Modal Logics for Computer Science

Zusammenfassung der wissenschaftlichen Arbeiten

eingereicht bei der
Fakultät Informatik
der
Technischen Universität Dresden

anstelle einer Habilitationsschrift

von Dr. rer. nat. Carsten Lutz
aus Hamburg

Februar 2006

Contents

1 Introduction

The main use of logic in computer science is to achieve a solid mathematical underpinning of the concepts that are central to this scientific discipline: on the one hand, logic provides a theoretical foundation for many applied subfields of computer science, as is probably most evident in the area of databases [1]. On the other hand, logic has close ties with many subjects of theoretical computer science such as automata theory and complexity theory. For example, the basic idea of descriptive complexity is to study complexity theory by relating it to questions of expressiveness in logic [116]. Due to this central standing, the role of logic in computer science has been compared to the role that calculus plays in physics [104].

A plethora of different logics has been studied in computer science. Without claiming completeness, we can achieve a rough categorization by dividing into propositional logic, modal logic, first-order logic, and higher-order logic. This thesis is concerned with modal logic which, not only in this enumeration, is located between propositional logic and first-order logic: syntactically, modal logic resembles propositional logic as it does not include (explicit) quantifiers and variables. Semantically, modal logic is interpreted in relational structures and very close to first-order logic. The name "modal logic" is not used very often in computer science, but we can find a large number of logics that can be conceived as modal, and also a multitude of applications of these formalisms. To name a concrete example, one of the most classical applications of logic in computer science is the specification and verification of software and hardware systems, where modal logic appears, e.g., in the guise of temporal logic and the propositional μ-calculus [54].

The main reason for modal logic to be an effective tool in computer science is its careful balance between expressiveness and computational complexity of reasoning. On the one hand, modal logic provides enough expressive power to capture the relevant aspects of many applications. On the other hand, reasoning in modal logic is usually more feasible than reasoning in first- and higher-order logic: first, deciding satisfiability and validity is usually decidable in modal logic, whereas it is often undecidable in first-order logics. And second, model checking is usually a polynomial-time problem in modal logic, but intractable (PSPACE-complete) in first-order logic. Because of this characteristics, modal logic can usually be found in applications in which there is an emphasis on automated reasoning. One example is verification of software and hardware systems, where reasoning is (for example) required to verify whether a given implementation of a system meets its specification. Another example is provided by description logics, a family of knowledge representation languages that are essentially modal logics. In the area of DLs, automatically deciding subsumption (which roughly corresponds to validity) is central to most applications.

In this thesis, we study a number of modal logics with applications in computer science. Many of these logics and their applications are quite different, which nicely illustrates the diversity of modal logics and their applications in computer science. More specifically, we study description logics, whose applications are knowledge representation in artificial intelligence, reasoning about conceptual database schemas,

and formal ontology. We study dynamic logics such as PDL and the μ-calculus, whose main application is the specification and verification of software systems. We study quantitative temporal logic for reasoning about real-time systems and spatial modal logics which provide a formal basis for geographic information systems (GISs). Finally, we consider modal logic for use in multi-agent systems that have a number of applications such as the verification of distributed open systems.

When studying a modal logic, we shall mainly be interested in the computational complexity of reasoning, in the expressive power, and in the succinctness of these logics. One of our primary technical tools for understanding modal logic will be complexity theory. There are two main reasons for choosing this tool. First, we have already argued that modal logic is often used in applications where automated reasoning plays a central role. In such applications, it is usually the ultimate goal to put logical reasoning to work in implemented computer programs. Obviously, complexity theory can provide us with insights about the feasibility of this goal. However, we should be careful: since we are studying worst-case complexity, theoretical results can hardly ever prove or disprove that there exists an implemented algorithm whose time and space consumption is acceptable for practical applications. Instead, theoretical results give us a first clue concerning the utility of a logic in practice, and non-trivial research into optimization techniques as well as empirical evaluations have to follow. To give an example, modern DL reasoners are usually based on algorithms that, in the worst case, require at least exponential time. Still, due to a number of very effective optimization techniques, these reasoners turn out to be very well-behaved in most applications. A similar point is made by Gurevich in [100].

The second reason for choosing complexity theory as a technical tool is that determining the computational complexity of reasoning in a logic can provide crucial insights regarding the mathematics of this logic. We can learn, e.g., about its expressive capabilities and shortcomings. From this perspective, what is most interesting about complexity results for a logic is their *proofs* and the techniques used in the proofs. This perspective is also taken e.g. by Gabbay et al. in [80] and (again) by Gurevich in [101]. Of course, a similar statement can rightfully be made about, for example, game theory, model theory, and proof theory. We prefer complexity theory and, in general, advocate a liberal perspective on this issue.

This thesis is structured as follows. In Section 2, we introduce modal logic, give a brief survey of their pre-computer science history, and then discuss the importance of modal logic for computer science, and the importance of computer science for modal logic. We then summarize the submitted papers proceeding in the same order as above: we study description logics in Section 3, dynamic logics in Section 4, temporal and spatial logics in Section 5, and logics for multi-agent systems in Section 6. Finally, Section 7 is concerned with a genuinely modal way of integrating specialized logics into more general ones such that desirable properties such as decidability are preserved.

2 Modal Logic

We give a brief introduction to modal logic in general, and to modal logic in computer science in particular. Then, we analyze the tight links that modal logic has to other logical disciplines and formalisms and discuss the (technical) aspects of modal logic that are most relevant for computer science.

2.1 History

Modal logic originated in philosophy as an attempt to resolve the philosophical paradoxes induced by the implication operator of classical logic, and as a logical tool for studying the notions of necessity and possibility. The most influential early work has been carried out by Lewis since the 1910s, and his writings [132, 133] are often viewed as the birth of modal logic. While philosophers continued to use and study modal logics, in the 1930s it was also adopted by mathematicians such as Gödel [88] and Orlov [158]. Their aim was to interpret the intuitionistic logic of Brouwer by extending classical propositional logic with an operator "it is provable that". Today, the language used by Gödel is still regarded as the basic modal language. In modern form, it can be defined as follows.

Definition 1 *Let* PV *be a set of* propositional variables *and* AP *a set of* atomic programs*. The set of modal formulas is the smallest set such that*

- *every* $p \in \mathsf{PV}$ *is a modal formula;*
- *if* φ *and* ψ *are modal formulas and* $a \in \mathsf{AP}$*, then the following are modal formulas:* $\neg\varphi$*,* $\varphi \wedge \psi$*, and* $\Box_a\varphi$*.*

It is common to use $\varphi \vee \psi$ *as an abbreviation for* $\neg(\neg\varphi \wedge \neg\psi)$*,* $\varphi \to \psi$ *for* $\neg\varphi \vee \psi$*, and* $\Diamond_a\varphi$ *for* $\neg\Box_a\neg\varphi$*.* △

The modal language introduced in Definition 1 is more accurately described as the *propositional multi*-modal language. If AP is a singleton, we obtain the *unimodal* language and omit the index to diamonds and boxes. There are also first-order and higher-order versions of modal logic [42], but they will not be considered in this thesis.

The modal logic of Gödel uses the unimodal language, and assigns to $\Box\varphi$ the intuitive meaning that "φ is provable". As the use of modal logic in mathematics continued, other intuitive meanings have been given to $\Box\varphi$. For example, this happened when a topological semantics for modal logic was developed by McKinsey and Tarski [150, 151], in which $\Box\varphi$ is interpreted as topological interior. Among the many meanings that have been considered for $\Box\varphi$ in philosophy, we find "it is necessary that φ is true" [111], "φ is true at all time point in the future" [167], and "it is obligatory that φ is true" [146].

A characterizing feature of the work carried out in the first half of the twentieth century is that modal logic was understood as a purely syntactic endeavor. In particular, a modal logic was commonly defined by devising a set of axioms and a set of inference rules. The main tools for studying modal logics were syntax manipulation

and proof theory. This axiomatic approach is also reflected by the modern definition of a modal logic.

Definition 2 *A* modal logic *is a set* Γ *of modal formulas that*

1. *contains all propositional tautologies;*
2. *is closed under modus ponens, i.e., if* $\varphi \in \Gamma$ *and* $\varphi \to \psi \in \Gamma$*, then* $\psi \in \Gamma$*;*
3. *is closed under uniform substitution, i.e., if* $\varphi \in \Gamma$*, then all formulas obtained from* φ *by uniformly substituting propositional variables with modal formulas is also in* Γ*.*

A modal logic is normal *if it contains the formula*

(K) $\Box(p \to q) \to (\Box p \to \Box q)$

and is closed under generalization, i.e., $\varphi \in \Gamma$ *implies* $\Box\varphi \in \Gamma$*.* $\triangle$

Most of the modal logics used in computer science are normal. As will be discussed in more detail below, there is a close connection between normal modal logics and modal logics that have a relational semantics

To define a modal logic, we can *generate* it using a set of modal axioms, where an axiom is simply a modal formula.[1] To do this, we unite the axioms with the set of propositional tautologies and close off under modus ponens and uniform substitution. A normal modal logic can be generated by additional adding the (K) axiom and closing off under generalization. For example, the normal modal logic generated by the following axioms is known as $\mathsf{S4}$:

(T) $\Box p \to p$

(4) $\Box p \to \Box\Box p$.

$\mathsf{S4}$ is one of the modal logics that was already defined by Lewis in his 1918 paper [132], and this is also where the name derives from (it is the fourth of five modal "systems" defined by Lewis). Studying the lattice of modal logics that is generated by relating the logics admitted by Definition 2 via set inclusion is a major research issue in pure modal logic, see for example [44, 169, 207].

In the early 1960s, Kripke proposed a semantics for modal logic that is based on relational structures [124, 125]. Although Kripke was not the first to define such a semantics, his work fundamentally changed the world of modal logic by alleviating the focus on syntax that was formerly characterizing the field.

Definition 3 *A* Kripke structure *is a triple* $K = (W, (R_a)_{a \in \mathsf{AP}}, V)$*, where*

- W *is a non-empty set of* worlds*;*

[1] Often, it is more convenient to consider an axiom as a formula *schema*, i.e., a formula in which each variable represents a modal formula. However, since modal logics are closed under uniform substitution, technically it suffices that an axiom is a formula.

- $R_a \subseteq W \times W$ *is an* accessibility relation *on* W*;*
- $V : W \to 2^{\mathsf{PV}}$ *is a* valuation *assigning to each world a set of propositional variables.*

Given a Kripke structure $K = (W, (R_a)_{a \in \mathsf{AP}}, V)$*, the satisfaction relation "*$\models$*" between worlds of* K *and modal formulas* φ *is defined as follows:*

$$\begin{array}{lll} K, w \models p & \textit{iff} & p \in V(w) \textit{ for all } p \in \mathsf{PV} \\ K, w \models \neg\varphi & \textit{iff} & K, w \not\models \varphi \\ K, w \models \varphi \wedge \psi & \textit{iff} & K, w \models \varphi \textit{ and } K, w \models \psi \\ K, w \models \Box_a \varphi & \textit{iff} & K, w' \models \varphi \textit{ for all } w' \in W \textit{ with } w R_a w' \end{array}$$

We write $K \models \varphi$ *iff* $K, w \models \varphi$ *for all* $w \in W$*. A* frame *is a Kripke structure without a valuation, i.e., a pair* $F = (W, (R_a)_{a \in \mathsf{AP}})$*. A Kripke structure* $K = (W, (R_a)_{a \in \mathsf{AP}}, V)$ *is* based on *a frame* F *if* F *is obtained from* K *by dropping* V*. We write* $F \models \varphi$ *if* $K \models \varphi$ *for all Kripke structures* K *based on* F*.* $\triangle$

The advent of relational semantics had a fundamental impact on research in modal logic. In the first years after Kripke's papers, relational semantics was mainly viewed as a technical tool, and studying the connection between axiomatic definitions of modal logics and their relational semantics was (and to some extent still is) a major research issue. Regarding this connection, the basic observation is that, for any class $\mathcal{C}$ of frames, the set

$$\mathsf{Log}(\mathcal{C}) := \{\varphi \mid \forall F \in \mathcal{C} : F \models \varphi\}$$

is a normal modal logic according to Definition 2. For example, if $\mathcal{T}$ is the class of all unimodal frames whose accessibility relation is reflexive and transitive, then $\mathsf{Log}(\mathcal{T}) = \mathsf{S4}$. We say that $\mathsf{S4}$ is *determined* by the class of reflexive and transitive frames. However, the connection syntax–relational semantics is far from trivial. For example, there are (uncountably) many modal logics that are not determined by any class of frames. Such logics are called *Kripke incomplete*. In fact, research on the relation between syntactically and semantically defined modal logics has occupied modal logicians for many decades.

The existence of Kripke incomplete logics clearly means that relational semantics has serious limitations. Other semantics such as the algebraic semantics surveyed in [12] do not suffer from such a limited applicability. Still, relational semantics was extremely successful: today, giving a relational semantics is one of the most standard ways to define a modal logic and, as a field, modal logic has long abandoned its focus on pure syntax manipulation and axiomatics. It seems that the success of relational semantics is largely due to three reasons. First, it is a very intuitive semantics that is grasped fairly easily. Second, in a large number of modern applications of modal logic, the relational semantics suffices and is even suggested by the application. We will see many examples of such applications in the subsequent sections of this thesis. And third, compared to the axiomatic approach, relational semantics often allows dramatically simpler proofs, e.g. when showing that two modal logics are not identical.

2.2 Computer Science

In the 1970s, computer scientists started to adopt modal logic for various purposes. The most influential early work has been concerned with reasoning about the correctness of computer programs and with the verification of discrete-state systems. Still today, these are arguably the most important applications of modal logic in computer science. Using modal logic for reasoning about programs was first suggested by Burstall [45], and then showed its full potential in the papers of Pratt [163] and Fischer and Ladner [74, 75] on propositional dynamic logic (PDL), and in the paper of Pnueli [162] on linear-time temporal logic (LTL). Despite aiming at a similar application, the two proposals PDL and LTL are quite different and gave rise to two distinct lines of research that are still very active today. In PDL, a program is viewed as being composed from atomic programs via the constructs of a programming language. Then, the behavior of (atomic and complex) programs is described in terms of their input/output relation. The syntax of PDL offers explicit names for atomic programs, operators for constructing complex programs out of simpler ones, and operators for talking about the input and output of programs. In contrast, LTL does not make programs explicit in its syntax. It aims at describing the evolution of the state that an (implicit) program takes while running. The operators offered by LTL allow to state that certain events do or do not happen in the future, and that a certain property remains true until another property is eventually established. While PDL is designed for reasoning about programs that are supposed to terminate (matching the classical idea of an algorithm), LTL is intended for the verification of reactive systems, i.e., systems that never terminate and interact in a specified way with their environment while running. Examples of reactive systems include operating systems, microprocessors, aviation software, and the like. Here, we introduce PDL as an example for a modal logic that originated in computer science.

Definition 4 *The set of* PDL formulas *and* programs *are defined by simultaneous induction as follows*

- *each atomic program is a program and each propositional variable is a formula;*
- *if α and β are programs and φ is a formula, then the following are also programs:*

$$\alpha \cup \beta,\ \alpha;\beta,\ \alpha^*,\ \varphi?$$

- *if φ and ψ are formulas and α is a program, then the following are also formulas:*

$$\neg\varphi,\ \varphi \wedge \psi,\ [\alpha]\varphi.$$

We use $\langle\alpha\rangle\varphi$ as an abbreviation for $\neg[\alpha]\neg\varphi$. The semantics of PDL is defined in terms of Kripke structures $K = (W, (R_a)_{a\in\mathsf{AP}}, V)$, where the elements of W are called states. *We simultaneously define the extension of R to complex programs and satisfaction of*

PDL formulas in K:

$$
\begin{array}{lcl}
R(\alpha_1 \cup \alpha_2) & = & R(\alpha_1) \cup R(\alpha_2) \\
R(\alpha_1; \alpha_2) & = & R(\alpha_1) \circ R(\alpha_2) \\
R(\alpha^*) & \multicolumn{2}{l}{\textit{is the reflexive-transitive closure of } R(\alpha)} \\
R(\varphi?) & = & \{(w,w) \in W^2 \mid K, w \models \varphi\}
\end{array}
$$

$$
\begin{array}{lll}
K, w \models p & \textit{iff} & w \in V(p) \textit{ for } p \in \mathsf{PV} \\
K, w \models \neg\varphi & \textit{iff} & K, w \not\models \varphi \\
K, w \models \varphi_1 \wedge \varphi_2 & \textit{iff} & K, w \models \varphi_1 \textit{ and } K, w \models \varphi_2 \\
K, w \models [\alpha]\varphi & \textit{iff} & K, w' \models \varphi \textit{ for all } w' \in W \textit{ with } (w, w') \in R(\alpha)
\end{array}
$$

$\triangle$

Let us briefly illustrate the use of PDL for reasoning about programs. Propositional variables are used to represent properties that hold before and after the execution of a program. Though the details are abstracted from, propositional variables may thus stand for, e.g., "variable x has a non-zero value" and "list ℓ is empty". Again abstracting from the details, atomic programs represent atomic operations such as variable incrementation, list concatenation, etc. With this intuition in mind, the PDL formula

$$\varphi \rightarrow [\alpha]\psi$$

makes a statement about the input/output behavior of the program α: it expresses that if φ holds before the execution of α, then ψ holds afterwards. Note that programs are not interpreted as functional relations, and thus PDL assumes that programs are non-deterministic.[2] The following examples show how the regular expressions available to construct complex PDL programs can be used to describe standard constructs of programming languages:

if φ then α else β	$(\varphi?; \alpha) \cup (\neg\varphi?; \beta)$
while φ do α	$(\varphi?; \alpha)^*; \neg\varphi?$
repeat α until φ	$\alpha; (\neg\varphi?; \alpha)^*; \varphi?$

It is interesting to observe that PDL is a blend of classical modal logic with Kleene algebra, and thus extends the classical modal language from Definition 1. This is true for many of the modal logics used in computer science. For example, LTL is based on a temporal language that extends the classical modal language from Definition 1 with the binary operator "until" as first proposed by the mathematician Kamp in his celebrated thesis [118]. A precise definition of the syntax and semantics of LTL can be found in Section 5.1. For relatively up-to-date surveys of PDL and LTL, we refer to the monographs [107] and [79], respectively.

The combination of modal logic and computer science has proved extraordinarily fruitful. Apart from PDL and LTL and their classical application of reasoning about programs and discrete-state systems, a plethora of modal formalisms and applications

[2] There are also deterministic versions of PDL, see e.g. [24].

have been suggested. The current thesis may serve as a witness for this claim: it presents a number of very different modal logics that have applications in a variety of subfields of computer science. In the following, we list some seminal contributions to modal logic in computer science. It is by no means to be understood as a complete list of formalisms and applications.

Branching time logic. When reasoning about time, we may either view the flow of time as linear or as branching. In the latter case, the different branches emerging from a point in time are viewed as different possible futures. It has been observed already during the 1960s by philosophers such as Prior that branching time semantics suggests dedicated logics that, in particular, admit quantification on branches [167]. Later, such formalisms have blossomed in computer science mainly for two reasons: first, they allow to relate the different possible evolutions of a reactive system; and second, they sometimes enjoy significantly more attractive computational properties than their linear time relatives. Most notably, the interest of computer science in branching time has led to the CTL family of temporal logics whose most prominent family members are CTL [51] and CTL* [66]. Intuitively, CTL* enriches LTL with universal and existential path quantifiers, allowing a very liberal use of these quantifiers. CTL requires that path quantifiers are used in a more controlled way, and unlike CTL* does not comprise LTL as a fragment (with all LTL operators implicitly universally path quantified). The most important effect of the restrictedness of CTL is that model checking can be done in linear time [52], while it is PSPACE-complete in LTL [181] and CTL* [68]. A good though slightly outdated survey of linear and branching temporal logic in computer science can be found in [63].

The modal μ-calculus. When dynamic logics such as PDL, LTL, and CTL were developed in the 1970s and 1980s, one main research goal was to push the expressive power as far as possible while still retaining good computational properties. For example, termination of programs cannot be expressed in PDL unless a well-foundedness operator is added [186]. This quest for expressive modal logics ultimately led to the proposal of the modal μ-calculus by Pratt [164] and, in its modern form, by Kozen [122]. Arguably, the three most exciting features of the μ-calculus are the following. First, despite its considerable expressive power, it can be defined in a very short and elegant way: it is simply propositional logic extended with an operator for smallest fixpoints and with an operator for greatest fixpoints (plus some natural syntactic restrictions). Second, it is computationally rather well-behaved. For example, satisfiability can be decided in EXPTIME [67]. And third, its formulas are notorious for being extremely unreadable. In particular, understanding formulas with nested fixpoints can give a hard time even to experts. Since its invention, the μ-calculus has been used in many areas of computer science such as reasoning about programs and reactive systems, and in process algebra. Other modal logics are often translated into the μ-calculus to obtain upper complexity bounds and to derive model theoretic properties. For example, this is possible both for PDL and LTL. Finally, one of the most important open questions from modal logic in computer science is concerned with the μ-calculus: model checking is known to be in NP $\cap$ *co*-NP [30], but it is unknown whether model checking can is tractable. A good survey with introductory character is [40].

