{B})$, such that $M, s \models \eta(\mathcal{B})$, for some state $s \in \mathcal{S}$. We show how to obtain a new model M' of $\eta(\mathcal{B})$ in which for every individual in $\mathcal{B}$ there is exactly one individual-alias in M'.

For each individual α_i, we randomly choose among the individual-aliases x such that $(s, x) \in \mathcal{R}_{create}$, a distinguished one denoted by s_{α_i} and we define $\mathcal{R}''_{create} = \{(s, s_{\alpha_i}) \mid \alpha_i \text{ is an individual}\}$. For each atomic program P we define $\mathcal{R}''_P$ as follows:

- if $(x, y) \in \mathcal{R}_P$, $M, x \models \wedge_i \neg A_i$, and $M, y \models \wedge_i \neg A_i$, then we put $(x, y) \in \mathcal{R}''_P$;
- if $(x, y) \in \mathcal{R}_P$, $M, x \models \wedge_i \neg A_i$, and $M, y \models A_j$ then we put $(x, s_{\alpha_j}) \in \mathcal{R}''_P$;
- if $(x, y) \in \mathcal{R}_P$, $M, x \models A_j$, and $M, y \models \wedge_i \neg A_i$, then we put $(s_{\alpha_j}, y) \in \mathcal{R}''_P$;
- if $(x, y) \in R_P$, $M, x \models A_i$, and $M, y \models A_j$, then we put $(s_{\alpha_i}, s_{\alpha_j}) \in \mathcal{R}''_P$.

The structure $M' = (\mathcal{S}', \{\mathcal{R}'_P\}, \Pi')$ is now defined as follows:

- $\mathcal{S}' = \{s\} \cup \{x \in \mathcal{S} \mid (s, x) \in \mathcal{R}''_{create} \circ (\bigcup_P (\mathcal{R}''_P \cup \mathcal{R}''^{-}_P)^*\}$
- $\mathcal{R}'_{create} = \mathcal{R}''_{create}$
- $\mathcal{R}'_P = \mathcal{R}''_P \cap (\mathcal{S}' \times \mathcal{S}')$
- $\Pi'(x) = \Pi(x)$, for each state $x \in \mathcal{S}'$.

Note that, in contrast to the construction in the previous section, the one above does not preserve the number of edges involving the chosen individual-aliases, hence it does not preserve either local or global functionality. The main properties of M' are stated in the following two lemmas.

Lemma 46 *Let M be a model of $\eta(B)$, let M' be defined as above, and let $f : \mathcal{S} \to \mathcal{S}'$ be a mapping defined as follows:*

$$f(x) = \begin{cases} s_{\alpha_i} & \textit{if } M, x \models A_i \textit{ (for some } A_j\textit{)} \\ x & \textit{otherwise.} \end{cases}$$

[1]The proof is much simpler in this case, witness the absence of constraints analogous to (7.3) and (7.4).

Then for every formula $\phi \in CL(\eta_1(\mathcal{B}))$*, for every state* x *of* M*:*

$$M, x \models \phi \;\; \textit{iff} \;\; M', f(x) \models \phi.$$

Proof We prove the lemma by induction on the formation of ϕ (called formula induction in the following). We assume, without loss of generality, $\vee, [\cdot]$ to be expressed by means of $\neg, \wedge, < \cdot >$, and that the converse operator is applied only to atomic programs.

- $\phi = A$.

  $M, x \models A$ iff $M', f(x) \models A$ by construction of M'.

- $\phi = \phi_1 \wedge \phi_2$.

  $M, x \models \phi_1 \wedge \phi_2$ iff $M, x \models \phi_1 \wedge M, x \models \phi_2$ iff (by formula induction hypothesis) $M', f(x) \models \phi_1 \wedge M', f(x) \models \phi_2$ iff $M', f(x) \models \phi_1 \wedge \phi_2$.

- $\phi = \neg\phi'$.

  $M, x \models \neg\phi'$ iff $M, x \not\models \phi'$ iff (by formula induction hypothesis) $M', f(x) \not\models \phi'$ iff $M', f(x) \models \neg\phi'$.

- $\phi =< r > \phi'$.

  $\Rightarrow$. Let $M, x \models< r > \phi'$ and let $(x = x_0, \ldots, x_q) \in Paths_M(r)$ such that $M, x_q \models \phi'$. We prove $M', f(x) \models< r > \phi'$, by induction on the length q of the path (called path induction, in the following).

  $q = 0$. By Proposition 4, there exists a formula $< \phi_1?; \ldots; \phi_g? > \phi'$, with $g \geq 0$, such that:

  - all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;
  - $M, x \models< \phi_1?; \ldots; \phi_g? > \phi'$;
  - $< \phi_1?; \ldots; \phi_g? > \phi \Rightarrow< r > \phi$ is valid.

  By formula induction hypothesis, for $\psi = \{\phi_1, \ldots, \phi_q, \phi'\}$, $M, x \models \phi_i$ implies $M', f(x) \models \phi_i$.

  $q > 0$. By Proposition 5, there exists a formula $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi'$, with $g \geq 0$, such that:

  - all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;
  - $r' \in Post(r)$, and hence the formula $< r' > \phi'$ is equivalent to ψ for some $\psi \in CL(< r > \phi') \subseteq CL(\eta_1(\mathcal{B}))$;
  - $(x_0, x_1) \in \mathcal{R}_a$;
  - $(x_1, \ldots, x_q) \in Paths_M(r')$;
  - $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi' \Rightarrow< r > \phi'$ is valid.

By formula induction hypothesis, for $\phi_i = \{\phi_1, \ldots, \phi_q\}$, $M, x \models \phi_i$ implies $M', f(x) \models \phi_i$. By construction of M', $(x, x_1) \in \mathcal{R}_a$ implies $(f(x), f(x_1)) \in \mathcal{R}'_a$. By path induction hypothesis, since $(x_1, \ldots, x_q) \in Paths_M(r')$ is shorter then $(x_0, \ldots, x_q) \in Paths_M(r)$, we can conclude that $M, x_1 \models < r' > \phi'$ implies $M', f(x_1) \models < r' > \phi'$. Hence $M', x \models < r > \phi'$.

$\Leftarrow$. Let $M', f(x) \models < r > \phi'$ and let $(f(x) = y_0, \ldots, y_q) \in Paths_{M'}(r)$ such that $M', y_q \models \phi'$. We prove $M, x \models < r > \phi'$, by induction on the length q of the path (called path induction, in the following).

$q = 0$. By Proposition 4, there exists a formula $< \phi_1?; \ldots; \phi_g? > \phi'$, with $g \geq 0$, such that:

- all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;
- $M, f(x) \models < \phi_1?; \ldots; \phi_g? > \phi'$;
- $< \phi_1?; \ldots; \phi_g? > \phi \Rightarrow < r > \phi$ is valid.

By formula induction hypothesis, for $\psi = \{\phi_1, \ldots, \phi_q, \phi'\}$, $M', f(x) \models \phi_i$ implies $M, x \models \phi_i$.

$q > 0$. By Proposition 5, there exists a formula $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi'$, with $g \geq 0$, such that:

- all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;
- $r' \in Post(r)$, and hence the formula $< r' > \phi'$ is equivalent to ψ for some $\psi \in CL(< r > \phi') \subseteq CL(\eta_1(\mathcal{B}))$;
- $(y_0, y_1) \in \mathcal{R}'_a$;
- $(y_1, \ldots, y_q) \in Paths_{M'}(r')$;
- $< \phi_1?; \ldots; \phi_g?; a >< r' > \phi' \Rightarrow < r > \phi'$ is valid.

By formula induction hypothesis, for $\phi_i = \{\phi_1, \ldots, \phi_q\}$, $M', f(x) \models \phi_i$ implies $M, x \models \phi_i$. By construction of M', if $(f(x), y_1) \in \mathcal{R}'_a$ then there exists a state x_1 such that $f(x_1) = y_1$ and $(x, x_1) \in \mathcal{R}_a$. By path induction hypothesis, since $(y_1, \ldots, y_q) \in Paths_{M'}(r')$ is shorter then $(y_0, \ldots, y_q) \in Paths_{M'}(r)$, we can conclude that $M', f(x_1) \models < r' > \phi'$ implies $M, x_1 \models < r' > \phi'$. Hence $M, x \models < r > \phi'$.

□

Lemma 47 *Let M be a model of $\eta(\mathcal{B})$ such that $M, s \models \Phi$, and let M' be derived from M' as specified above. Then, $M', s \models \eta(\mathcal{B})$.*

Proof By Lemma 46 $M, s \models \eta_1(\mathcal{B})$ implies $M', s \models \eta_1(\mathcal{B})$. On the other hand, $M', s \models \eta_2(\mathcal{B})$, since it has one individual-alias of each individual. Hence the thesis holds. □

We can now state the main theorem of this section.

Theorem 48 *A $\mathcal{CI}$ knowledge base $\mathcal{B}$ is satisfiable iff its $\mathcal{DI}$-counterpart $\eta(\mathcal{B})$ is satisfiable.*

Proof Similar to the proof of Theorem 43. $\Box$

Theorem 49 *Satisfiability and logical implication for $\mathcal{CI}$ knowledge bases (TBox and ABox) are EXPTIME-complete problems.*

Proof By Theorem 48, satisfiability for $\mathcal{CI}$ knowledge bases is polynomially related to satisfiability in $\mathcal{DI}$, which is EXPTIME-complete. $\Box$

7.3 Discussion

Observe that in devising the results in Section 7.1 and Section 7.2, we did not exploit the fact that knowledge about individuals is organized in membership assertions. We exploited only the fact that the number of individuals occurring in a knowledge base is finite.

This observation allows us to rephrase the results in those sections in a more general form. Let us introduce the description logics $\mathcal{CNO}$ and $\mathcal{CIO}$, which are obtained by adding to $\mathcal{CN}$ and $\mathcal{CI}$ special atomic concepts A_α, called *names*, having exactly a single instance α, i.e. the individual they name. Names may occur in concepts exactly as atomic concepts, and hence constitute one of the most flexible way to express knowledge about single individuals.

By using names we can capture the construct ONE-OF, having the form $\{\alpha_1, \ldots, \alpha_n\}$, denoting the concept made of exactly the enumerated individuals $\alpha_1, \ldots, \alpha_n$ [2]; as well as the construct FILLS, having the form $R : \alpha$, denoting those individuals having the individual α as a *role filler* of R [3] (see [107] and references therein for further discussion on these constructs).

The result in Section 7.1 can be generalized as follows: satisfiability in $\mathcal{CNO}$ knowledge bases can be polynomially reduced to satisfiability of $\mathcal{CN}$ formulae, hence is decidable, and EXPTIME-complete. Similarly, the result in Section 7.2 can be generalized as follows: satisfiability in $\mathcal{CIO}$ knowledge bases can be polynomially reduced to satisfiability of $\mathcal{CI}$ formulae, hence is decidable, and EXPTIME-complete.

It is straightforward to define the propositional dynamic logics $\mathcal{DNO}$ and $\mathcal{DIO}$, corresponding to the description logics $\mathcal{CNO}$ and $\mathcal{CIO}$ respectively. It is also straightforward to define $\mathcal{CFO}$ and $\mathcal{DFO}$, the description logic and propositional dynamic logic obtained from $\mathcal{CNO}$ and $\mathcal{DNO}$ by allowing only functional restrictions, instead of full qualified number restrictions.

The notion of names introduced above has a correspondent in modal logic in the notion of *nominals*. Nominals have a tradition in modal logic that dates back to [96, 17], recent papers on nominals are [11, 58, 10]. Nominals have also been studied

[2] Actually, names and the ONE-OF construct are essentially equivalent, since a name A_α is equivalent to $\{\alpha\}$ and $\{\alpha_1 \ldots, \alpha_n\}$ is equivalent to $A_{\alpha_1} \sqcup \ldots \sqcup A_{\alpha_n}$.

[3] The FILLS construct $R : \alpha$ is captured by $\exists R.A_\alpha$.

within the setting of propositional dynamic logics in [89, 59, 90]. In the following, we focus on two such logics.

The first is called *deterministic combinatory propositional dynamic logic*, DcPDL, and is essentially deterministic propositional dynamic logic augmented with nominals. In [89] its decidability is established, by a finite model argument, and it is proved that satisfiability can be checked in nondeterministic double exponential time, i.e. it is in the complexity class 2NEXPTIME. Since that paper, this upper bound hasn't be improved (see [90]). Obviously since DcPDL contains deterministic PDL, its satisfiability is EXPTIME-hard. Thus the computational complexity of satisfiability is not fully characterized yet. Now it is easy to check that every DcPDL formula can be polynomially translated into a $\mathcal{DFO}$ formula. From the discussion above we know that satisfiability in $\mathcal{DFO}$ is EXPTIME-complete. Hence the results in this chapter allow us to precisely characterize the complexity of satisfiability (and thus of validity and logical implication) of DcPDL as EXPTIME-complete, closing the previous gap between the upper bound and the lower bound.

The second logic we consider is called *converse combinatory propositional dynamic logic*, CcPDL, and is essentially converse propositional dynamic logic with nominals. Such logic is not known to be decidable yet, see [90]. Now it is easy to check that every CcPDL can be polynomially translated into a $\mathcal{DIO}$ formula, preserving satisfiability, where $\mathcal{DIO}$ is the propositional dynamic logic corresponding to $\mathcal{CIO}$. From the discussion above we know that satisfiability in $\mathcal{DIO}$ is EXPTIME-complete. Hence the results in this chapter allow us to establish the decidability of CcPDL and to precisely characterize the computational complexity of satisfiability (and hence validity and logical implication) as EXPTIME-complete.

Finally, we remark that, to the best of our knowledge, $\mathcal{DNO}$ is the first logic in which both nominals and graded modalities (qualified number restrictions) are present.

