

Milenko Mosurović, Tatjana Stojanović
and Ana Kaplarević-Mališić

REASONING IN BASIC DESCRIPTION LOGICS AND DESCRIPTION LOGICS WITH MODAL OPERATORS

Abstract. Description logics are a family of knowledge representation languages constructed for a wide area of application domains. They are based on the notion of concepts and roles, and are mainly characterized by constructors that allow complex concepts and roles to be built from atomic ones. Constructor selection depends on the application domain the representation formalism is designed for. This paper concerns a special type of DLs extended with temporal operators. After an overview of basic description logics, original results related to the temporal extensions of DLs, precisely $DLR_{\Delta, S}$, which has been modeled with an aim to overcome problems of reasoning over conceptual schemas and queries in temporal databases are given.

Mathematics Subject Classification (2000): 03B70, 68T30, 68T27, 68T35, 03B44, 03B45, 03B35, 03B60, 03B80, 03F20

1. Introduction	114
2. Basic description logics	116
2.1. Definition of the basic formalism	117
2.2. Inferences	124
2.3. Reasoning algorithms	128
2.4. Language extensions	135
3. Description logics with modal operators	140
3.1. Preliminaries	140
3.2. The Temporal Description Logic	142
3.3. Temporal queries	144
3.4. Conceptual Schema and Query Examples	145
3.5. Decidability and complexity	147
4. Conclusion	154
References	155

1. Introduction

Description logics (DL) are a family of knowledge representation languages which can be used to represent the terminological knowledge of an application domain in a structured and formal way [Siu04]. Besides representation enabling, they have the task to provide tools for reasoning about the knowledge described by them. They lie on the tracks of the research in the field of knowledge representation.

DL were established with a motivation of providing a formal foundation on network-based knowledge representation systems. In the 1970's research in the field of knowledge representation was very intensive. It gave a wide spectrum of ideas and solutions which were more or less usable or GENERAL. Roughly speaking, there were two types of knowledge representation approaches [Baa et al. 02]: logic-based formalisms as more general and formal and non-logic-based representations, as specialized and, often, ad hoc approaches.

Among these specialized non-logic-based representations there were semantic networks and frames, broadly used in practice. Although they were significantly different, they could both be regarded as network structures, where the structure of the network aims at representing domain knowledge as a set of individuals and their relationships [Baa et al. 02]. Hence, they were often referred to as network-based structures (see [Leh92]). Owing to their more human-centered origins, the network-based systems were often considered as more usable in practice than the logical systems. On the other hand, their less precise semantic characterization

[Baa et al. 02], i.e., concepts as general classes and individuals as instances of concepts were mixed in one vocabulary, which resulted in the absence of general reasoning functionalities.

Attempt on making the system better was representing basic elements of network-based systems by relying on the first-order logic. It turned out that some constraints were not describable. Moreover, in many cases first-order theorem provers were a too big machinery. However, using only fragments of the first-order logic, depending on features of representation language, was good enough. These conclusions were made in big part owing to development of KL-ONE system [BraSch85], the first realized system of the so-called "structured inheritance networks" [Bra77, Bra78, SchSm09]. KL-ONE family of languages are considered as DL ancestors.

The following three ideas, induced by work on KL-ONE systems, have largely shaped the subsequent development of DLs [Baa et al. 02]:

- The basic syntactic building blocks are atomic concepts (unary predicates), atomic roles (binary predicates), and individuals (constants).
- The expressive power of the language is restricted in using a rather small set of (epistemologically adequate) constructors for building complex concepts and roles.
- Implicit knowledge about concepts and individuals can be inferred automatically with help of inference procedures. In particular, subsumption relationships between concepts and instance relationships between individuals and concepts play an important role: unlike IS-A links in Semantic Networks, which are explicitly introduced by the user, subsumption relationships and instance relationships are inferred from the definition of the concepts and the properties of the individuals.

Having above in mind it is clear why the research in the area of Description Logics began under the label of terminological systems. "Later, the emphasis was on the set of concept-forming constructs admitted in the language, giving rise to the name concept languages. In more recent years, after attention was further moved towards the properties of the underlying logical systems, the term Description Logics became popular" [Baa et al. 02].

Major characteristics of description logics are:

- emphasis on reasoning;
- formal logic-based semantics;
- inference patterns;
- subsumption relations between concepts of a terminology;
- hierarchy of concepts derived from subsumption relations.

Reasoning procedures in DL must be decidable and their complexity depends on expressiveness.

All improvements that were brought by DL were the consequences of the fact that they were, in most cases, developed with formal background and with a concrete area of application in mind. Today, there are various implemented systems based on Description Logics which are used in various application domains. Depending on domains and system requirements necessary description formalisms differ

by expressive power and, consequently, by formal and computational properties of reasoning. With the same motivation different extensions of DL were investigated, too. Although semantics of extensions could be interesting for studying, most of the researches associated with language extensions are focused on finding reasoning procedures for the extended languages. Within these, extensions depending on application domain constructs for non-monotonic, epistemic, and temporal reasoning, and constructs for representing belief and uncertain and vague knowledge could be interesting.

The conventional description logics were designed to represent knowledge only about static application domains. To capture various dynamic features, for instance, intensional knowledge (in multi-agent systems), dependence on time or actions (in distributed systems), description logics are combined with suitable "modal" (propositional) logics, say epistemic, temporal, or dynamic. Again, there is a variety of possible combinations (see e.g. [Sch93, Lanx94, Baalau93, BaaOhl93]). Some of them are rather simple and do not increase substantially the complexity of the combined logics (for example, the temporal description logic of Schild [Sch93] is ExpTime-complete); others are too expressive and undecidable (e.g. the multi-dimensional description logic of Baader and Ohlbach [BaaOhl93]).

An optimal compromise between the expressive power and decidability was found in the series of papers [WolZak198, WolZak199c, WolZak199, MosZak199], where various expressive and yet decidable description logics with epistemic, temporal, and dynamic operators were constructed.

This paper gives an overview of basic description logics as well as original results, which concern the temporal extensions of Description Logics. The paper is organized as follows. Section 2 is based on [Baa et al. 02] and gives an introduction to description logics as a formal language for representing knowledge and reasoning based on that knowledge. It gives bases of syntax and semantics, and the typical reasoning tasks are described. At the end of the section some extensions of basic language are given. Section 3 mainly refers to modal extensions of description logics with emphasis on temporal extensions of description logics, precisely *DL_{TR}* as temporal extension of non-temporal description logic *DL_R*. It also gives an example of how the presented logics can be applied in temporal databases.