Reasoning about Knowledge. As discussed already in Section 2.1, reasoning about knowledge is one of the most classical applications of modal logic. This theme also occurs rather frequently in computer science, in particular in the context of distributed systems. For example, if we want to reason about cryptographic protocols, bargaining sessions, or communicating robots, then the knowledge of the involved agents is clearly of prime importance. Based on this observation, a large number of logics for reasoning about knowledge have been proposed in computer science. Perhaps most notably, computer scientists have put an emphasis on describing the *temporal evolution* of the knowledge of agents. To obtain logics suitable for this purpose, classical modal logics for reasoning about knowledge such as multi-modal S5 are combined with temporal logics such as LTL, see [103] for an example. In the resulting two-dimensional modal logics, there are several options for the interaction of knowledge and time such as perfect recall vs. imperfect recall and synchronous systems (where all agents have access to a shared clock) vs. asynchronous systems. A comprehensive survey can be found in the monograph [70]. A particularly strong emphasis on reasoning about knowledge can be observed in the area of multi-agent systems, where a multitude of modal logics has been proposed. Notable examples include alternating temporal logic (ATL) for reasoning about the collaborative capabilities of agents [11] and logics for reasoning about the change of knowledge that is provoked by announcements made to the agents either in public or in private [194].

Process theory is concerned with the study of processes, which describe the evolution of a system. Systems are modelled in a variety of ways including algebraic descriptions, Petri nets, and state-transition diagrams. To talk about the behavior of processes, it is possible to use modal logics. The modal formalisms dominating the area of process logic are mainly (different variations of) Henessy-Milner logic [109], temporal logics such as CTL, and the modal μ-calculus. One focus of the field is on identifying appropriate equivalence relations between systems such as weak and strong bisimulations. Not surprisingly, there is a close connection between such relations and modal logic. For example, it has been proved in [109] that bisimulation equivalence coincides with distinguishability by formulas of the basic modal language on the class of finitely-branching models, but not on infinitely-branching ones.

Modal logic in Artificial Intelligence. A wide range of modal logics have been introduced in artificial intelligence for vastly different purposes. We do not attempt to give a comprehensive survey, but mention only two selected applications. First, description logics are used in knowledge representation to formally describe the terminology of an application domain. They were developed independently of modal logics, but later found to be notational variants [177]. In description logic, a propositional letter is used to describe a class of objects that gives rise to a terminological notion, and an accessibility relation describes the relation between instances of these classes. Complex formulas are then used to describe complex terminological notions in terms of more basic ones. In the area of description logic, there is a strong emphasis on deciding satisfiability and subsumption, where the latter is essentially validity. We refer to the handbook [20] and Section 3 for more details. Second, several modal logics have been used to formalize non-monotonic reasoning. The most prominent such modal

logic is the autoepistemic logic (AEL) of Moore [154], which is based on uni-modal K45, i.e., the modal logic determined by a transitive and Euclidean accessibility relation. In AEL, the modal box $\Box\varphi$ is read as "the agent believes that φ is true". A set of modal formulas is used to describe the initial beliefs of the agent, and the actual beliefs are then characterized by means of a fixpoint equation in terms of the initial beliefs. It is well-known that there exist faithful translations between autoepistemic logic and other major non-monotonic logics such as Reiter's default logic [121]. More information can be found for example in the textbook [43].

As a further witness for the importance that modal logic has gained in computer science, we may cite an article called "On the unusual effectiveness of logic in computer science" [104]. It is the outcome of a workshop with the same name and analyzes the importance that logic has gained in computer science. The article exemplarily presents five of the most important applications of logic in computer science. Out of these five, two applications are based on first-order logic, one is based on higher-order logic, and two are based on modal logic.

As stressed in [31], the adoption of modal logic by computer science has also led to considerable changes in the field of modal logic itself. For a start, computer science has helped to overcome the status of relational semantics as a mere technical tool, and thus contributed to viewing modal logic (also) as a semantic discipline. In fact, axiomatizations of modal logics play a role also in computer science, but the view in terms of relational semantics is often the more natural and more important one. The main reason for this emphasis on relational semantics is that relational structures are ubiquitous in computer science: they occur as the state-transition graphs of discrete systems, as decorated trees in computational linguistics, as semantic networks in artificial intelligence, etc. Therefore, a genuinely semantic approach to defining modal logics is often suggested by the application. An axiomatization may be undertaken as a later step to get a better understanding of the modal logic at hand, but in many cases it is not essential for the application.

Another key difference between classical modal logic and modal logic in computer science is that computer scientists have been much less reluctant to deviate from the standard modal language of Definition 1, and also from the standard Kripke semantics of Definition 3. For example, the logic PDL introduced above goes beyond the standard modal language, and so do LTL, CTL, and the μ-calculus. An application-driven generalization of Kripke semantics can be found in Section 6.2, where we consider alternating transition systems. This flexibility in defining the syntax and semantics raises the question what actually characterizes a logic that we call "modal". Some clues to this question are provided in Section 2.3.

Perhaps the most important impact of computer science to modal logic is that, in the words of [31], "computer scientists brought a whole new array of questions to the study of modal logic". Indeed, the classical subjects studied in modal logic such as axiomatics and frame definability have been complemented with (at least) the following subjects that originated in computer science.

Computational complexity. The question *how hard* it is to reason in a certain modal logic. Here, "reasoning" can mean a lot of different things such as deciding the satisfi-

ability or validity of a given formula, deciding whether a given formula implies another one, and even checking whether a given set of modal formulas is a conservative extension of another such set. Computer science also contributed new reasoning problems whose computational complexity can be studied, most notably model checking as surveyed in [53]. In model checking, the question is whether a distinguished state of a given Kripke structure satisfies a given modal formula. Especially in reasoning about finite state systems, model checking is a highly relevant reasoning problem. Determining "how hard" a reasoning problem is usually means checking whether there exists an algorithm solving the problem at all (decidability), and if so, what are the resources in terms of time and space that such algorithms require.

Expressivity. The study of what can and what cannot be expressed in a given modal logic. When studying expressivity, the challenge usually is to identify the limitations of a given logic by determining relevant properties that provably cannot be expressed in the logic, and to relate different logics in terms of their expressive power. Often, the identification of expressive shortcomings of a logic has led to the introduction and study of new, more powerful logics. For example, Wolper's observation that "p is true at all even time points" cannot be expressed in LTL [206] has ultimately led to the definition of the temporal μ-calculus [196].

Succinctness is the question how "efficient" a logic is in expressing properties, i.e., what is the minimal length of formulas describing a given property. One reason for studying succinctness is a further discrimination of logics that have the same expressive power. For example, consider the following two logics: the restricted version LTL^r of LTL that has "always in the future" as its only temporal operators; and $\mathrm{FO}_2^<$, which is first-order logic with only two variables and the only binary predicate "$<$". These logics are known to have the same expressive power if we consider only Kripke structures that are based on the frame $(\mathbb{N}, <)$ [69]. However, there are natural properties that can be expressed exponentially more succinct in $\mathrm{FO}_2^<$ than in LTL^u [69]. In general, there is an intimate interplay between computational complexity, expressivity, and succinctness. In the concrete example of the frame $(\mathbb{N}, <)$, the difference in succinctness explains the fact that, despite identical expressivity, there is a huge difference in the computational complexity of satisfiability: NP-complete for LTL^r [181] vs. NExpTime-complete for $\mathrm{FO}_2^<$ [69].

The new array of questions raised by computer scientists also gave rise to a new array of techniques that have been developed to answer these questions. Arguably, the most relevant technical contribution of computer science to modal logic lies is the discovery and exploitation of the fruitful connection between modal logic and automata theory, see for example [200] and [197]. This connection provides many elegant tools such as non-deterministic ω-automata and alternating tree automata which are well-suited for proving upper complexity bounds, and often also for analyzing the expressivity and succinctness of modal logics. For example, a succinctness proof based on alternating automata can be found in [205]. Another development in modal logic that was fostered by computer science is the use of techniques based on games. Historically, the main use of games was to characterize and compare the expressive power of logics in the style of Ehrenfeucht and Fraïsse, see e.g. the textbook [134]. In the modal setting,

these games are very closely related to the notion of bisimulation. Nowadays, many different types of games are used in many different ways, in particular also to determine the computational complexity and succinctness of logics. For example, see [2] for a succinctness result based on games.

2.3 The Rest of the (logical) World

To get a complete picture of the role that modal logic plays in computer science, it is advisable to relate it to the other major logical formalisms that are applied there: propositional logic, first-order logic and higher-order logic.

Propositional Logic

There is an obvious syntactic similarity between propositional logic and modal logic: modal logic can be conceived as the extension of propositional logic with additional unary operators. This is the historic view on modal logic, and it explains why modal logics have initially been defined only in a syntactic way: it is simply not possible to extend the *semantics* of propositional logic in a straightforward way to the modal operators since these are not truth-functional. The similarity of propositional and modal logic also shows up in the algebraic treatment of these two formalisms, where propositional logic corresponds to Boolean algebra and modal logic corresponds to Boolean algebra with operators.

When a (relational) semantic perspective is taken on modal logic, then it is in many aspects much closer to first-order logic than to propositional logic. Nevertheless, it is sometimes possible to apply semantic techniques from propositional logic to modal logic, see e.g. the satisfiability solver $*$-SAT for modal logic that is based on a propositional satisfiability solver [87], and the BDD-based algorithm for deciding satisfiability in modal logic presented in [160]. A completely different connection between propositional logic and modal logic is provided by intuitionistic versions of propositional logic as introduced by Heyting in [110]. The purpose of such logics is to admit only "constructive" mathematical reasoning, and they are usually defined syntactically by dropping from classical propositional logic the law of the excluded middle. As first observed by Gödel, there exists a natural translation of such logics into the modal logic S4 [88]. Nowadays, it is standard to conceive intuitionistic logics (which do not have explicit modal operators) as modal logics.

First-Order Logic

Syntactically, the similarities between modal logic and first order logic are limited. In particular, first-order logic provides explicit variables for quantification over states, while modal logic does not. The latter is true even for the many extended modal languages that have been proposed in computer science. Thus, variable freeness is one of the distinguishing features of modal logic. Arguments have been put forward by the modal logic community that the presence of variables leads to unreadable formulas, and by the first-order community that many properties can only be expressed in an awkward way without variables. We express the belief that both is true at times,

and that there only exists a question of which approach is more well-suited *for what purpose*.

Despite the differences in syntax, there is a close semantical connection between modal logic and first-order logic: in view of Kripke semantics, it is obvious that formulas of the basic modal language can be translated into equivalent formulas of first-order logic with exactly one free variable. This is known as the *standard translation*, which is inductively defined as follows, assuming that there exists one unary predicate P of first-order logic for every propositional variable p, and one binary predicate R_a for every program $a \in \mathsf{AP}$:

$$\begin{aligned} ST_x(p) &= P(x) \\ ST_x(\neg\varphi) &= \neg ST_x(\varphi) \\ ST_x(\varphi \wedge \psi) &= ST_x(\varphi) \wedge ST_x(\psi) \\ ST_x(\Box_a\varphi) &= \forall y.R_a(x,y) \rightarrow ST_y(\varphi) \end{aligned}$$

In the translation, a fresh variable y is introduced for every box operator that is encountered. The standard translation identifies modal logic as a fragment of first-order logic. From the first-order perspective, the identification of this fragment, which is suggested by the variable free syntax of modal logic and has surprisingly attractive properties (to be discussed below), is perhaps the main achievement of modal logic.

Despite the existence of the standard translation, modal logic suggests a genuine view on relational structures that is quite different from the view suggested by first-order logic. An explanation for this phenomenon is obtained by considering the standard translation. First, it produces first-order formulas with only one free variable. And second, quantification occurs only in a syntactically restricted form where the bound variable y occurs in a binary predicate R_a together with the free variable x. Intuitively, this means that modal formulas have a *local* flavor: they are evaluated at a single state, and to evaluate them it suffices to consider only those states that are reachable from the current state by a binary predicate R_a. In contrast, first-order formulas are global since they may contain an arbitrary number of free variables; there simply is no "current state". Moreover, when evaluating first-order formulas, we may have to consider states that are arbitrarily far away in the structure.

It was already mentioned that the modal fragment of first-order logic has surprisingly attractive properties. From the viewpoint of computer science, the most attractive property certainly is a balanced compromise between expressiveness and computational complexity. On the one hand, it is often possible to tailor modal logics that have sufficient expressivity for the application at hand. For example, we have seen in Section 2.2 that PDL was tailored for reasoning about programs. On the other hand, if tailored carefully, then modal logics usually exhibit a much better computational complexity than the corresponding first-order logic. The modal logics described in the main part of this thesis have all been engineered with the goal of achieving such a balance. The most classical examples are the following:

- On the class of all Kripke structures, satisfiability in first-order logic is undecidable whereas satisfiability in the basic modal language is decidable and

PSPACE-complete [31]. Model checking is PSPACE-complete in the first-order case, and in PTIME for the basic modal language.

- Both first-order logic and LTL are decidable on structured based on the temporal frame $(\mathbb{N}, <)$. However, satisfiability in first-order logic is of non-elementary complexity [185] while satisfiability in LTL is PSPACE-complete [181].

However, the penalty that one has to pay for the computational well-behavedness of modal logic is that, usually (depending on the class of structures that is considered), modal logics are less expressive than first-order logics. And even if a property is expressible in both formalisms, then they can sometimes be expressed much more succinctly in first-order logic than in modal logic. For example, consider once more the temporal structures based on the frame $(\mathbb{N}, <)$. For all $n > 0$, every formula of modal logic equivalent to the first-order formula

$$\forall x \forall y. \big(\bigwedge_{i<n} (P_i(x) \leftrightarrow P_i(y)) \rightarrow (P_n(x) \leftrightarrow P_n(y)) \big)$$

is of length at least 2^i [69]. In practical applications, however, the lack in succinctness of modal logic seems hardly ever to be problematic.

Several explanations for the attractive computational properties of modal logic have been brought forward. Historically, the first relevant observation was made by Gabbay [77]: we can modify the standard translation such that it produces first-order formulas with only two variables. The way to achieve this is to alternate the two variables with the nesting depth of the quantifiers. The two-variable fragment FO^2 of first-order logic enjoys much better computational properties than full first-order logic. Most notably, it is decidable (and NEXPTIME-complete) on the class of all structures [95]. While this is a possible explanation for the good computational properties of basic modal logic, it falls short of explaining the computational well-behavedness enjoyed by extensions of the basic modal language such as PDL. While PDL is decidable and EXPTIME-complete [75] despite including a transitive closure operator "$\cdot^*$", the extension of FO^2 with a transitive closure operator on binary relations is Σ^1_1-complete and thus highly undecidable [97]. More information on the relation between modal logic and the two-variable fragment of first-order logic can be found in Section 4.3 of this thesis.

A more sustainable explanation is offered by Vardi [198], who states that the robust decidability of modal logic is due to it having the *tree model property*: every satisfiable modal formula has a model that has the shape of a (possibly infinite) tree. Indeed, this property is enjoyed by basic modal logic *and* the vast majority of its extensions. It also unlocks the door to using powerful and elegant automata-theoretic techniques for proving decidability results and tight upper complexity bounds. Yet another explanation has its origin in the syntactically restricted form in which quantification occurs in formulas that are obtained from the standard translation. As already pointed out, the bound variable y always occurs in a binary predicate R_a together with the free variable x. Intuitively, the quantifier is *guarded* by the predicate R_a which results in localized quantification. This observation has led to the introduction of the *guarded*

fragment of first-order logic [13]. The guarded fragment is a generalization of the class of formulas obtained through the standard translation, and it allows only guarded quantification. It turns out that the guarded fragment is computationally quite attractive. For example, satisfiability is decidable and 2-ExpTime-complete [93]. It even becomes ExpTime-complete when a bound is imposed on the arity of predicates (as in modal logic, where the bound is 2). These complexity bounds are retained in relevant extensions of the guarded fragment, such as with fixpoints [98]. Therefore, the computational well-behavedness of modal logics can be explained with the fact that they fall inside the guarded fragment [94]. On the other hand, the computational well-behavedness of the guarded fragment, in turn, can be explained with it having some form of tree model property. For this reason, we feel that Vardi's explanation is the most basic and convincing one.

The connection between modal logic and first-order logic is certainly not limited to the standard translation. In fact, it goes far deeper. In the following, we provide a brief overview of the relevant results. Van Benthem has precisely characterized the class of first-order formulas that are equivalent to a formula of modal logic: these are exactly the formulas that are invariant under bisimulation. More details on this interesting result can be found in the monograph [190]. Another connection to first-order logic is provided through the notion of frame definability and via Sahlqvist formulae, which are discussed in some more detail in the subsequent section about second-order logic. Finally, it is even possible to understand first-order logic as a multidimensional (propositional) modal logic [201]. The basic idea is to reserve one accessibility relation for each first-order variable, and then to understand the existential and universal quantification on a variable as the diamond and box operators for the corresponding accessibility relation. Then, first-order logic with n variables (roughly) corresponds to the n-dimensional product of the modal logic $\mathsf{S5}$. This product logic can be defined by setting $\mathsf{AP} := \{1, \ldots, n\}$, requiring that the set of worlds W of Kripke structures is the cross-product of n sets $W_1, \ldots, W_n$, and assigning to each box operator $\Box_i$ the following semantics:

$$K, \langle w_1, \ldots, w_n \rangle \models \Box_i \varphi \text{ iff } K, \langle w_1, \ldots, w_{i-1}, w', w_{i+1}, \ldots, w_n \rangle \models \varphi \text{ for all } w' \in W_i.$$

A similar propositional view on first-order logic is adopted when using cylindric algebra, see for example [12, 108]. More information on the interesting and extensive field of multi-dimensional modal logic can be found in the comprehensive monograph [80].

Second-Order Logic

When extending the standard translation from the basic modal language to PDL and the μ-calculus, it is no longer possible to use first-order logic as the target language. In the case of PDL, the problem is due to the transitive closure operator "$\cdot^*$" and the fact that transitive closure cannot be expressed in first-order logic, see e.g. [134]. In the case of the μ-calculus, the fixpoint operators are a form of second-order quantification on unary predicates. Similar problems are encountered with LTL, where the modal box and the until operator are defined using transitive closure. In all three cases, however, it is unproblematic to extend the standard translation if we use second-order

logic as the target language. Thus, many of the modal logics that have been proposed in computer science have a second-order flavor rather than only a first-order one.

In contrast to first-order logic, second-order logic on the class of all structures is undecidable even if we restrict ourselves to only two variables. Therefore, the good computational behavior of modal logics such as PDL cannot be explained by carrying over the argument based on the decidability of two-variable first-order logic to the second-order case. The remaining two arguments, however, can be carried over: the tree model property and guarded quantification. Concerning the former, it can be observed that even expressive modal logics such as PDL and the μ-calculus have the tree model property. This opens up two possibilities for proving decidability. First, it enables the automata-theoretic approach to reasoning in modal logic. And second, it allows to embed modal logic into the monadic second-order theory of trees, which is decidable due to a classical result of Rabin [153]. In contrast to the automata theoretic approach, such an embedding does usually not yield tight complexity bounds since the monadic second-order theory of trees is of non-elementary complexity. Nevertheless, this theory is often used to obtain decidability results for modal logic. An example of this approach can be found in Section 4.1 of this thesis.

Concerning guarded quantification, it has been shown by Grädel and Walukiewicz that the extension of the guarded fragment of first-order logic with fixpoint operators is still decidable [98]. Even better, they show that, computationally, this extension behaves just as good as the guarded fragment without fixpoints: it is 2-ExpTime-complete in general and only ExpTime-complete if a bound is imposed on the arity of predicates. Thus, guarded fixpoint logic is capable of explaining also the low computational complexity of modal logics with second-order features. However, it should not come as a surprise that the good computational behavior of this logic, in turn, is due to it having some form of tree model property.

As in the first-order case, the connection between modal logic and second-order logic is not limited to the standard translation. In fact, Janin and Walukiewicz have proved that there is an intimate connection between the μ-calculus and monadic second-order logic: a formula of the latter is equivalent to a formula of the μ-calculus if and only if it is invariant under bisimulation [117]. Thus, the μ-calculus relates to monadic second-order logic in precisely the same way as the basic modal language relates to first-order logic! Another connection between modal logics using the classical modal language and second-order logic is established by the notion of frame definability. We say that a class $\mathcal{F}$ of frames is *modally definable* is there exists a modal formula φ such that

$$\mathcal{F} = \{F \mid F \text{ is a frame and } F \models \varphi\}.$$

Since "$F \models \varphi$" is defined by universally quantifying over all valuations V for F, it should not be surprising that modal definability is closely related to definability of frame classes in monadic second-order logic. In fact, every modal formula defines a frame class that can also be defined using a monadic second-order formula, and there are simple modal formulas that define frame classes not definable by a first-order formula. For example, the McKinsey formula $\Box\Diamond p \to \Diamond\Box p$ defines the class of frames (W, R) such that R is transitive and R's converse is well-founded, and this frame

property cannot be defined in first-order logic [190]. An interesting line of research is concerned with identifying large classes of modal formulas that define frame classes which can also be defined using first-order logic. For example, this has led to the identification of the class of Sahlqvist formulas [173].

3 Description Logic

Description logics (DLs) are a family of knowledge representation (KR) formalisms that allow to represent the terminology of an application domain in a structured way. Historically, the first DL system was the seminal KL-ONE system of Brachman [39], which aimed at improving earlier KR formalisms such as frame-based systems [152] and semantic networks [168] by actually having a formal semantics. About 15 years after the development of KL-ONE, Schild realized that many DLs can be viewed as modal logics in disguise [177]. To illustrate this connection, we now introduce the expressive description logic $\mathcal{ALCQI}$. It can be viewed as the core of OWL-DL, a description logic that has recently been standardized by the W3C (the World Wide Web consortium) as the ontology language to be used on the web [23]. OWL-DL and its expressive fragments such as $\mathcal{ALCQI}$ are widely used in applications of description logics, see for example the habilitation thesis [174] and references therein.