Chapter 8

Recursive Definitions: Fixpoints

There are basically two ways of using and describing classes (concepts). In the first one, which we can call the *prescriptive approach*, the description formalism allows for expressing several properties of a class, thus prescribing constraints that the instances of the class must satisfy. In the second one, which we can call the *definitional approach*, the formalism allows for providing the definition of a class, i.e. a set of properties that precisely characterize the instances of the class. While the prescriptive approach is quite well understood and established, the definitional approach is still the subject of an interesting debate, regarding both its nature and its semantic foundation. In particular, it is well known that there are various possibilities of assigning a meaning to a class definition when it contains some sort of recursion [2, 3, 84, 8, 5].

In this chapter, we are concerned with the semantical problems related to the definitional approach, arguing that, instead of choosing a single style of semantics for the knowledge representation formalism, we achieve better results by adopting a more general formalism that allows for different semantics to coexist.

8.1 Fixpoints

In this section, we briefly recall some notions on fixpoints. The reader is referred to [33] for an introduction to the subject.

Consider the equation $X = f(X)$ where f is an operator from $2^{\mathcal{S}}$ to $2^{\mathcal{S}}$ ($2^{\mathcal{S}}$ denotes the set of all subsets of a set $\mathcal{S}$). The solutions $\mathcal{E}$ of such an equation are called *fixpoints* of the operator f. In general an equation as the one above may have either no solution, a finite number of solutions, or infinite number of them. Among the various solutions, the smallest and the greatest solutions (with respect to set-inclusion) have a prominent position, if they exist. A fundamental result due to Tarski [123] guarantees the existence and the uniqueness of both such solutions in case f is monotonic wrt set-inclusion ($\subseteq$), where f is *monotonic* wrt $\subseteq$ whenever $\mathcal{E}_1 \subseteq \mathcal{E}_2$ implies $f(\mathcal{E}_1) \subseteq f(\mathcal{E}_2)$.

Theorem 50 (Tarski) *Let $\mathcal{S}$ be a set, and f an operator from $2^{\mathcal{S}}$ to $2^{\mathcal{S}}$ that is monotonic wrt $\subseteq$. Then, there is a unique least fixpoint of f given by*

$$\bigcap\{\mathcal{E} \subseteq \mathcal{S} \mid f(\mathcal{E}) \subseteq \mathcal{E}\}$$

and a unique greatest fixpoint of f given by

$$\bigcup\{\mathcal{E} \subseteq \mathcal{S} \mid \mathcal{E} \subseteq f(\mathcal{E})\}.$$

8.2 Concept definitions as equations

It is widely recognized that the notion of TBox as introduced in Chapter 2 can be made more powerful if we allow some sort of concept definitions to be expressed. Let us call *definition statement* (or simply definition), statements of the form:

$$A =_{def} C$$

where A is an atomic concept and C is a concept expression (A cannot occur in the left-hand part of more then one definition). Intuitively, the above definition statement is intended to provide a precise account of A in terms of C. When we specify the semantics of definitions, we need to distinguish between two different types of atomic concepts, namely, *primitive concepts* and *defined concepts*: given a set $\mathcal{D}$ of definitions, primitive concepts are the atomic concepts that do not appear on the left of any definition of $\mathcal{D}$, whereas defined concepts are those that have an associated definition in $\mathcal{D}$. An interpretation $\mathcal{I}$ satisfies a set of definitions if, for each $A =_{def} C$ in the set, $\mathcal{I}$ assigns the same subset of $\Delta^{\mathcal{I}}$ to the defined concept A and to concept C.

We call *recursive definition statements*[1] (or simply recursive definitions), definition statements of the form

$$A =_{def} F(A),$$

where $F(A)$ stands for a concept that has A as a subconcept[2]. From a semantical point of view, a recursive definition $A =_{def} F(A)$ is a sort of equation specifying that, for any interpretation $\mathcal{I}$, the subset of $\Delta^{\mathcal{I}}$ that can be tied to the concept A must satisfy the equation $A^{\mathcal{I}} = (F(A))^{\mathcal{I}}$, i.e. must be one of its *solutions*. Notice that, in general, either none, one, or several subsets of $\Delta^{\mathcal{I}}$ may exist which are solutions of the above equation. For example, it is easy to see that two interpretations that satisfy the statement $A =_{def} P \sqcap \forall R.A$ and that agree on both the concept P and the role R, may differ in the extension assigned to the defined concept A. Notice also that we can associate to a definition statement an operator from subsets of $\Delta^{\mathcal{I}}$ to subsets of $\Delta^{\mathcal{I}}$, such that the solutions of the equation correspond to the fixpoints of the operator. For example to the definition $A =_{def} P \sqcap \forall R.A$ we can associate, for any interpretation $\mathcal{I}$, the operator $\lambda\mathcal{S}.\{s \in \Delta^{\mathcal{I}} \mid s \in P^{\mathcal{I}}$ and $\forall t.(s,t) \in R^{\mathcal{I}}$ implies $t \in \mathcal{S}\}$.

In the literature on concept languages, three semantics for recursive definitions, have been proposed (see [84]):

[1]Terminological cycles in [2, 3, 84]. Note that, for the moment, we do not consider mutual recursive definitions, as $A =_{def} F(B)$, $B =_{def} F'(A)$.

[2]A *subconcept* of a concept C is any substring of C (including C itself) that is a concept, according to the syntax rules.

- the descriptive semantics,
- the least fixpoint semantics,
- the greatest fixpoint semantics.

Let us recall their properties using some examples. According to the descriptive semantics, a recursive definition $A =_{def} F(A)$ is a *constraint* stating that, for any $\mathcal{I}$ satisfying the definition, $A^{\mathcal{I}}$ has to be *any* solution of the equation $A^{\mathcal{I}} = (F(A))^{\mathcal{I}}$. In other words, the meaning assigned to $A =_{def} F(A)$ is the same as that assigned to the equivalence assertion $A \equiv F(A)$. In our example, $A =_{def} P \sqcap \forall R.A$ states that the individuals in the class A are those in the class P that are related by means of R to individuals in A itself, and vice versa, *where A is no better specified.* In fact, the descriptive semantics is not appropriate to properly define recursive concepts. Instead, it is suitable to specify a set of necessary and sufficient conditions that individuals must satisfy in order to be instances of a concept. For example [84], we can express the fact that humans are mammals having two parents that are humans, and, on the converse, that mammals having two parents that are humans are humans themselves, in terms of the equivalence assertion

$$human \equiv mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.human.$$

It is interesting to observe that we may state an analogous property for horses $horse \equiv mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.horse$ without implying any mutual relationship between *human* and *horse*. We will see later on, this is not true if we use a fixpoint semantics for defining these two concepts.

According to the least (greatest) fixpoint semantics, a definition statement of the form $A =_{def} F(A)$ specifies that, in any interpretation $\mathcal{I}$, A is to be interpreted as the smallest (greatest) solution, if it exists, of $A^{\mathcal{I}} = (F(A))^{\mathcal{I}}$. In other words, in order to consider an interpretation $\mathcal{I}$ adequate to give a meaning to $A =_{def} F(A)$, any other interpretation $\mathcal{J}$, agreeing with $\mathcal{I}$ on the primitive concepts and roles, must assign to A a superset (subset) of $A^{\mathcal{I}}$. Let us consider some examples illustrating the differences in the two fixpoint semantics. In our running example $A =_{def} P \sqcap \forall R.A$, the least fixpoint semantics leads to identify A with $\bot$, (indeed the empty set satisfies the statement, and it is obviously the smallest solution), while the greatest fixpoint semantics interprets A as the largest class satisfying the definition, which can be proven to be equivalent to $\forall R^*.P$, where R^* denotes the reflexive and transitive closure of R.

Although the least fixpoint semantics does not help in the above example, it is particularly suitable for providing *inductive definitions* of concepts. Consider the case of a single source finite directed acyclic graph (DAG) defined as follows[3]:

- an EMPTY-DAG is a DAG (base step);

[3]We assume that a leaf of a DAG is a NODE with all arcs leading to a special node called EMPTY-DAG, as opposed to a NODE having no connection at all. Indeed, in the latter case, the definition of *dag* would simplify to $dag =_{def} node \sqcap \forall arc.dag$, hiding the general form of inductive definitions, i.e. base case and inductive case.

- a NODE that has connections and all connections are DAGs, is a DAG (inductive step);
- nothing else is a DAG.

We can write a natural definition statement denoting the class of DAGs, namely

$$dag =_{def} emptydag \sqcup (node \sqcap \exists arc.\top \sqcap \forall arc.dag),$$

as long as we interpret it according to the least fixpoint semantics. Similarly, we can model the class of LISTs (defined inductively as: an EMPTY-LIST is a LIST; a NODE that has exactly one successor that is a LIST is a LIST; nothing else is a LIST) by

$$list =_{def} emptylist \sqcup (node \sqcap (\leq 1 succ.\top) \sqcap \exists succ.list).$$

The greatest fixpoint semantics is well suited for defining classes of individuals whose structure is *non-well-founded* or *co-inductive*. An example is the class of STREAMs, modeling the well-known linear data structure having a NODE as first element, and such that the rest of the structure is a STREAM itself. Note that streams, differently from lists, are infinite sequences of nodes. A natural statement for the definition of stream is

$$stream =_{def} node \sqcap (\leq 1\, succ.\top) \sqcap \exists succ.stream$$

with the proviso that, for every $\mathcal{I}$, we need to associate to $stream^{\mathcal{I}}$ the greatest solution of the corresponding equation.

Notice however that, if we interpret the definition statements

$$\begin{array}{ll} human =_{def} mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.human, \\ horse \;\; =_{def} mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.horse \end{array}$$

by the greatest fixpoint semantics, as well as with least fixpoint semantics, we obtain a rather non-intuitive result: for any interpretation $\mathcal{I}$ satisfying the above definition statements, $human^{\mathcal{I}} = horse^{\mathcal{I}}$.

The above considerations show that the three semantics capture different intuitions, and hence we may need all of them in the same TBox in order to properly model different concepts. Our proposal in this paper is exactly in the direction of reconciling the various semantics in the same TBox. This is pursued by means of a language that incorporates two constructs, $\mu X.F(X)$ and $\nu X.F(X)$ (the symbols $X, Y, \ldots$ stand for concept variables), denoting, respectively, the least fixpoint and the greatest fixpoint of the operator associated with the definition $X =_{def} F(X)$, that is, for every $\mathcal{I}$ satisfying the definition, the smallest solution and the greatest solution of the equation $X^{\mathcal{I}} = (F(X))^{\mathcal{I}}$.

In our approach, definition statements will never appear in a TBox. Instead, as usual a TBox will be simply a set of inclusion assertions that may involve fixpoint constructs. For example, in order to specify the properties of the concepts of *human*, *horse*, *dag*, *list* and *stream*, we can use the equivalence assertions:[4]

[4]Notice that, if we add to this TBox the equivalence assertion $sm \equiv \nu X \,.\, mam \sqcap (\leq 2\, par.\top) \sqcap (\geq 2\, par.\top) \sqcap \forall par.X$, defining the concept sm (sexual mammal), then it turns out that both $human$ and $horse$ are subsumed by sm.

$$\begin{array}{ll} \mathit{dag} & \equiv \mu X\,.\,\mathit{emptydag} \sqcup (\mathit{node} \sqcap \exists \mathit{arc}.\top \sqcap \forall \mathit{arc}.X) \\ \mathit{list} & \equiv \mu X\,.\,\mathit{emptylist} \sqcup (\mathit{node} \sqcap (\leq 1\, \mathit{succ}.\top) \sqcap \exists \mathit{succ}.X) \\ \mathit{stream} & \equiv \nu X.\mathit{node} \sqcap (\leq 1\, \mathit{succ}.\top) \sqcap \exists \mathit{succ}.X \\ \mathit{human} & \equiv \mathit{mam} \sqcap (\leq 2\, \mathit{par}.\top) \sqcap (\geq 2\, \mathit{par}.\top) \sqcap \forall \mathit{par}.\mathit{human} \\ \mathit{horse} & \equiv \mathit{mam} \sqcap (\leq 2\, \mathit{par}.\top) \sqcap (\geq 2\, \mathit{par}.\top) \sqcap \forall \mathit{par}.\mathit{horse}. \end{array}$$

The availability of least and greatest fixpoint constructs not only allows different semantics to be used in the same TBox, but also increases the expressive power of concept definitions. On the one hand, it makes it possible to model not only abstract classes, but also inductively and co-inductively defined data structures, such as dags, lists and streams. This is particularly important if our objective is to integrate class-based representation formalisms and programming systems (declarative or procedural), in order to make these formalisms more useful in practice. On the other hand, we have the possibility of nesting fixpoints, thus going beyond the simple equational format by which we motivated their introduction. As an example, consider the following: Among the inhabitants of the planet "Plonk", a disease called "foo" is quite common. Such a disease manifests itself in two forms: a "visible" one and a "latent" (not visible) one, and it has a rather intricate hereditary pattern. Individuals that have the visible form transmit the visible form to at least one direct descendant (obviously, if there is any direct descendant), these ill descendants in turn do the same, and so on, *until* someone transmits the latent form of the disease. All direct descendants (if any) of an individual that has the latent form inherit the visible form. The pattern goes on like this, generation after generation, *forever*. Notice that, along any chain of descendants, the visible form of the disease sooner or latter is interrupted, because either an individual has no direct descendant or an individual transmits to some descendant the latent form. The hereditary pattern ($\mathit{foo\_hp}$) of the above disease can be defined as follows:

$$\begin{array}{l} \mathit{foo\_hp} \equiv \nu X.\mu Y.((\mathit{visible} \sqcap (\exists \mathit{child}.Y \sqcup \forall \mathit{child}.\bot)) \sqcup \\ \qquad\qquad\qquad (\neg \mathit{visible} \sqcap \forall \mathit{child}.(\mathit{visible} \sqcap X))) \end{array}$$

where $\mathit{visible}$ denotes the visible form of the disease, while $\neg\mathit{visible}$ denotes the latent form.

8.3 The description logic $\mu\mathcal{ALC}$

The first description logic involving fixpoints that we shall study is called $\mu\mathcal{ALC}$, and is obtained by adding the fixpoint constructs to $\mathcal{ALC}$.

In the sequel we make use of notions of scope, bound and free occurrences of variables, closed formulas, etc. The definitions of these notions are the same as the analogues in first-order logic, treating μ and ν as quantifiers.