2. Basic description logics

Description logic (DL) is a common name¹ for a family of knowledge representation formalisms applied on a domain (the "world") by defining relevant domain terminology. They are based on a common family of languages, called description languages, which provide a set of constructors to build class (concept) and properties (role) descriptions. Such description can be used in axioms and assertions of DL knowledge bases and can be reasoned about with respect to DL knowledge bases by DL systems.

¹Previously used names where *terminological knowledge representation languages*, *concept languages*, *term subsumption languages*, *K_L-O_{YB}-based knowledge representation languages*

2.1. Definition of the basic formalism. Knowledge base generated from a knowledge representation system based on DL has two components: TBox and ABox. TBox introduces terminology, i.e., vocabulary of an application domain, while ABox gives assertions about named individuals of concepts from introduced terminology.

The terminology consists of concepts and roles. Concepts denote sets of individuals, while roles denote binary relations between individuals. Complex descriptions of concepts and roles can be built by users in all DL systems. Description language for building these descriptions has model-theoretic semantics. Statements in the TBox and the ABox can be translated into first-order logic or an extension of it.

DL system also offers to reason about terminologies, individuals and assertions. Typical tasks for reasoning on a TBox level are

- determining satisfiability of terminology and
 - subsumption relations of concepts.
- Important reasoning problems on a ABox level are:
- determining consistency of sets of assertions (i.e., if ABox has a model) and
 - whether a set of assertions entails that an individual is an instance of a given concept.

These checks can help to determine whether a knowledge base is meaningful or to organize concepts into a hierarchy according to their generality.

Knowledge representation (KR) system is integrated into a wider environment of an application. Other components interact with KR system by querying and modifying the knowledge base by adding and retracting concepts, roles and assertions. Rules present unlimited mechanism for adding assertions. They represent an extension of a logic core of formalism that can be logically interpreted.

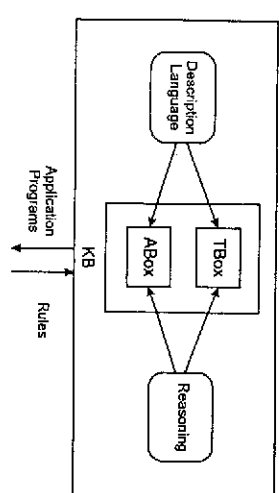


FIGURE 1. Architecture of KR system based on DL.

2.1.1. The basic description language *AL*. Elementary descriptions are atomic concepts and atomic roles. Complex descriptions can be built from them inductively with concept constructors. In abstract notation, we use the letter *A* for

atomic concepts, the letter R for atomic roles, and the letters C and D for concept descriptions. We shall discuss various languages from the family of \mathcal{AL} -languages².

Concept descriptions in \mathcal{AL} are formed according to the following syntax rule:

$$\begin{aligned} C, D &\rightarrow A \mid (\text{atomic concept}) \\ &\quad \top \mid (\text{universal concept}) \\ &\quad \perp \mid (\text{bottom concept}) \\ &\quad \neg A \mid (\text{atomic negation}) \\ &\quad C \sqcap D \mid (\text{intersection}) \\ &\quad \forall R.C \mid (\text{value restriction}) \\ &\quad \exists R.\top \mid (\text{limited existential quantification}) \end{aligned}$$

In \mathcal{AL} only the \top is allowed in the scope in an existential quantification over a role and negation can only be applied to atomic concept. The sublanguage of \mathcal{AL} obtained by disallowing atomic negation is \mathcal{FL}^- . The sublanguage of \mathcal{FL}^- obtained by disallowing limited existential quantification is \mathcal{FL}_0 .

Example 1. Let Person and Female be atomic concepts. Then $\text{Person} \sqcap \text{Female}$ and $\text{Person} \sqcap \neg \text{Female}$ are \mathcal{AL} -concepts describing those persons that are female and those that are not. If hasChild is an atomic role, then the concept $\text{Person} \sqcap \text{hasChild}.\top$ denotes those persons that have a child, and the concept $\text{Person} \sqcap \text{hasChild}.\text{Female}$ denotes those persons all of whose children are female. Using the bottom concept we describe persons without a child by the concept $\text{Person} \sqcap \neg \text{hasChild}.\top$.

In order to define a formal semantics of \mathcal{AL} -concepts, we introduce interpretations \mathcal{I} that consist of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function, which assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to concept descriptions by the following inductive definitions:

$$\begin{aligned} \top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\ \perp^{\mathcal{I}} &= \emptyset \\ (\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid (\forall b)(a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\ (\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid (\exists b)(a, b) \in R^{\mathcal{I}}\} \end{aligned}$$

Two concepts C and D are equivalent ($C \equiv D$) if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretation \mathcal{I} . For example, it is easy to verify that concepts $\text{hasChild}.\text{Female} \sqcap \text{hasChild}.\text{Student}$ and $\text{hasChild}.\text{Female} \sqcap \text{Student}$ are equivalent.

2.1.2. The family of \mathcal{AL} -languages. More expressive languages are obtained by adding further constructors to \mathcal{AL} .

²The language $\mathcal{AL}_{\text{S}} = \{\text{attributive language}\}$ has been introduced in [SchSm91] as a minimal language that is of practical interest.

The union of concepts (indicated by the letter U) is written as $C \sqcup D$, and interpreted as $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$.

Full existential quantification (indicated by the letter \mathcal{E}) is written by $\exists R.C$, and interpreted as $(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid (\exists b)(a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$. Note that $\exists R.C$ differs from $\exists R.\top$ in that arbitrary concepts are allowed to occur in the scope of the existential quantifier.

Number restrictions (indicated by the letter N) are written as $\geq nR$ (at-least restriction) and as $\leq nR$ (at-most restriction), where n represents a nonnegative integer. They are interpreted as

$$\begin{aligned} (\geq nR)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a, b) \in R^{\mathcal{I}}\}| \geq n\}, \\ (\leq nR)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a, b) \in R^{\mathcal{I}}\}| \leq n\} \end{aligned}$$

respectively, where " $|\cdot|$ " denotes the cardinality of a set.

The negation of arbitrary concepts (indicated by the letter C , for "complement") is written by $\neg C$, and interpreted as $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$.

With the additional constructors concept:

$$\text{Person} \sqcap (\leq 1\text{hasChild} \sqcup (\geq 3\text{hasChild} \sqcap \exists\text{hasChild.Female}))$$

describes those persons that have either not more than one child or at least three children, one of which is female.