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be sets of *concept names* and *role names*. The set of $\mathcal{ALCQI}$ *roles* is defined as $\mathsf{N_R} \cup \{r^- \mid r \in \mathsf{N_R}\}$. The set of $\mathcal{ALCQI}$ *concepts* is the smallest set such that

- every concept name is a concept;
- if C and D are concepts, r is a role, and $n \geq 0$, then the following are also concepts:
$$\neg C,\ C \sqcap D,\ C \sqcup D,\ (\leqslant n\ r\ C),\ (\geqslant n\ r\ C)$$

We use $(= n\ r\ C)$ to abbreviate $(\geqslant n\ r\ C) \sqcap (\leqslant n\ r\ C)$, $\exists r.C$ for $(\geqslant 1\ r\ C)$, $\forall r.C$ for $(\leqslant 0\ r\ \neg C)$, $\top$ for an arbitrary (but fixed) propositional tautology, and $\bot$ for $\neg\top$. A *TBox* is a finite set of *concept implications* $C \sqsubseteq D$. We use $C \doteq D$ to abbreviate the two concept implications $C \sqsubseteq D$ and $D \sqsubseteq C$. Description logics are equipped with a Tarski-style set-theoretic semantics. An *interpretation* $\mathcal{I}$ is a pair $(\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty set (the *domain*), and $\cdot^\mathcal{I}$ an *interpretation function* assigning

- to each $A \in \mathsf{N_C}$ a subset $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$;
- to each $r \in \mathsf{N_R}$ a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$.

The interpretation function can be inductively extended to complex roles and concepts as follows:

$$\begin{aligned}
(r^-)^\mathcal{I} &= \{(e,d) \mid (d,e) \in r^\mathcal{I}\} \\
(\neg C)^\mathcal{I} &= \Delta^\mathcal{I} \setminus C^\mathcal{I} \\
(C \sqcap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\
(C \sqcup D)^\mathcal{I} &= C^\mathcal{I} \cup D^\mathcal{I} \\
(\leqslant n\ r\ C)^\mathcal{I} &= \{d \mid \#\{e \in C^\mathcal{I} \mid (d,e) \in r^\mathcal{I}\} \leq n\} \\
(\geqslant n\ r\ C)^\mathcal{I} &= \{d \mid \#\{e \in C^\mathcal{I} \mid (d,e) \in r^\mathcal{I}\} \geq n\}
\end{aligned}$$

where $\#S$ denotes the cardinality of the set S. An interpretation $\mathcal{I}$ is a *model* of a concept C if $C^\mathcal{I} \neq \emptyset$. It is a *model* of a TBox $\mathcal{T}$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$ for all $C \sqsubseteq D \in \mathcal{T}$.

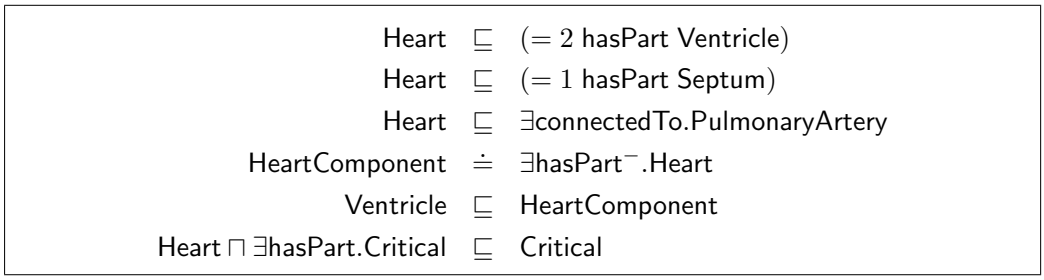

Figure 1: An example TBox

The TBox formalism introduced above is sometimes referred to as *general TBoxes* since there are also several weaker variants. To illustrate the use of description logics for representing the terminology of an application domain, consider the TBox displayed in Figure 1. It shows an excerpt from a medical terminology. The first three lines state some necessary conditions that objects must satisfy in order to be a heart. For example, they must have exactly two parts that are ventricles. The fourth line gives necessary and sufficient conditions for an object to be a heart component, the fifth line states that ventricles do not occur outside the heart, and the last line enforces that any heart with a critical component is critical itself. While the toy example given in Figure 1 is fictional, describing medical terminology is actually a major application of description logics. For example, description logics underly the medical ontologies SNOMED [56, 183] and GALEN [170].

To understand the connection between description logics and modal logics, first consider the fragment $\mathcal{ALC}$ of $\mathcal{ALCQI}$ that is obtained by disallowing the use of the inverse role constructor "r^-", and by replacing the number restrictions $(\leqslant n\ r\ C)$ and $(\leqslant n\ r\ C)$ with the value restrictions $\exists r.C$ and $\forall r.C$. An $\mathcal{ALC}$ concept C can be viewed as a formula φ_C of the basic modal language by identifying

- concept names with propositional variables;
- role names with atomic programs;
- the Boolean operators $\neg, \sqcap, \sqcup$ with $\neg, \wedge, \vee$;
- the constructors $\exists r.C$ and $\forall r.C$ with $\Diamond_r C$ and $\Box_r C$ respectively.

On the semantic level, an interpretation $\mathcal{I}$ can be viewed as the Kripke structure $K_\mathcal{I} := (\Delta^\mathcal{I}, (r^\mathcal{I})_{r \in \mathsf{N_R}}, V)$, where V maps each $d \in \Delta^\mathcal{I}$ to the set of concept names A such that $d \in A^\mathcal{I}$. Conversely, it is easy to convert a Kripke structure into a DL interpretation. It is not difficult to see that this translation is faithful in the sense that we have $d \in C^\mathcal{I}$ iff $K_\mathcal{I}, d \models \varphi_C$ for all interpretations $\mathcal{I}$, $d \in \Delta^\mathcal{I}$, and $\mathcal{ALC}$ concepts C. The sketched correspondence between modal logic and DLs has been used by Schild [176–178] and De Giacomo et al. [86] to transfer results between the two fields. The correspondence can also be extended to the description logic $\mathcal{ALCQI}$: the modal counterpart of the inverse roles r^- available in $\mathcal{ALCQI}$ are backwards modalities, as

considered e.g. in the context of the μ-calculus in [199]. The counterpart of number restrictions are graded modalities that have been discussed for example in [72,73,191].

We call an $\mathcal{ALCQI}$ concept *satisfiable* w.r.t. a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \emptyset$. We say that a concept D *is subsumed by* a concept E w.r.t. $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds in all models $\mathcal{I}$ of $\mathcal{T}$. The most important reasoning problems in DL are deciding the satisfiability of a given concept C w.r.t. a given TBox $\mathcal{T}$, and checking whether there exists a subsumption relationship between two given concepts w.r.t. a given TBox. Intuitively, satisfiability is important to (automatically) verify whether a concept description makes sense from a logical perspective, i.e., whether it is contradictory in itself or to a given TBox. Satisfiability also plays an important role because many other inference problems can be reduced to it. Subsumption can be used to check whether a concept D is more general than a concept C, i.e., whether each instance of C also is an instance of D. For example, the concept name Heart is subsumed by the concept $\forall$hasPart.HeartComponent w.r.t. the TBox in Figure 1. The main use of subsumption is to compute a hierarchy of the concept names occurring in a TBox w.r.t. their generality. This hierarchy can then be presented to developers and users of the TBox for browsing and inspection. It is easy to see that unsatisfiability and subsumption are polynomially inter-reducible: first, we have $C \sqsubseteq_{\mathcal{T}} D$ if and only $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{T}$; and second, C is satisfiable w.r.t. $\mathcal{T}$ if and only if $C \not\sqsubseteq_{\mathcal{T}} \bot$. Because of this close connection, in the following we will often concentrate on satisfiability and assume that subsumption is treated by means of the above reduction.

3.1 Finite Model Reasoning

A key difference between expressive DLs such as $\mathcal{ALCQI}$ and less expressive ones such as $\mathcal{ALC}$ is that the former are capable of enforcing infinite models. For example, the concept $\neg A$ is satisfiable w.r.t. the following $\mathcal{ALCQI}$ TBox only in infinite models, but not in finite ones:

$$\begin{array}{rcl} \neg A & \sqsubseteq & \exists r.A \\ A & \sqsubseteq & \exists r.A \sqcap (\leqslant 1\, r^{-}\, \top) \end{array}$$

In contrast, every $\mathcal{ALC}$ concept that is satisfiable w.r.t. a TBox $\mathcal{T}$ is satisfiable in a finite model of $\mathcal{T}$. This property is commonly called the *finite model property (FMP)*. The fact that $\mathcal{ALCQI}$ lacks the FMP cannot be ignored in many applications. For example, an important application of DLs is reasoning about conceptual database models such as ER diagrams and UML diagrams [49]. In this application, a DL interpretation represents a database and a domain element of a DL interpretation represents an object in a database. Since databases are considered to be finite, we should be interested in deciding satisfiability and subsumption in finite models instead of in unrestricted ones. Even in standard ontology applications such as in medical informatics, finite model reasoning should probably be preferred over unrestricted model reasoning: if $C \sqcap \neg D$ is satisfiable in a medical terminology such as the one shown in Figure 1, but only in infinite models, do we really want to conclude that C is not subsumed by D? After all, the counterexamples witnessing non-subsumption require that there are infinitely many anatomical parts around. This example also suggests

that finite model reasoning can be used in conjunction with standard reasoning to exhibit modelling flaws. If a concept C is satisfiable in unrestricted models, but not finitely satisfiable, this may point to a problem and should be indicated to the ontology designer.

Unfortunately, finite model reasoning in DLs is much less developed than standard reasoning. The prime reason seems to be that it is technically much more involved. The only relevant result is proved by Calvanese in [46], where it is shown that finite satisfiability in $\mathcal{ALCQI}$ is decidable in 2-ExpTime. In contrast, unrestricted satisfiability in $\mathcal{ALCQI}$ is known to be ExpTime-complete [86], and the lower bound carries over to the finite case. Therefore, it was considered an important open question whether or not finite model reasoning in $\mathcal{ALCQI}$ is more difficult than unrestricted reasoning. In [140, 141], we answer this question positively: like their unrestricted counterparts, finite satisfiability and finite subsumption in $\mathcal{ALCQI}$ are "only" ExpTime-complete. To prove the ExpTime upper bound for satisfiability in $\mathcal{ALCQI}$, we show how to translate a given concept C and TBox $\mathcal{T}$ into a system of inequalities $\mathcal{E}_{C,\mathcal{T}}$ such that C is finitely satisfiable w.r.t. $\mathcal{T}$ if and only if $\mathcal{E}_{C,\mathcal{T}}$ admits an integer solution with certain characteristics (an *admissible* solution). Deciding solvability of systems of inequalities over the integers is known as *integer programming* [180]. Since integer programming is an NP-complete problem, the translation approach only yields a NExpTime upper bound for finite satisfiability in $\mathcal{ALCQI}$. However, we additionally show that the systems of inequalities obtained by the translation satisfy a monotonicity condition, and that under this condition the existence of admissible solutions can be decided in polynomial time. Therefore, we obtain the desired ExpTime upper bound.

It is interesting to note that the translation approach described above works only if the numbers inside number restrictions are coded in unary. Note that the coding has a severe impact on succinctness: while the length of $(\leqslant n\ r\ C)$ is $\mathcal{O}(n)$ under unary coding, it is only $\mathcal{O}(\log(n))$ under binary coding. Therefore, reasoning is potentially more difficult under binary coding than under unary coding of numbers. Also in [140, 141], we show that this is actually not the case. We exhibit a polynomial reduction from finite satisfiability in $\mathcal{ALCQI}$ under binary coding to finite satisfiability in $\mathcal{ALCFI}$, which is the variant of $\mathcal{ALCQI}$ that admits only the numbers 0 and 1 inside number restrictions. Since the coding of numbers obviously plays no role in the case of $\mathcal{ALCFI}$, we obtain an ExpTime upper bound also for finite satisfiability in $\mathcal{ALCQI}$ under binary coding of numbers. In contrast to existing similar reductions such as the one in [86], ours works also in the case of finite models and, for this reason, is technically a lot more involved. In [140, 141], we additionally show ExpTime-completeness of the reasoning task *ABox consistency* which we refrain from discussing in detail here. Interesting results that are related to ours (but have been proved later) are NExpTime-completeness of finite satisfiability in C^2, the two variable fragment of first-order logic with counting [96, 166], and ExpTime-completeness of finite emptiness of two-way alternating tree automata [32].

3.2 Tractable Description Logics

The most important reasoning services offered by modern description logic reasoners such as FaCT [114] and RACER [102] are the computation of concept satisfiability and concept subsumption, both w.r.t. TBoxes. To realize these services, DL reasoners usually employ tableau algorithms for expressive DLs such as $\mathcal{ALCQI}$ and its extensions [21]. Despite the fact that satisfiability and subsumption in expressive DLs is usually ExpTime-complete and that the implemented tableau algorithms are even 2-NExpTime ones, DL reasoners exhibit a surprisingly good runtime behavior that is sufficient for many applications. Still, the high computational complexity of modern DLs also has a number of drawbacks, of which we mention only two: first, the high complexity obviously implies that there exist inputs on which the reasoner needs an unacceptable amount of time. Although such inputs are not encountered too frequently in applications, there *are* cases when they occur. If this happens, the user can only attempt to reformulate her knowledge base. Since the internals of modern DL reasoners are far from trivial and a convincing theoretic explanation for the "usually" good runtime of DL reasoners is missing, the average user is often left to guessing how to achieve a better runtime behavior. This is in contrast to the otherwise purely declarative approach to knowledge representation using DLs. Second, there are applications that require knowledge bases of massive size, such as the SNOMED systematized nomenclature of human and veterinary medicine [56, 183] which is a TBox involving ∼400.000 concept names. Currently, such inputs are simply too large for modern DL reasoners based on tableau algorithms.

Due to these drawbacks of expressive DLs, it has always been a goal of DL research to identify small, yet useful description logics for which satisfiability and subsumption are tractable [17, 37, 60, 61, 155]. Early attempts concentrated on variants of $\mathcal{FL}_0$, which is the DL that comprises only the operators conjunction and universal value restriction ($\forall r.C$). Although subsumption in $\mathcal{FL}_0$ is indeed tractable, it becomes co-NP-complete when extended with a very weak form of TBoxes [156] (*acyclic* TBoxes) and is even PSpace-complete with a stronger form [15, 119] (*cyclic* TBoxes) that is still weaker than the general TBoxes introduced above. Surprisingly, the sibling $\mathcal{EL}$ of $\mathcal{FL}_0$ that comprises only the operators conjunction and existential value restriction ($\exists r.C$) has received much less attention. Only in 2004, it was observed by Brandt that subsumption in $\mathcal{EL}$ is tractable even w.r.t. the general TBoxes introduced in this section [41].[3] Since several recent applications of DLs such as SNOMED are based on variations of $\mathcal{EL}$, this result has the potential to bring the quest for tractable and useful DLs to a positive ending.

In the joint paper [16] with Baader and Brandt, we perform a detailed analysis of the computational complexity of extensions of $\mathcal{EL}$. In particular, we show that subsumption in $\mathcal{EL}$ w.r.t. TBoxes remains tractable if we (simultaneously) add the following means of expressivity:

- the bottom concept $\bot$ (and thus disjointness between concept names through concept implications $A \sqcap B \sqsubseteq \bot$);

[3] Satisfiability is trivial in $\mathcal{EL}$ and $\mathcal{FL}_0$ as there are no unsatisfiable concepts.

- nominals, i.e., a special sort of concept names that have to be interpreted in singleton sets;
- a restricted form of concrete domains that allow reference to numbers and strings in concept expressions;
- role inclusions $r_1 \circ \cdots \circ r_k \sqsubseteq s$ to be used in TBoxes, where $r_1, \ldots, r_k, s$ are role names, and the inclusion $r_1 \circ \cdots \circ r_k \sqsubseteq s$ is satisfied by an interpretation $\mathcal{I}$ if $r_1^{\mathcal{I}} \circ \cdots \circ r_k^{\mathcal{I}} \subseteq s^{\mathcal{I}}$. Observe that role inclusion can be used to express transitivity of roles via $r \circ r \sqsubseteq r$, and also right identities $r \circ s \sqsubseteq s$ which are important in medical applications such as SNOMED.

The resulting extension of $\mathcal{EL}$ is called $\mathcal{EL}$++. This result should be compared with the fact that, as we also show in [16], subsumption in $\mathcal{FL}_0$ w.r.t. general TBoxes is ExpTime-complete. The polynomial time algorithm for $\mathcal{EL}$++ given in [16] borrows ideas from tableau algorithms for DLs, from algorithms for propositional Horn logic, and from the filtration technique for modal logic.

We also consider other expressive means that are common in DLs and show that subsumption in $\mathcal{EL}$ extended with any of the following is ExpTime-complete:

- atomic negation, i.e., negation that can only be applied to concept names but not to complex concepts;
- disjunction;
- at-least restriction ($\geqslant n\ r\ C$), even if n is restricted to 2 and C to $\top$;
- at-most restrictions ($\leqslant n\ r\ C$), even if n is restricted to 1 and C to $\top$;
- functional roles;
- each of the following constructors for forming complex roles: negation, union, and transitive closure.

Additionally, we show that, in $\mathcal{EL}$ extended with existential value restrictions on inverse roles ($\exists r^-.C$), subsumption w.r.t. TBoxes is PSpace-hard. The best known upper bound is an ExpTime one.

The polytime subsumption algorithm for $\mathcal{EL}$++ has been implemented in a DL reasoner called CEL. An empirical evaluation is undertaken in [19], where CEL is used the reason on SNOMED and on the Gene Ontology [187], a TBox describing genes and gene products that involves ∼40.000 concept names. The main result of the evaluation is that even the relatively naive implementation of the $\mathcal{EL}$++ subsumption algorithm in CEL can outperform existing and highly optimized reasoners for expressive DLs. This could not be taken as granted: before CEL, Classic was the most advanced DL reasoner that uses a polytime algorithm [37]. Still, reasoners based on expressive DLs such as FaCT and RACER usually outperform Classic if used on TBoxes formulated in Classic's language. It is also interesting to note that CEL is able to compute the subsumptions in SNOMED (currently in ∼40 minutes), whereas reasoners based on

expressive DLs cannot handle SNOMED at all.[4] The developers of SNOMED have recently signaled that they are interested in using $\mathcal{EL}$++ and CEL for the further development of SNOMED.

3.3 Conservative Extensions

Modern applications of DLs require terminologies of large scale and complex structure whose design and maintenance is a challenging task. Moreover, terminologies and their applications evolve over time, and therefore it is frequently necessary to maintain, refine, customize, and integrate terminologies. These tasks, in turn, rely on basic operations on TBoxes such as revision, extension, and merging with other TBoxes. When such operations are performed on well-established and tested TBoxes, it is of prime importance to have control of the resulting consequences. To illustrate this issue, we discuss two example scenarios.

First, suppose that a knowledge engineer maintains a well-tested TBox $\mathcal{T}$ that formalizes the terminology of an application domain. Assume that the engineer wants to extend $\mathcal{T}$ with a number of additional concept implications that describe the terminology of a part of the domain that was not yet covered by $\mathcal{T}$. Moreover, the extended TBox is used in an application that requires computing subsumptions between concepts, and for which the TBox $\mathcal{T}$ was used before. To avoid unexpected results when using the extended TBox, the existing part of $\mathcal{T}$ should not be compromised by the new axioms. In particular, the extended TBox should not entail new subsumptions between concepts that are formulated in the signature of the old TBox, where the term "signature" refers to the concept and role names used.

Second, assume that there are two well-established TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$ that describe different and largely independent aspects of an application domain, but nevertheless have an overlap in signature. To use $\mathcal{T}_1$ and $\mathcal{T}_2$ together in the same application, one would like to merge them by simply taking their union. Similarly to the case of TBox extensions, it is then important to know whether the merging operation compromises the component TBoxes: are there any subsumptions in the signature of $\mathcal{T}_1$ entailed by $\mathcal{T}_1 \cup \mathcal{T}_2$ that are not entailed by $\mathcal{T}_1$ alone, and likewise for $\mathcal{T}_2$. Intuitively, the entailment of such subsumptions means that there may be unexpected results when using the merged terminology in place of the component terminologies.

The reasoning problems suggested by these two examples can be conveniently formalized using the notion of a conservative extension, which has been widely studied and applied in mathematical logic and in the philosophy of science. Formally, a TBox $\mathcal{T} \cup \mathcal{T}'$ is a *conservative extension* of a TBox $\mathcal{T}$ iff $C \sqsubseteq_{\mathcal{T} \cup \mathcal{T}'} D$ implies $C \sqsubseteq_{\mathcal{T}} D$ for all concepts C and D formulated in the signature of $\mathcal{T}$. Equivalently, $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$ iff every concept in the signature of $\mathcal{T}$ that is unsatisfiable w.r.t. $\mathcal{T} \cup \mathcal{T}'$ is already unsatisfiable w.r.t. $\mathcal{T}$. Now, the reasoning problem that is relevant for the examples above is the following: given TBoxes $\mathcal{T}$ and $\mathcal{T}'$, is $\mathcal{T} \cup \mathcal{T}'$ a

[4] We have been informed that the FaCT++ reasoner, which is based on an expressive DL, is able to compute the subsumptions in SNOMED within 3 hours. However, we have not yet been able to reproduce these results.

conservative extension of $\mathcal{T}$? In the following, we will refer to this problem simply as *conservativeness*. Obviously, conservativeness captures both TBox extension (where $\mathcal{T}'$ contains the additional concept implications) and TBox merging.