The primitive symbols in $\mu\mathcal{ALC}$ are atomic concepts, *(concept) variables* (denoted by $X, Y, \ldots$), and atomic roles which are the only roles admitted in the language.

Concepts in $\mu\mathcal{ALC}$ are formed inductively according to the following abstract syntax:

$$C ::= A \mid \top \mid \bot \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \mu X.C \mid \nu X.C \mid X$$

where A denotes an atomic concept, R an atomic role, X a variable. We implicitly assume the restriction that every free occurrence of a variable X is in the scope of an even number of negation signs ($\neg$).

Not all the constructs introduced are independent. The following equalities hold: $\bot = \neg\top$, $\forall R.C = \neg\exists R.\neg C$, $\top = \nu X.X$, $\nu X.C = \neg\mu X.\neg C[X/\neg X]$ (where $C[X/\neg X]$ is the concept obtained substituting all free occurrences of X by the concept $\neg X$).

As usual, an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a domain of interpretation $\Delta^{\mathcal{I}}$, and a interpretation function $\cdot^{\mathcal{I}}$, which maps every atomic concept to a subset of $\Delta^{\mathcal{I}}$, and every atomic role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. But the presence of free variables does not allow us to extend the interpretation function $\cdot^{\mathcal{I}}$ directly to every concept of the language. For this reason we introduce valuations. A *valuation* ρ on an interpretation $\mathcal{I}$, is a mapping from variables to subsets of $\Delta^{\mathcal{I}}$.

Given a valuation ρ, we denote by $\rho[X/\mathcal{E}]$ the valuation identical to ρ except for $\rho[X/\mathcal{E}](X) = \mathcal{E}$. In other words, for every variable Y,

$$\rho[X/\mathcal{E}](Y) = \begin{cases} \mathcal{E} & \text{if } Y = X \\ \rho(Y) & \text{if } Y \neq X \end{cases}$$

Let $\mathcal{I}$ be an interpretation and ρ a valuation on $\mathcal{I}$. We assign meaning to concepts of the language by associating to $\mathcal{I}$ and ρ an *extension function* $\cdot^{\mathcal{I}}_{\rho}$, mapping concepts to subsets of $\Delta^{\mathcal{I}}$, defined as follows:

$$\begin{array}{lcl}
X^{\mathcal{I}}_{\rho} & = & \rho(X) \subseteq \Delta^{\mathcal{I}} \\
A^{\mathcal{I}}_{\rho} & = & A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \\
\top^{\mathcal{I}}_{\rho} & = & \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}}_{\rho} & = & \emptyset \\
(\neg C)^{\mathcal{I}}_{\rho} & = & \Delta^{\mathcal{I}} - C^{\mathcal{I}}_{\rho} \\
(C_1 \sqcap C_2)^{\mathcal{I}}_{\rho} & = & (C_1)^{\mathcal{I}}_{\rho} \cap (C_2)^{\mathcal{I}}_{\rho} \\
(C_1 \sqcup C_2)^{\mathcal{I}}_{\rho} & = & (C_1)^{\mathcal{I}}_{\rho} \cup (C_2)^{\mathcal{I}}_{\rho} \\
(\exists R.C)^{\mathcal{I}}_{\rho} & = & \{s \in \Delta^{\mathcal{I}} \mid \exists s'.\ (s,s') \in R^{\mathcal{I}} \text{ and } s' \in C^{\mathcal{I}}_{\rho}\} \\
(\forall R.C)^{\mathcal{I}}_{\rho} & = & \{s \in \Delta^{\mathcal{I}} \mid \forall s'.\ (s,s') \in R^{\mathcal{I}} \text{ implies } s' \in C^{\mathcal{I}}_{\rho}\} \\
(\mu X.C)^{\mathcal{I}}_{\rho} & = & \bigcap\{\mathcal{E} \subseteq \Delta^{\mathcal{I}} \mid \ C^{\mathcal{I}}_{\rho[X/\mathcal{E}]} \subseteq \mathcal{E}\ \} \\
(\nu X.C)^{\mathcal{I}}_{\rho} & = & \bigcup\{\mathcal{E} \subseteq \Delta^{\mathcal{I}} \mid \ \mathcal{E} \subseteq \ C^{\mathcal{I}}_{\rho[X/\mathcal{E}]}\ \}
\end{array}$$

We remark that, in the last two cases $C^{\mathcal{I}}_{\rho[X/\mathcal{E}]}$ is interpreted as an operator from subsets $\mathcal{E}$ of $\Delta^{\mathcal{I}}$ to subsets of $\Delta^{\mathcal{I}}$. By the syntactic restriction enforced on variables, such an operator is guaranteed to be monotonic wrt $\subseteq$. Notice also that free variables appearing in a concept are interpreted more or less as atomic concepts.

A concept C is satisfiable, if there exists an interpretation $\mathcal{I}$ and a valuation ρ on $\mathcal{I}$ such that $C^{\mathcal{I}}_{\rho} \neq \emptyset$, otherwise the concept is unsatisfiable. A concept C_1 is subsumed by a concept C_2, written as $C_1 \sqsubseteq C_2$, if for every interpretation $\mathcal{I}$ and every valuation ρ on $\mathcal{I}$, $(C_1)^{\mathcal{I}}_{\rho} \subseteq (C_2)^{\mathcal{I}}_{\rho}$.

A $\mu\mathcal{ALC}$ TBox is a finite set (possibly empty) of inclusion assertions $C_1 \sqsubseteq C_2$ where C_1 and C_2 are *closed* concepts of $\mu\mathcal{ALC}$. [5]

[5] As usual, we use equivalence assertions of the form $C_1 \equiv C_2$ as an abbreviation for $\{C_1 \sqsubseteq C_2, C_2 \sqsubseteq C_1\}$.

An interpretation $\mathcal{I}$ satisfies an inclusion assertion $C_1 \sqsubseteq C_2$, if $(C_1)^{\mathcal{I}}_{\rho} \subseteq (C_2)^{\mathcal{I}}_{\rho}$, where ρ is any valuation on $\mathcal{I}$ (being C_1 and C_2 closed, and hence independent from valuations). $\mathcal{I}$ is a model of a TBox $\mathcal{K}$, if $\mathcal{I}$ satisfies all inclusion assertions in $\mathcal{K}$. We say that a TBox $\mathcal{K}$ is satisfiable, if it has a model. Observe that inclusion assertions in $\mathcal{K}$ are interpreted according to the descriptive semantics.

We say that a TBox $\mathcal{K}$ logically implies an inclusion assertion $C_1 \sqsubseteq C_2$, written $\mathcal{K} \models C_1 \sqsubseteq C_2$, if for every model $\mathcal{I}$ of $\mathcal{K}$ and every valuation ρ on $\mathcal{I}$, $(C_1)^{\mathcal{I}}_{\rho} \subseteq (C_2)^{\mathcal{I}}_{\rho}$.

8.4 Properties of the fixpoint constructs

In the following, we use the notation $C(X)$ to indicate that the variable X occurs free in the concept C (other variables could occur free in C as well), and the notation $C(D)$, where D is a concept, as a shorthand for $C(X)[X/D]$. In addition, we use the symbol σ as an abstraction for either μ or ν.

Let us comment briefly on some simple properties of the logic. First, the concept $\sigma X.C(X)$ is equivalent to the concept $\sigma Y.C(Y)$, as long as Y is free for X in $C(X)$. Second, the extension function $\cdot^{\mathcal{I}}_{\rho}$ gives to a closed concept a value which is independent of the actual valuation ρ. Hence $\sigma X.C$, where X does not occur in C, is equivalent to C. Third, since $\sigma X.C(X)$ is a fixpoint we have that $C(\sigma X.C(X))$ is equivalent to $\sigma X.C(X)$. Furthermore, we have that the concept $\mu X.C(X)$ is always subsumed by the concept $\nu X.C(X)$.

The next property is more substantial. Consider a $\mu\mathcal{ALC}$ TBox $\mathcal{K}$ containing the two equivalence assertions

$$dag\_of\_student \equiv \mu X\ .\ emptydag \sqcup (student \sqcap \exists arc.\top \sqcap \forall arc.X)$$

$$dag\_of\_person \equiv \mu X\ .\ emptydag \sqcup (person \sqcap \exists arc.\top \sqcap \forall arc.X)$$

defining the concepts *dag_of_student* and *dag_of_person* as the classes of DAGs whose nodes are students and persons respectively. Assuming that students are persons, we want to be able to infer that DAGs of students are DAGs of persons as well. That is we want

$$\mathcal{K} \models student \sqsubseteq person \ \text{ implies } \ \mathcal{K} \models dag\_of\_student \sqsubseteq dag\_of\_person.$$

It turns out that for $\mu\mathcal{ALC}$ such a property holds. To prove this we introduce the following lemma, first.

Lemma 51 *Let $\mathcal{K}$ be a $\mu\mathcal{ALC}$ TBox, and C and D two $\mu\mathcal{ALC}$ concepts in which a variable X may occur free. Then*

$$\mathcal{K} \models C \sqsubseteq D \quad \textit{implies} \quad \mathcal{K} \models \sigma X.C \sqsubseteq \sigma X.D.$$

Proof We proceed by contradiction.[6] Assume that $C^{\mathcal{I}}_{\rho} \subseteq D^{\mathcal{I}}_{\rho}$ holds for all models $\mathcal{I}$ of $\mathcal{K}$ and all valuations ρ on $\mathcal{I}$. And suppose that there exists a model $\mathcal{I}$ of $\mathcal{K}$ and a valuation ρ on $\mathcal{I}$ such that $(\sigma X.C)^{\mathcal{I}}_{\rho} \not\subseteq (\sigma X.D)^{\mathcal{I}}_{\rho}$.

[6]For uniformity, we do not distinguish if X occurs free or not. Obviously if X does not occur free, the result is trivial.

First we prove the result for $\sigma = \mu$. Let s be an individual in $(\mu X.C)^{\mathcal{I}}_{\rho}$ but not in $(\mu X.D)^{\mathcal{I}}_{\rho}$. Now, we have

$$s \in (\mu X.C)^{\mathcal{I}}_{\rho} \text{ iff } \forall \mathcal{E} \subseteq \Delta^{\mathcal{I}} . \, (C^{\mathcal{I}}_{\rho[X/\mathcal{E}]} \subseteq \mathcal{E} \text{ implies } s \in \mathcal{E}) \tag{8.1}$$

$$s \notin (\mu X.D)^{\mathcal{I}}_{\rho} \text{ iff } \exists \mathcal{E}' \subseteq \Delta^{\mathcal{I}} . \, (D^{\mathcal{I}}_{\rho[X/\mathcal{E}']} \subseteq \mathcal{E}' \text{ and } s \notin \mathcal{E}'). \tag{8.2}$$

For the set $\mathcal{E}'$ in (8.2), the following expression holds:

$$C^{\mathcal{I}}_{\rho[X/\mathcal{E}']} \subseteq D^{\mathcal{I}}_{\rho[X/\mathcal{E}']} \subseteq \mathcal{E}'$$

hence by (8.1) we have $s \in \mathcal{E}'$ and by (8.2) we have $s \notin \mathcal{E}'$, which is impossible.

The proof for $\sigma = \nu$ is similar. Let s be an individual in $(\nu X.C)^{\mathcal{I}}_{\rho}$ but not in $(\nu X.D)^{\mathcal{I}}_{\rho}$. Now, we have

$$s \in (\nu X.C)^{\mathcal{I}}_{\rho} \text{ iff } \exists \mathcal{E}'' \subseteq \Delta^{\mathcal{I}} . \, (\mathcal{E}'' \subseteq C^{\mathcal{I}}_{\rho[X/\mathcal{E}'']} \text{ and } s \in \mathcal{E}'') \tag{8.3}$$

$$s \notin (\nu X.D)^{\mathcal{I}}_{\rho} \text{ iff } \forall \mathcal{E} \subseteq \Delta^{\mathcal{I}} . \, (\mathcal{E} \subseteq D^{\mathcal{I}}_{\rho[X/\mathcal{E}]} \text{ implies } s \notin \mathcal{E}). \tag{8.4}$$

For the set $\mathcal{E}''$ in (8.3), the following expression holds:

$$\mathcal{E}'' \subseteq C^{\mathcal{I}}_{\rho[X/\mathcal{E}'']} \subseteq D^{\mathcal{I}}_{\rho[X/\mathcal{E}'']}$$

hence by (8.3) we have $s \in \mathcal{E}''$ and by (8.4) we have $s \notin \mathcal{E}''$, which is impossible. $\Box$

By using this lemma we can prove the result we are looking for.

Theorem 52 *Let $\mathcal{K}$ be a $\mu\mathcal{ALC}$ TBox, and $D(X)$ a $\mu\mathcal{ALC}$ concept such that every occurrence of the variable X in $D(X)$ is in the scope of an even number of negation signs. Then, for any $\mu\mathcal{ALC}$ concepts C_1 and C_2:*

$$\mathcal{K} \models C_1 \sqsubseteq C_2 \textit{ implies } \mathcal{K} \models D(C_1) \sqsubseteq D(C_2).$$

Proof First, we transform $D(X)$ in "negation normal form", that is we push the negations occurring in $D(X)$ all way in, getting an equivalent concept $D^n(X)$ where negations occur only in front of atomic concepts and no negation occur in front X.

Now we prove the result by induction on the formation of $D^n(X)$. Base case. If $D^n(X) = X$, the result holds trivially.