By extension by \mathcal{AL} any subset of the above constructors generates a particular \mathcal{AL} -language. Each \mathcal{AL} -language is named by a string of the form $\mathcal{AL}[\mathcal{U}][\mathcal{E}][N][C]$, where each letter represents the corresponding constructor.

From the semantic point of view, not all of these languages are distinct. The semantics enforces the equivalences $(C \sqcup D) \equiv \neg(\neg C \sqcap \neg D)$ and $\exists R.\neg C \equiv \neg\forall R.C$ (union and full existential quantification can be expressed using negation). We assume that union and full existential quantification are available in every language that contains negation and vice versa (\mathcal{ALC} is used instead of \mathcal{ALUE} and \mathcal{ALCN} instead of \mathcal{ALUEV}).

2.1.3. Description languages as fragments of predicate logic. Since an interpretation \mathcal{I} assigns to every atomic concept (role) a unary (binary) relation over $\Delta^{\mathcal{I}}$, we can view atomic concepts (roles) as unary (binary) predicates. Then:

- any concept C can be translated into a predicate logic formula $\phi_C(x)$ with one free variable x such that for every interpretation \mathcal{I} the set of elements of $\Delta^{\mathcal{I}}$ satisfying $\phi_C(x)$ is exactly $C^{\mathcal{I}}$
- an atomic concept A is translated into the formula $A(x)$
- the constructors intersection, union, and negation are translated into logical conjunction, disjunction, and negation, respectively
- if C is already translated into $\phi_C(x)$ and R is an atomic role, then value restriction and existential quantification are captured by the formulae

$$\begin{aligned} \phi_{\forall R.C}(y) &= (\forall x)(R(y, x) \rightarrow \phi_C(x)) \\ \phi_{\exists R.C}(y) &= (\exists x)(R(y, x) \wedge \phi_C(x)) \end{aligned}$$

where y is a new variable

- number restrictions are expressed by the formulae

$$\phi_{\geq n} R(x) = (\exists y_1, \dots, y_n) R(x, y_1) \wedge \dots \wedge R(x, y_n) \wedge \bigwedge_{i < j} y_i \neq y_j$$

$$\phi_{\leq n} R(x) = (\forall y_1, \dots, y_{n+1}) R(x, y_1) \wedge \dots \wedge R(x, y_{n+1}) \rightarrow \bigvee_{i < j} y_i = y_j$$

2.1.4. Terminologies. In the sequel, we will introduce:

- terminological axioms, which make statements about relations between concepts or roles,
- definitions as specific axioms and
- terminologies as sets of definitions by which we introduce atomic concepts as abbreviations or names for complex concepts.

Terminological axioms. In the most general case, terminological axioms have the form

$$C \subseteq D \quad (R \sqsubseteq S) \quad \text{or} \quad C \equiv D \quad (R \equiv S)$$

where C , D are concepts (and R , S are roles). Axioms of the first kind are called inclusions, while axioms of the second kind are called equalities.

An interpretation \mathcal{I} satisfies an inclusion $C \subseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies an equality $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. If \mathcal{T} is a set of axioms, then \mathcal{I} satisfies \mathcal{T} iff \mathcal{I} satisfies each element of \mathcal{T} . If \mathcal{I} satisfies an axiom (resp. a set of axioms), then it is a model of this axiom (resp. set of axioms). Two axioms or two sets of axioms are equivalent if they have the same models.

Definitions. An equality whose left-hand side is an atomic concept is a definition. Definitions are used to introduce symbolic names for complex descriptions.

A set of definitions should be unequivocal. A finite set of definitions \mathcal{T} is a terminology or TBox if no symbolic name is defined more than once, that is, if for every atomic concept A there is at most one axiom in \mathcal{T} whose left-hand side is A .

Example 2. A terminology (TBox) with concepts about family relationships can be introduced as follows:

Woman \equiv Person \sqcap Female
 Man \equiv Person \sqcap \neg Woman
 Mother \equiv Woman \sqcap HasChild, Person
 Father \equiv Man \sqcap HasChild, Person
 Parent \equiv Father \sqcup Mother
 Grandmother \equiv Mother \sqcap HasChild, Parent
 MotherWithManyChildren \equiv Mother $\sqcap \geq 3$ HasChild
 MotherWithoutDaughter \equiv Mother $\sqcap \nexists$ HasChild, \neg Woman
 Wife \equiv Woman \sqcap HasHusband, Man \square

Suppose, that \mathcal{T} is a terminology. We divide the atomic concepts occurring in \mathcal{T} into two sets, the name symbols $N_{\mathcal{T}}$ (defined concepts) that occur on the left-hand side of some axiom and the base symbols $B_{\mathcal{T}}$ (primitive concepts) that occur only

on the right-hand side of axioms. Based on this, terminologies define name symbols using base symbols.

A base interpretation for \mathcal{T} is an interpretation that interprets only the base symbols. Let \mathcal{J} be a base interpretation. An interpretation \mathcal{I} that interprets also the name symbols is an extension of \mathcal{J} if it has the same domain as \mathcal{J} , i.e., $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$, and if it agrees with \mathcal{J} for the base symbols. We say that \mathcal{T} is *definitional* if every base interpretation has exactly one extension that is a model of \mathcal{T} . In other words, if we know what the base symbols stand for, and \mathcal{T} is definitional, then the meaning of the name symbols is completely determined. If a terminology is definitional, then every equivalent terminology is also definitional.

The question whether a terminology is definitional or not is related to the question whether or not its definitions are cyclic.

(1) $\text{Human}' \equiv \text{Animal} \sqcap \text{hasParent, Human}'$

Let A , B be atomic concepts occurring in \mathcal{T} . We say that A directly uses B in \mathcal{T} if B appears on the right-hand side of the definition of A . The transitive closure of the relation "directly uses" is called "uses". Then \mathcal{T} contains a cycle iff there exists an atomic concept in \mathcal{T} that uses itself. Otherwise, \mathcal{T} is called *acyclic*.

If a terminology \mathcal{T} is acyclic, then it is definitional. Definitions in terminology \mathcal{T} can be expanded by replacing each occurrence of a name on the right-hand side of a definition with the concepts that it represents. If \mathcal{T} is a cyclic this process eventually stops giving a terminology \mathcal{T}' containing solely definitions of the form $A \equiv C'$, where C' contains only base symbols and no name symbols. \mathcal{T}' is the expansion of \mathcal{T} . Size of the expansion can be exponential in the size of the original terminology.