In the joint paper [85] with Ghilardi and Wolter, we propose conservativeness as a relevant reasoning problem for description logics and analyze its decidability and computational complexity in the description logic $\mathcal{ALC}$. The basic results is that deciding conservativeness in $\mathcal{ALC}$ is 2-ExpTime-complete. The lower bound is proved by a reduction of the word problem for exponentially space-bounded alternating Turing machines as introduced in [50]. The upper bound is obtained by an algorithm that can be viewed as an elaborate kind of type elimination procedure as first proposed by Pratt in the context of PDL [163]. More precisely, to check whether $\mathcal{T} \cup \mathcal{T}'$ is *not* a conservative extension of $\mathcal{T}$, we have to check whether there is a model $\mathcal{I}$ of $\mathcal{T}$ such that no model $\mathcal{J}$ bisimilar to $\mathcal{I}$ can be extended to a model of $\mathcal{T} \cup \mathcal{T}'$, where *extended* means adding an interpretation of the concept and role names occurring in $\mathcal{T}'$ but not in $\mathcal{T}$. With $\mathcal{T}$-type, we mean a set of subconcepts of concepts in $\mathcal{T}$ satisfying some basic Boolean closure conditions. $\mathcal{T}'$-types are defined analogously. To check the existence of a model as just described, we use an algorithm that considers *type pairs*, i.e., pairs (t, T) where t is a $\mathcal{T}$-type and T a set of $\mathcal{T}'$-types. Intuitively, T contains all the $\mathcal{T}'$-types to which t can be extended in an interpretation of $\mathcal{T}$ to obtain an interpretation of $\mathcal{T} \cup \mathcal{T}'$. The algorithm starts with all type pairs and then repeatedly eliminates pairs that cannot be realized in an interpretation. It stops if no more elimination is possible or it finds a type pair (t, T) with $T = \emptyset$. In the latter case, $\mathcal{T} \cup \mathcal{T}'$ is not a conservative extension of $\mathcal{T}$.

We then perform a more refined complexity analysis revealing that there exists an algorithm for deciding conservativeness that takes only exponential time in the size of $\mathcal{T}$ and double-exponential time in the size of $\mathcal{T}'$. Especially in the case of TBox extensions, where $\mathcal{T}'$ typically is very small compared to $\mathcal{T}$, this is a relevant observation showing that, in such cases, deciding conservativeness is not substantially more complex than deciding the standard reasoning problems satisfiability and subsumption (which are ExpTime-complete in $\mathcal{ALC}$).

While conservativeness already provides the TBox designer with relevant information, we can be even more informative: assume that the extension of a TBox turns out *not* to be a conservative extension. This may or may not be intended, and it is up to the developer to decide whether the new TBox faithfully represents the domain under consideration. To support her in this decision, it is useful to compute *witness concepts*, i.e., examples of concepts that were satisfiable before, but are unsatisfiable after the extension. Out algorithm for deciding conservativeness can also be used to compute witness concepts. We prove that the length of (the smallest) witness concepts can be triple exponential in the size of the original TBoxes (but not worse).

We also consider a further refinement of conservativeness that is motivated by the observation that non-conservative extensions and mergings of TBoxes are sometimes intended and completely acceptable. Suppose the original TBox $\mathcal{T}$ consists of a core terminology (e.g., an upper-level ontology [99]) formulated in a signature Γ that by no means should be corrupted, and of other parts for which a non-conservative extension

is acceptable or even intended. In this case, the required reasoning problem is the following: given a set Γ of concept names and roles and terminologies $\mathcal{T}$ and $\mathcal{T}'$, is there a concept in the signature Γ that is unsatisfiable w.r.t. $\mathcal{T} \cup \mathcal{T}'$, but satisfiable w.r.t. $\mathcal{T}$? We prove that, in $\mathcal{ALC}$, deciding this refined version of conservativeness is also 2-ExpTime-complete. However, there is a significant difference to the basic reasoning problem: in the refined version, we cannot get an algorithm whose runtime is only exponential in the size of $\mathcal{T}$: even for a fixed TBox $\mathcal{T}'$, the complexity of deciding whether $\mathcal{T} \cup \mathcal{T}'$ is a conservative extension of $\mathcal{T}$ w.r.t. Γ is 2-ExpTime-hard.

4 Expressive Modal and Dynamic Logics

In this section, we propose and analyze several very expressive modal logics: an extension of PDL, several extensions of the μ-calculus, and some so-called Boolean modal logics that admit the Boolean operators on programs. Historically, members of the PDL and μ-calculus family of modal logics are often called *dynamic logics* because their purpose is to reason about the changes that programs and actions in finite-state systems provoke in the truth of propositions. We are interested in expressive variants of such logics mainly for the reason that, in computer science logic, it has always been a major research goal to identify logics that are as expressive as possible and still decidable. This is witnessed e.g. by research on the two-variable fragment of first-order logic [95, 96, 159, 166] the guarded fragment [13, 93, 98], the μ-calculus [122, 126, 175], and monadic theories of trees [153]. In general, research in this direction is interesting both from a theoretical and practical perspective because it helps to understand the limits of decidability. Moreover, expressive and decidable logics are useful as a target formalism for embedding other, more specialized logics with the aim of transferring complexity theoretic and model theoretic results such as upper complexity bounds and the tree model property. As will be discussed in more detail below, this is also the main motivation for considering the logics analyzed in this section.

4.1 Enriched PDL

Since the invention of PDL in the 1970s for reasoning about programs, the adaptation to a growing number of applications has led to many modifications and extensions. Nowadays, these additional applications are often the main driving force behind the continuing interest in the PDL family of logics, see e.g. [3, 7, 28, 58, 86]. An important family of variations of PDL is obtained by adding an intersection operator "$\cap$" on programs, and possibly additional program operators. The semantics of this operator is as expected:

$$R(\alpha_1 \cap \alpha_2) = R(\alpha_1) \cap R(\alpha_2).$$

Unfortunately, the extension of PDL with intersection (IPDL) is notorious for being "theoretically difficult". This is mostly due to an intricate model theory: in contrast to most other extensions of PDL, the addition of intersection destroys the tree model property in a rather serious way. This is witnessed e.g. by the formula

$$\neg p \wedge [b]\bot \wedge \langle (a; p?; a) \cap b^* \rangle \top$$

which enforces a cycle of length 2. It is easy to modify this formula such that it enforces a cycle whose length is exponential in the length of the formula. Original PDL and many of its extensions can be decided using automata on infinite trees or an embedding into the alternation-free fragment of the μ-calculus. By adding intersection to PDL and destroying the tree model property, we leave this framework and thus lose the toolkit of results and techniques that have been established over the last twenty years. Consequently, the results obtained for IPDL are sparse. The first result about the computational properties of PDL with intersection is due to Harel, who proved that satisfiability in IPDL with deterministic programs is undecidable [106].

In 1984, Danecki showed that dropping determinism regains decidability [57]. He also establishes a 2-ExpTime upper bound.

Since 1984, it was unknown whether Danecki's upper bound is tight. In 2004, Martin Lange and I were able to prove a 2-ExpTime lower bound for satisfiability in IPDL, thus showing that Danecki's upper bound cannot be improved. The result is presented in [131] and rests on a reduction of exponentially space bounded alternating Turing machines. More precisely, we give three variations of the reduction: the first one uses the test operator "$\varphi?$" of IPDL and shows that satisfiability is ExpTime-hard even if we restrict ourselves to tree structures. The second reduction is a slight variation of the first one and does not require the test operator. Since there are relevant applications of PDL that do not require this operator, a lower bound is more convincing if it does not rely on test. The drawback of the second reduction is that it does not work on tree structures. Finally, the third reduction uses a sophisticated encoding to establish 2-ExpTime hardness of satisfiability in IPDL even without the test operator and on tree structures. It is interesting to note that the expressive power of PDL and IPDL coincides on tree structures since we can simply remove the intersection operator from IPDL programs by computing the intersection of regular expressions (this does not work on non-tree structures). Still, satisfiability in PDL and IPDL on tree structures is of quite different complexity: ExpTime-complete vs. 2-ExpTime-complete.

It is interesting that, until recently, virtually nothing was known about further extensions of IPDL. Most strikingly, the natural extension of IPDL with a converse operator "$\cdot^-$" on programs has never been investigated. This operator has the following semantics:

$$R(\alpha^-) = \{(v, w) \mid (w, v) \in R(\alpha)\}.$$

The extension of IPDL with this operator is called *ICPDL*. I addressed ICPDL in my paper [136], where I prove that satisfiability in ICPDL is decidable by developing a satisfiability preserving translation into the monadic second-order theory of trees. This result has several interesting consequences:

First, decidability of ICPDL implies decidability of the information logic DAL (*Data Analysis Logic*), a problem that has been open since DAL was proposed in 1985 [71]. The purpose of DAL is to aggregate data into sets that can be characterized using given properties, and, dually, to determine properties that best characterize a given set of data. Technically, DAL may be viewed as the variant of IPDL obtained by requiring all relations to be equivalence relations and admitting only the program operators $\cap$ and $\cup^*$, where the latter is a combination of PDL's operators $\cup$ and $\cdot^*$. In ICPDL, equivalence relations can be simulated using $(a \cup a^-)^*$ for some atomic program a. Thus, DAL can be viewed as a fragment of ICPDL.

Second, there is a close correspondence between variants of PDL and description logics. In particular, the description logic $\mathcal{ALC}_{\mathsf{reg}}$ [14,86] is a syntactic variant of PDL without the test operator [177], and the intersection operator of IPDL corresponds to the intersection role constructor in description logics. The latter is a traditional constructor that is present in many DL formalisms, see e.g. [47,61,138,149]. Decidability and complexity results play a central role in DL, but have never been obtained for

the natural extension $\mathcal{ALC}^{\cap}_{\mathsf{reg}}$ of $\mathcal{ALC}_{\mathsf{reg}}$ with role intersection. Clearly, $\mathcal{ALC}^{\cap}_{\mathsf{reg}}$ is a syntactic variant of test-free ICPDL, and thus my decidability result carries over.

Third, ICPDL can be applied to obtain results for modal logics of knowledge. The basic observation is as in the case of DAL: ICPDL can simulate equivalence relations by writing $(a \cup a^-)^*$. Since union and transitive closure of programs can be combined to express the common knowledge operator of epistemic logic, and intersection of programs corresponds to the distributed knowledge operator, decidability of ICPDL can be used to obtain decidability for epistemic logic with both common knowledge and distributed knowledge. We should admit, however, that this approach is rather brute force: since the common knowledge and distributed knowledge operators of epistemic logic cannot be nested to build up more complex operations on relations, epistemic logic lacks much of the complexity of ICPDL. Therefore and as noted in [70], decidability can also be obtained using more standard techniques.

As already noted above, the decidability result is obtained by a translation into the monadic second-order theory of trees. This involves a tricky encoding of the non-tree models if ICPDL into tree-shaped abstractions. Due to the high complexity of monadic second order logic and the structure of the formulas obtained by the translation (in which the alternation depth of quantifiers is not constant), we obtain only a non-elementary upper bound for satisfiability in ICPDL. The best known lower bound is the 2-ExpTime one for IPDL proved in [131]. It seems likely that satisfiability in ICPDL is actually also contained in 2-ExpTime. A proof, however, is far from trivial and seems to require a combination of the techniques used in [57] and [136].

4.2 Enriched μ-calculus

The μ-calculus extends the basic modal language from Definition 1 by adding fixpoint operators $\mu x.\varphi$ and $\nu x.\varphi$, where x is from a fixed set Var of *fixpoint variables*. To define the semantics of the additional operators, we introduce valuations for fixpoint variables. Given a Kripke structure $K = (W, R, V)$ and a set $\{x_1, \dots x_n\} \subseteq \mathsf{Var}$, a *valuation* for this set is an assignment $\mathcal{V} : \{x_1, \dots x_n\} \to 2^W$. For a valuation $\mathcal{V}$, a fixpoint variable x, and a set $W' \subseteq W$, we denote by $\mathcal{V}[x \leftarrow W']$ the valuation obtained from $\mathcal{V}$ by assigning W' to x. We then define the consequence relation "$\models^{\mathcal{V}}$" between worlds in Kripke structures and formulas of the μ-calculus with reference to a valuation $\mathcal{V}$. The Boolean operators and the modal operators are treated as in Definition 3, where the valuation $\mathcal{V}$ is simply passed on from the left-hand side to the right-hand side of the "iff" clauses. The remaining cases are as follows:

$$
\begin{array}{lll}
K, w \models^{\mathcal{V}} x & \text{iff} & w \in \mathcal{V}(x) \text{ for all } x \in \mathsf{Var} \\
K, w \models^{\mathcal{V}} (\mu x.\varphi) & \text{iff} & \bigcap\{W' \subseteq W \mid \{v \in W \mid K, v \models^{\mathcal{V}[x \leftarrow W']} \varphi\} \subseteq W')\} \\
K, w \models^{\mathcal{V}} (\nu x.\varphi) & \text{iff} & \bigcup\{W' \subseteq W \mid W' \subseteq \{v \in W \mid K, v \models^{\mathcal{V}[x \leftarrow W']} \varphi\}\}.
\end{array}
$$

One of the main uses of the μ-calculus is as a target formalism for embedding temporal and modal logics with the goal of transferring computational and model theoretic properties such as the ExpTime upper complexity bound. It has been advocated by several authors that also description logics should be embedded into the μ-calculus,

mainly to identify DLs that are of very high expressive power, but computationally still well-behaved [48, 126, 175]. When putting this idea to work, we face the problem that modern DLs include several constructs that cannot easily be translated into the μ-calculus. Most importantly, these constructs are inverse roles (corresponding to inverse programs in the μ-calculus), number restrictions (corresponding to graded modalities), and nominals (recall that nominals are a special sort of concept names/propositional variables that have to be interpreted in a singleton set).

This observation has led to the enrichment of the μ-calculus with backwards programs, graded modalities, and nominals. The μ-calculus simultaneously enriched with all these means of expressivity is called the *fully enriched μ-calculus*. More precisely, this logic is obtained by extending the μ-calculus as follows: first, a *program* of the fully enriched μ-calculus is either an atomic program or the converse a^- of an atomic program a. Second, we introduce nominals, which are a special kind of propositional variable. In contrast to standard propositional variables, nominals are true in exactly one world of each Kripke structure. And third, we replace the diamond and box operators with graded modalities, i.e., an *atleast operator* $\langle n, \alpha\rangle\varphi$ and an *allbut operator* $[n, \alpha]\varphi$, where n is a non-negative integer and α a (possibly converse) program. The semantics of the new operators is as follows:

$$\begin{array}{lll} K, w \models^{\mathcal{V}} \langle n, \alpha\rangle\varphi & \text{iff} & \#\{v \in W \mid (w, v) \in R(\alpha) \wedge K, v \models^{\mathcal{V}} \varphi\} > n \\ K, w \models^{\mathcal{V}} [n, \alpha]\varphi & \text{iff} & \#\{v \in W \mid (w, v) \in R(\alpha) \wedge K, v \models^{\mathcal{V}} \varphi\} \leq n \end{array}$$

Thus, $\langle n, \alpha\rangle\varphi$ states that atleast $n + 1$ α-successors satisfy φ, and dually, $[n, \alpha]\neg\varphi$ expresses that all but at most n α-successors satisfy φ. Note that the modalities $\langle\alpha\rangle\varphi$ and $[\alpha]\varphi$ of the standard μ-calculus can be expressed as $\langle 0, \alpha\rangle\varphi$ and $[0, \alpha]\varphi$, respectively.

The fully enriched μ-calculus is expressive enough to accommodate a very large class of modal and description logics. Alas, it is too expressive: as shown by Bonatti and Peron in [34], satisfiability in the fully enriched μ-calculus is undecidable. In contrast, some fragments of it are known to be decidable: the μ-calculus extended with inverse programs and nominals (full hybrid μ-calculus) and the μ-calculus extended with inverse programs and graded modalities (full graded μ-calculus) are both decidable and ExpTime-complete, see [175] and [48, 126]. In the latter case, this result is known only when the numbers in graded modalities are coded in unary or when inverse programs are not admitted.

The above results raise the question of maximal decidable fragments of the fully enriched μ-calculus. In the joint paper [33] with Bonatti, Murano, and Vardi, we study this question in a systematic way by considering all fragments of the fully enriched μ-calculus that are obtained by dropping at least one of inverse programs, graded modalities, and nominals. It turns out that dropping *any* of these three means of expressivity regains decidability and ExpTime-completeness. More precisely, we prove that this is the case for the μ-calculus extended with nominals and graded modalities (the hybrid graded μ-calculus) and for the μ-calculus extended with inverse programs and graded modalities where the numbers in graded modalities are coded in binary.

Our results are based on the automata-theoretic approach. We introduce *fully enriched automata (FEAs)*, which work on infinite forests and use a parity acceptance condition. Intuitively, these automata generalize alternating automata on infinite trees in a similar way as the fully enriched μ-calculus extends the standard μ-calculus: FEAs can move up to a node's predecessor (by analogy with inverse programs), move down to at least n or all but n successors (by analogy with graded modalities), and jump directly to the roots of the input forest (which are the analogues of nominals). We prove that the emptiness problem is decidable for fully enriched automata and then show how to reduce to this problem satisfiability in the hybrid graded and the full graded μ-calculi, exploiting the forest model property enjoyed by these logics. Observe that decidability of the emptiness problem for FEAs does not contradict the undecidability of the fully enriched μ-calculus: the latter does not enjoy a forest model property, and hence satisfiability cannot be decided using forest-based FEAs.

To show that the emptiness problem for FEAs is in ExpTime, we introduce an additional automata model: *two-way graded parity tree automata (2GAPTs)*. These automata are interesting in their own right because they generalize in a natural way two existing, but incomparable automata models: two-way alternating tree automata (2APT) [199] and graded parity tree automata (GAPT) [126]. We give a polynomial reduction of the emptiness problem for FEAs to that for 2GAPTs, and then show containment in ExpTime for the 2GAPT emptiness problem by a reduction to the emptiness of graded nondeterministic parity tree automata (GNPT) as introduced in [126].

4.3 Boolean Modal Logic

Modal logics such as PDL extend the basic modal language by allowing to construct complex programs from atomic ones using program operators such as composition and union. The basic idea of Boolean modal logic (BML) as studied for example in [82] is to admit the Boolean operators $\neg$, $\cap$, and $\cup$ on programs (with the obvious semantics), and thus to alleviate at least to some extent the asymmetry between propositional variables (unary predicates) and programs (binary predicates) that can be observed in the basic modal language.

An interesting property of BML is that, in this logic, we can define the so-called window operator. To understand this operator, recall that $\Box_a\varphi$ holds at a world w iff w' being accessible from w implies that φ holds at w'. Thus, it is obviously quite natural to define an operator $\boxplus_a$ that is symmetric to $\Box_a$: $\boxplus_a\varphi$ holds at a world w iff φ holding at a world w' implies that w' is accessible from w. This operator is called the window operator, where the name is derived from the symbol used for it. In contrast to the standard modal box, which can be thought of as expressing necessity, the window operator can be understood as expressing sufficiency. Logics with this operator were investigated from different viewpoints by, e.g., Humberstone, Gargov et al., and Goranko [83, 89, 90, 115]. In BML, the window operator $\boxplus_a\varphi$ can simply be expressed as $[\neg a]\neg\varphi$. As pointed out in a joint paper with Sattler [138], the window operator is quite useful in the context of description logics. There, it can be used to make statements such as "fighters for animal rights love *all* animals", which is not

	no negation	atomic negation	full negation
$-$	PSpace-compl.	ExpTime-compl.	
$\cup$	PSpace-compl.	ExpTime-compl.	NExpTime-compl.
$\cap$	PSpace-compl.	NExpTime-compl.	
$\cap$ and $\cup$	PSpace-compl.		

Figure 2: Complexity of various fragments of BML.

expressible in standard DLs.

In the joint paper [139] with Sattler, we investigate the complexity of reasoning in Boolean modal logic and its natural fragments. We find that satisfiability in full BML is NExpTime-complete, in contrast to PSpace-completeness of the basic modal language. The upper bound is immediate by an obvious translation into the two-variable fragment of first-order logic, and the lower bound is established using a NExpTime-complete variant of the domino problem. In contrast to the standard (undecidable) domino problem of [29, 120], in the NExpTime-complete variant the task is to tile a torus of exponential size [35]. We then show that there are (at least) two ways to lower the complexity of satisfiability in BML to ExpTime-completeness. The first is to drop union and intersection, retaining negation as the only program operator. Then, the window operator is still definable and we can show an ExpTime upper bound using non-deterministic automata on infinite trees. To do this, we introduce a tree abstraction of models. This is necessary since the addressed logic does not have the tree model property, as witnessed for example by the formula

$$p \wedge [\neg a]\neg p$$

which enforces a reflexive loop and can be modified to enforce cycles of exponential length. The lower bound is easily obtained by reducing satisfiability in the basic modal logic enriched with a universal modality, c.f. [91].

The second way to lower the complexity of satisfiability to ExpTime-completeness is to put a finite bound on the number of accessibility relations. Indeed, our NExpTime lower bound relies on the fact that an unbounded number of relations are available. When such a bound is imposed, we can use a sequence of reduction steps to reduce satisfiability in BML to satisfiability in the basic modal language enriched with the universal modality. Since the latter is ExpTime-complete, we obtain the desired result. In [139], we also analyze some other fragments of BML. The results are summarized in Figure 2, where the grey entries have already been known, and "atomic negation" means that negation can only be applied to (formulas and) atomic programs.