Inductive case. We assume that the result holds for every subconcept of $D^n(X)$, and we show that $\mathcal{K} \models D^n(C_1) \sqsubseteq D^n(C_2)$ holds as well. Indeed this easily follows from the semantics, for $D^n(X)$ of the forms

$$D^n_1(X) \sqcap D^n_2(X) \mid D^n_1(X) \sqcup D^n_2(X) \mid \exists R.D^n_1(X) \mid \forall R.D^n_1(X).$$

It remains to prove the result for $D^n(X) = \sigma Y.D^n_1(X)$ $(Y \neq X)$, but by Theorem 51 we have

$$\mathcal{K} \models D^n_1(C_1) \sqsubseteq D^n_1(C_2) \text{ implies } \mathcal{K} \models \sigma Y.D^n_1(C_1) \sqsubseteq \sigma Y.D^n_1(C_2),$$

hence we are done. $\Box$

Going back to our the example, we can, in fact, infer that DAGs of students are also DAGs of persons. Indeed, by Theorem 52, we have that $\mathcal{K} \models student \sqsubseteq person$ implies $\mathcal{K} \models \mu X.emptydag \sqcup (student \sqcap \exists arc.\top \sqcap \forall arc.X) \sqsubseteq \mu X.emptydag \sqcup (person \sqcap \exists arc.\top \sqcap \forall arc.X)$.

Even though it does not include any role construct, $\mu\mathcal{ALC}$ is actually an extension of $\mathcal{C}$. Indeed we can translate a $\mathcal{C}$ concept into a $\mu\mathcal{ALC}$ concept by resorting to the following equivalences:

$$\begin{array}{l} \exists R_1 \circ R_2.C = \exists R_1.\exists R_2.C \\ \exists R_1 \sqcup R_2.C = \exists R_1.C \sqcup \exists R_2.C \\ \exists R^*.C = \mu X.(C \sqcup \exists R.X) \\ \exists id(D).C = C \sqcap D. \end{array}$$

Note that $\forall R^*.C = \nu X.(C \sqcap \forall R.X)$.

8.5 Reasoning in $\mu\mathcal{ALC}$

In this section, we focus on the problem of reasoning in $\mu\mathcal{ALC}$ TBoxes. We start our discussion by showing that logical implication in $\mu\mathcal{ALC}$ TBoxes (thus also satisfiability of $\mu\mathcal{ALC}$ TBoxes) is reducible to unsatisfiability of *a single $\mu\mathcal{ALC}$ concept.* To prove this result, we introduce the notions of generated sub-interpretation and sub-valuation.[7]

Let $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ be an interpretation, ρ a valuation on $\mathcal{I}$, and $s \in \Delta^\mathcal{I}$ an individual. We define the interpretation $\mathcal{I}^s = (\Delta^{\mathcal{I}^s}, \cdot^{\mathcal{I}^s})$, and the valuation ρ^s on $\mathcal{I}^s$, as follows:

- $\Delta^{\mathcal{I}^s} = \{s' \in \Delta^\mathcal{I} \mid (s, s') \in (R_1^\mathcal{I} \cup \ldots \cup R_m^\mathcal{I})^*\}$.
- For each atomic role R_i, we have $R_i^{\mathcal{I}^s} = R_i^\mathcal{I} \cap (\Delta^{\mathcal{I}^s} \times \Delta^{\mathcal{I}^s})$.
- For each atomic concept A, we have $A^{\mathcal{I}^s} = A^\mathcal{I} \cap \Delta^{\mathcal{I}^s}$.
- For each variable X, we have $\rho^s(X) = \rho(X) \cap \Delta^{\mathcal{I}^s}$.

We call $\mathcal{I}^s$ *the sub-interpretation of* $\mathcal{I}$ *generated by* s, and ρ^s *the sub-valuation of* ρ *generated by* s.

For generated sub-interpretations and sub-valuations we can state the following lemma.

Lemma 53 *Let C be a $\mu\mathcal{ALC}$ concept. Then for any interpretation $\mathcal{I}$, any valuation ρ on $\mathcal{I}$, and any individual $s \in \Delta^\mathcal{I}$, we have: $s \in C_\rho^\mathcal{I}$ iff $s \in C_{\rho^s}^{\mathcal{I}^s}$.*

Proof Without loss of generality, we consider concepts formed according to the following simplified abstract syntax: $C ::= A \mid \bot \mid \neg C \mid C_1 \sqcap C_2 \mid \exists R.C \mid \mu X.C \mid X$.

[7] Together these notions play the same role as that of generated sub-model in modal logics.

We prove the result by induction on the number of nested fixpoint constructs. Base case. If in C there are no fixpoint constructs, the thesis can be proven by induction on the formation of C.

Inductive case. We assume that the thesis holds for concepts C with n nested fixpoint constructs, and we prove it for concepts $\mu X.C$ with $n+1$. We recall that, by Tarski-Knaster Theorem on fixpoints [123], $s \in (\mu X.C)^{\mathcal{I}}_{\rho}$ iff there exists an ordinal α such that $s \in (\mu_{\alpha} X.C)^{\mathcal{I}}_{\rho}$, where $(\mu_{\alpha} X.C)^{\mathcal{I}}_{\rho}$ is defined by transfinite induction as

- $(\mu_0 X.C)^{\mathcal{I}}_{\rho} = \emptyset$
- $(\mu_{\alpha+1} X.C)^{\mathcal{I}}_{\rho} = C^{\mathcal{I}}_{\rho[X/(\mu_{\alpha} X.C)^{\mathcal{I}}_{\rho}]}$
- $(\mu_{\lambda} X.C)^{\mathcal{I}}_{\rho} = \bigcup_{\alpha<\lambda} (\mu_{\alpha} X.C)^{\mathcal{I}}_{\rho}$, if λ is a limit ordinal.

Hence we proceed by transfinite induction on ordinals α.

Base case of the transfinite induction. $\mu_0 X.C$ is defined as $\bot$, thus trivially we have $s \in (\mu_0 X.C)^{\mathcal{I}}_{\rho}$ iff $s \in (\mu_0 X.C)^{\mathcal{I}^s}_{\rho^s}$.

Successor case of the transfinite induction. We want to show that $s \in (\mu_{\alpha+1} X.C)^{\mathcal{I}}_{\rho}$ iff $s \in (\mu_{\alpha+1} X.C)^{\mathcal{I}^s}_{\rho^s}$, which reduces to

$$s \in C^{\mathcal{I}}_{\rho[X/(\mu_{\alpha} X.C)^{\mathcal{I}}_{\rho}]} \text{ iff } s \in C^{\mathcal{I}^s}_{\rho^s[X/(\mu_{\alpha} X.C)^{\mathcal{I}^s}_{\rho^s}]}. \tag{8.5}$$

To prove this, we start by showing that

$$s \in C^{\mathcal{I}^s}_{\rho^s[X/(\mu_{\alpha} X.C)^{\mathcal{I}^s}_{\rho^s}]} \text{ iff } s \in C^{\mathcal{I}^s}_{(\rho[X/(\mu_{\alpha} X.C)^{\mathcal{I}}_{\rho}])^s}. \tag{8.6}$$

Notice that the two valuations above may differ only on the value of X. If it holds that

$$s \in X^{\mathcal{I}^s}_{\rho^s[X/(\mu_{\alpha} X.C)^{\mathcal{I}^s}_{\rho^s}]} \text{ iff } s \in X^{\mathcal{I}^s}_{(\rho[X/(\mu_{\alpha} X.C)^{\mathcal{I}}_{\rho}])^s}, \tag{8.7}$$

then by straightforward induction on the formation of C we have that (8.6) holds as well. Let us prove (8.7). We can write it as

$$s \in \rho^s[X/(\mu_{\alpha} X.C)^{\mathcal{I}^s}_{\rho^s}](X) \text{ iff } s \in (\rho[X/(\mu_{\alpha} X.C)^{\mathcal{I}}_{\rho}])^s(X),$$

and since $s \in \Delta^{\mathcal{I}^s}$, this reduces to

$$s \in (\mu_{\alpha} X.C)^{\mathcal{I}^s}_{\rho^s} \text{ iff } s \in (\mu_{\alpha} X.C)^{\mathcal{I}}_{\rho}.$$

which holds by transfinite inductive hypothesis.

Now, since C contains n fixpoint constructs, by inductive hypothesis on n, we have

$$s \in C^{\mathcal{I}}_{\rho[X/(\mu_{\alpha} X.C)^{\mathcal{I}}_{\rho}]} \text{ iff } s \in C^{\mathcal{I}^s}_{(\rho[X/(\mu_{\alpha} X.C)^{\mathcal{I}}_{\rho}])^s}.$$

Hence, considering (8.5) and (8.6), it follows that indeed $s \in (\mu_{\alpha+1} X.C)^{\mathcal{I}}_{\rho}$ iff $s \in (\mu_{\alpha+1} X.C)^{\mathcal{I}^s}_{\rho^s}$.

Limit case of the transfinite induction. Let λ be a limit ordinal, then $s \in (\mu_\lambda X.C)^{\mathcal{I}}_\rho$ iff there exists an ordinal $\alpha < \lambda$ such that $s \in (\mu_\alpha X.C)^{\mathcal{I}}_\rho$. By transfinite induction hypothesis, it holds that $s \in (\mu_\alpha X.C)^{\mathcal{I}}_\rho$ iff $s \in (\mu_\alpha X.C)^{\mathcal{I}^s}_{\rho^s}$, and thus

$$s \in (\mu_\lambda X.C)^{\mathcal{I}}_\rho \text{ iff } s \in (\mu_\lambda X.C)^{\mathcal{I}^s}_{\rho^s}.$$

This completes the transfinite induction. So for all ordinals α it holds that

$$s \in (\mu_\alpha X.C)^{\mathcal{I}}_\rho \text{ iff } s \in (\mu_\alpha X.C)^{\mathcal{I}^s}_{\rho^s}.$$

The induction on the nesting of fixpoint constructs is completed as well, hence we have proven the lemma. □

Now we are ready to state the result mentioned above.

Theorem 54 *Let $\mathcal{K} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$ be a $\mu\mathcal{ALC}$ TBox, and C and D two $\mu\mathcal{ALC}$ concepts. Then $\mathcal{K} \models C \sqsubseteq D$ if and only if the $\mu\mathcal{ALC}$ concept*

$$\nu X.(\forall R_1.X \sqcap \ldots \sqcap \forall R_m.X \sqcap C_{\mathcal{K}}) \sqcap C \sqcap \neg D \qquad (8.8)$$

is unsatisfiable, where $R_1, \ldots, R_m$ are all the atomic roles appearing in $\mathcal{K}$, and $C_{\mathcal{K}} = (\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$.

Proof If part. By contradiction. Assume that (8.8) is not satisfiable, and suppose that $\mathcal{K} \not\models C \sqsubseteq D$, i.e. there exists an interpretation $\mathcal{I}$, and a valuation ρ on $\mathcal{I}$, such that $\mathcal{I}$ is a model of $\mathcal{K}$ and $C^{\mathcal{I}}_\rho \not\subseteq D^{\mathcal{I}}_\rho$. It follows that, there exists an individual $s \in \Delta^{\mathcal{I}}$ such that $s \in C^{\mathcal{I}}_\rho$ and $s \in (\neg D)^{\mathcal{I}}_\rho$. On the other hand, the fact that $\mathcal{I}$ is a model of $\mathcal{K}$ implies that $(C_{\mathcal{K}})^{\mathcal{I}}_\rho = \Delta^{\mathcal{I}}$, and thus that $(\nu X.(\forall R_1.X \sqcap \ldots \sqcap \forall R_m.X \sqcap C_{\mathcal{K}}))^{\mathcal{I}}_\rho = \Delta^{\mathcal{I}}$. So we have $s \in (\nu X.(\forall R_1.X \sqcap \ldots \sqcap \forall R_m.X \sqcap C_{\mathcal{K}}) \sqcap C \sqcap \neg D)^{\mathcal{I}}_\rho$, i.e. (8.8) is satisfiable, contradicting the hypotheses.

Only If part. Again we proceed by contradiction. Assume $\mathcal{K} \models C \sqsubseteq D$. And suppose that (8.8) is satisfiable, i.e. there exists an interpretation $\mathcal{I}$, a valuation ρ on $\mathcal{I}$, and an individual $s \in \Delta^{\mathcal{I}}$, such that $s \in (\nu X.(\forall R_1.X \sqcap \ldots \sqcap \forall R_m.X \sqcap C_{\mathcal{K}}) \sqcap C \sqcap \neg D)^{\mathcal{I}}_\rho$.

Now consider the sub-interpretation $\mathcal{I}^s = (\Delta^{\mathcal{I}^s}, \cdot^{\mathcal{I}^s}_{\rho^s})$ and the sub-valuation ρ^s on $\mathcal{I}^s$ generated by s. On the one hand, we clearly have that $(C_{\mathcal{K}})^{\mathcal{I}^s}_{\rho^s} = \Delta^{\mathcal{I}^s}$, hence $\mathcal{I}^s$ is a model of $\mathcal{K}$. On the other hand by Lemma 53 $s \in (\nu X.(\forall R_1.X \sqcap \ldots \sqcap \forall R_m.X \sqcap C_{\mathcal{K}}) \sqcap C \sqcap \neg D)^{\mathcal{I}^s}_{\rho^s}$, so it follows that $\mathcal{I}^s$ and ρ^s do not satisfy the subsumption $C \sqsubseteq D$, contradicting the hypotheses. □

This result allows us to limit our attention to concept unsatisfiability only. In order to devise a method to check a $\mu\mathcal{ALC}$ concept for unsatisfiability, we exhibit a correspondence between $\mu\mathcal{ALC}$ and a well-known logic of programs called *modal mu-calculus* ([71, 73, 121, 122]), which has been recently investigated for expressing temporal properties of reactive and parallel processes ([118, 75, 28, 132, 31]).