Example 3. The expansion of the Family TBox previously introduced is:

Woman \equiv Person \sqcap Female
 Man \equiv Person $\sqcap \neg$ (Person \sqcap Female)
 Mother \equiv (Person \sqcap Female) \sqcap HasChild, Person
 Father \equiv (Person $\sqcap \neg$ (Person \sqcap Female)) \sqcap HasChild, Person
 Parent \equiv ((Person $\sqcap \neg$ (Person \sqcap Female)) \sqcap HasChild, Person)
 \sqcup ((Person \sqcap Female) \sqcap HasChild, Person)
 Grandmother \equiv ((Person \sqcap Female) \sqcap HasChild, Person)
 \sqcap HasChild, ((Person $\sqcap \neg$ (Person \sqcap Female))
 \sqcap HasChild, Person)
 \sqcup ((Person \sqcap Female) \sqcap HasChild, Person)
 MotherWithManyChildren \equiv ((Person \sqcap Female) \sqcap HasChild, Person) $\sqcap \geq 3$ HasChild
 MotherWithoutDaughter \equiv ((Person \sqcap Female) \sqcap HasChild, Person)
 \sqcap HasChild, (\neg (Person \sqcap Female))
 Wife \equiv (Person \sqcap Female)
 \sqcap HasHusband, (Person $\sqcap \neg$ (Person \sqcap Female))

Lemma 1. Let \mathcal{T} be an acyclic terminology and \mathcal{T}' be its expansion. Then

- (1) \mathcal{T} and \mathcal{T}' have the same name and base symbols;
- (2) \mathcal{T} and \mathcal{T}' are equivalent;
- (3) \mathcal{T} and \mathcal{T}' are definitorial.

Proof. Let \mathcal{T}_1 be a terminology. Suppose $A \equiv C$ and $B \equiv D$ are definitions in \mathcal{T}_1 such that B occurs in C . Let C' be the concept obtained from C by replacing each occurrence of B in C with D , and let \mathcal{T}_2 be the terminology obtained from \mathcal{T}_1 by replacing the definition $A \equiv C$ with $A \equiv C'$. Then both terminologies have the same name and base symbols. Moreover, since \mathcal{T}_2 has been obtained from \mathcal{T}_1 by replacing equals by equals, both terminologies have the same models. Since \mathcal{T}' is obtained from \mathcal{T} by a sequence of replacement steps like the ones above, this proves statements (1) and (2).

Suppose now that \mathcal{J} is an interpretation of the base symbols. We extend it to an interpretation \mathcal{I} that covers also the name symbols by setting $A^{\mathcal{I}} = C^{\mathcal{J}}$, if $A \equiv C$ is the definition of A in \mathcal{T}' . Clearly, \mathcal{I} is a model of \mathcal{T}' , and it is the only extension of \mathcal{J} that is a model of \mathcal{T}' . This shows that \mathcal{T}' is definitorial. Moreover, \mathcal{T} is definitorial as well, since it is equivalent to \mathcal{T}' . \square

Of course, there are also terminologies with cycles that are definitorial, but:

Theorem 1. Every definitorial ACC-terminology is equivalent to an acyclic terminology.

The theorem is a reformulation of Both's Definability Theorem [Gab72] for the modal propositional logic K_n .

2.1.5. Terminologies with inclusion axioms. In the case of concepts that cannot be defined completely necessary conditions for concept membership are still stated using an inclusion. Inclusion whose left-hand side is atomic is a specialization.

For example, concept "Women" from TBox in Example 2 can be described in less detail with the specialization

$$(2) \quad \text{Woman} \sqsubseteq \text{Person}$$

If specialization is allowed in a terminology, then the terminology loses its definitorial impact, even if it is acyclic. A set of axioms \mathcal{T} is a generalized terminology if the left-hand side of each axiom is an atomic concept and for every atomic concept there is at most one axiom where it occurs on the left-hand side.

Generalized terminology \mathcal{T} can be transformed into a regular terminology $\tilde{\mathcal{T}}$, containing definitions only, such that $\tilde{\mathcal{T}}$ is equivalent to \mathcal{T} in a sense specified below. $\tilde{\mathcal{T}}$ is obtained from \mathcal{T} by choosing a new base symbol A for every specialization $A \sqsubseteq C$ in \mathcal{T} and by replacing the specialization $A \sqsubseteq C$ with the definition $A \equiv A \sqcap C$. The terminology $\tilde{\mathcal{T}}$ is the normalization of \mathcal{T} .

If a TBox contains the specialization (2), then the normalization contains the definition $\text{Woman} \equiv \text{Woman} \sqcap \text{Person}$. The additional base symbol Woman stands for the qualities that distinguish a woman among persons.

Lemma 2. Let \mathcal{T} be a generalized terminology and $\tilde{\mathcal{T}}$ its normalization. Then

- (1) Every model of $\tilde{\mathcal{T}}$ is a model of \mathcal{T} .
- (2) For every model \mathcal{I} of \mathcal{T} there is a model $\tilde{\mathcal{I}}$ of $\tilde{\mathcal{T}}$ that has the same domain as \mathcal{I} and agrees with \mathcal{I} on the atomic concepts and roles in \mathcal{T} .

Proof. The first statement holds because a model $\tilde{\mathcal{I}}$ of $\tilde{\mathcal{T}}$ satisfies $A^{\tilde{\mathcal{I}}} = (A \sqcap C)^{\tilde{\mathcal{I}}} = A^{\tilde{\mathcal{I}}} \sqcap C^{\tilde{\mathcal{I}}}$, which implies $A^{\tilde{\mathcal{I}}} \subseteq C^{\tilde{\mathcal{I}}}$. Conversely, if \mathcal{I} is a model of \mathcal{T} , then the extension $\tilde{\mathcal{I}}$ of \mathcal{I} , defined by $A^{\tilde{\mathcal{I}}} = A^{\mathcal{I}}$, is a model of $\tilde{\mathcal{T}}$, because $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ implies $A^{\tilde{\mathcal{I}}} = A^{\mathcal{I}} \sqcap C^{\mathcal{I}} = A^{\mathcal{I}} \sqcap C^{\tilde{\mathcal{I}}}$, and therefore $\tilde{\mathcal{I}}$ satisfies $A \equiv A \sqcap C$. \square

In theory, inclusion axioms do not extend the expressivity of terminologies, while, in practice, they are a convenient means to introduce terms into a terminology that cannot be defined completely.

2.1.6. World Descriptions. The second component of a knowledge base, in addition to the terminology or TBox, is the world description or ABox.