In terms of expressive power, the inclusion of Boolean program operators brings modal logic closer to FO^2, the two-variable fragment of first-order logic. The reason for this is that the latter includes the Boolean operators and does not distinguish between their application to unary and binary predicates. It is therefore a natural

question to ask what we have to add to BML to reach *exactly* the expressive power of FO^2, and how the computational complexity and succinctness of the resulting modal logic relates to the complexity and succinctness of FO^2. We investigate these questions in the joint paper [142] with Sattler and Wolter. It turns out that adding to BML a backwards program operator "α^-" (as in Section 4.2) and an identity program id with

$$R(\mathsf{id}) = \{(w, w) \mid w \in W\}$$

suffices to reach the expressive power of FO^2. Let us call the resulting logic *enriched BML*. Finding an equivalence preserving translation from enriched BML to FO^2 is an easy task. For the converse direction, we use a translation inspired by a similar one of Etessami, Vardi, and Wilke in [69].

Given that enriched BML and FO^2 have the same expressive power, it is interesting to compare their succinctness and computational complexity. In particular, modal logics are usually computationally simpler than (finite variable fragments of) first-order logic, but also less expressive. Thus, the reason for the computational simplicity of modal logics is not quite clear: is it their reduced expressive power or is it that they talk about relational structures in a less succinct way than first-order logic. Our analysis of enriched BML, which has *exactly* the same expressive power as FO^2, provides some evidence in favour of the second explanation.

Regard the succinctness, we forst observe that the translation from BML to FO^2 produces formulas whose length is linear in that of the original formula, but the converse translation involves an exponential blowup in formula length. Indeed, such a blowup cannot be avoided since, as we show in [142], there are properties that can be expressed exponentially more succinct in enriched BML than in FO^2. More precisely, let φ_n be the following formula of FO^2, for $n \geq 1$:

$$\begin{aligned}\forall x \exists y \Big(& \bigwedge_{k=0..n-1} \Big(\bigwedge_{j=0..k-1} P_j(x) \Big) \rightarrow (P_k(x) \leftrightarrow P_k(y)) \\ \wedge & \bigwedge_{k=0..n-1} \Big(\bigvee_{j=0..k-1} \neg P_j(x) \Big) \rightarrow (P_k(x) \leftrightarrow P_k(y)) \Big)\end{aligned}$$

Using the fact that this formula enforces a domain of cardinality at least 2^n, it can be shown that every formula of enriched BML that is equivalent to φ_n is of length at least $2^n/2$.

As we show in [142], this difference in succinctness between enriched BML and FO^2 indeed has an impact on computational complexity. First, satisfiability in FO^2 is NExpTime-complete [76, 95]. This holds regardless of the number of binary predicates (even if there are none at all). In contrast, the complexity of enriched BML is sensitive to the number of accessibility relations. If an unbounded number is available, enriched BML inherits NExpTime-hardness from plain BML. Together with the linear translation to FO^2, we thus get NExpTime-completeness. On the other hand, if we impose a finite bound on the number of accessibility relations, then satisfiability in enriched BML is ExpTime-complete. The lower bound is easy to show. In [142], we establish the upper bound by first using a series of non-trivial reductions

to simplify the problem, and then applying a Pratt-style type elimination procedure as first proposed in [163]. Thus, we observe the interesting effect that the difference in succinctness between modal logic and first-order logic induces a difference in computational complexity only if we impose a bound on the number of variables.

It is interesting to note that our results from [142] generalize the main results from Etessami et al. [69], who consider only structures based on the frame $(\mathbb{N}, <)$. It also improves a result by Borgida [36], who identifies a description logic that has the same expressive power as FO^2. In contrast to Borgidas logic, enriched BML has a weaker set of program operators, and this makes the expressive equivalence with FO^2 considerably less obvious.

5 Temporal and Spatial Logic

Modal logics of time and space have several interesting applications in computer science. As has been discussed in Section 2.2, the main application of temporal logic is reasoning about the behavior or reactive systems, i.e., systems that run continuously and are supposed to interact with their environment in a specified way. Examples include microprocessors, operating systems, and communication protocols. Spatial logics are mainly used in geographic information systems (GIS) and for various applications in artificial intelligence. The term GIS refers to a family of technologies that allow to store, manage, and analyze spatial data [135]. Most GIS have deductive capabilities that allow to infer new spatial knowledge from the information that is stored explicitly, and these reasoning capabilities are often based on a spatial logic. More information on reasoning in GIS can for example be found in the book [62]. In artificial intelligence, spatial reasoning is essential in subfields such as image understanding and robot navigation. The literature is somewhat scattered, but the book [184] covers many relevant subjects. There are several different ways to use modal logic for spatial reasoning. Some of them are discussed in more detail in Section 5.2.

5.1 Quantitative Temporal Logic

The classical approach to reasoning about reactive systems is based on purely qualitative logics such as LTL, CTL, and CTL*. However, when real-time properties play a crucial role in the description of the system behavior, this rather abstract approach is no longer feasible. For example, if we are modelling a network router or even aviation software, it may not be sufficient to know that the software will eventually react to a critical situation. What we want to enforce is that it reacts within a concrete time interval, say within 10 milliseconds. Such concrete distances between events cannot be adequately described in qualitative logics such as LTL. Consequently, the basic logical tool for reasoning about real-time systems is provided by quantitative logics, which are usually extensions of LTL with metric operators. To obtain a realistic model of time, such logics are commonly interpreted in the real line. Examples of such quantitative logics can be found in [8–10, 112, 179].

To introduce real-time temporal logics, we start with the purely qualitative logic LTL. The language of LTL is obtained by extending the basic modal language from Definition 1 with a binary until operator $\varphi \mathcal{U} \psi$. We do not include a "next-time" operator $\bigcirc\varphi$ since we will interpret LTL in structures based on the real numbers and, in such structures, the next point in time is clearly not a meaningful concept. A Kripke structure $K = (W, R, L)$ is a *real-time structure* if $W = \mathbb{R}^+$ is the set of non-negative real numbers, and R is the standard "$<$" relation on the reals. The consequence relation is defined as usual for the operators of the basic modal language, and as follows for the until operator:

$$K, w \models \varphi \mathcal{U} \psi \text{ iff there is a } v \in W \text{ such that } wRv,\ K, v \models \psi,$$
$$\text{and } K, u \models \varphi \text{ for all } u \text{ with } w < u < v.$$

When reasoning about real-time systems, it is often natural to consider only real-time structures such that, in every bounded interval, each propositional variable changes

its truth value only finitely often. This condition is called the Zeno condition (after Zeno's paradox) or the *finite variability assumption (FVA)*. It reflects the fact that most reactive systems cannot change their state infinitely often while approaching some fixed point in time. However, there are also significant cases in which it is useful to give up the FVA, see e.g. [27, 59, 105] for discussions of this subject.

Moving from qualitative to quantitative logics is often accompanied by a considerable increase in computational complexity of the satisfiability problem. The most important example demonstrating this effect is LTL. Satisfiability of LTL over the non-negative reals as introduced above is PSPACE-complete [172] both with and without FVA. Consider now an additional operator $\varphi \mathcal{U}_{[n,m]} \psi$, where n and m are rational numbers coded in binary, and the semantics is as follows:

$$K, w \models \varphi \mathcal{U}_{[n,m]} \psi \text{ iff there is a } v \in W \text{ such that } w + n \leq v \leq w + m,\ K, v \models \psi, \text{ and } K, u \models \varphi \text{ for all } u \text{ with } w < u < v.$$

We use $\Diamond_{[n,m]}\varphi$ to abbreviate $\top \mathcal{U}_{[n,m]} \varphi$. Let mLTL denote the extension of LTL with this metric version of the until operator. In mLTL, satisfiability is EXPSPACE-complete if the case $n = m$ is not admitted in the metric until operator, and even undecidable if it is [8,10,112]. Thus, the complexity is considerably higher than in the corresponding quantitative logic LTL. However, not all linear-time quantitative temporal logics are computationally as hard as mLTL. In the joint paper [143] with Walther and Wolter, our goal is to identify quantitative real-time logics whose complexity is not worse than that of qualitative logics, i.e., in PSPACE or below.

Some such computationally well-behaved quantitative logics have already been known before our investigations. In particular, it was known that the complexity of mLTL can be brought down to PSPACE by requiring that the lower parameter n to the metric until operator has the value zero [8]. We use mLTL^- to refer to this fragment of mLTL. In contrast to EXPSPACE-completeness and undecidability of mLTL, PSPACE-completeness of satisfiability in mLTL^- had only been proved under the FVA. In our paper [143], we propose a new technique for polynomially reducing satisfiability in metric temporal logics with numbers coded in binary to satisfiability in the same logic with numbers coded in unary. A notable feature of this technique is that it does not depend on the FVA. Since satisfiability in mLTL^- with unary coding of numbers is known to be PSPACE-complete also without the FVA [112], we can use our reduction to prove containment in PSPACE of the same logic with numbers coded in binary. Thus, we identify a relevant quantitative temporal logic for which reasoning is not harder than in the qualitative version even without the FVA. The proof is presented in [143] as an example application of the reduction technique. In that paper, we also present another application of our technique. This application is RTCTL, a metric version of CTL that is equipped with a discrete-time semantics. This logic has been proposed by Emerson et al. [64], who also show EXPTIME-completeness. Using our technique, we can reprove the upper bound in a much simpler way than Emerson et al.: due to the discrete time semantics, we obtain a direct reduction of satisfiability in RTCTL (with numbers coded in binary) to satisfiability in *qualitative* CTL and thus get the desired EXPTIME upper bound.

If we drop the until operator from LTL retaining the modal box as the only temporal operator, satisfiability in the resulting logic $\mathrm{LTL}_{\Box}$ is NP-complete (with the real-line based semantics) [182]. In our paper [143], we also try to identify quantitative counterparts of this computationally rather well-behaved temporal logic. First, we establish three negative results showing that all of the following fragments of mLTL^{-} are still PSPACE-hard:

1. the only temporal operators are $\Diamond\varphi$ and $\Diamond_{[0,n]}\varphi$, with n coded in unary;
2. the only temporal operator is $\Diamond_{[0,n]}\varphi$, with n coded in binary;
3. the only temporal operator is $\varphi\,\mathcal{U}_{[0,1]}\,\psi$.

Observe that the second and third fragments do not include any qualitative temporal operators. The three PSPACE lower bounds are proved using reductions from LTL interpreted on the natural numbers. An analysis of the possible fragments of mLTL^{-} reveals the following logic as the only natural remaining candidate for a fragment of mLTL^{-} for which satisfiability is in NP:

4. the only temporal operator is $\Diamond_{[0,n]}\varphi$, with n coded in unary.

Indeed, we are able to show containment in NP of satisfiability in this logic by devising an algorithm that first guesses a set of types (sets of subformulas of the input formula satisfying certain Boolean closure conditions) that is of polynomial cardinality, and then constructs and solves a system of linear inequalities with rational coefficients over the real numbers to deal with the metric operators. We give two separate algorithms for the case with and without FVA. Thus, Logic 4 as described above appears to be the right quantitative counterpart of LTL with the modal box as only temporal operator. Observe that this logic can only make statements about a bounded prefix of the future. Still, such a logic can be useful for reasoning about the evolution of real-time systems whose runtime is a priori bounded.

As an interesting side note, note that our results for Logics 2 and 4 illustrate that the coding of numbers can make a difference in computational complexity for quantitative real-time logics.

5.2 Logics of Topological Relations

It has already been noted that there are several different ways to use modal logic for reasoning about space. As always in modal logic, the main issue is to choose a suitable meaning for the states, the accessibility relations, and the propositional variables. The most standard way of using modal logic for spatial reasoning is based on topological spaces. Recall that a topological space is a pair $\mathfrak{T} = (U, \mathbb{I})$, where U is a set and $\mathbb{I}$ is an *interior operator* on U, i.e., for all $s, t \subseteq U$, we have

$$\begin{array}{rclcrcl} \mathbb{I}(U) & = & U & & \mathbb{I}(s) & \subseteq & s \\ \mathbb{I}(s) \cap \mathbb{I}(t) & = & \mathbb{I}(s \cap t) & & \mathbb{II}(s) & = & \mathbb{I}(s). \end{array}$$

As suggested already by Gödel, it is possible to use U as the set of states of a uni-modal logic and interpret the box operator as the interior operation. Then, a propositional

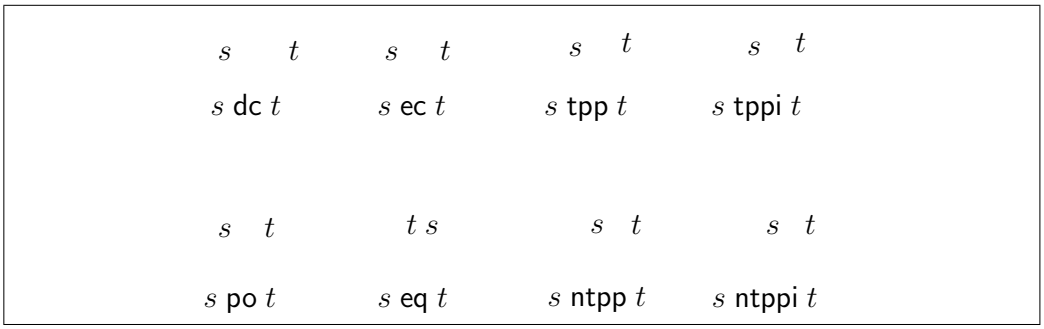

Figure 3: The eight relations between regions.

variable identifies a subset of U, which intuitively corresponds to a spatial region. Gödel clearly did not think of spatial reasoning in computer science when proposing a topological semantics for modal logic. However, this connection was established later by other researchers. Most notably, Bennett has shown that constraint satisfaction problems based on the RCC8 set of topological relations can be reduced to satisfiability in modal logic [25]. This connection has proved rather fruitful and was exploited in a still ongoing line of research, see for example [26, 81, 157, 165]. Other approaches to spatial reasoning that are based on modal logic with the box operator interpreted as topological interior can be found in [5, 6]. An approach to spatial reasoning using modal logics based on distances and metric spaces is presented in [128].

In this section, we consider a different approach to spatial reasoning with modal logic that is also based on topological spaces. We start with introducing the RCC8 set of topological relations. Given a topological space $\mathfrak{T} = (U, \mathbb{I})$, a *region* is a non-empty, regular closed subset of U, where $s \subseteq U$ is *regular closed* if $\mathbb{CI}(s) = s$. Intuitively, requiring regular closedness excludes pathological cases of regions such as points and lines in the standard topology of $\mathbb{R}^2$. The extension of the eight RCC8 relations dc ('disconnected'), ec ('externally connected'), tpp ('tangential proper part'), tppi ('inverse of tangential proper part'), po ('partial overlap'), eq ('equal'), ntpp ('non-tangential proper part'), and nttpi ('inverse of non-tangential proper part') is defined as follows on the set of regions of a topological space $\mathfrak{T}$:

$$\begin{array}{rll}
(s,t) \in \mathsf{dc}^{\mathfrak{T}} & \text{iff} & s \cap t = \emptyset \\
(s,t) \in \mathsf{ec}^{\mathfrak{T}} & \text{iff} & \mathbb{I}(s) \cap \mathbb{I}(t) = \emptyset \ \wedge \ s \cap t \neq \emptyset \\
(s,t) \in \mathsf{po}^{\mathfrak{T}} & \text{iff} & \mathbb{I}(s) \cap \mathbb{I}(t) \neq \emptyset \ \wedge \ s \not\subseteq t \ \wedge \ t \not\subseteq s \\
(s,t) \in \mathsf{eq}^{\mathfrak{T}} & \text{iff} & s = t \\
(s,t) \in \mathsf{tpp}^{\mathfrak{T}} & \text{iff} & s \subseteq t \ \wedge \ s \not\subseteq \mathbb{I}(t) \ \wedge \ s \neq t \\
(s,t) \in \mathsf{ntpp}^{\mathfrak{T}} & \text{iff} & s \subseteq \mathbb{I}(t) \ \wedge \ s \neq t \\
(s,t) \in \mathsf{tppi}^{\mathfrak{T}} & \text{iff} & (t,s) \in \mathsf{tpp}^{\mathfrak{T}} \\
(s,t) \in \mathsf{ntppi}^{\mathfrak{T}} & \text{iff} & (t,s) \in \mathsf{ntpp}^{\mathfrak{T}}.
\end{array}$$

Figure 3 gives examples of the eight RCC8 relations in the real plane $\mathbb{R}^2$.

We now define a modal logic $\mathrm{ML}_{\mathsf{RCC8}}$ as follows. For the syntax, we define a set

of relation symbols

$$\mathsf{RCC8} := \{\mathsf{dc}, \mathsf{ec}, \mathsf{tpp}, \mathsf{tppi}, \mathsf{po}, \mathsf{eq}, \mathsf{ntpp}, \mathsf{nttpi}\}$$

and introduce a box operator $[\mathsf{r}]\varphi$ for each topological relation $\mathsf{r} \in \mathsf{RCC8}$. For the semantics, we use Kripke structures $K = (W, (R_\mathsf{r})_{\mathsf{r}\in\mathsf{RCC8}}, V)$ induced by a topological space $\mathfrak{T} = (U, \mathbb{I})$, i.e.,

- $W = U$;
- $R_\mathsf{r} = \mathsf{r}^\mathfrak{T}$ for all $\mathsf{r} \in \mathsf{RCC8}$;
- V is such that, for all $p \in \mathsf{PV}$, the set $\{w \in W \mid p \in V(w)\}$ is a region in $\mathfrak{T}$.

Our logic $\mathrm{ML}_{\mathsf{RCC8}}$ is well-suited for spatial reasoning in a GIS-like context. We only give a simple example. In the example, we use the *universal box* $\Box_u\varphi$, which states that φ holds at all states in the model. Since the RCC8 relations are jointly exhaustive, the universal box can be defined as follows:

$$\Box_u\varphi := \bigwedge_{\mathsf{r}\in\mathsf{RCC8}} [\mathsf{r}]\varphi.$$

Our example makes several (very simplified) statements about the relationship of cities, harbors, rivers, and the sea. Based on this "background theory", it then describes the relationship of the city of Dresden and the river Elbe.

$$\begin{aligned}
&\Box_u(\mathit{harbor\text{-}city} \leftrightarrow (\mathit{city} \wedge (\langle\mathsf{tppi}\rangle \mathit{harbor} \vee \langle\mathsf{ntppi}\rangle \mathit{harbor})))\\
&\Box_u(\mathit{harbor} \rightarrow (\langle\mathsf{ec}\rangle \mathit{river} \vee \langle\mathsf{ec}\rangle \mathit{sea}))\\
&\Box_u(\mathit{Dresden} \rightarrow \mathit{harbor\text{-}city})\\
&\Box_u(\mathit{Elbe} \rightarrow \mathit{river})\\
&\Box_u(\mathit{Dresden} \rightarrow \textstyle\bigwedge_{\mathsf{r}\in\mathsf{RCC8}-\{\mathsf{dc}\}} [\mathsf{r}]\neg \mathit{sea})\\
&\Box_u(\mathit{Dresden} \rightarrow (\langle\mathsf{po}\rangle \mathit{Elbe} \wedge \textstyle\bigwedge_{\mathsf{r}\in\mathsf{RCC8}-\{\mathsf{dc}\}} [\mathsf{r}](\mathit{river} \rightarrow \mathit{Elbe})))
\end{aligned}$$

From these formulas, it follows that Dresden has a tangential or non-tangential part that is a harbor and is related via ec to the river Elbe.

The modal logic $\mathrm{ML}_{\mathsf{RCC8}}$ has first been proposed by Cohn in 1993 [55], but its computational properties turned out to be very hard to analyze and even decidability was an open problem for a long time. Only in 2004, in the joint papers [144, 145] with Frank Wolter we managed to settle this question. More precisely, we analyze the expressive power and decidability of $\mathrm{ML}_{\mathsf{RCC8}}$. Regarding the expressivity, we show the following main results:

- Constraint satisfaction problems based on the RCC8 relations can be embedded into $\mathrm{ML}_{\mathsf{RCC8}}$ in a straightforward way.
- $\mathrm{ML}_{\mathsf{RCC8}}$ has the same expressive power as the corresponding two variable fragment of first-order logic, but the latter is exponentially more succinct. Technically, these results are closely related to those presented in Section 4.3.

These results already indicate that the expressive power of $\mathrm{ML}_{\mathsf{RCC8}}$ is rather high. In fact, it turns out to be too high for decidability. In [144,145], we settle the complexity of $\mathrm{ML}_{\mathsf{RCC8}}$ by giving a very general undecidability proof that applies to $\mathrm{ML}_{\mathsf{RCC8}}$ and many of its variants. For example, the result also holds if we consider only Kripke structures induced by a fixed topological space such as $\mathbb{R}^2$ and $\mathbb{R}^3$, which are the most natural ones for spatial reasoning. The undecidability result also applies if we modify our notion of a region by requiring that regions are, for example, a convex set or a hyper-rectangle. The undecidability proof is based on a sophisticated, linearized encoding of the domino problem that was inspired by [147, 171]. By modifying the construction, we are able to strengthen our result to a proof of Σ^1_1-hardness, although this stronger result does not apply to all cases that are captured by the undecidability proof. In a further modification, we show that even the restriction to finite (but unbounded) topological spaces does not regain decidability.