Formulas $\Phi, \Psi, \ldots$ of modal mu-calculus are formed inductively from atomic formulas $A, \ldots$ and variables $X, \ldots$ according to the following abstract syntax:

$$\Phi, \Psi ::= A \mid \top \mid \bot \mid \neg\Phi \mid \Phi \wedge \Psi \mid \Phi \vee \Psi \mid < a > \Phi \mid [a]\Phi \mid \mu X.\Phi \mid \nu X.\Phi \mid X$$

where a is the generic element of a set of labels $\mathcal{L}$, and every occurrence of any variable X must be in the scope of an even number of negation signs. The semantics of modal mu-calculus is based on the notions of (Kripke) structure and valuation. A *Kripke structure* $\mathcal{M}$ is a triple $(\mathcal{S}, \{\mathcal{R}_i \mid i \in \mathcal{L}\}, \mathcal{V})$, where $\mathcal{S}$ is a set of states, each $\mathcal{R}_i$ is a binary relation, and $\mathcal{V}$ is a mapping from atomic formulas to subsets of $\mathcal{S}$. A *valuation* ρ on $\mathcal{M}$ is a mapping from variables to subsets of $\mathcal{S}$. To a Kripke structure $\mathcal{M}$ and a valuation ρ on $\mathcal{M}$, we associate an extension function $\cdot_\rho^{\mathcal{M}}$ defined inductively as follows:

$$\begin{array}{lcl}
X_\rho^{\mathcal{M}} & = & \rho(X) \subseteq \mathcal{S} \\
A_\rho^{\mathcal{M}} & = & \mathcal{V}(A) \subseteq \mathcal{S} \\
\top_\rho^{\mathcal{M}} & = & \mathcal{S} \\
\bot_\rho^{\mathcal{M}} & = & \emptyset \\
(\neg\Phi)_\rho^{\mathcal{M}} & = & \mathcal{S} - \Phi_\rho^{\mathcal{M}} \\
(\Phi \wedge \Psi)_\rho^{\mathcal{M}} & = & \Phi_\rho^{\mathcal{M}} \cap \Psi_\rho^{\mathcal{M}} \\
(\Phi \vee \Psi)_\rho^{\mathcal{M}} & = & \Phi_\rho^{\mathcal{M}} \cup \Psi_\rho^{\mathcal{M}} \\
(< a > \Phi)_\rho^{\mathcal{M}} & = & \{s \in \mathcal{S} \mid \exists s'.\ (s, s') \in \mathcal{R}_a \text{ and } s' \in \Phi_\rho^{\mathcal{M}}\} \\
([a]\Phi)_\rho^{\mathcal{M}} & = & \{s \in \mathcal{S} \mid \forall s'.\ (s, s') \in \mathcal{R}_a \text{ implies } s' \in \Phi_\rho^{\mathcal{M}}\} \\
(\mu X.\Phi)_\rho^{\mathcal{M}} & = & \bigcap\{\mathcal{E} \subseteq \mathcal{S} \mid \quad \Phi_{\rho[X/\mathcal{E}]}^{\mathcal{M}} \subseteq \mathcal{E}\ \} \\
(\nu X.\Phi)_\rho^{\mathcal{M}} & = & \bigcup\{\mathcal{E} \subseteq \mathcal{S} \mid \quad \mathcal{E} \subseteq \ \Phi_{\rho[X/\mathcal{E}]}^{\mathcal{M}}\ \}
\end{array}$$

A formula Φ is *satisfiable* if there exists a Kripke structure $\mathcal{M}$ and a valuation ρ on $\mathcal{M}$ such that $\Phi_\rho^{\mathcal{M}} \neq \emptyset$.

The following theorem is the basis for the correspondence between $\mu\mathcal{ALC}$ and the modal mu-calculus.

Theorem 55 *There exists a one-to-one linear-time function q mapping concepts of $\mu\mathcal{ALC}$ to formulas of modal mu-calculus such that for any $\mu\mathcal{ALC}$ concept C, C is satisfiable if and only if $q(C)$ is satisfiable.*

Proof We can define q in the following way: $q(A) = A$ (atomic concepts are mapped to atomic formulas), $q(X) = X$, $q(\top) = \top$, $q(\bot) = \bot$, $q(\neg C) = \neg q(C)$, $q(\exists R.C) =< R > q(C)$ (atomic roles are mapped to labels), $q(\forall R.C) = [R]q(C)$, $q(\mu X.C) = \mu X.q(C)$, and $q(\nu X.C) = \nu X.q(C)$.

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is equivalent to a Kripke structure $\mathcal{M} = (\mathcal{S}, \{\mathcal{R}_i \mid i \in \mathcal{L}\}, \mathcal{V})$ such that: $\mathcal{S} = \Delta^{\mathcal{I}}$; $\{\mathcal{R}_i \mid i \in \mathcal{L}\}$ is equal to the part of $\cdot^{\mathcal{I}}$ interpreting atomic roles; and $\mathcal{V}$ is equal to the part of $\cdot^{\mathcal{I}}$ interpreting atomic concepts. We also have that a valuation ρ on $\mathcal{I}$ is equivalent to a valuation ρ' on $\mathcal{M}$. Now both the extension function associated with $\mathcal{I}$ and ρ, and the extension function associated with $\mathcal{M}$ and ρ' map, respectively, any concept C and the corresponding formula $q(C)$ to the same subset of $\Delta^{\mathcal{I}} = \mathcal{S}$. Hence the thesis follows. $\square$

It follows that we may transfer both decidability and complexity results ([73, 51, 106]) for the modal mu-calculus to $\mu\mathcal{ALC}$. Thus, we can immediately state what is the complexity of reasoning with $\mu\mathcal{ALC}$ concepts and $\mu\mathcal{ALC}$ TBoxes.

Theorem 56 *Satisfiability of $\mu\mathcal{ALC}$ concepts, satisfiability of $\mu\mathcal{ALC}$ TBoxes, and logical implication in $\mu\mathcal{ALC}$ TBoxes are EXPTIME-complete problems.*

Proof Since the satisfiability problem for modal mu-calculus is EXPTIME-complete ([51]), by Theorem 55 the satisfiability of $\mu\mathcal{ALC}$ concepts can be checked in deterministic exponential time (tight bound). Hence, by Theorem 54, the thesis follows. $\Box$

8.6 The description logic $\mu\mathcal{ALCN}$

In this section, study the description logic $\mu\mathcal{ALCN}$, obtained from $\mu\mathcal{ALC}$ by including qualified number restrictions (see Chapter 4).

Concepts in $\mu\mathcal{ALCN}$ are formed inductively according to the following abstract syntax:

$$\begin{array}{rcl} C & ::= & A \mid \top \mid \bot \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \\ & & (\leq n\, R.C) \mid (\geq n\, R.C) \mid \mu X.C \mid \nu X.C \mid X \end{array}$$

where A denotes an atomic concept, R an atomic role, X a variable. We implicitly assume the restriction that every free occurrence of variables X is in the scope of an even number of negations, considering concepts C in $(\leq n\, R.C)$ in the scope of one negation.

Qualified number restrictions are interpreted as follows. Let $\mathcal{I}$ be an interpretation and ρ a valuation on $\mathcal{I}$, and $\cdot^{\mathcal{I}}_{\rho}$ the extension function associated with $\mathcal{I}$ and ρ.

$$\begin{array}{rcl} (\leq n\, R.C)^{\mathcal{I}}_{\rho} & = & \{s \in \Delta^{\mathcal{I}} \mid \text{ there exists at most } n \;\; s' \text{ such that} \\ & & (s,s') \in R^{\mathcal{I}} \text{ and } s' \in C^{\mathcal{I}}_{\rho}\}, \\ (\geq n\, R.C)^{\mathcal{I}}_{\rho} & = & \{s \in \Delta^{\mathcal{I}} \mid \text{ there exists at least } n \;\; s' \text{ such that} \\ & & (s,s') \in R^{\mathcal{I}} \text{ and } s' \in C^{\mathcal{I}}_{\rho}\} \end{array}$$

The other constructs are interpreted as in $\mu\mathcal{ALC}$.

Next we investigate the decidability and the complexity of satisfiability of $\mu\mathcal{ALCN}$ concepts and logical implication in $\mu\mathcal{ALCN}$ TBoxes (and thus of satisfiability of $\mu\mathcal{ALCN}$ TBoxes). As for $\mu\mathcal{ALC}$ these two reasoning tasks are not distinct. Indeed, we can prove the analogue of Theorem 54.

Theorem 57 *Let $\mathcal{K} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$ be a $\mu\mathcal{ALCN}$ TBox, and C and D two $\mu\mathcal{ALCN}$ concepts. Then $\mathcal{K} \models C \sqsubseteq D$ if and only if the $\mu\mathcal{ALCN}$ concept*

$$\nu X.(\forall R_1.X \sqcap \ldots \sqcap \forall R_m.X \sqcap C_{\mathcal{K}}) \sqcap C \sqcap \neg D$$

is unsatisfiable, where $R_1, \ldots, R_m$ are all the roles appearing in $\mathcal{K}$, and $C_{\mathcal{K}} = (\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$.

In order to devise a (effective) method for checking a $\mu\mathcal{ALCN}$ concept for unsatisfiability, we exhibit a correspondence between $\mu\mathcal{ALCN}$ and a variant of modal mu-calculus, called *deterministic modal mu-calculus*, which has the same syntax as the modal mu-calculus, but is interpreted on *deterministic Kripke structures*, that is Kripke structures in which the relations $\mathcal{R}_i$ are partial functions ([121]).

We show that there is a function t mapping concepts of $\mu\mathcal{ALCN}$ to deterministic modal mu-calculus formulae, such that C is satisfiable if and only if $t(C)$ is satisfiable.

The function t is defined inductively. The mapping from A, X, $C \sqcap D$, $C \sqcup D$, $\neg C$, and $\sigma X.C$ is simply $t(A) = A$, $t(X) = X$, $t(C \sqcap D) = t(C) \wedge t(D)$, $t(C \sqcup D) = t(C) \vee t(D)$, $t(\neg C) = \neg t(C)$, $t(\sigma X.C) = \sigma X.t(C)$. The mapping form $\forall R.C$ and $\exists R.C$ is based on a technique developed for propositional dynamic logic to map non-deterministic PDL formulae to deterministic PDL formulae preserving satisfiability ([87, 121]), namely:

$$\begin{array}{rcl} t(\exists R.C) & = & < R > (\mu X.(t(C) \vee < R_{new} > X)), \\ t(\forall R.C) & = & [R](\nu X.(t(C) \wedge [R_{new}]X)), \end{array}$$

where R_{new} is a new role. Finally, $(\leq n\, R.C)$ and $(\geq n\, R.C)$ are mapped to the following formulae (we use the abbreviations $[R^*]\Phi$ for $\nu X.(\Phi \wedge [R]X)$, $[R^+]\Phi$ for $[R][R^*]\Phi$, $< R^* > \Phi$ for $\mu X.(\Phi \vee < R > X)$, and $< R^+ > \Phi$ for $< R >< R^* > \Phi$):

$$\begin{array}{l} t((\leq n\, R.C)) = [R][R^*_{new}](t(C) \Rightarrow [R^+_{new}](t(C) \Rightarrow \\ \qquad [R^+_{new}](\ldots(t(C) \Rightarrow [R^+_{new}]\neg t(C))\ldots)) \end{array}$$

where the number of nested formulae of the form $t(C) \Rightarrow [R^+_{new}]\Phi$ is n, and

$$\begin{array}{l} t((\geq n\, R.C)) = < R >< R^*_{new} > (t(C) \wedge < R^+_{new} > (t(C) \wedge \\ \qquad < R^+_{new} > (\ldots(t(C) \wedge < R^+_{new} > t(C))\ldots)) \end{array}$$

where the number of nested formulae of the form $t(C) \wedge < R^+_{new} > \Phi$ is $n - 1$. These formulae express constraints on the number of states satisfying C along the chain $R \circ R^*_{new}$. For example, consider the concept $(\leq 2\, R.A)$, where A is an atomic concept, $t((\leq 2\, R.A)) = [R][R^*_{new}](A \Rightarrow [R^+_{new}](A \Rightarrow [R^+_{new}]\neg A))$ that means "everywhere along the chain $R \circ R^*_{new}$ there are at most two states where A holds" (see Chapter 4).

Theorem 58 *Let C be a $\mu\mathcal{ALCN}$ concept, and t the function defined above. Then, C is satisfiable if and only if $t(C)$ is satisfiable.*

It is known that satisfiability in deterministic modal mu-calculus is an EXPTIME-complete problem ([121, 51, 106]). Since $t(C)$ is clearly polynomial in the size of C (assuming numbers in C coded in unary), from the above theorem we can derive the decidability and the computational complexity of reasoning with $\mu\mathcal{ALCN}$ TBoxes.

Theorem 59 *Satisfiability of $\mu\mathcal{ALCN}$ concepts, satisfiability of $\mu\mathcal{ALCN}$ TBoxes, and logical implication in $\mu\mathcal{ALCN}$ TBoxes are EXPTIME-complete problems.*

8.7 Discussion

We already noticed that fixpoint constructs allow for representing not only abstract classes, but also several data structures extensively used in application programs. We believe that this characteristic is an important step towards a satisfactory integration of concept languages with both traditional and declarative programming systems.

Indeed the description logics introduced in this chapter provide powerful mechanisms for data structure modeling. In particular, the properties stated in Section 8.4

can be the base to formulate a notion of *parametric concept*[8]. For instance, the expression (named $dag\_of[Z]$)

$$\mu X\ .\ emptydag \sqcup (Z \sqcap \exists arc.\top \sqcap \forall arc.X)$$

where Z is a formal parameter, denotes the class of DAGs whose nodes are left unspecified. This class can be used in several ways in the TBox. For example, it can be instantiated by binding the formal parameter to actual parameters, thus getting, say, $dag\_of[student], dag\_of[person], \ldots$, which are concepts inheriting the properties of $dag\_of[Z]$.

Our proposal of allowing for fixpoint construct explicitly in the formalism is shared by a recent work independently carried out by Schild [109].[9] The main goal of that work is to study both the expressive power and the computational complexity of subsumption and satisfiability for TBoxes expressed in $\mathcal{ALC}$ (no fixpoint constructs), that allow for *mutually* recursive definitions. To this end, a concept language is defined that corresponds to a variant of the modal mu-calculus ([130]) in which *mutual fixpoints* are allowed but some restrictions on nested fixpoints are enforced. It is well known that mutual fixpoints can be re-expressed by means of nested ones (see, for example, [33, 109]). As a consequence of this observation it follows that both logics introduced in this chapter, are actually more expressive than the one analyzed in [109].