Assertions about individuals. In the ABox, individuals are introduced, by giving them names, and properties of these individuals are asserted. We denote individual names by a, b, c . Using concepts C and roles R , one can make assertions of the following two kinds in an ABox: $C(a)$, and $R(b, c)$. The first kind are concept assertions, and they state that a belongs to (the interpretation of) C . The second kind are role assertions, and they state that c is a filler of the role R for b . An ABox, denoted as A , is a finite set of such assertions.

Example 4. If JOHN, PAUL, and MARY are individual names, then Father(JOHN) means that John is a father, and hasChild(MARY, PAUL) means that Paul is a child of Mary. An example of an ABox for TBox from Example 2:

```

MotherWithoutDaughter(MARY)   Father(JOHN)
hasChild(MARY, JOHN)           hasChild(JOHN, HARRY)
hasChild(MARY, PAUL)

```

In a simplified view, an ABox can be seen as an instance of a relational database with only unary and binary relations. Contrary to the "closed-world semantics" of classical databases, the semantics of ABoxes is an "open-world semantics", since normally knowledge representation systems can be applied in situations where it cannot be assumed that the knowledge in the KB is complete. The TBox also imposes semantic relations between the concepts and roles in the ABox that do not have counterparts in database semantics.

ABoxes are given semantics by extending interpretations to individual names. From this point on, an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ not only maps atomic concepts and roles to sets and relations, but also maps each individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. This mapping is constructed with respect to the unique name assumption (UNA), that is, if a, b are distinct names, then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. The interpretation \mathcal{I} satisfies the concept assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and it satisfies the role assertion $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. An interpretation satisfies the ABox A if it satisfies each assertion in A . In this case we say that \mathcal{I} is a model of the assertion or of the

ABox. Finally, \mathcal{I} satisfies an assertion α or an ABox \mathcal{A} with respect to a TBox \mathcal{T} if in addition to being a model of α or of \mathcal{A} , it is a model of \mathcal{T} . Thus, a model of \mathcal{A} and \mathcal{T} is an abstraction of a concrete world where the concepts are interpreted as subsets of the domain as required by the TBox and where the membership of the individuals to concepts and their relationships with one another in terms of roles respect the assertions in the ABox.

Individual names in the description language. Sometimes, it is convenient to allow individual names (also called nominals) not only in the ABox, but in the description language as well. The most basic constructor employing individual is the "set" (or one-of), written by $\{a_1, \dots, a_n\}$, where a_1, \dots, a_n are individual names. As one could expect, such a set concept is interpreted as

$$(3) \quad \{a_1, \dots, a_n\}^{\mathcal{I}} = \{a_1^{\mathcal{I}}, \dots, a_n^{\mathcal{I}}\}$$

With sets in the description language one can, for instance, define the concept of permanent members of the UN security council as $\{\text{CHINA, FRANCE, RUSSIA, UK, US}\}$.

Another constructor involving individual names is the "fills" constructor $R : a$ for a role R . The semantics of this constructor is:

$$(4) \quad (R : a)^{\mathcal{I}} = \{d \in \Delta^{\mathcal{I}} \mid (d, a^{\mathcal{I}}) \in R^{\mathcal{I}}\}$$

that is, $R : a$ stands for the set of those objects that have a as a filler of the role R .

2.2. Inferences. A knowledge representation system can perform specific types of reasoning. Knowledge base, containing TBox and ABox has semantics that makes it equivalent to a set of axioms in first-order predicate logic. Like any other set of axioms, it contains implicit knowledge that can be made explicit through inferences.

Further discussion shows that the main problem with inference is consistency check for ABox, to which all other inferences can be reduced.

2.2.1. Reasoning tasks for concepts. During the modeling of a domain terminology \mathcal{T} is constructed by defining new concepts. It is important to check if new concepts are contradictory or not. A concept is meaningful if there is an interpretation that satisfies the axioms of \mathcal{T} , such that the concept denotes a nonempty set in that interpretation. Such a concept is satisfiable with respect to \mathcal{T} , otherwise it is unsatisfiable.

To check whether a domain model is contradictory or not, or to optimize queries that are formulated as concepts, it might be needed to know whether a concept is more general than another one (the subsumption problem). A concept C is subsumed by a concept D if in every model of \mathcal{T} the set denoted by C is a subset of the set denoted by D . The algorithms that check the subsumption may also be used for organizing concepts of a TBox in a taxonomy according to their generality. Two more relationships between concepts are equivalence and disjointness. These properties are formally defined as follows. Let \mathcal{T} be a TBox.

Satisfiability: A concept C is satisfiable with respect to \mathcal{T} if there exists a model \mathcal{I} of \mathcal{T} such that $C^{\mathcal{I}}$ is nonempty – \mathcal{I} is a model of C .

Subsumption: A concept C is subsumed by a concept D with respect to \mathcal{T} if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model \mathcal{I} of \mathcal{T} – $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$.

Equivalence: Two concepts C and D are equivalent with respect to \mathcal{T} if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model \mathcal{I} of \mathcal{T} – $C \equiv_{\mathcal{T}} D$ or $\mathcal{T} \models C \equiv D$.

Disjointness: Two concepts C and D are disjoint with respect to \mathcal{T} if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ for every model \mathcal{I} of \mathcal{T} .

If the TBox is empty, we simply write $\models C \sqsubseteq D$ if C is subsumed by D , and $\models C \equiv D$ if C and D are equivalent.

Lemma 3 (Reduction to Subsumption). *For concepts C , D we have*

- (1) C is unsatisfiable $\Leftrightarrow C$ is subsumed by \perp ;
- (2) C and D are equivalent $\Leftrightarrow C$ is subsumed by D and D is subsumed by C ;
- (3) C and D are disjoint $\Leftrightarrow C \sqcap D$ is subsumed by \perp .

The statements also hold with respect to a TBox.

Most DL systems that can check subsumption can perform all four inferences defined above, because almost all of description languages implemented in actual DL systems contain an unsatisfiable concept and all of them include the intersection operator " \sqcap ".

Subsumption, equivalence, and disjointness of concepts can be reduced to the satisfiability problem if in addition to intersection, a system allows forming of the negation of a description [Smol88].

Lemma 4 (Reduction to Unsatisfiability). *For concepts C , D we have*

- (1) C is subsumed by $D \Leftrightarrow C \sqcap \neg D$ is unsatisfiable;
- (2) C and D are equivalent \Leftrightarrow both $(C \sqcap \neg D)$ and $(\neg C \sqcap D)$ are unsatisfiable;
- (3) C and D are disjoint $\Leftrightarrow C \sqcap D$ is unsatisfiable.