Finally, we consider the RCC5 set of spatial relations that can be obtained from the RCC8 set by coarsening (1) the tpp and ntpp relations into a new "proper-part of" relation pp; (2) the tppi and ntppi relations into a new "has proper-part" relation ppi; and (3) the dc and ec relations into a new disjointness relation dr. By a reduction of satisfiability in the 3-dimensional product of the modal logic S5, we show that even the counterpart $\mathrm{ML}_{\mathsf{RCC5}}$ of $\mathrm{ML}_{\mathsf{RCC8}}$ that is obtained by replacing the RCC8 relations with the RCC5 relations is undecidable. This result does not subsume the undecidability result for $\mathrm{ML}_{\mathsf{RCC8}}$ since it does not apply to as many variations of the logic as the $\mathrm{ML}_{\mathsf{RCC8}}$ result. Indeed, in the RCC5 case some relatively natural logics remain that could turn out to be decidable.

A decidable logic based on the RCC8 relations whose semantics is very different to that of $\mathrm{ML}_{\mathsf{RCC8}}$ is presented as an example in Section 7.

6 Logic for Multi-agent Systems

The agent paradigm has been extensively used in computer science and artificial intelligence. While a generally accepted definition of what an agent is seems to be difficult to attain, there are some properties that are considered typical for agents: they operate autonomously, have control over their actions and internal state, and they interact with other agents via input and output channels. This general approach has been applied in a large number of quite different applications such as autonomous trading [202], the synthesis of communication protocols [4], and the specification and verification of distributed systems [70]. There are many properties of agents that may be relevant for an application and are useful to capture in a logic. Arguably, most attention has been devoted to the knowledge of agents, and to the evolution of knowledge over time.

Epistemic logic for reasoning about the knowledge of agents is a classical subject of modal logic in philosophy, c.f. the work of von Wright [208] and Hintikka [111]. In epistemic logic, one uses the modal language from Definition 1, and there is one modal box $\Box_a$ for each agent a. The formula $\Box_a\varphi$ is read as "agent a knows φ". The basic epistemic logic is usually defined axiomatically as the normal modal logic generated by the following axioms (c.f. Definition 2 and the following example):

(T) $\Box_a p \to p$

(4) $\Box_a p \to \Box_a \Box_a p$

(5) $\neg\Box_a p \to \Box_a \neg\Box_a p$.

Intuitively, the (T) axiom states that an agent can only know true things, the (4) axiom describes positive introspection (agents know what they know), and the (5) axiom describes negative introspection (agents know what they don't know). In the modal logic literature, this basic epistemic logic is usually called S5.

Semantically, S5 is the modal logic determined by the class of frames whose accessibility relations are equivalence relations. In the context of knowledge, the basic semantic intuitions are as follows: in a Kripke structure $K = (W, R, V)$, each world $s \in W$ describes a possible state of affairs $\{\varphi \mid K, s \models \varphi\}$. Note that what we called "state of affairs" also includes complete information about the knowledge of agents. Concerning the accessibility relations, if sR_as' for some $s, s' \in W$ and some agent a, then in state of affairs s, agent a considers the state of affairs s' possible. Thus, knowing φ means that φ is true in all state of affairs that are considered possible.

6.1 Public Announcement Logic

The knowledge of agents in multi-agent systems is usually not static, but evolves over time. To use epistemic logic in computer science applications, it is therefore usually necessary to add expressive means for speaking about the dynamic aspects of knowledge. The most standard approach as described by Fagin, Halpern, Moses, and Vardi in the monograph [70] consists in adding temporal operators to the basic epistemic

logic and using a product semantics. A different approach is taken by *dynamic epistemic logics (DELs)*, in which epistemic logic is extended with dynamic operators that allow to describe the ramifications of knowledge-changing actions. Although DELs are a relatively young field, a large number of formalisms have been proposed, and the various proposals differ considerably in expressive power [22, 84, 161, 188, 189, 192, 193]. However, there is a dynamic operator that is included in almost all proposed logics: the *public announcement* operator that has first been introduced in [161]. This operator allows to state that, after some announcement that is publicly made by an outsider to all agents simultaneously, some property holds true. Both the announcement and the property may include epistemic statements such as "agent a knows fact F" or "agent a knows that agent b does not know fact F". The announcement is assumed to be truthful, i.e., the person making the announcement does not lie. The effect of the announcement being public is that everybody knows the announced fact, everybody knows that everybody knows it, and so forth. It is interesting to note that the announced fact is not necessarily true anymore *after* the announcement. For example, this is the case if the announced fact is "agent a knows fact F, but agent b doesn't know that" (because, after the announcement, agent b knows that agent a knows F).

Formally, the public announcement operator is a binary modal operator $[\varphi]\psi$ that is supposed to express "if φ is true and publically announced, then ψ is true afterwards". The operator has a conditional meaning since we want to be able to make statements about the consequences of the public announcement of a formula φ without making assumptions about (or even having knowledge of) the truth of φ. The new operator has the following semantics:

$$K, s \models [\varphi]\psi \quad \text{iff} \quad K, s \models \varphi \text{ implies } K|\varphi, s \models \psi$$

where the model $K|\psi := (W', R', V')$ is defined as follows:

$$\begin{aligned} W' &:= \{t \in W \mid K, t \models \psi\} \\ R'_a &:= R_a \cap (W' \times W') \\ V'(p) &:= V(p) \cap W'. \end{aligned}$$

Intuitively, the semantics can be understood as follows. Once that φ has been announced, no agent will consider a state w possible in which $\neg\varphi$ is true. Hence, these states can be dropped from the model which is what happens through the transition from K to $K|\varphi$.

Existing research about logics including the public announcement operator has mainly concentrated on expressiveness and axiomatics. In [137], I analyze the computational complexity of reasoning in epistemic logics extended with the public announcement operator and also investigate the succinctness of such logics. I start with *public announcement logic (PAL)*, the extension of epistemic logic (as introduced above) with the public announcement operator. It is well-known that the expressive power of PAL is identical to the expressive power of EL: there exists an equivalence preserving translation from the former to the latter [161, 194]. Computationally, this translation is only moderately useful: it yields decidability of reasoning in PAL, but

it does not produce tight complexity bounds due to an exponential blowup in formula size.

In [137], I show that reasoning in PAL is of the same complexity as reasoning in EL. To this end, I first propose a novel, equivalence preserving translation from PAL to EL. Like the existing one, this translation induces an exponential blowup in formula size. However, the advantage of the new translation is that it can easily be modified such that it becomes only satisfiability preserving, but avoids the exponential blowup in formula size. The modified translation takes formulas of single-agent PAL to formulas of single-agent EL, and formulas of multi-agent PAL to formulas of multi-agent EL extended with an "everybody knows" operator. Thus, I can use the translation to prove that (i) single-agent PAL is NP-complete, and (ii) multi-agent PAL is PSPACE-complete. I then extend the equivalence-preserving translation and its satisfiability preserving modification to PAL extended with (two variants of) common knowledge. In this case, the target language of the translation is PDL, and I obtain EXPTIME-completeness results.

Due to the fact that PAL and EL are equally expressive and of the same computational complexity, one may be tempted to think that the addition of the public announcement operator to EL is only syntactic sugar and not of much interest. However, it has been convincingly argued in [188, 194] that PAL is a much more intuitive and natural formalism for talking about the dynamics of knowledge than EL. In [137], I identify another, more formal argument in favor of PAL: I prove that there are properties that can be expressed exponentially more succinct in PAL than in EL. Of course, this succinctness result also implies that one cannot hope to find an *equivalence preserving* translation from PAL to EL that avoids an exponential blowup. A limitation of my current succinctness result is that it applies only to the class of all Kripke structures (where the accessibility relations can be any relation), and not to the class of epistemic structures (where accessibility relations have to be equivalence relations).

6.2 Alternating Temporal Logic

In many agent-based systems, the agents have the ability to execute actions in order to bring about a desired state of the system. Usually, different actions are available to different agents, and agents can execute actions concurrently so that the resulting system state depends on the combination of actions that were executed. In such a situation, it is rewarding for agents to form coalitions: a coalition of agents may be able to bring about a state of the system that cannot be achieved by any single agent. In [11], Alur, Henzinger, and Kupferman propose *alternating temporal logic (ATL)* as a formalism for the specification and verification of open distributed systems. It was soon realized that ATL is also a highly appropriate formalism for reasoning about the strategic abilities of coalitions of agents [113]. Technically, ATL can be viewed as a game-theoretic generalization of CTL. Using the path quantifiers of CTL, one is essentially restricted to stating that some state of the system is either inevitable or possible. In ATL, we can describe how coalitions of agents can control the state of the system, such as "agents 1 and 2 can cooperate to ensure that, no matter what the

other agents do, the system will not enter an invalid state". Semantically, ATL is based on *alternating transition systems (ATSs)*, which can be viewed as a generalization of Kripke structures that emphasizes the game-like nature of multi-agent systems.

Let AG be a countably infinite set of *agents*. A *coalition* is a finite set $A \subseteq \mathsf{AG}$ of agents. The set of ATL formulas is the smallest set such that

- every propositional letter is an ATL formula;
- if φ and ψ are ATL formulas and A is a coalition, then the following are ATL formulas: $\neg\varphi$, $\varphi \wedge \psi$, $\langle\langle A\rangle\rangle \bigcirc \varphi$, $\langle\langle A\rangle\rangle \Box \varphi$, $\langle\langle A\rangle\rangle \varphi \,\mathcal{U}\, \psi$.

The modality $\langle\langle A\rangle\rangle$ is called a *path quantifier* and $\bigcirc$ ("next"), $\Box$ ("always"), and $\mathcal{U}$ ("until") are called *temporal operators*. We use $\langle\langle A\rangle\rangle \Diamond \varphi$ as an abbreviation for $\langle\langle A\rangle\rangle \top \,\mathcal{U}\, \varphi$.

We now introduce the semantics of ATL. An *alternating transition system (ATS)* for $\Sigma \subseteq \mathsf{AG}$ is a tuple $\mathcal{S} = \langle \Pi, \Sigma, Q, \pi, \delta \rangle$ where

- $\Pi \subseteq \mathsf{PV}$ is a finite, non-empty set of *propositional variables*,
- $\Sigma = \{a_1, \ldots, a_n\} \subseteq \mathsf{AG}$ is a (finite) set of $n > 0$ *agents*,
- Q is a finite, non-empty set of *states*,
- $\pi : Q \to \mathbf{2}^{\Pi}$ is a *valuation function* which assigns to every state a set of propositional variables that are true there, and
- $\delta : Q \times \Sigma \to \mathbf{2}^{\mathbf{2}^Q}$ is a *transition function* which maps a state $q \in Q$ and an agent $a \in \Sigma$ to a set of *choices* $\delta(q, a)$ available to a at q such that the following condition is satisfied: for every state $q \in Q$ and every set $Q_{a_1}, \ldots, Q_{a_n}$ of choices $Q_{a_i} \in \delta(q, a_i)$, $1 \leq i \leq n$, the intersection $Q_{a_1} \cap \cdots \cap Q_{a_n}$ is a singleton set.

Intuitively, $\delta(q, a)$ describes the choices available to agent a in state q: when in state q, agent a chooses a set from $\delta(q, a)$ to ensure that the "next state" will be among those in the chosen set. It is natural to generalize this notion to *A-choices* for coalitions A: let $\mathcal{S} = \langle \Pi, \Sigma, Q, \pi, \delta \rangle$ be an ATS. For each state $q \in Q$ and each coalition $A \subseteq \Sigma$, set

$$\delta(q, A) := \begin{cases} \{Q_A \subseteq Q \mid Q_A = \bigcap_{a \in A} Q_a \text{ where for each } a \in A,\ Q_a \in \delta(q, a)\} & \text{if } A \neq \emptyset \\ \{\bigcup \delta(q, \Sigma)\} & \text{if } A = \emptyset \end{cases}.$$

When in state q, the coalition A may jointly choose a set from $\delta(q, A)$ to ensure that the next state is from this set. Observe that $\delta(q, \Sigma)$ is a set of singletons. The states appearing in singletons in $\delta(q, \Sigma)$ are the *successors* of q, i.e., whatever choices the individual agents make, the next state of the system will be from $\bigcup \delta(q, \Sigma)$. This explains the definition of $\delta(q, \emptyset)$: the empty set of agents cannot influence the behavior of the system, so the only choice that the empty coalition has is the set of all successors.

The definition of satisfaction of ATL formulas in ATSs relies on the notions of a computations and a strategy, which are introduced in the following. An infinite sequence $\lambda = q_0 q_1 q_2 \cdots \in Q^\omega$ of states is a *computation* if, for all positions $i \geq 0$,

q_{i+1} is a successor of q_i. As a notational convention, for any finite or infinite sequence $\lambda = \lambda_0\lambda_1\cdots$ and $i \geq 0$, we use $\lambda[i]$ to denote the i-th component λ_i and $\lambda[0,i]$ to denote the initial sequence $\lambda_0\cdots\lambda_i$. A *strategy* for an agent $a \in \Sigma$ is a mapping $f_a : Q^+ \to 2^Q$ such that for all $\lambda \in Q^*$ and all $q \in Q$, $f_a(\lambda \cdot q) \in \delta(q,a)$. Intuitively, a strategy f_a maps each history of states $\lambda \cdot q$ to a choice in $\delta(q,a)$ available to agent a at the current state q. A *strategy* for agents in a coalition $A \subseteq \Sigma$ is a set of strategies $F_A = \{f_a \mid a \in A\}$, one for each agent in A. The set $\mathsf{out}(q, F_A)$ of *outcomes* of a strategy F_A starting at a state $q \in Q$ is the set of all computations $\lambda = q_0q_1q_2\cdots \in Q^\omega$ such that $q_0 = q$ and $q_{i+1} \in \bigcap_{f_a \in F_A} f_a(\lambda[0,i])$ for all $i \geq 0$.

We can now define the semantics of ATL. Given an ATS $\mathcal{S} = \langle \Pi, \Sigma, Q, \pi, \delta\rangle$, the satisfaction relation $\models$ between states $q \in Q$ and ATL formulas is defined as follows:

- $\mathcal{S}, q \models p$ iff $p \in \pi(q)$ for all propositions $p \in \Pi$;
- $\mathcal{S}, q \models \neg\varphi$ iff $\mathcal{S}, q \not\models \varphi$;
- $\mathcal{S}, q \models \varphi_1 \vee \varphi_2$ iff $\mathcal{S}, q \models \varphi_1$ or $\mathcal{S}, q \models \varphi_2$;
- $\mathcal{S}, q \models \langle\langle A\rangle\rangle \bigcirc \varphi$ iff there is a strategy F_A such that for all computations $\lambda \in \mathsf{out}(q, F_A)$, it holds that $\mathcal{S}, \lambda[1] \models \varphi$;
- $\mathcal{S}, q \models \langle\langle A\rangle\rangle \Box \varphi$ iff there is a strategy F_A such that for all computations $\lambda \in \mathsf{out}(q, F_A)$, it holds that $\mathcal{S}, \lambda[i] \models \varphi$ for all positions $i \geq 0$;
- $\mathcal{S}, q \models \langle\langle A\rangle\rangle \varphi_1 \,\mathcal{U}\, \varphi_2$ iff there is a strategy F_A such that for all computations $\lambda \in \mathsf{out}(q, F_A)$, there is a position $i \geq 0$ such that $\mathcal{S}, \lambda[i] \models \varphi_2$ and $\mathcal{S}, \lambda[j] \models \varphi_1$ for all positions j with $0 \leq j < i$.

Note that there is an intimate relationship between CTL and ATL. Let Σ be the set of all agents in an ATS $\mathcal{S}$. On $\mathcal{S}$, we can then simulate CTL's existential path quantifier E as the cooperation expression $\langle\langle\Sigma\rangle\rangle$, while we can simulate CTL's universal path quantifier A as the cooperation expression $\langle\langle\emptyset\rangle\rangle$. Clearly, this translation only works if the set of agents Σ is finite and known in advance.

Here are some simple example formulas in ATL. The formula $\langle\langle C\rangle\rangle\Box\varphi$ asserts the *controllability* of the overall system by some coalition C with respect to property φ. That is, it states that the coalition C can cooperate to ensure that the property φ *always* holds in the system, no matter how the components of the system outside C behave. The formula $\langle\langle a\rangle\rangle\Diamond see_b(msg)$ says that agent a can guarantee that agent b eventually sees the message msg (where $see_b(msg)$ is an atomic proposition). Finally, $\langle\langle a\rangle\rangle\Box\neg see_b(msg)$ expresses that agent a can ensure that agent b never sees the message msg.

The complexity of satisfiability in ATL was first addressed by van Drimmelen in [195]. The main result proved in [195] is ExpTime-completeness of ATL satisfiability, where the lower bound stems from the simple reduction of satisfiability in CTL sketched above, and the upper bound is established by a reduction to the emptiness of alternating automata. However, as we observe in the joint paper [204] with Walther, Wolter, and Wooldridge, the result of van Drimmelen only covers the case where the

set of agents is finite and fixed, i.e., not viewed as part of the input: first, the reduction of CTL satisfiability requires that we can form the path quantifier $\langle\langle\Sigma\rangle\rangle$ and thus Σ has to be finite and known; and second, the automata used in the proof of the upper bound work on trees whose outdegree is exponential in $|\Sigma|$, and for this reason the runtime of the obtained algorithm is double exponential if $|\Sigma|$ is considered as part of the input, and single exponential only if Σ is assumed to be constant. To illustrate that the outdegree cannot easily be reduced, we exhibit the following sequence of ATL formulas $(\varphi_i)_{i\in\mathbb{N}}$. This sequence is such that, for any ATS $\mathcal{S}$, state q, and $i \geq 0$, $\mathcal{S}, q \models \varphi_i$ implies that q has at least 2^i successors in $\mathcal{S}$:

$$\varphi_i := \bigwedge_{1\leq j\leq i} \left(\langle\langle a_j\rangle\rangle \bigcirc p_j \wedge \langle\langle a_j\rangle\rangle \bigcirc \neg p_j\right)$$

As every agent a_i may choose the propositional letter p_i to be true or false at a successor state, jointly the agents $a_1, \ldots, a_k$ may choose any possible valuation of $p_1, \ldots, p_k$ for a successor state. As there are 2^i such valuations, there must be as many successors.

In [204], we consider the following three variations of satisfiability in ATL.

(a) *Satisfiability over given sets of agents*:

Given a finite set Σ of agents and a formula φ using only agents from Σ, is φ satisfiable in an ATS over Σ?

(b) *Satisfiability over arbitrary sets of agents*:

Given a formula φ, is there a finite set Σ of agents (containing the agents referred to in φ) such that φ is satisfiable in an ATS over Σ?

(c) *Satisfiability over formula-defined sets of agents*:

Given a formula φ, is φ satisfiable in an ATS over exactly the agents which occur in φ?

Notice that, in none of these problems, the set of agents is assumed to be fixed. Our first observation is that it suffices to consider Problem (a) for upper bounds and Problem (b) for lower bounds: Problems (a) and (c) are polynomially reducible to each other, while Problem (b) is polynomially reducible to (a). The reductions between (a) and (c) are fairly obvious. For the reduction from (b) to (a), we show that, for each formula φ and each set of agents $\Sigma \supset \Sigma_\varphi$, φ is satisfiable in an ATS for Σ iff it is satisfiable in an ATS for $\Sigma_\varphi \cup \{a\}$, for one fresh agent a.

We then prove that Problem (a) is in ExpTime by blending the techniques used by van Drimmelen in [195] with the standard "tableau algorithm" for CTL [63,65] which, although often called a tableau algorithm, has more resemblance with a Pratt-style elimination procedure [163]. The basic idea is to start with the set of all types, where a type is a set of subformulas of the input formula satisfying certain Boolean closure conditions. Then, we repeatedly eliminate types that cannot be realized in an ATS. To check realizability, we have to consider all coalition formulas $\langle\langle A\rangle\rangle\Diamond\varphi$ and $\langle\langle A\rangle\rangle\varphi\,\mathcal{U}\,\psi$

occurring in the type and try to construct a tree-shaped fragment of an ATS that witnesses satisfiability of the formula and uses only types not yet been eliminated. These trees have exponential outdegree, but still the overall runtime of the procedure is only exponential. A similar construction as used in the correctness proof of our algorithm has been independently developed by Goranko and van Drimmelen in [92], where it is used to prove completeness of an ATL axiomatization. Our technique has been extended to a version of ATL enriched with epistemic operators in [203].

We establish an ExpTime lower bound for Problem (b) by reducing the global consequence problem in the modal logic K, i.e., the problem to decide, given two formulas φ and ψ of the basic modal language, whether every Kripke structure K that makes φ true in every state also makes ψ true in every state. It follows that all of the Problems (a) to (c) introduced above are ExpTime-complete.

7 A Modal Approach to the Combination of Logics

Many of the modal logics that have been developed in computer science single out a particular aspect of the application domain such as the temporal evolution of the system state, and then aim at representing the chosen aspect as faithful as possible. When complex formal models are required in applications, there arises a need for logics that can represent multiple aspects of the application domain in an integrated way. For example, to analyze the dynamic aspects of multi-agent systems, we need a logic containing both temporal operators and operators for talking about the knowledge of agents. In general, there are basically two ways to address this shortcoming: first, we can construct for each application a new logical system that offers all the required operators; and second, we can use general combination methodologies to combine existing specialized logics into more general ones that subsume all the component logics w.r.t. expressive power. Several combination techniques have been analyzed in the literature including fusions [18, 123], fibring [78], and product constructions [80, 148]. In all these combination methods, a main concern is the transfer of "good" properties such as decidability, axiomatizations, and upper complexity bounds from the component logics to the combination.