We conclude by noting that although the proposed language is very powerful, it lacks the construct for *inverse roles* which is needed for example to correctly capture the notions of (finite) TREE, BINARY-TREE, etc. Indeed, to define the concept of TREE (an EMPTY-TREE is a TREE; a NODE that has at most one parent, some children, and all children are TREEs, is a TREE; nothing else is a TREE) we can write $tree \equiv \mu X\ .\ empty\_tree \sqcup (node \sqcap (\leq 1\, child^{-}.\top) \sqcap \exists child.\top \sqcap \forall child.X$. Notice that the introduction of inverse roles does not pose any difficulty from the semantical point of view; however, its impact on the reasoning method needs to be investigated.

[8]Note that parametric concepts can be introduced also in simpler logics which do not include fixpoint constructs.

[9]In [109] number restrictions are not considered.

Appendix

Eliminating $\mathcal{I}$ from $\mathcal{DI}$

In this appendix we consider two well known propositional dynamic logics, namely $\mathcal{D}$ and $\mathcal{DI}$. $\mathcal{D}$ is the original propositional dynamic logic defined in [56], whereas $\mathcal{DI}$, also defined in [56], extends $\mathcal{D}$ by including the construct to denote the "converse" of a program.

We show that is possible to eliminate the "converse" operator from $\mathcal{DI}$, without compromising the soundness and completeness of inference for it. Specifically we present an elegant reduction of $\mathcal{DI}$ formulae into $\mathcal{D}$ formulae that eliminates the converse programs from a $\mathcal{DI}$ formula but adds enough information so as not to destroy its original meaning with respect to satisfiability, validity, and logical implication. Notably the resulting formula, which is a $\mathcal{D}$ formula, is polynomially related to the original one.

This reduction on the one hand helps in better understanding the nature of the converse operator. On the other hand it puts the basis to build efficient -in practical cases- inference procedures for $\mathcal{DI}$. In fact the reduction, being polynomial, allows one to build efficient inference procedures for $\mathcal{DI}$, by translating $\mathcal{DI}$ formulae into $\mathcal{D}$, and then running an efficient inference procedure for $\mathcal{D}$. We discuss this issue further at the end of the appendix.

The general technique used for deriving the reduction is analogous to the one introduced in Chapter 3 and used to prove many results in this thesis. However the present reduction is probably the best illustration of the technique, since every step is intuitive, and proofs go through without major complexities, thus exhibiting the key features of the technique in a neat way.

A.1 Reducing $\mathcal{DI}$ to $\mathcal{D}$

We now show the reduction from $\mathcal{DI}$ to $\mathcal{D}$. More precisely, we exhibit a mapping ζ from $\mathcal{DI}$ formulae to $\mathcal{D}$ formulae such that, for any $\mathcal{DI}$ formula Φ, Φ is satisfiable if and only if $\zeta(\Phi)$ is satisfiable. The formula $\zeta(\Phi)$, whose size is polynomial in the size of Φ, is said to be the *$\mathcal{D}$-counterpart of* Φ.

We assume without loss of generality that in Φ the converse operator is applied to atomic programs only. It is easy to check that any $\mathcal{DI}$ formula can be transformed in linear time in the size of the formula so that such an assumption is fulfilled.

Definition Let Φ be a $\mathcal{DI}$ formula with the converse operator applied to atomic programs only. We define the *$\mathcal{D}$-counterpart* $\zeta(\Phi)$ of Φ as the conjunction of two formulae, $\zeta(\Phi) = \zeta_1(\Phi) \wedge \zeta_2(\Phi)$, where:

- $\zeta_1(\Phi)$ is obtained from the original formula Φ by replacing each occurrence of P^- with a new atomic program P^c, for all atomic programs P occurring in Φ.
- $\zeta_2(\Phi) = [(P_1 \cup \ldots \cup P_m \cup P_1^c \cup \ldots \cup P_m^c)^*]\zeta_2^1 \wedge \ldots \wedge \zeta_2^g$, where $P_1, \ldots, P_m$ are all atomic programs appearing in Φ, and with one conjunct ζ_2^i of the form

  $$(\phi \Rightarrow [P] < P^c > \phi) \wedge (\phi \Rightarrow [P^c] < P > \phi)$$

  for every $\phi \in CL(\zeta_1(\Phi))$ and $P \in \{P_1, \ldots, P_m\}$.

□

Theorem 60 *Let Φ be a $\mathcal{DI}$ formula, and $\zeta(\Phi)$ its $\mathcal{D}$-counterpart. Then $\zeta(\Phi)$ is a $\mathcal{D}$ formula, and its size is polynomially related to the size of Φ.*

Proof Straightforward. □

The purpose of $\zeta_1(\Phi)$ it to replace the converse of atomic programs (the only converse programs) in Φ with new atomic programs. Each new atomic program P^c is intended to represent P^- (the converse of the atomic program P) in $\zeta_1(\Phi)$.

The purpose of $\zeta_2(\Phi)$ is to force the models M of $\zeta(\Phi)$ so that, for all $\phi \in CL(\zeta_1(\Phi))$, for all states s of M, if ϕ holds in s then all the P-successors of s have a P^c-successor in which ϕ holds, and similarly all the P^c-successors of s have a P-successor in which ϕ holds. We shall show that, as far as satisfiability (but also validity and logical implication) is concerned, this allows us to faithfully represent the converse of P by means of P^c.

First of all, observe that if instead of $\zeta_2(\Phi)$, we imposed the two axiom schemas (ϕ any formula):

$$\phi \Rightarrow [P] < P^c > \phi$$
$$\phi \Rightarrow [P^c] < P > \phi$$

then the models of $\zeta_1(\Phi)$ would be isomorphic to the models of Φ. In fact, the above axiom schemas are identical to the ones used in the axiomatization of $\mathcal{DI}$ to force programs P^- to be the converse of the programs P. However the resulting logics would not be $\mathcal{D}$ but trivially $\mathcal{DI}$.

Instead, $\zeta_2(\Phi)$ can be thought as a finite instantiation of the above two axiom schemas: one instance for each formula in $CL(\Phi)$[1]. Although imposing the validity of such a finite instantiation does not suffice to guarantee the isomorphism of the models

[1]Actually, $\zeta_2(\Phi)$ already takes into account the reduction from logical implication to satisfiability of Theorem 1.

of $\zeta_1(\Phi)$ and Φ, we show that it suffices to guarantee that $\zeta_1(\Phi)$ has a model if and only if Φ has a model.

It is a standard result that if a $\mathcal{DI}$-formula Φ has a model, then it has a connected model, where a model $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ of Φ is a *connected model*, if for some $ss \in \mathcal{S}$:

- $M, ss \models \Phi$;
- $\mathcal{S} = \{t \mid (ss, t) \in (\bigcup_P \mathcal{R}_P \cup \mathcal{R}_{P^-})^*\}$.

Let Φ be either a $\mathcal{DI}$ formula or a $\mathcal{D}$ formula. We call a structure $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ a *structure of* Φ, if every atomic program P and every atomic proposition A occurring in Φ is interpreted in M, i.e. $\mathcal{R}_P$ appears in M, and A appears in the co-domain of Π, respectively.

In the following we use π as an abstraction for both P and P^c. Moreover, π^c denotes P^c if $\pi = P$, and it denotes P if $\pi = P^c$.

Let $M = (\mathcal{S}, \{\mathcal{R}_\pi\}, \Pi)$ be a connected model of $\zeta(\Phi)$. We call the *c-closure* of M, the structure $M' = (\mathcal{S}', \{\mathcal{R}'_\pi\}, \Pi')$ of $\zeta(\Phi)$, defined as follows:

- $\mathcal{S}' = \mathcal{S}$;
- $\mathcal{R}'_\pi = \mathcal{R}_\pi \cup \{(t, s) \mid (s, t) \in \mathcal{R}_{\pi^c}\}$, for each atomic program π in $\zeta(\Phi)$;
- $\Pi' = \Pi$.

Note that in the c-closure M' of a model M, each $\mathcal{R}'_P$ of M' is obtained from $\mathcal{R}_P$ of M by including, for each pair (s, t) in $\mathcal{R}_{P^c}$, the pair (t, s) in $\mathcal{R}'_P$, and similarly each $\mathcal{R}'_{P^c}$ is obtained from R_{P^c} by including, for each pair (s, t) in R_P, the pair (t, s) in $\mathcal{R}'_{P^c}$. As a result in the c-closure of a model each atomic program P^c is interpreted as the converse of P.

The next lemma is the core of the results in the present section. Intuitively it says that the c-closure of a connected model is equivalent to the original model wrt the formulae in $CL(\gamma_1(\Phi))$.

Lemma 61 *Let $M = (\mathcal{S}, \{\mathcal{R}_P\}, \Pi)$ be a connected model of $\zeta(\Phi)$, and $M' = (\mathcal{S}', \{\mathcal{R}'_P\}, \Pi')$ its c-closure. Then, for every $s \in \mathcal{S}\,(= \mathcal{S}')$, and every $\phi \in CL(\zeta_1(\Phi))$:*

$$M, s \models \phi \ \ \textit{iff} \ \ M', s \models \phi.$$

Proof We prove the lemma by induction on the formation of ϕ (called formula induction in the following).

- $\phi = A$.

  $M, s \models A$ iff $A \in \Pi(s)$ iff, by construction of M', $A \in \Pi'(s)$ iff $M', s \models A$.
- $\phi = \neg\phi'$.

  $M, s \models \neg\phi'$ iff $M, s \not\models \phi'$ iff, by formula induction hypothesis, $M', s \not\models \phi'$ iff $M', s \models \neg\phi'$.

- $\phi = \phi_1 \wedge \phi_2$.

  $M, s \models \phi_1 \wedge \phi_2$ iff $M, s \models \phi_1$ and $M, s \models \phi_2$ iff, by formula induction hypothesis, $M', s \models \phi_1$ and $M', s \models \phi_2$ iff $M', s \models \phi_1 \wedge \phi_2$.

- $\phi =< r > \phi'$.

  $\Rightarrow$. $M, s \models< r > \phi'$ iff there is a path $(s = s_0, \ldots, s_q) \in Paths_M(r)$ such that $M, s_q \models \phi'$. We show that $M', s \models< r > \phi'$, by induction on the length of the path (called path induction in the following).

  $q = 0$. In this case $(s = s_0) \in Paths_M(r)$ and $M, s \models \phi'$. Then, by Proposition 4, there exists a formula $< \phi_1?; \ldots; \phi_g? > \phi'$ such that:

  - all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;
  - $M, s \models< \phi_1?; \ldots; \phi_g? > \phi'$;
  - $< \phi_1?; \ldots; \phi_g? > \phi' \Rightarrow< r > \phi'$ is valid.

  By formula induction hypothesis, for every $\phi_x \in \{\phi_1, \ldots, \phi_g, \phi'\}$, we have that $M, s \models \phi_x$ iff $M', s \models \phi_x$. Hence $M', s \models< r > \phi'$.

  $q > 0$. In this case, by Proposition 5, there exists a formula $< \phi_1?; \ldots; \phi_g?; \pi >< r' > \phi'$ such that:

  - all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;
  - $r' \in Post(r)$, and hence $< r' > \phi' \in CL(< r > \phi') \subseteq CL(\zeta_1(\Phi))$;
  - $(s_0, s_1) \in \mathcal{R}_\pi$;
  - $M, s_1 \models< r' > \phi'$;
  - $(s_1, \ldots, s_q) \in Paths_M(r')$;
  - $< \phi_1?; \ldots; \phi_g?; \pi >< r' > \phi' \Rightarrow< r > \phi'$ is valid.

  By formula induction hypothesis, for every $\phi_x \in \{\phi_1, \ldots, \phi_g\}$, we have $M, s_0 \models \phi_x$ iff $M', s_0 \models \phi_x$.

  By construction of M', $(s_0, s_1) \in \mathcal{R}_\pi$ implies $(s_0, s_1) \in \mathcal{R}'_\pi$.

  Considering that $< r' > \phi' \in CL(< r > \phi') \subseteq CL(\zeta_1(\Phi))$, by path induction hypothesis, $M, s_1 \models< r' > \phi'$ and $(s_1, \ldots, s_q) \in Paths_M(r')$ implies $M', s_1 \models< r' > \phi'$.

  Hence $M', s \models< r > \phi'$.

  $\Leftarrow$. $M', s \models< r > \phi'$ iff there is a path $(s = s_0, \ldots, s_q) \in Paths_{M'}(r)$ such that $M', s_q \models \phi'$. We prove that $M, s \models< r > \phi'$, by induction on the length of the path (called path induction in the following).

  $q = 0$. In this case $(s = s_0) \in Paths_{M'}(r)$ and $M', s \models \phi'$. Then, by Proposition 4, there exists a formula $< \phi_1?; \ldots; \phi_g? > \phi'$ such that:

- all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;
- $M', s \models < \phi_1?; \dots; \phi_g? > \phi'$;
- $< \phi_1?; \dots; \phi_g? > \phi' \Rightarrow < r > \phi'$ is valid.

By formula induction hypothesis, for every $\phi_x \in \{\phi_1, \dots, \phi_g, \phi'\}$, we have that $M', s \models \phi_x$ iff $M, s \models \phi_x$. Hence $M, s \models < r > \phi'$.

$q > 0$. In this case, by Proposition 5, there exists a formula $< \phi_1?; \dots; \phi_g?; \pi >< r' > \phi'$ such that:

- all tests $\phi_i?$ occur in r, and hence all ϕ_i are subformulae of $< r > \phi'$;
- $r \in Post(r)$, and hence $< r' > \phi' \in CL(< r > \phi') \subseteq CL(\zeta_1(\Phi))$;
- $(s_0, s_1) \in \mathcal{R}'_\pi$;
- $M', s_1 \models < r' > \phi'$;
- $(s_1, \dots, s_q) \in Paths_{M'}(r')$;
- $< \phi_1?; \dots; \phi_g?; \pi >< r' > \phi' \Rightarrow < r > \phi'$ is valid.

By formula induction hypothesis, for every $\phi_x \in \{\phi_1, \dots, \phi_g\}$, we have $M', s_0 \models \phi_x$ iff $M, s_0 \models \phi_x$.