The statements also hold with respect to a TBox.

Since, for sets M , N we have $M \subseteq N$ iff $M \setminus N = \emptyset$, then the reduction of subsumption becomes apparent and easy to understand. The reduction of equivalence is correct because C and D are equivalent, if and only if C is subsumed by D and D is subsumed by C . Finally, the reduction of disjointness is just a rephrasing of the definition.

In an AC-language without full negation, subsumption and equivalence cannot be reduced to unsatisfiability in the way shown in Lemma 4. The complexity of such inferences is somewhat different.

As seen in Lemma 3, from the viewpoint of worst-case complexity, subsumption is the most general inference for any AC-language. Lemma 5 shows that unsatisfiability is a special case of each of the other problems. Lemma 3 and 5 show that, in order to obtain complexity bounds for inferences on concepts in AC-languages (more precisely, for the complexity of the unsatisfiability, the equivalence, and the disjointness problem), it suffices to assess lower bounds for unsatisfiability and upper bounds for subsumption.

Lemma 5 (Reducing Unsatisfiability). *Let C be a concept. Then the following statements are equivalent.*

- (1) C is unsatisfiable;
- (3) C and \perp are equivalent;

(2) C is subsumed by A ; (4) C and T are disjoint.

The statements also hold with respect to a TBox.

2.2.2. Eliminating the TBox. This section shows that, if T is an acyclic TBox, it is always possible to reduce reasoning problems with respect to T to problems with respect to the empty TBox. As seen in Lemma 1, T is equivalent to its expansion T' . Recall that in the expansion every definition is of the form $A \equiv D$ such that D contains only base symbols, but no name symbols. For each concept C we define the expansion of C with respect to T as the concept C' that is obtained from C by replacing each occurrence of a name symbol A in C by the concept D , where $A \equiv D$ is the definition of A in T' ; the expansion of T .

Since the expansion C' is derived from C by replacing names with descriptions in such a way that both are interpreted in the same way in any model of T' , it follows that

- $C \equiv T C'$.

Thus, C is satisfiable with respect to T iff C' is satisfiable with respect to T' . However, C' contains no defined names, and thus C' is satisfiable with respect to T' iff it is satisfiable. This yields that

- C is satisfiable with respect to T iff C' is satisfiable.

If D is another concept, then $D \equiv T D'$, and this yields that $C \sqsubseteq T D$ iff $C' \sqsubseteq T D'$ and $C \equiv T D$ iff $C' \equiv T D'$. Since C' and D' contain only base symbols, this implies

- $T \models C \sqsubseteq D$ iff $T \models C' \sqsubseteq D'$
- $T \models C \equiv D$ iff $T \models C' \equiv D'$.

With similar arguments we can show that

- C and D are disjoint with respect to T iff C' and D' are disjoint.

Expanding concepts with respect to an acyclic TBox allows removing the TBox from reasoning problems.

Expanding concepts may substantially increase computational complexity, since in the worst case the size of T' is exponential in the size of T . A complexity analysis of the difficulty of reasoning with respect to TBoxes shows that the expansion of definitions is a source of complexity that cannot always be avoided.

2.2.3. Reasoning tasks for ABoxes. After designed a terminology and using the reasoning services of DL system to check that all concepts are satisfiable and that the expected subsumption relations hold, the ABox can be filled with assertions about individuals. An ABox contains two types of assertions: concept assertions of the form $C(a)$ and role assertions of the form $R(a, b)$. It is understandable that the representation of such knowledge has to be consistent.

An ABox A is consistent with respect to a TBox T , if there is an interpretation that is a model of both A and T . It is simply said that A is consistent if it is consistent with respect to the empty TBox.

For example, the set of assertions $\{\text{Mother}(\text{MERY}), \text{Father}(\text{MERY})\}$ is consistent with respect to the empty TBox, however, the assertions are not consistent with respect to the Family TBox.

Similarly as for concepts, checking the consistency of an ABox with respect to an acyclic TBox can be reduced to checking an expanded ABox. The expansion of A with respect to T is defined as the ABox A' that is obtained from A by replacing each concept assertion $C(a)$ in A with the assertion $C'(a)$, where C' is the expansion of C with respect to T . In every model of T , a concept C and its expansion C' are interpreted in the same way. Therefore, A' is consistent with respect to T iff A is consistent with respect to T . However, since A' does not contain a name symbol defined in T it is consistent with respect to T iff it is consistent. The conclusion is:

- A is consistent with respect to T iff its expansion A' is consistent.

Other inferences that are going to be introduced can also be defined with respect to a TBox or for an ABox alone. As in the case of consistency, reasoning tasks for ABoxes with respect to acyclic TBoxes can be reduced to reasoning on expanded ABoxes.

Over an ABox A , queries can be posed about the relationships between concepts, roles and individuals. The prototypical ABox inference on which such queries are based is the instance check, or the check whether an assertion is entailed by an ABox. An assertion α is entailed by A and we write $A \models \alpha$ if every interpretation that satisfies A , that is, every model of A , also satisfies α . If α is a role assertion, the instance check is easy, since description language does not contain constructors to form complex roles. If α is of the form $C(a)$, the instance check can be reduced to the consistency problem for ABoxes because there is the following connection:

- $A \models C(a)$ iff $A \cup \{ \neg C(a) \}$ is inconsistent.

Reasoning about concepts can also be reduced to consistency checking. We have seen in Lemma 4 that the important reasoning problems for concepts can be reduced to the one to decide whether a concept is satisfiable or not. Similarly, concept satisfiability can be reduced to ABox consistency because for every concept C it holds:

- C is satisfiable iff $\{C(a)\}$ is consistent,

where a is an arbitrarily chosen individual name.

Conversely, in [Sch94] it has been shown that ABox consistency can be reduced to the concept satisfiability in languages with the "set" and the "fills" constructors.

If knowledge bases are considered as means to store information about individuals, it may be needed to know all individuals that are instances of a given concept description C , that is, the description language is used to formulate queries. Given an ABox A and a concept C , the retrieval problem is to find all individuals a such that $A \models C(a)$. A non-optimized algorithm for a retrieval query can be realized by testing whether each individual occurring in the ABox is an instance of the query concept C .

The dual inference to retrieval is the realization problem: given an individual a and a set of concepts, find the most specific concepts C from the set such that $A \models C(a)$. Here, the most specific concepts are those that are minimal with respect to the subsumption ordering \sqsubseteq .