In the joint paper [127] with Kutz, Wolter, and Zakharyaschev, we introduce a new combination method called $\mathcal{E}$*-connection* that has a distinguishedly modal flavor: the integration of the component logics is achieved by introducing additional modal operators that, intuitively, allow to switch from one component logic to another. This combination methodology is sufficiently "loose" so that decidability transfers in most relevant cases from the component logics to the combination. To formally talk about the combination of logics, we first need to define what a logic is. In [127], the notion of a logic is identified with what we call an *abstract description system (ADSs)*. These systems are semi-algebraic descriptions of logics that capture modal logics, description logics, and even many first- and higher-order logics. Since ADSs are heavy machinery whose definition is somewhat involved, in this summary we only introduce $\mathcal{E}$-connections using an example.

Suppose we want to develop a logic for a geographic information system about Europe that contains both political and spatial information. For representing the political information, we use the description logic $\mathcal{ALCO}$ and for the spatial information, we use the modal logic $\mathsf{S4}_u$. Both are introduced in the following.

$\mathcal{ALCO}$ is the extension of $\mathcal{ALC}$ (c.f. Section 3) obtained by introducing a set of *individual names* $\mathsf{N_I}$ as a new syntactic sort and allowing expressions $\{o\}$, with $o \in \mathsf{N_I}$, as additional atomic concepts. Regarding the semantics, an interpretation $\mathcal{I}$ is required to (additionally) assign to each individual name $o \in \mathsf{N_I}$ an element $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Then, the semantics of the new atomic concepts is defined by setting $\{o\}^{\mathcal{I}} := \{o^{\mathcal{I}}\}$. Concepts of the form $\{o\}$ are often called *nominals* and can be understood as concept names whose extension is required to be a singleton, see Section 4.2.

$\mathsf{S4}_u$ uses the basic modal language as introduced in Definition 1, where the set of programs is $\{i, u\}$. The semantics of $\mathsf{S4}_u$ is based on topological spaces as introduced in Section 5.2: a *topological structure* is a triple $(U, \mathbb{I}, V)$ such that $(U, \mathbb{I})$ is a topological

$\mathsf{dc}(p,q):$	$\neg\Diamond_u(p \wedge q)$
$\mathsf{eq}(p,q):$	$\Box_u(p \leftrightarrow q)$
$\mathsf{ec}(p,q):$	$\Diamond_u(p \wedge q) \wedge \neg\Diamond_u(\boldsymbol{I}p \wedge \boldsymbol{I}q)$
$\mathsf{po}(p,q):$	$\Diamond_u(\boldsymbol{I}p \wedge \boldsymbol{I}q) \wedge \Diamond_u(\boldsymbol{I}p \wedge \neg q) \wedge \Diamond_u(\boldsymbol{I}q \wedge \neg p)$
$\mathsf{tpp}(p,q):$	$\Box_u(\neg p \vee q) \wedge \Diamond_u(p \wedge \neg\boldsymbol{I}q) \wedge \Diamond_u(\neg p \wedge q)$
$\mathsf{ntpp}(p,q):$	$\Box_u(\neg p \vee \boldsymbol{I}q) \wedge \Diamond_u(\neg p \wedge q)$

Figure 4: The RCC8 relations in $\mathsf{S4}_u$

space and V maps each propositional variable to a subset of U. Then, we inductively extend the valuation V to complex formulas as follows:

$$\begin{aligned} V(\neg\varphi) &= U \setminus V(\varphi) \\ V(\varphi \wedge \psi) &= V(\varphi) \cap V(\psi) \\ V(\Box_i\varphi) &= \mathbb{I}(V(\varphi)) \\ V(\Box_u\varphi) &= \begin{cases} U & \text{if } V(\varphi) = U \\ \emptyset & \text{otherwise} \end{cases} \end{aligned}$$

Note that the above implies $V(\Diamond_i\varphi) = \mathbb{C}(V(\varphi))$, where $\mathbb{C}$ is the topological closure operator. In the following, we will use the symbol $\boldsymbol{I}$ for $\Box_i$ and $\boldsymbol{C}$ for $\Diamond_i$. To utilize $\mathsf{S4}_u$ for spatial reasoning, we can describe the RCC8 relations (as introduced in Section 5.2) using formulas of $\mathsf{S4}_u$ as shown in Figure 4. In the figure, tppi and ntppi are omitted since they are symmetric to tpp and ntpp. To ensure that propositional variables identify regions that are regular closed sets, we can add the assertion $\Box_u(\boldsymbol{CI}p = p)$ for each propositional variable p.

We now integrate $\mathcal{ALCO}$ and $\mathsf{S4}_u$ into a single logic $\mathcal{ALCO}$-$\mathsf{S4}_u$ using $\mathcal{E}$-connections. The following exposition is informal at times, please refer to [127] for full details. The basic ideas of $\mathcal{E}$-connections are (i) to interpret the component logics in disjoint domains, (ii) to link these domains by means of *connection relations*, and (iii) to introduce additional modal operators for the connection relations. The syntax of $\mathcal{ALCO}$-$\mathsf{S4}_u$ comprises concepts for the $\mathcal{ALCO}$ part and formulas for the $\mathsf{S4}_u$ part as distinct sorts. Formally, concepts and formulas of $\mathcal{ALCO}$-$\mathsf{S4}_u$ are obtained by uniting the concept and formula formation rules of the two component logics, and adding the following:

- if φ is a formula, the $\langle se\rangle\varphi$ is a concept;
- if C is a concept, then $\langle se^-\rangle C$ is a formula.

Here, *se* is a name for the only connection relation used in this example and stands for spatial extension. A *structure* for $\mathcal{ALCO}$-$\mathsf{S4}_u$ is a triple $(\mathcal{I}, T, r)$, where $\mathcal{I}$ is an $\mathcal{ALCO}$ interpretation, $T = (U, \mathbb{I}, V)$ a topological structure for $\mathsf{S4}_u$, and $r \subseteq \Delta^{\mathcal{I}} \times U$ a *connection relation*. Intuitively, the set $\{x \in U \mid (a,x) \in r\}$ describes the spatial extension of the object $a \in \Delta^{\mathcal{I}}$. The extension $C^{\mathcal{I}}$ of concepts C and $V(\varphi)$ of formulas

φ is then defined inductively using the same clauses as for $\mathcal{ALCO}$ and $\mathsf{S4}_u$, as well as the following ones:

- $(\langle se\rangle\varphi)^{\mathcal{I}} := \{d \in \Delta^{\mathcal{I}} \mid \exists x \in U : (d,x) \in r \wedge x \in V(\varphi)\}$
- $V(\langle se^{-}\rangle C) := \{x \in U \mid \exists d \in \Delta^{\mathcal{I}} : (d,x) \in r \wedge d \in C^{\mathcal{I}}\}$

Thus, $\langle se\rangle\varphi$ behaves like a modal diamond on the relation r, and $\langle se^{-}\rangle C$ behaves like a modal diamond on the converse of r. In general, an $\mathcal{E}$-connection can connect an arbitrary number of logics using an arbitrary number of connection relations. The intuitive meaning of the connection relation depends on the logics that it connects and on the application.

Let us now indicate how to use $\mathcal{ALCO}$-$\mathsf{S4}_u$ for constructing the geographic information system about Europe. We start with formalizing political information in an $\mathcal{ALCO}$ TBox. We take concept names Country, Treaty, etc., individual names *EU*, *Schengen_treaty*, *Spain*, *Luxembourg*, *UK*, etc., and a role member, and then write statements such as

$$\begin{array}{rcl}
\{Luxembourg\} & \sqsubseteq & \exists\mathsf{member\text{-}of}.\{EU\} \sqcap \exists\mathsf{member\text{-}of}.\{Schengen_treaty\} \\
\{Iceland\} & \sqsubseteq & \exists\mathsf{member\text{-}of}.\{Schengen_treaty\} \sqcap \neg\exists\mathsf{member\text{-}of}.\{EU\} \\
\{France\} & \sqsubseteq & \mathsf{Country} \\
\{Schengen_treaty\} & \sqsubseteq & \mathsf{Treaty} \\
\mathsf{Country} & \sqsupseteq & \exists\mathsf{member\text{-}of}.\{Schengen_treaty\}
\end{array}$$

To capture the spatial knowledge about the countries and areas introduced in the above TBox, we use the $\mathsf{S4}_u$ part of $\mathcal{ALCO}$-$\mathsf{S4}_u$ together with the connection operators. For example, we add the following three formulas:

$$\begin{array}{l}
\mathsf{eq}(\langle se^{-}\rangle\{EU\}, \langle se^{-}\rangle(\{Portugal\} \sqcup \{Spain\} \sqcup \cdots \sqcup \{UK\}) \\
\mathsf{ec}(\langle se^{-}\rangle\{France\}, \langle se^{-}\rangle\{Luxembourg\}) \\
\mathsf{ntpp}(\langle se^{-}\rangle\{Luxembourg\}, \langle se^{-}\rangle(\exists\mathsf{member\text{-}of}.\{Schengen_Treaty\}))
\end{array}$$

stating that the space occupied by the EU is the space occupied by its members, France and Luxembourg have a common border, and if you cross the border of Luxembourg, then you enter a member of the Schengen treaty. The propositional variables of $\mathsf{S4}_u$ can be used to identify regions that have a geographic meaning, but not a political one such as Sea and Mountains. Then, we could express that there are mountains in Germany by writing

$$\{Germany\} \sqsubseteq \langle se\rangle\mathsf{Mountains}.$$

In [127], we show that if satisfiability is decidable in the component logics, then satisfiability in the $\mathcal{E}$-connection is also decidable. Applied to a TBox $\mathcal{T}$ of the example $\mathcal{E}$-connection $\mathcal{ALCO}$-$\mathsf{S4}_u$ from above, the algorithm developed in [127] attempts to construct

- a set Γ of concept types (sets of concepts satisfying some Boolean closure conditions),

- a set Θ of formula types (the analogue of concept types), and
- a relation r between concept types and formula types

such that Γ, Θ, and r can be realized in a model $(\mathcal{I}, T, r')$, i.e., the set of (concept) types occurring in $\mathcal{I}$ is exactly Γ, the set of (formula) types occurring in T is exactly Θ, and r' connects an element of $\mathcal{I}$ with an element of T iff their types are connected by r. It is central to our algorithm that the realizability of Γ, Θ, and r can be reduced to deciding satisfiability in the component logics. The time complexity of the resulting algorithm is one exponential higher than the time complexity of the decision procedures for the component logics. We also show that, when $\mathcal{E}$-connecting logics, an increase in complexity cannot always be avoided. This is done by exhibiting an ADS for which satisfiability is NP-complete, but whose $\mathcal{E}$-connection with itself is ExpTime-complete.

We then introduce several variations of the basic $\mathcal{E}$-connection formalism, obtained for example by

- allowing to apply the Boolean operators to link relations, thus introducing connection operators such as $\langle (r \cap r') \cup r'' \rangle \varphi$, c.f. Section 4.3.
- adding counting capabilities to the link operators so that they can express "there are at least/at most n instances connected via r and satisfying φ", c.f. Sections 3 and 4.2.

In the first case, we obtain transfer results for decidability that are as general as in the basic case, though the proofs are considerably more involved. Here, we can prove that our algorithm is optimal by giving a (natural) example of an ADS that is NP-complete, but whose $\mathcal{E}$-connection with itself is NExpTime-complete. In the second case, the component logics have to satisfy certain conditions for decidability to transfer. Intuitively, the problem is that, when counting on link relations is enabled, cardinality statements can be "exported" from one component to another. We also show that distributed description logics (DDLs), a formalism proposed in [38] for the integration of ontologies, can be conceived as a special case of $\mathcal{E}$-connections.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] M. Adler and N. Immerman. An $n!$ lower bound on formula size. *ACM Transactions on Computational Logic*, 4(3):296–314, 2003.

[3] L. Afanasiev, P. Blackburn, I. Dimitriou, B. Gaiffe, E. Goris, M. Maarx, and M. de Rijke. PDL for ordered trees. *Journal of Applied Non-Classical Logic*, 15(2):115–135, 2005.

[4] F. Afrati, C. H. Papadimitriou, and G. Papageorgiou. The synthesis of communication protocols. *Algorithmica*, 3(3):451–472, 1988.

[5] M. Aiello and J. van Benthem. A modal walk through space. *Journal of Applied Non-Classical Logics*, 12(3-4):319–364, 2002.

[6] M. Aiello, J. van Benthem, and G. Bezhanishvili. Reasoning about space: The modal way. *Journal of Logic and Computation*, 13(6):889–920, 2003.

[7] N. Alechina, S. Demri, and M. de Rijke. A modal perspective on path constraints. *Journal of Logic and Computation*, 13(6):939–956, 2003.

[8] R. Alur, T. Feder, and T. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43:116–146, 1996.

[9] R. Alur and T. Henzinger. Logics and models of real time: a survey. In *Real Time: Theory and Practice*, Lecture Notes in Computer Science, pages 74–106, Berlin, 1992. Springer-Verlag.

[10] R. Alur and T. Henzinger. A really temporal logic. *Journal of the ACM*, 41:181–204, 1994.

[11] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[12] H. Andréka, I. Németi, and I. Sain. *Handbook of Philosophical Logic*, volume 2, chapter Algebraic Logic, pages 133–247. Kluwer Academic Publishers, second edition, 2001.

[13] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[14] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 446–451, Sydney, Australia, 1991.

[15] F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 18(2–4):175–219, 1996.

[16] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 364–369. Morgan Kaufmann, 2005.

[17] F. Baader and C. Lutz. Description logic. In J. van Benthem, P. Blackburn, and F. Wolter, editors, *Handbook of Modal Logic*. Elsevier. To appear.

[18] F. Baader, C. Lutz, H. Sturm, and F. Wolter. Fusions of description logics and abstract description systems. *Journal of Artificial Intelligence Research (JAIR)*, 16:1–58, 2002.

[19] F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Proceedings of the Methods for Modalities Workshop (M4M-05)*, Berlin, Germany, 2005.

[20] F. Baader, D. L. McGuiness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press, 2003.

[21] F. Baader and U. Sattler. Tableau algorithms for description logics. In R. Dyckhoff, editor, *Proceedings of the International Conference on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*, volume 1847 of *Lecture Notes in Artificial Intelligence*, pages 1–18. Springer-Verlag, 2000.

[22] A. Baltag, L. S. Moss, and S. Solecki. The logic of public announcements and common knowledge for distributed applications (extended abstract). In I. Gilboa, editor, *Theoretical Aspects of Reasoning about Knowledge: Proceedings of the Seventh Conference (TARK 1998)*, pages 43–56. Morgan Kaufmann, 1998.

[23] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schner, and L. A. Stein. OWL web ontology language reference. W3C Recommendation, 2004.

[24] 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(3):402–417, 1982.

[25] B. Bennett. Modal logics for qualitative spatial reasoning. *Journal of the Interest Group in Pure and Applied Logic*, 4(1), 1997.

[26] B. Bennett, A. G. Cohn, F. Wolter, and M. Zakharyaschev. Multi-dimensional modal logic as a framework for spatio-temporal reasoning. *Applied Intelligence*, 17(3):239–251, 2002.

[27] B. Bérard and C. Picaronny. Accepting zeno words: A way toward timed refinements. *Acta Informaticae*, 37(1):45–81, 2000.

[28] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In *Proceedings of the 1st International Conference on Service Oriented Computing (ICSOC 2003)*, volume 2910 of *Lecture Notes in Computer Science*, pages 43–58. Springer, 2003.

[29] R. Berger. The undecidability of the dominoe problem. *Memoirs of the American Mathematical Society*, 66, 1966.

[30] O. Bernholtz, M. Y. Vardi, and P. Wolper. An automata–theoretic approach to branching–time model checking. In D. L. Dill, editor, *Proceedings of the 6th International Conference on Computer–Aided Verification (CAV'94)*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155. Springer, 1994.

[31] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[32] M. Bojanczyk. The finite graph problem for two-way alternating automata. *Theoretical Computer Science*, 3(298):511–528, 2003.

[33] P. A. Bonatti, A. Murano, C. Lutz, and M. Vardi. Complexity of enriched μ-calculi. In *33rd International Colloquium on Automata, Languages and Programming (ICALP 2006)*. Submitted.

[34] P. A. Bonatti and A. Peron. On the undecidability of logics with converse, nominals, recursion and counting. *Artificial Intelligence*, 158(1):75–96, 2004.

[35] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer-Verlag, 1997.

[36] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.

[37] A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the classic description logic. *Journal of Artificial Intelligence Research*, pages 277–308, 1994.

[38] A. Borgida and L. Serafini. Distributed description logics: Assimilating information from peer sources. *Journal of Data Semantics*, 1:153–184, 2003.

[39] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216, 1985.

[40] J. Bradfield and C. Stirling. Modal logics and μ-calculi: an introduction. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra*, pages 293–330. Elsevier, 2001.

[41] S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In R. L. de Mantáras and L. Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-2004)*, pages 298–302. IOS Press, 2004.

[42] T. Braüner and S. Ghilardi. First-order modal logic. In J. van Benthem, P. Blackburn, and F. Wolter, editors, *Handbook of Modal Logic*. Elsevier. To appear.

[43] G. Brewka, J. Dix, and K. Konolige. *Nonmonotonic Reasoning: An Overview*, volume 73. CSLI Publications, 1997.

[44] R. A. Bull. That all normal extensions of S4.3 have the finite model property. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 12:341–344, 1966.

[45] R. M. Burstall. Program proving as hand simulation with a little induction. In *International congress of the International Federation for Information Processing (IFIP '74)*, pages 308–312, 1974.

[46] D. Calvanese. Finite model reasoning in description logics. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 292–303. Morgan Kaufmann Publishers, 1996.

[47] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proceedings of the 17th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[48] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, 1999.

[49] D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 229–263. Kluwer Academic Publisher, 1998.

[50] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[51] E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of the Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.

[52] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[53] E. M. Clarke and H. Schlingloff. Model checking. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 24, pages 1635–1790. Elsevier Science, 2001.

[54] J. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.

[55] A. G. Cohn. Modal and non modal qualitative spatial logics. In F. D. Anger, H. M. Guesgen, and J. van Benthem, editors, *Proceedings of the Workshop on Spatial and Temporal Reasoning*. IJCAI-93, IJCAI-93, 1993.

[56] R. Cote, D. Rothwell, J. Palotay, R. Beckett, and L. Brochu. The systematized nomenclature of human and veterinary medicine. Technical report, SNOMED International, Northfield, IL: College of American Pathologists, 1993.

[57] R. Danecki. Nondeterministic propositional dynamic logic with intersection is decidable. In A. Skowron, editor, *Proceedings of the Fifth Symposium on Computation Theory*, volume 208 of *Lecture Notes in Computer Science*, pages 34–53, Zaborów, Poland, Dec. 1984. Springer.

[58] 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 (AI*IA'95)*, volume 992, pages 103–114. Springer, 1995.

[59] S. Demri and D. Nowak. Reasoning about transfinite sequences (extended abstract). In D. A. Peled and Y.-K. Tsay, editors, *Proceedings of the 3rd International Symposium on Automated Technology for Verification and Analysis (ATVA'05)*, volume 3707 of *Lecture Notes in Computer Science*, pages 248–262. Springer, 2005.

[60] F. M. Donini, B. Hollunder, M. Lenzerini, A. M. Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.

[61] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. *Information and Computation*, 134(1):1–58, 1997.

[62] M. J. Egenhofer and R. G. Golledge, editors. *Spatial and Temporal Reasoning in Geographic Information Systems*. Oxford University Press, 1998.

[63] E. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science. Volume B*, pages 995–1072. North-Holland, Amsterdam, 1990.

[64] E. Emerson, A. Mok, A. Sistla, and J. Srinivasan. Quantitative temporal reasoning. *Real-Time Systems*, 4:331 – 352, 1992.

[65] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. In *STOC '82: Proceedings of the fourteenth*

annual ACM symposium on Theory of computing, pages 169–180. ACM Press, 1982.

[66] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[67] E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, FOCS'88*, pages 328–337. IEEE, 1988.

[68] E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time strikes back. In B. K. Reid, editor, *Conference Record of the 12th Annual ACM Symposium on Principles of Programming Languages*, pages 84–96. ACM Press, 1985.

[69] K. Etessami, M. Y. Vardi, and T. Wilke. First-order logic with two variables and unary temporal logic. *Information and Computation*, 179(2):279–295, 2002.

[70] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[71] L. Farinas Del Cerro and E. Orlowska. DAL-a logic for data analysis. *Theoretical Computer Science*, 36(2-3):251–264, 1985.

[72] M. Fattorosi-Barnaba and F. de Caro. Graded modalities I. *Studia Logica*, 44:197–221, 1985.

[73] K. Fine. In so many possible worlds. *Notre Dame Journal of Formal Logic*, 13:516–520, 1972.

[74] M. J. Fischer and R. E. Ladner. Propositional modal logic of programs. In *Conference record of the ninth annual ACM Symposium on Theory of Computing*, pages 286–294. ACM Press, 1977.

[75] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[76] M. Fürer. The computational complexity of the unconstrained limited domino problem (with implications for logical decision problems). In *Logic and Machines: Decision problems and complexity*, pages 312–319. Springer-Verlag, 1984.

[77] D. Gabbay. Expressive functional completeness in tense logic. In U. Mönnich, editor, *Aspects of Philosophical Logic*, pages 91–117. Reidel, Dordrecht, 1981.

[78] D. M. Gabbay. *Fibring Logics*, volume 38 of *Oxford Logic Guides*. Oxford University Press, 1999.

[79] D. M. Gabbay, I. M. Hodkinson, and M. A. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects, Volume 1*. Oxford University Press, Logic Guides 28, 1994.

[80] D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Number 148 in Studies in Logic and the Foundations of Mathematics. Elsevier, 2003.