Considering that $< r' > \phi' \in CL(< r > \phi') \subseteq CL(\zeta_1(\Phi))$, by path induction hypothesis, $M', s_1 \models < r' > \phi'$ and $(s_1, \dots, s_q) \in Paths_{M'}(r')$ implies $M, s_1 \models < r' > \phi'$.

Since $(s_0, s_1) \in \mathcal{R}'_\pi$, by construction of M', we have that either $(s_0, s_1) \in \mathcal{R}_\pi$, or $(s_0, s_1) \notin \mathcal{R}_\pi$ and $(s_1, s_0) \in \mathcal{R}_{\pi^c}$.

- If $(s_0, s_1) \in \mathcal{R}_\pi$, then we can immediately conclude that $M, s_0 \models < r > \phi'$.
- If $(s_0, s_1) \notin \mathcal{R}_\pi$ and $(s_1, s_0) \in \mathcal{R}_{\pi^c}$, then considering that $< r' > \phi'$ is equivalent to a formula $\psi \in CL(\zeta_1(\Phi))$, by $\zeta_2(\Phi)$ we have that

$$M, s_1 \models < r' > \phi' \Rightarrow [\pi^c] < \pi >< r' > \phi'.$$

  Thus there exists a state $s'_1 \in \mathcal{S}$ (different from s_1) such that $(s_0, s'_1) \in \mathcal{R}_\pi$ and $M, s'_1 \models < r' > \phi'$. Hence, also this case, we can conclude that $M, s_0 \models < r > \phi'$.

□

The previous lemma has the following consequence.

Lemma 62 *Let M be a connected model of $\zeta(\Phi)$ and M' its c-closure. Then M' is a model of $\zeta(\Phi)$ as well.*

Proof Let $M = (\mathcal{S}, \{\mathcal{R}_\pi\}, \Pi)$ and $M' = (\mathcal{S}', \{\mathcal{R}'_\pi\}, \Pi')$. By Lemma 61, for all $s \in S = S'$ and all $\phi \in CL(\zeta_1(\Phi))$:

$$M, s \models \phi \text{ iff } M', s \models \phi.$$

Furthermore, by definition of M', $(s,s') \in \mathcal{R}'_\pi$ implies $(s',s) \in \mathcal{R}'_{\pi^c}$. Thus, for all $s \in \mathcal{S}'$ and all $\phi \in CL(\zeta_1(\Phi))$:

$$M', s \models \phi \Rightarrow [P] < P^c > \phi$$
$$M', s \models \phi \Rightarrow [P^c] < P > \phi.$$

Hence we can conclude that the thesis holds. $\square$

We can now formulate the main result of this appendix.

Theorem 63 *A $\mathcal{DI}$ formula Φ is satisfiable iff its $\mathcal{D}$-counterpart $\zeta(\Phi)$ is satisfiable.*

Proof $\Rightarrow$. Let $M^{\mathcal{DI}} = (\mathcal{S}^{\mathcal{DI}}, \{\mathcal{R}_P^{\mathcal{DI}}\}, \Pi^{\mathcal{DI}})$ be a model of Φ. We define a structure $M^{\mathcal{D}} = (\mathcal{S}^{\mathcal{D}}, \{\mathcal{R}_\pi^{\mathcal{D}}\}, \Pi^{\mathcal{D}})$ of $\zeta(\Phi)$ as follows:

- $\mathcal{S}^{\mathcal{D}} = \mathcal{S}^{\mathcal{DI}}$;
- $\mathcal{R}_P^{\mathcal{D}} = \mathcal{R}_P^{\mathcal{DI}}$ and $\mathcal{R}_{P^c}^{\mathcal{D}} = \{(t,s) \mid (s,t) \in \mathcal{R}_P^{\mathcal{DI}}\}$, for all atomic programs P occurring in Φ;
- $\Pi^{\mathcal{D}} = \Pi^{\mathcal{DI}}$.

It is easy to verify that $M^{\mathcal{D}}$ is a model of $\zeta(\Phi)$.

$\Leftarrow$. Let $M^{\mathcal{D}} = (\mathcal{S}^{\mathcal{D}}, \{\mathcal{R}_\pi^{\mathcal{D}}\}, \Pi^{\mathcal{D}})$ be connected model of $\zeta(\Phi)$ and $M^{\mathcal{D}'} = (\mathcal{S}^{\mathcal{D}'}, \{\mathcal{R}_\pi^{\mathcal{D}'}\}, \Pi^{\mathcal{D}'})$ its c-closure. By Lemma 62, M' is a model of $\zeta(\Phi)$ as well.

Observe that, by definition, M' is such that, for each atomic program π, $\mathcal{R}_{\pi^c}^{\mathcal{D}}{}' = (\mathcal{R}_\pi^{\mathcal{D}'})^-$. We define a structure $M^{\mathcal{DI}} = (\mathcal{S}^{\mathcal{DI}}, \{\mathcal{R}_P^{\mathcal{DI}}\}, \Pi^{\mathcal{DI}})$ of $\zeta(\Phi)$ as follows:

- $\mathcal{S}^{\mathcal{DI}} = \mathcal{S}^{\mathcal{D}'}$;
- $\mathcal{R}_P^{\mathcal{DI}} = \mathcal{R}_P^{\mathcal{D}'}$ for all atomic programs P occurring in Φ;
- $\Pi^{\mathcal{DI}} = \Pi^{\mathcal{D}'}$.

It is easy to verify that $M^{\mathcal{DI}}$ is a model of Φ. $\square$

A.2 Discussion

The logics $\mathcal{D}$ and $\mathcal{DI}$ share many characteristics, and many results for $\mathcal{D}$ extend to $\mathcal{DI}$ with no difficulties. For instance the proofs of finite model property and decidability for $\mathcal{D}$ in [56] are easily extended to $\mathcal{DI}$, as well as the proof of EXPTIME-completeness in [94]. However, while efficient – in practical cases – inference procedures have been successfully developed for $\mathcal{D}$, extending them to $\mathcal{DI}$ has proved to be a difficult task, and to the best of our knowledge has been unsuccessful till now.

To be more precise, the inference procedures for $\mathcal{D}$ based on the enumeration of models such as those in [56, 94] can be easily modified to accommodate converse programs. But these procedures are better suited for proving theoretical results than for being used in practice, since they are inherently exponential, not only in the worst-case.

In contrast, inference procedures for $\mathcal{D}$ such as those in [93, 95], based on tableaux methods, which are much more efficient in practical cases, are difficult to modify to cope with converse programs.

The difficulty can be intuitively grasped by observing how these procedures attempt to build a model of a $\mathcal{D}$ formula in order to check its satisfiability. They start by introducing an initial state, and try to make it satisfy the formula. At first, reasoning is carried out locally, i.e. considering subformulae that involve state transitions, simply as atomic propositions. Next, when no more local reasoning is possible, the successor states, introduced by atomic programs, are generated, and the relevant formulae that these states ought to satisfy are propagated. The two steps above are recursively repeated for each successor state until certain termination conditions are met. The key point is that once the successors of a given state have been generated, no more reasoning involving that state will be carried out. Thus, to check satisfiability of a $\mathcal{D}$ formula, a tableaux based procedure can be organized so as to work "forward" only. This feature turns out to be essential in order to ensure efficient termination criteria.

The presence of converse programs does not allow us to extend the above approach in an obvious way. Indeed, reasoning on a state may not be completely carried out locally, i.e. without generating its successors, because, through converse programs, some successors may require further properties to be satisfied by the original state. Therefore, to check satisfiability of a $\mathcal{DI}$ formula, a procedure has to work both "forward" and "backward", thus losing efficiency, since at any point reasoning may involve the whole piece of model built so far.

Is there a way out of this problem? One possible solution is trying to single out a small set of additional formulae to be checked in every state, that, in some sense, anticipate the properties its successors may require at a later stage of the computation.

The reduction from $\mathcal{DI}$ to $\mathcal{D}$ presented in this appendix singles out a set of additional formulae of the kind mentioned above. Hence the reduction can be used as the basis to develop better reasoning procedure for $\mathcal{DI}$, on top of inference procedures for $\mathcal{D}$. In fact, the reduction allows us to build a satisfiability procedure for $\mathcal{DI}$ by simply translating a $\mathcal{DI}$ formula to a $\mathcal{D}$ formula and then running on it a $\mathcal{D}$ satisfiability procedure. Therefore, considering that the reduction is polynomial, by employing an efficient satisfiability procedure for $\mathcal{D}$ we get an efficient satisfiability procedure for $\mathcal{DI}$.

Bibliography

[1] G. Attardi and M. Simi. Consistency and completeness of OMEGA, a logic for knowledge representation. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI-81)*, pages 504–510, 1981.

[2] F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, pages 621–626, 1990.

[3] F. Baader. Augmenting concept languages by transitive closure of roles: an alternative to terminological cycles. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, 1991.

[4] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of the Workshop on Processing Declarative Knowledge (PDK-91)*, Lecture Notes In Artificial Intelligence 567, pages 67–86. Springer-Verlag, 1991.

[5] C. Beeri. A formal approach to object-oriented databases. *Data and Knowledge Engineering*, 5:353–382, 1990.

[6] D. Bell and R. Greenes. Building a semantic network for radiologic records. In *Proceedings of 1993 Spring Congress of the American Medical Informatics Association*, volume 58, 1993.

[7] M. Ben-Ari, J. Y. Halpern, and A. Pnueli. Deterministic propositional dynamic logic: finite models, complexity, and completeness. *Journal of Computer and System Sciences*, 25:402–417, 1982.

[8] D. Beneventano and S. Bergamaschi. Subsumption for complex object data models. In *Proceedings of the 4th International Conference on Database Theory*, Lecture Notes in Computer Science 646, pages 357–375. Springer-Verlag, 1992.

[9] J. Bernauer. Conceptual graphs as an operational model for descriptive findings. In *Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care*, pages 214–218. McGraw-Hill, 1991.

[10] P. Blackburn. Nominal tense logic. *Notre Dame Journal of Formal Logic*, 34(1):56–83, 1993.

[11] P. Blackburn and E. Spaan. A modal perspective on computational complexity of attribute value grammar. *Journal of Logic, Language and Information*, 2:129–169, 1993.

[12] Board of Directors of the American Medical Informatics Association. Standards for medical identifiers, codes, and message needed to create efficient computer-stored medical records. *Journal of the American Medical Informatics Association*, 1(1):1–7, 1994.

[13] A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[14] R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI-84)*, pages 34–37, 1984.

[15] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. Alperin Resnick, and A. Borgida. Living with CLASSIC: when and how to use a KL-ONE-like language. In J. F. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, Los Altos, 1991.

[16] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[17] R. Bull. An approach to tense logic. *Theoria*, 12:171–182, 1970.

[18] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In *Proceedings of the 4th International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 109–120, 1994. Morgan Kaufmann, Los Altos.

[19] K. Campell, A. Das, and M. Musen. A logical foundation for representation of clinical data. *Journal of the American Medical Informatics Association*, 1(3):218–232, 1994.

[20] K. Campell and M. Musen. Creation of a systematic domain for medical care: the need for a comprehensive patient-description vocabulary. In *Proceedings of MEDINFO-92*, pages 1437–1442. North-Holland: Elsevier Science Publishers, 1992.

[21] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[22] B. F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, 1980.

[23] J. Cimino. Controlled medical vocabulary construction: methods from the Canon group. *Journal of the American Medical Informatics Association*, 1(3):296–297, 1994.

[24] J. Cimino and G. Barnett. Automated translation between terminologies using semantic definitions. *MD Comput.*, 7:104–109, 1990.

[25] J. Cimino, R. Barrows, and B. Allen. Adapting ICD9-CM for clinical decision support (abstract). In *Proceedings of 1992 Spring Congress of the American Medical Informatics Association*, 1992.

[26] J. Cimino, P. Clayton, G. Hripcsak, and S. Johonson. Knowledge based approaches to the mantainance of a large controlled medical terminology. *Journal of the American Medical Informatics Association*, 1(1):35–50, 1994.

[27] J. Cimino, G. Hripcsak, S. Johonson, and P. Clayton. Designing an introspective, controlled medical vocabulary. In *Proceedings of the 13th Annual Symposium on Computer Applications in Medical Care*, pages 513–518. IEEE Computer Society Press, 1989.

[28] R. Cleaveland. Tableaux-based model checking in the propositional mu-calculus. *Acta Informatica*, 27:725–747, 1990.

[29] R. Ĉoté, D. Protti, and J. Scherrer. *Role of Informatics in health data coding and classification systems.* North Holland, 1984.

[30] R. Ĉoté, D. Rothwell, J. Palotay, R. Beckett, and L. Broch (eds.). *The Systematized Nomenclature of Medicine: SNOMED International.* College of American Pathologists, 1993.

[31] M. Dam. CTL* and ECTL* as fragments of the modal mu-calculus. Lecture Notes in Computer Science 581, pages 145–164. Springer-Verlag, 1992.

[32] R. Danecki. Nondeterministic propositional dynamic logic with intersection is decidable. In *Proceedings of the 5th Symposium on Computational Theory*, Lecture Notes in Computer Science 208, pages 34–53. Springer-Verlag, 1984.

[33] J. de Bakker. *Mathematical Theory of Program Correctness.* Prentice-Hall, 1980.

[34] G. De Giacomo. Reconciling different semantics for concept definition (extended abstract). In *Proceedings of the 1st COMPULOG net Meeting on Knowledge Representation and Reasoning Systems (CNKRR-93)*, pages 1–5, 1993.

[35] G. De Giacomo. *Decidibilità di formalismi per la rappresentazione della conoscenza basati su classi e loro applicazione ai Medical Terminology Server.* PhD thesis, Università di Roma "La Sapienza", 1995.

[36] G. De Giacomo and M. Lenzerini. A uniform framework for concept definitions. Unpublished manuscript, July 1993.

[37] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pages 205–212. AAAI-Press/The MIT-Press, 1994.

[38] G. De Giacomo and M. Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with mu-calculus. In *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94)*, pages 411–415. John Wiley and Sons, 1994.

[39] G. De Giacomo and M. Lenzerini. Converse, local determinism, and graded nondeterminism in propositional dynamic logics. Technical Report 11-94, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", June 1994.