2.3. Reasoning algorithms. As it was shown in the previous section, if conjunction and negation are allowed in certain DL, then all relevant inference problems can be reduced to consistency problem for ABoxes. If negation is not allowed, then subsumption of concepts can be computed by so-called structural subsumption algorithms, i.e., algorithms that compare the syntactic structure of (possibly normalized) concept descriptions. Such algorithms are, usually, very efficient, but they are only complete for rather simple languages with little expressivity. In particular, DLs with (full) negation and disjunction cannot be handled by structural subsumption algorithms. For such languages, tableau-based algorithms are often used.

Designing new algorithms for reasoning in DLs can be unnecessary in many cases. Trying to reduce the problem to a known inference problem in logics is a good way. For example, decidability of the inference problems for \mathcal{ALC} and many other DLs can be obtained as a consequence of the known decidability result for the two variable fragment of the first-order predicate logic. The language \mathcal{L}^2 consists of all formulae of the first-order predicate logic that can be built with the help of predicate symbols (including equality) and constant symbols (but without function symbols) using only the variables x, y [Mor73]. By appropriately renaming variable names, any concept description of the language \mathcal{ALC} can be translated into an \mathcal{L}^2 -formula with one free variable [Bot96]. This connection between \mathcal{ALC} and \mathcal{L}^2 shows that any extension of \mathcal{ALC} by constructors that can be expressed with the help of only two variables yields a decidable DL. Number restrictions and composition of roles are examples of constructors that cannot be expressed within \mathcal{L}^2 , but number restrictions can be expressed in \mathcal{L}^2 , the extension of \mathcal{L}^2 by counting quantifiers, which has recently been shown to be decidable [Grä et al. 97, Pac97]. However, the complexity of the decision procedures obtained in this way is usually higher than necessary: for example, the satisfiability problem for \mathcal{L}^2 is NEXPTIME-complete, whereas satisfiability of \mathcal{ALC} -concept descriptions is "only" PSPACE-complete.

Lower complexity decision procedures can be obtained by using the connection between DLs and propositional modal logics. \mathcal{ALC} is a syntactic variant of the propositional multi-modal logic \mathbf{K} [Sch91] and the extension of \mathcal{ALC} by transitive closure of roles corresponds to Propositional Dynamic Logic (PDL) [Ba91]. Some of the algorithms used in propositional modal logics for deciding satisfiability are very similar to the tableau-based algorithms newly developed for DLs. Instead of using tableau-based algorithms, decidability of certain propositional modal logics (and thus of the corresponding DLs), can also be shown by establishing the finite model property [Fit93] of the logic (i.e., showing that a formula/concept is satisfiable iff it is satisfiable in a finite interpretation) or by using tree automata [VarWol86].

2.3.1. Structural subsumption algorithms. These algorithms usually proceed in two phases. First, the descriptions to be tested for subsumption are normalized, and then the syntactic structure of the normal forms is compared. Ideas underlying this approach will be shown for the language \mathcal{FL}_0 , which allows for conjunction ($C \sqcap D$) and value restrictions ($\forall R.C$). Then the bottom concept (\perp), atomic negation ($\neg A$) and number restrictions ($\leq nR$ and $\geq nR$) handling will be presented.

An \mathcal{FL}_0 -concept description is in a normal form iff it is of the form

$$A_1 \sqcap \dots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \dots \sqcap \forall R_n.C_n$$

where A_1, \dots, A_m are distinct concept names, R_1, \dots, R_n are distinct role names, and C_1, \dots, C_n are \mathcal{FL}_0 -concept descriptions in normal form. Using associativity, commutativity and idempotence of \sqcap , and the fact that the descriptions $\forall R.(C \sqcap D)$ and $(\forall R.C) \sqcap (\forall R.D)$ are equivalent, it is easy to see that any description can be transformed into an equivalent one in the normal form.

Lemma 6. Let $A_1 \sqcap \dots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \dots \sqcap \forall R_n.C_n$ be the normal form of the \mathcal{FL}_0 -concept description C , and $B_1 \sqcap \dots \sqcap B_k \sqcap \forall S_1.D_1 \sqcap \dots \sqcap \forall S_l.D_l$ the normal form of the \mathcal{FL}_0 -concept description D . Then $C \sqsubseteq D$ iff the following two conditions hold:

- (1) for all i , $1 \leq i \leq k$ there exists j , $1 \leq j \leq m$ such that $R_i = A_j$
- (2) for all i , $1 \leq i \leq l$ there exists j , $1 \leq j \leq n$ such that $S_i = R_j$ and $C_j \sqsubseteq D_i$

Having this lemma in mind, it is easy to construct recursive algorithm for computing subsumption. That algorithm has a polynomial time complexity [LevBra87].

If \mathcal{FL}_0 is extended by language constructors that can express unsatisfiable concepts, then the definition of the normal form must be changed. On the other hand, the structural comparison of the normal forms must take into account that an unsatisfiable concept is subsumed by every concept. The simplest DL where this occurs is \mathcal{FL}_\perp , the extension of \mathcal{FL}_0 by the bottom concept \perp .

An \mathcal{FL}_\perp -concept description is of the normal form iff it is \perp or of the form

$$A_1 \sqcap \dots \sqcap A_m \sqcap \forall R_1.C_1 \sqcap \dots \sqcap \forall R_n.C_n$$

where A_1, \dots, A_m are distinct concept names different from \perp , R_1, \dots, R_n are distinct role names, and C_1, \dots, C_n and \mathcal{FL}_\perp -concept descriptions in the normal form. Such a normal form can easily be computed. In principle, one just computes the \mathcal{FL}_0 -normal form of the description (where \perp is treated as an ordinary concept name): $B_1 \sqcap \dots \sqcap B_k \sqcap \forall R_1.D_1 \sqcap \dots \sqcap \forall R_n.D_n$. If one of the D_i s is \perp then replace the whole description by \perp . Otherwise, apply the same procedure recursively to the D_i s. The structural subsumption algorithm for \mathcal{FL}_\perp works just like the one for \mathcal{FL}_0 with the only difference that \perp is subsumed by any description.

Extension of \mathcal{FL}_\perp by atomic negation can be treated similarly. During the computation of the normal form, negated concept names are treated like concept names. If a name and its negation occur on the same level of the normal form, then \perp is added, which can then be treated as described above. The structural comparison of the normal forms treats negated concept names just like concept names.