[81] D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyaschev. Combining spatial and temporal logics: Expressiveness vs. complexity. *Journal of Artificial Intelligence Research*, 23:167–243, 2005.

[82] G. Gargov and S. Passy. A note on Boolean modal logic. In D. Skordev, editor, *Mathematical Logic and Applications*, pages 253–263. Plenum Press, 1987.

[83] G. Gargov, S. Passy, and T. Tinchev. Modal environment for Boolean speculations. In D. Skordev, editor, *Mathematical Logic and Applications*, pages 253–263, New York, USA, 1987. Plenum Press.

[84] J. Gerbrandy. *Bisimulations on Planet Kripke*. ILLC Dissertation Series, Amsterdam, 1999.

[85] S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? A case for conservative extensions in description logics. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, 2006.

[86] G. D. Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI'94). Volume 1*, pages 205–212. AAAI Press, 1994.

[87] E. Giunchiglia, A. Tacchella, and F. Giunchiglia. SAT-based decision procedures for classical modal logics. *Journal of Automated Reasoning*, 28(2):143–171, 2002.

[88] K. Gödel. Eine Interpretation des intuitionistischen Aussagenkalküls. *Ergebnisse eines mathematischen Kolloquiums*, pages 39–40, 1933.

[89] V. Goranko. Completeness and incompleteness in the bimodal base $\mathrm{L}(R, -R)$. In *Proceedings of the Conference on Mathematical Logic "Heyting '88", Chaika, Bulgaria*. Plenum Press, 1987.

[90] V. Goranko. Modal definability in enriched languages. *Notre Dame Journal of Formal Logic*, 31(1):81–105, 1990.

[91] V. Goranko and S. Passy. Using the universal modality: Gains and questions. *Journal of Logic and Computation*, 2(1):5–30, 1992.

[92] V. Goranko and G. van Drimmelen. Decidability and complete axiomatization of the alternating-time temporal logic. *Theoretical Computer Science*. To appear.

[93] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64:1719–1742, 1999.

[94] E. Grädel. Why are modal logics so robustly decidable? *Bulletin of the European Association for Theoretical Computer Science*, 68:90–103, 1999.

[95] E. Grädel, P. Kolaitis, and M. Vardi. On the Decision Problem for Two-Variable First-Order Logic. *Bulletin of Symbolic Logic*, 3:53–69, 1997.

[96] E. Grädel, M. Otto, and E. Rosen. Two-Variable Logic with Counting is Decidable. In *Proceedings of Twelfth IEEE Symposium on Logic in Computer Science (LICS'97)*, 1997.

[97] E. Grädel, M. Otto, and E. Rosen. Undecidability results on two-variable logics. In *14th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1200 of *Lecture Notes in Computer Science*, pages 249–260. Springer Verlag, 1997.

[98] E. Grädel and I. Walukiewicz. Guarded Fixed Point Logic. In *Proceedings of Fourteenth IEEE Symposium on Logic in Computer Science (LICS'99)*, pages 45–54, 1999.

[99] N. Guarino. Formal ontologies and information systems. In *Proceedings of FOIS'1998*, pages 3–15. IOS Press, 1998.

[100] Y. Gurevich. Feasible functions. *London Mathematical Society Newsletter*, (206):6–7, 1993.

[101] Y. Gurevich. The value, if any, of decidability. *Bulletin of the EATCS*, 55, 1995.

[102] V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR'01)*, number 2083 in Lecture Notes in Artifical Intelligence, pages 701–705. Springer-Verlag, 2001.

[103] J. Y. Halpern and R. Fagin. Modelling knowledge and action in distributed systems. *Distributed Computing*, 3(4):159–177, 1989.

[104] J. Y. Halpern, R. Harper, N. Immerman, P. G. Kolaitis, M. Y. Vardi, and V. Vianu. On the unusual effectiveness of logic in computer science. *The Bulletin of Symbolic Logic*, 7(2):213–236, June 2001.

[105] M. R. Hansen, P. K. Pandya, and Z. Chaochen. Finite divergence. *Theoretical Computer Science*, 138(1):113–139, 1995.

[106] D. Harel. Dynamic logic. In D. M. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Volume II*, pages 496–604. D. Reidel Publishers, 1984.

[107] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[108] L. Henkin, J. D. Monk, and A. Tarski. *Cylindric Algebras, Part I*. North-Holland, Amsterdam, 1971.

[109] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–161, 1985.

[110] A. Heyting. Die formalen Reglen der intuitionistischen Logik. *Sitzungsberichete der Preussischen Akademie der Wissenschaften, Physikalische-Mathematische Klasse*, pages 42–56, 1930.

[111] J. Hintikka. *Time and Necessity: Studies in Aristotle's Theory of Modality*. Oxford University Press, 1973.

[112] Y. Hirshfeld and A. Rabinovich. Logics for real time: Decidability and complexity. *Fundamenta Informaticae*, 62:1–28, 2004.

[113] W. Hoek and M. Wooldridge. Time, knowledge, and cooperation: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[114] I. Horrocks. Using an expressive description logic: Fact or fiction? In *Proceedings of the Sixth International Conference on the Principles of Knowledge Representation and Reasoning (KR98)*, pages 636–647, 1998.

[115] I. L. Humberstone. Inaccessible worlds. *Notre Dame Journal of Formal Logic*, 24(3):346–352, 1983.

[116] N. Immerman. *Descriptive Complexity*. Springer-Verlag, 1999.

[117] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *CONCUR '96: Concurrency Theory, 7th International Conference*, volume 1119 of *Lecture Notes in Computer Science*, pages 263–277. Springer-Verlag, 1996.

[118] H. Kamp. *On tense logic and the theory of order*. PhD thesis, UCLA, 1968.

[119] Y. Kazakov and H. de Nivelle. Subsumption of concepts in $\mathcal{FL}_0$ for (cyclic) terminologies with respect to descriptive semantics is PSpace-complete. In E. F. Diego Calvanese, Giuseppe De Giacomo, editor, *Proceedings of the International Workshop in Description Logics 2003 (DL2003)*, number 81 in CEUR-WS (http://ceur-ws.org/), 2003.

[120] D. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, 1968.

[121] K. Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence*, 35(3):343–382, 1988.

[122] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(1):333–354, 1983.

[123] M. Kracht and F. Wolter. Properties of independently axiomatizable bimodal logics. *The Journal of Symbolic Logic*, 56(4):1469–1485, 1991.

[124] S. A. Kripke. Semantic considerations on modal logic. *Acta Philosophica Fennica*, 24:83–94, 1963.

[125] S. A. Kripke. Semantical analysis of modal logic I. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[126] O. Kupferman, U. Sattler, and M. Y. Vardi. The complexity of the graded mu-calculus. In *Proceedings of the Conference on Automated Deduction*, volume 2392 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 2002.

[127] O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. $\mathcal{E}$-connections of abstract description systems. *Artificial Intelligence*, 156(1):1–73, 2004.

[128] O. Kutz, F. Wolter, H. Sturm, N.-Y. Suzuki, and M. Zakharyaschev. Logics of metric spaces. *ACM Trans. Comput. Log*, 4(2):260–294, 2003.

[129] O. Kutz, F. Wolter, and M. Zakharyaschev. Connecting abstract description systems. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, pages 215–226. Morgan Kaufmann, 2002.

[130] M. Lange. A lower complexity bound for propositional dynamic logic with intersection. In R. A. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *Advances in Modal Logic Volume 5*. King's College Publications, 2005.

[131] M. Lange and C. Lutz. 2-EXPTIME lower bounds for propositional dynamic logics with intersection. *Journal of Symbolic Logic*, 70(5):1072–1086, 2005.

[132] C. I. Lewis. *A survey of symbolic logic*. Berkeley University Press, 1918.

[133] C. I. Lewis and C. H. Langford. *Symbolic Logic*. Dover, 1932.

[134] L. Libkin. *Elements of Finite Model Theory*. Springer Verlag, 2004.

[135] P. Longley, M. Goodchild, D. Maguire, and D. Rhind. *Geographic Information Systems and Science*. Wiley and sons, 2001.

[136] C. Lutz. PDL with intersection and converse is decidable. In C.-H. L. Ong, editor, *Computer Science Logic (CSL'05)*, volume 3634 of *Lecture Notes in Computer Science*, pages 413–427. Springer-Verlag, 2005.

[137] C. Lutz. Complexity and succinctness of public announcement logic. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2006.

[138] C. Lutz and U. Sattler. Mary likes all cats. In F. Baader and U. Sattler, editors, *Proceedings of the 2000 International Workshop in Description Logics (DL2000)*, number 33 in CEUR-WS (http://ceur-ws.org/), pages 213–226, 2000.

[139] C. Lutz and U. Sattler. The complexity of reasoning with boolean modal logics. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, editors, *Advances in Modal Logics Volume 3*. CSLI Publications, Stanford, CA, USA, 2001.

[140] C. Lutz, U. Sattler, and L. Tendera. The complexity of finite model reasoning in description logics. In *Proceedings of the 19th Conference on Automated Deduction (CADE-19)*, Lecture Notes in Artificial Intelligence, pages 60–74. Springer Verlag, 2003.

[141] C. Lutz, U. Sattler, and L. Tendera. The complexity of finite model reasoning in description logics. *Information and Computation*, 199:132–171, 2005.

[142] C. Lutz, U. Sattler, and F. Wolter. Modal logic and the two-variable fragment. In L. Fribourg, editor, *Computer Science Logic*, number 2142 in Lecture Notes in Computer Science, pages 247–261. Springer-Verlag, 2001.

[143] C. Lutz, D. Walther, and F. Wolter. Quantitative temporal logics: PSpace and below. In *Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning (TIME-05)*, pages 138–146. IEEE Computer Society Press.

[144] C. Lutz and F. Wolter. Modal logics of topological relations. In R. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *Advances in Modal Logic (AiML'04)*, pages 249–263, 2004.

[145] C. Lutz and F. Wolter. Modal logics of topological relations. *Logical Methods in Computer Science*, 2005. Submitted.

[146] E. Mally. *Grundgesetze des Sollens: Elemente der Logik des Willens*. Leuschner und Lubensky, Universitäts-Buchhandlung, 1926.

[147] M. Marx and M. Reynolds. Undecidability of compass logic. *Journal of Logic and Computation*, 9(6), 1999.

[148] M. Marx and Y. Venema. *Multi-Dimensional Modal Logic*. Kluwer Academic Publishers, 1997.

[149] F. Massacci. Decision procedures for expressive description logics with role intersection, composition and converse. In B. Nebel, editor, *Proceedings of the seventeenth International Conference on Artificial Intelligence (IJCAI-01)*, pages 193–198. Morgan Kaufmann, Aug. 4–10 2001.

[150] J. McKinsey. A solution of the decision problem for the lewis systems S2 and S4, with an application to topology. *Journal of Symbolic Logic*, 6(4):117–134, 1941.

[151] J. McKinsey and A. Tarski. The algebra of topology. *Annals of Mathematics*, 45(1):141–191, 1944.

[152] M. Minsky. A framework for representating knowledge. In P. H. Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, New York, USA, 1975.

[153] M.O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[154] R. C. Moore. Possible-world semantics for autoepistemic logic. In *Proceedings of the 1st International Workshop on Nonmonotonic Reasoning*, pages 344–354, 1984.

[155] B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34(3):371–383, 1988.

[156] B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[157] W. Nutt. On the translation of qualitative spatial reasoning problems into modal logics. In W. Burgard, T. Christaller, and A. B. Cremers, editors, *KI-99: Advances in Artificial Intelligence*, volume 1701 of *Lecture Notes in Artificial Intelligence*, pages 113–124. Springer-Verlag, 1999.

[158] I. Orlov. The calculus of compatibility of propositions. *Mathematics of the USSR*, 35:263–286, 1928.

[159] L. Pacholski, W. Szwast, and L. Tendera. Complexity results for first-order two-variable logic with counting. *SIAM Journal on Computing*, 29(4):1083–1117, Aug. 2000.

[160] G. Pan, U. Sattler, and M. Y. Vardi. BDD-based decision procedures for K. In *Proceedings of the Conference on Automated Deduction*, volume 2392 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 2002.

[161] J. A. Plaza. Logics of public communications. In M. L. Emrich, M. Pfeifer, M. Hadzikadic, and Z. M. Ras, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216, 1989.

[162] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on the Foundations of Computer Science (FOCS-77)*, pages 46–57. IEEE Computer Society Press, 1977.

[163] V. R. Pratt. Models of program logics. In *Proceedings of the Twentieth Annual Symposium on Foundations of Computer Science*, San Juan, Puerto Rico, 1979.

[164] V. R. Pratt. A decidable mu-calculus: Preliminary report. In *Proceedings of the 22nd Annual IEEE Symposium on Foundations of Computer Science, FOCS'81*, pages 421–427. IEEE, 1981.

[165] I. Pratt-Hartmann. A topological constraint language with component counting. *Journal of Applied Non-Classical Logics*, 12(3–4):441–467, 2002.

[166] I. Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language, and Information*, 14(3):369–395, 2005.

[167] A. N. Prior. *Past, present and future*. Oxford University Press, 1967.

[168] M. R. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*, pages 227–270. MIT Press, 1968.

[169] W. Rautenberg. Der Verband der normalen verzweigten Modallogiken. *Math. Zeitschrift*, 156:123–140, 1977.

[170] A. Rector and I. Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97)*. AAAI Press, 1997.

[171] Reynolds and Zakharyaschev. On the products of linear modal logics. *Journal of Logic and Computation*, 11:909–931, 2001.

[172] M. Reynolds. The complexity of the temporal logic over the reals. Manuscript, currently under submission.

[173] H. Sahlqvist. Completeness and correspondence in the first and second order semantics for modal logic. In S. Kanger, editor, *Proceedings of the Third Scandinavian Logic Symposium*, pages 110–143. North-Holland, 1975.

[174] U. Sattler. Description logics for ontologies. Habilitation thesis, Institute of Theoretical Computer Science, TU Dresden, 2003.

[175] U. Sattler and M. Vardi. The hybrid μ-calculus. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proceedings of the First International Joint Conference on Automated Reasoning (IJCAR 2001)*, number 2083 in Lecture Notes in Artifical Intelligence, pages 76–91. Springer-Verlag, 2001.

[176] K. Schild. Terminological cycles and the propositional μ-calculus. In P. T. Jon Doyle, Erik Sandewall, editor, *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 509–520. Morgan Kaufmann, 1994.

[177] K. D. Schild. A correspondence theory for terminological logics: Preliminary report. In J. Mylopoulos and R. Reiter, editors, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471. Morgan Kaufmann, 1991.

[178] K. D. Schild. Combining terminological logics with tense logic. In M. Filgueiras and L. Damas, editors, *Progress in Artificial Intelligence – 6th Portuguese Conference on Artificial Intelligence, EPIA'93*, volume 727 of *Lecture Notes in Artificial Intelligence*, pages 105–120. Springer-Verlag, 1993.

[179] P. Schobbens, J. Raskin, and T. Henzinger. Axioms for real-time logics. *Theoretical Computer Science*, 274:151–182, 2002.

[180] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, Chichester, UK, 1986.

[181] A. P. Sistla and E. M. Clarke. Complexity of propositional temporal logics. *Journal of the ACM*, 32:733–749, 1985.

[182] A. P. Sistla and L. D. Zuck. Reasoning in a restricted temporal logic. *Information and Computation*, 102(2):167–195, 1993.

[183] K. Spackman, K. Campbell, and R. Cote. SNOMED RT: A reference terminology for health care. *Journal of the American Medical Informatics Association*, pages 640–644, 1997. Fall Symposium Supplement.

[184] O. Stock, editor. *Spatial and Temporal Reasoning.* Kluwer Academic Publishers, Dordrecht, Holland, 1997.

[185] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *ACM Symposium on Theory of Computing (STOC '73)*, pages 1–9. ACM Press, 1973.

[186] R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Control*, 54(1–2):121–141, 1982.

[187] The Gene Ontology Consortium. Gene Ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

[188] J. van Benthem. Logics for information update. In J. van Benthem, editor, *Proceedings of TARK-VIII*, pages 51–88. Morgan Kaufmann, 2001.

[189] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. Unpublished Manuscript.

[190] J. F. A. K. van Benthem. *Modal Logic and Classical Logic.* Bibliopolis, Naples, Italy, 1983.

[191] W. van der Hoek and M. de Rijke. Counting objects. *Journal of Logic and Computation*, 5(3):325–345, 1995.

[192] H. van Ditmarsch, W. van der Hoek, and B. Kooi. Dynamic epistemic logic with assignment. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, M. Singh, and M. Wooldridge, editors, *Proceedings of AAMAS 2005 (Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems)*, pages 141–148. ACM Inc., 2005.

[193] H. van Ditmarsch, W. van der Hoek, and B. Kooi. Public announcements and belief expansion. In R. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *Advances in Modal Logic, Volume 5*, pages 335–346. King's College Publications, 2005.

[194] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Kluwer academic publishers, to appear.

[195] G. van Drimmelen. Satisfiability in alternating-time temporal logic. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS 2003)*. IEEE Computer Society, 2003.

[196] M. Y. Vardi. A temporal fixpoint calculus. In *Conference Record of the 15th Annual ACM Symposium on Principles of Programming Languages*, pages 250–259, 1988.

[197] M. Y. Vardi. Alternating automata and program verification. In J. van Leeuwen, editor, *Computer Science Today*, volume 1000 of *Lecture Notes in Computer Science*, pages 471–485. Springer-Verlag, 1995.

[198] M. Y. Vardi. Why is modal logic so robustly decidable? In N. Immerman and P. G. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1997.

[199] M. Y. Vardi. Reasoning about the past with two-way automata. In K. G. Larsen, S. Skyum, and G. Winskel, editors, *Proceedings of the 25th International Colloquium on Automata, Languages and Programming, ICALP'98*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer-Verlag, 1998.

[200] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logic of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.

[201] Y. Venema. *Many-Dimensional Modal Logic*. PhD thesis, Faculteit Wiskunde en Informatica, Universiteit van Amsterdam, Sept. 1991.

[202] I. A. Vetsikas and B. Selman. A principled study of the design tradeoffs for autonomous trading agents. In *Proceedings of the Second international joint conference on Autonomous Agents and Multi-Agent Systems (AAMAS'06)*, pages 473–480, 2003.

[203] D. Walther. ATEL with common and distributed knowledge is exptime-complete. In *Proceedings of the Fourth Workshop on Methods for Modalities (M4M-4)*, Humbolt University, Berlin, 2005.

[204] D. Walther, C. Lutz, F. Wolter, and M. Wooldridge. ATL satisfiability is indeed EXPTIME-complete. *Journal of Logic and Computation*, 2005. Accepted for publication.

[205] T. Wilke. CTL^+ is exponentially more succinct than CTL. In C. Pandu, Rangan, V. Raman, and R. Ramanujam, editors, *Proceedings of the 19th Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'99*, volume 1738 of *Lecture Notes in Computer Science*, pages 110–121. Springer Verlag, 1999.

[206] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–99, 1983.

[207] F. Wolter. The structure of lattices of subframe logics. *Annals of Pure and Applied Logic*, 86(1):47–100, 1997.

[208] G. H. v. Wright. *An Essay in Modal Logic*. North-Holland, Amsterdam, 1951.

A Eingereichte Arbeiten

[16] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 364–369. Morgan-Kaufmann Publishers, 2005.

[33] P. A. Bonatti, A. Murano, C. Lutz, and M. Vardi. Complexity of enriched μ-calculi. Submitted to the *33rd International Colloquium on Automata, Languages and Programming (ICALP'06)*.

[85] S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? A case for conservative extensions in description logics. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, 2006.

[127] O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. E-connections of abstract description systems. *Artificial Intelligence*, 156(1):1–73, 2004.

[131] M. Lange and C. Lutz. 2-ExpTime lower bounds for propositional dynamic logics with intersection. *Journal of Symbolic Logic*, 70(5):1072–1086, 2005.

[136] C. Lutz. PDL with intersection and converse is decidable. In C.-H. L. Ong, editor, *Computer Science Logic (CSL'05)*, number 3634 in Lecture Notes in Computer Science, pages 413–427. Springer-Verlag, 2005. Extended version to be submitted to the *Journal of Symbolic Logic*.

[137] C. Lutz. Complexity and succinctness of public announcement logic. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'06)*, 2006.

[139] C. Lutz and U. Sattler. The complexity of reasoning with Boolean modal logics. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, editors, *Advances in Modal Logics Volume 3*. CSLI Publications, Stanford, CA, USA, 2001.

[141] C. Lutz, U. Sattler, and L. Tendera. The complexity of finite model reasoning in description logics. *Information and Computation*, 199:132–171, 2005.

[142] C. Lutz, U. Sattler, and F. Wolter. Modal logic and the two-variable fragment. In L. Fribourg, editor, *Computer Science Logic (CSL'01)*, number 2142 in Lecture Notes in Computer Science, pages 247–261. Springer-Verlag, 2001.

[143] C. Lutz, D. Walther, and F. Wolter. Quantitative temporal logics: PSpace and below. In *Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning (TIME'05)*, pages 138–146. IEEE Computer Society Press. Extended version submitted to *Information and Computation*.

[144] C. Lutz and F. Wolter. Modal logics of topological relations. In R. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *Advances in Modal Logic (AiML'04)*, pages 249–263, 2004. Extended version submitted to *Logical Methods in Computer Science*.

[204] D. Walther, C. Lutz, F. Wolter, and M. Wooldridge. ATL satisfiability is indeed ExpTime-complete. Accepted for publication in the *Journal of Logic and Computation*.