[40] G. De Giacomo and M. Lenzerini. Description logics with inverse roles, functional restrictions, and n-ary relations. In *Proceedings of the 4th European Workshop on Logics in Artificial Intelligence (JELIA-94)*, Lecture Notes in Artificial Intelligence 838, pages 332–346. Springer-Verlag, 1994.

[41] G. De Giacomo and M. Lenzerini. On the correspondence between description logics and logics of programs (position paper). In *Proceedings of the Description Logics Workshop 1994 (DL-94)*, pages 1–4, 1994.

[42] G. De Giacomo and M. Lenzerini. Enhanced propositional dynamic logic for reasoning about concurrent actions (extended abstract). In *Proceedings of the AAAI 1995 Spring Symposium on Extending Theories of Action: Formal Theory and Practical Applications*, pages 62–67, 1995.

[43] G. De Giacomo and M. Lenzerini. Making $\mathcal{CATS}$ out of kittens: description logics with aggregates. In *Proceedings of the Description Logics Workshop 1995 (DL-95)*, pages 85–87, 1995.

[44] G. De Giacomo and M. Lenzerini. PDL-based framework for reasoning about actions. In *Proceedings of the 4th Congress of the Italian Association for Artificial Intelligence (AIIA-95)*, Lecture Notes in Artificial Intelligence 992, pages 103–114. Springer-Verlag, 1995.

[45] G. De Giacomo and M. Lenzerini. What's in an aggregate: foundations for description logics with tuples and sets. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 801–807, 1995.

[46] M. de Rijke. *Extending Modal Logic.* PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, 1993.

[47] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proceedings of the 2nd International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, pages 151–162. Morgan Kaufmann, Los Altos, 1991.

[48] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 458–463, 1991.

[49] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf. Adding epistemic operators to concept languages. In *Proceedings of the 3rd International Conference on the Principles of Knowledge Representation and Reasoning (KR-92)*, pages 342–353. Morgan Kaufmann, Los Altos, 1992.

[50] J. Doyle and R. Patil. Two theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence Journal*, 48:261–297, 1991.

[51] E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. In *Proceedings of the 20th Annual Symposium on the Foundations of Computer Science*, pages 328–337, 1988.

[52] D. Evans, C. Chute, J. Cimino, et al. CANON: towards a medical concept representation language for electronic medical records (abstract). In *Proceedings of 1993 Spring Congress of the American Medical Informatics Association*, page 26, 1993.

[53] D. Evans, J. Cimino, W. Hersh, S. Huff, D. Bell, and for the Canon Group. Towards a medical concept representation language. *Journal of the American Medical Informatics Association*, 1(3):207–217, 1994.

[54] M. Fattorosi-Barnaba and F. De Caro. Graded modalities I. *Studia Logica*, 44:197–221, 1985.

[55] K. Fine. In so many possible worlds. *Notre Dame Journal of Formal Logic*, 13(4):516–520, 1972.

[56] N. J. Fisher and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[57] C. Friedman, P. Alderson, A. Austin, J. Cimino, and S. Johonson. A general natural-language text processor for clinical radiology. *Journal of the American Medical Informatics Association*, 1(2):161–174, 1994.

[58] G. Gargov and V. Goranko. Modal logic with names. *Journal of Philosophical Logic*, 22:607–636, 1993.

[59] G. Gargov and S. Passy. Determinism and looping in combinatory PDL. *Theoretical Computer Science*, 61:259–277, 1988.

[60] R. Goldblatt. *Logics of time and computation*, volume 7 of *Lecture Notes*. Center for the Study of Language and Information, second edition, 1992.

[61] B. Gordon (ed.). *Current Medical Information and Terminology.* AMA, 1970.

[62] I. Haimowitz, R. Patil, and P. Szolovits. Representing medical knowledge in a terminological language is difficult. In *Proceedings of the 12th Annual Symposium on Computer Applications in Medical Care*, pages 101–105. IEEE Computer Science Press, 1988.

[63] J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[64] D. Harel. Dynamic logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, pages 497–603. D. Reidel Publishing Company, Oxford, 1984.

[65] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. Technical Report RR-91-03, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, 1991. An abridged version appeared in *Proc. of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning (KR-91)*.

[66] G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic*. Methuen, London, 1968.

[67] G. E. Hughes and M. J. Cresswell. *A Companion to Modal Logic*. Methuen, London, 1984.

[68] R. B. Hull and R. King. Semantic database modelling: survey, applications and research issues. *ACM Computing Surveys*, 19(3):201–260, Sept. 1987.

[69] S. Johonson, C. Friedman, J. Cimino, A. Clark, G. Hripcsak, and P. D. Clayton. A conceptual schema for a central patient database. In *Proceedings of the 15th Annual Symposium on Computer Applications in Medical Care*, pages 381–387. McGraw-Hill, 1991.

[70] S. Johonson, C. Friedman, J. Cimino, T. Clark, G. Hripcsak, and P. Clayton. Conceptual data model for a central patient database. In *Proceedings of 1992 Spring Congress of the American Medical Informatics Association*, pages 381–385, 1992.

[71] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–355, 1983.

[72] D. Kozen and R. Parikh. An elementary proof of the completeness of PDL. *Theoretical Computer Science*, 14:113–118, 1981.

[73] D. Kozen and R. Parikh. A decision procedure for the propositional mu-calculus. In *Proceedings of the 2nd Workshop on Logic of Programs*, Lecture Notes in Computer Science 164, pages 313–325. Springer-Verlag, 1983.

[74] D. Kozen and J. Tiuryn. Logics of programs. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 790–840. Elsevier Science Publishers, 1990.

[75] K. J. Larsen. Proof systems for satisfiability in Hennessy-Milner logic with recursion. *Theoretical Computer Science*, 72:265–288, 1990.

[76] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[77] D. Lindberg, B. Humphreys, and A. McClay. The Unified Medical Language System. *Methods in Inf. Med.*, 32:281–291, 1993.

[78] R. MacGregor. The evolving technology of classification-based knowledge representation systems. In J. F. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann, Los Altos, 1991.

[79] F. Masarie, R. Miller, O. Bouhaddou, N. Guise, and H. Warner. An interlingua for electronic interchange of medical information: using frames to map clinical vocabularies. *Comput. Biomed. Res.*, 24(4):379–400, 1991.

[80] National Library of Medicine. *Medical Subject Headings.* 1989.

[81] B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence Journal*, 34(3):371–383, 1988.

[82] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes In Artificial Intelligence. Springer-Verlag, 1990.

[83] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence Journal*, 43:235–249, 1990.

[84] B. Nebel. Terminological cycles: semantics and computational properties. In J. F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, Los Altos, 1991.

[85] W. Nowlan and A. Rector. Medical knowledge representation and predictive data entry. In *Proceedings of the 2nd Artificial Intelligence in Medicine Europe*, Lecture Notes In Medical Informatics 44, pages 105–116. Springer-Verlag, 1991.

[86] R. Parikh. The completeness of propositional dynamic logic. Lecture Notes in Computer Science 64, pages 403–415. Springer-Verlag, 1978.

[87] R. Parikh. Propositional dynamic logic of programs: a survey. In *Proceedings of the 1st Workshop on Logic of Programs*, Lecture Notes in Computer Science 125, pages 102–144. Springer-Verlag, 1981.

[88] D. Park. Fixpoint induction and proofs of program properties. In *Machine Intelligence*, volume 5, pages 59–78. Edinburgh University Press, 1970.

[89] S. Passy and T. Tinchev. PDL with data constraints. *Information Processing Letters*, 20:35–41, 1985.

[90] S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93:263–332, 1991.

[91] P. F. Patel-Schneider. A hybrid, decidable, logic-based knowledge representation system. *Computational Intelligence*, 3(2):64–77, 1987.

[92] C. Peltason. The BACK system – an overview. *SIGART Bulletin*, 2(3):114–119, 1991.

[93] V. R. Pratt. A practical decision method for propositional dynamic logic. In *Proceedings of the 10th Annual Symposium on Theory of Computing*, pages 326–337, 1978.

[94] V. R. Pratt. Models of program logics. In *Proceedings of the 20th IEEE Symposium on the Foundations of Computer Science*, pages 115–122, 1979.

[95] V. R. Pratt. A near-optimal method for reasoning about action. *Journal of Computer and System Sciences*, 20:231–255, 1980.

[96] A. Prior. *Past, Present and Future.* Oxford University Press, 1967.

[97] A. Rassinoux, R. Baud, and J. Scherrer. Conceptual graphs model extension for knowledge representation of medical text. In *Proceedings of MEDINFO-92*, pages 1368–1374. North-Holland: Elsevier Science Publishers, 1992.

[98] A. Rector et alt. *GALEN: Generalized Architecture for Language Encyclopedias and Nomenclatures in Medicine, the master notation, version 1*, 1993.

[99] A. Rector, W. Nowlan, and A. Glowinski. Goals for concept representation in the GALEN project. In *Proceedings of the 17th Annual Symposium on Computer Applications in Medical Care*, pages 414–418. McGraw-Hill, 1993.

[100] A. Rector, W. Nowlan, and S. Kay. Conceptual knowledge: the core of medical information systems. In *Proceedings of MEDINFO-92*, pages 1420–1426. North-Holland: Elsevier Science Publishers, 1992.

[101] A. Rector, W. Solomon, W. Nowlan, and T. Rush. A terminology server for medical language and medical information systems. In *IMIA-94*, page 14, 1994.

[102] A. Rossi Mori, J. Bernauer, V. Pakarinen, A. Rector, P. Robbè, W. Ceusters, P. Hurlen, A. Ogonowski, and H. Olsen. Models for representation of terminologies and coding systems in medicine. In *Proceedings of the Seminar Opportunities for European and U.S.Cooperation in Standardization in Health Care Informatics*, pages 1368–1374, 1992.

[103] A. Rossi Mori et alt. *Medical Informatics - Categorial structure of concept systems - Guideline and vocabularies (CEN/TC251/PT003)*, 1994.

[104] A. Rossi Mori, A. Gangemi, and M. Galanti. The coding cage. In *Proceedings of Medical Informatics Europe*, 1993.

[105] A. Rossi Mori, A. Thornton, and A. Gangemi. An entity-relationship model for a European machine-dictionary of medicine. In *Proceedings of the 14th Annual Symposium on Computer Applications in Medical Care*, pages 185–189. IEEE Computer Society Press, 1990.

[106] S. Safra. On the complexity of ω-automata. In *Proceedings of the 20th Annual Symposium on the Foundations of Computer Science*, pages 319–327, 1988.

[107] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.

[108] K. Schild. A correspondence theory for terminological logics: preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, 1991.

[109] K. Schild. Terminological cycles and the propositional μ-calculus. In *Proceedings of the 4th International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 509–520, 1994. Morgan Kaufmann, Los Altos.

[110] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with unions and complements. Technical Report SR-88-21, FB Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, 1988.

[111] A. Schmiedel. Modeling the medical data of the heart catheter report. Technical report, Project Group Medicine-Informatics, German Heart Center Berlin, 1991.

[112] A. Schmiedel. Using BACK in a medical domain. In *Terminological Logic Users Workshop.* KIT-report 95, University of Berlin, 1991.

[113] A. Schmiedel. Persistent maintenance of object descriptions using BACK. Technical Report KIT 112, University of Berlin, 1993.

[114] J. G. Schmolze. Terminological knowledge representation systems supporting n-ary terms. In *Proceedings of the 1st International Conference on the Principles of Knowledge Representation and Reasoning (KR-89)*, pages 432–443. Morgan Kaufmann, Los Altos, 1989.

[115] G. Schreiber, B. Wielinga, and J. Breuker. *KADS: A principled approach to knowledge-based system development.* Academic Press, 1993.

[116] E. Shortliffe and O. Barnett. Medical data: their acquisition, storage, and use. In E. Shortliffe and L. Perreault, editors, *Medical Informatics: Computer Applications in Health Care. Readings*, pages 37–69. Addison-Wesley, 1990.

[117] E. Spaan. *Complexity of Modal Logics.* PhD thesis, University of Amsterdam, 1993.

[118] C. Stirling. Modal and temporal logic. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 477–563. Clarendon Press, Oxford, 1992.

[119] R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*, 54:121–141, 1982.

[120] R. S. Streett. Fixpoints and program looping: reductions from the propositional mu-calculus into propositional dynamic logics of looping. In *Proceedings of the 4th Workshop on Logic of Programs*, Lecture Notes in Computer Science 193, pages 359–372. Springer-Verlag, 1985.

[121] R. S. Streett and E. A. Emerson. The propositional mu-calculus is elementary. In *Proceedings of the 6th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 172, pages 465–472. Springer-Verlag, 1984.

[122] R. S. Streett and E. A. Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Control*, 81:249–264, 1989.

[123] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[124] United States National Center for Health Statistics. *International Classification of Diseases, Ninth Revision, with Clinical Manifestation.* 1980.

[125] J. Van Benthem and J. Bergstra. Logic of transition systems. *Journal of Logic, Language and Information*, 3(4):247–283, 1995.

[126] J. Van Benthem, J. Van Eijck, and V. Stebletsova. Modal logic, transition systems and processes. *Journal of Logic and Computation*, 4(5):811–855, 1994.

[127] W. Van der Hoek. On the semantics of graded modalities. *Journal of Applied Non-Classical Logics*, 2(1):81–123, 1992.

[128] W. Van der Hoek and M. de Rijke. Counting objects. *Journal of Logic and Computation*, 5(3):325–345, 1995.

[129] M. Y. Vardi. The taming of converse: reasoning about two-way computation. In *Proceedings of the 4th Workshop on Logic of Programs*, Lecture Notes in Computer Science 193, pages 413–424. Springer-Verlag, 1985.

[130] M. Y. Vardi and P. Wolper. Automata theoretic techniques for modal logics of programs. In *Proceedings of the 16th Annual Symposium on the Foundations of Computer Science*, pages 446–456, 1984.

[131] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.

[132] G. Winsket. A note on model checking the modal ν-calculus. In *Proceedings of the 11th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 372, pages 761–772. Springer-Verlag, 1989.