Finally, if we consider the language \mathcal{ALN} , the additional presence of number restrictions leads to a new type of conflict. On one hand, as in the case of atomic negation, number restrictions may be conflicting with each other (e.g., $\geq 2R$ and $\leq 1R$). On the other hand, at least restrictions $\geq nR$ for $n \geq 1$ are in conflict with value restrictions $\forall R.\perp$. When computing the normal form, number restrictions can be treated like concept names. The next step is taking care of the new types of

conflicts by introducing \perp and using it for normalization as described above. During the structural comparison of normal forms, inherent subsumption relationships between number restrictions (e.g. $\geq nR \sqsubseteq \geq mR$ iff $n \geq m$) must also be taken into account [BorPai94].

Structural subsumption algorithms, like described above, usually fail to be complete for larger DLs. In particular, they cannot treat disjunction, full negation, and full existential restriction $\exists R.C$. These constructors can be more efficiently treated in subsumption algorithms that are constructed in a tableau-based style.

2.3.2. Tableau algorithms. Tableau algorithms use another idea to examine subsumption of concept descriptions. Precisely, as it was shown in Subsection 2.2, they use negation to reduce subsumption to (un)satisfiability of concept descriptions using: $C \sqsubseteq D$ iff $C \sqcap \neg D$ is unsatisfiable.

Before describing a tableau-based satisfiability algorithm for \mathcal{ALCN} in more detail, we illustrate the underlying ideas using a few basic rules:

- For any existential restriction the algorithm introduces a new individual as role filler, and this individual must satisfy the constraints expressed by the restriction.
- The algorithm uses value restrictions in interaction with already defined role relations to impose new constraints on individuals.
- For disjunctive constraints, the algorithm tries both possibilities in successive attempts. It must backtrack if it reaches an obvious contradiction, i.e., if the same individual must satisfy constraints that are obviously conflicting.
- If an at-most number restriction is violated, then the algorithm must identify different role fillers.

2.3.3. A tableau-based satisfiability algorithm for \mathcal{ALCN} . Describing the algorithm needs introducing an appropriate data structure which will be used for representing constraints like “ a belongs to (the interpretation of) C ” and “ b is an R -filler of a ”. Although many papers on tableau algorithms for DLs introduce the new notion of a constraint system for this purpose, considering the types of constraints that must be expressed, ABox assertions can be used for their representations. Since the presence of at-most number restrictions may lead to the identification of different individual names, the unique name assumption (UNA) will not be imposed on the ABoxes considered by the algorithm. Instead, explicit inequality assertions of the form $x \neq y$ for individual names x, y , with the obvious semantics that an interpretation I satisfies $x \neq y$ iff $x^I \neq y^I$ will be allowed. These assertions are assumed to be symmetric, i.e., saying that $x \neq y$ belongs to an ABox \mathcal{A} is the same as saying that $y \neq x$ belongs to \mathcal{A} .

Let C_0 be an \mathcal{ALCN} -concept. In order to test satisfiability of C_0 , the algorithm starts with the ABox $\mathcal{A}_0 = \{C_0(x_0)\}$, and applies consistency preserving transformation rules (see Figure 2) to \mathcal{A}_0 until no more rules apply. If the “complete” ABox obtained in this way does not contain an obvious contradiction (called clash), then \mathcal{A}_0 is consistent (and thus C_0 is satisfiable), and inconsistent

(unsatisfiable) otherwise. The transformation rules that handle disjunction and at-most restrictions are non-deterministic in the sense that a given ABox is transformed into finitely many new ABoxes such that the original ABox is consistent iff one of the new ABoxes is so. For this reason instead of single ABoxes finite sets of ABoxes $\mathcal{S} = \{\mathcal{A}_1, \dots, \mathcal{A}_k\}$ are considered. Such a set is consistent iff there is some i , $1 \leq i \leq k$ such that \mathcal{A}_i is consistent. A rule in Figure 2 is applied to a given finite set of ABoxes \mathcal{S} as follows: it takes an element \mathcal{A} of \mathcal{S} and replaces it by one ABox \mathcal{A}' , by two ABoxes \mathcal{A}' and \mathcal{A}'' , or by finitely many ABoxes $\mathcal{A}_{i,j}$.

The $\neg\cap$-rule	
Condition: \mathcal{A} contains $(C_1 \sqcap C_2)(x)$, but it does not contain both $C_1(x)$ and $C_2(x)$	
Action: $\mathcal{A}' = \mathcal{A} \cup \{C_1(x), C_2(x)\}$.	
The $\neg\cup$-rule	
Condition: \mathcal{A} contains $(C_1 \sqcup C_2)(x)$, but neither $C_1(x)$ nor $C_2(x)$	
Action: $\mathcal{A}' = \mathcal{A} \cup \{C_1(x)\}$, $\mathcal{A}'' = \mathcal{A} \cup \{C_2(x)\}$.	
The $\rightarrow\exists$-rule	
Condition: \mathcal{A} contains $(\exists R.C)(x)$, but there is no individual name z such that $C(z)$ and $R(x, z)$ are in \mathcal{A}	
Action: $\mathcal{A}' = \mathcal{A} \cup \{C(y), R(x, y)\}$ where y is an individual name not occurring in \mathcal{A} .	
The $\rightarrow\forall$-rule	
Condition: \mathcal{A} contains $(\forall R.C)(x)$ and $R(x, y)$, but it does not contain $C(y)$	
Action: $\mathcal{A}' = \mathcal{A} \cup \{C(y)\}$.	
The $\rightarrow\geq$-rule	
Condition: \mathcal{A} contains $(\geq nR)(x)$ and there are no individual names z_1, \dots, z_n such that $R(x, z_i)$ ($1 \leq i \leq n$) and $z_i \neq z_j$ ($1 \leq i < j \leq n$) are contained in \mathcal{A}	
Action: $\mathcal{A}' = \mathcal{A} \cup \{R(x, y_i) \mid 1 \leq i \leq n\} \cup \{y_i \neq y_j \mid 1 \leq i < j \leq n\}$ where y_1, \dots, y_n are distinct individual names not occurring in \mathcal{A} .	
The $\rightarrow\leq$-rule	
Condition: \mathcal{A} contains distinct individual names y_1, \dots, y_{n+1} such that $(\leq nR)(x)$ and $R(x, y_1), \dots, R(x, y_{n+1})$ are in \mathcal{A} and $y_i \neq y_j$ is not in \mathcal{A} for some $i \neq j$	
Action: For each pair y_i, y_j such that $i > j$ and $y_i \neq y_j$ is not in \mathcal{A} , the ABox $\mathcal{A}_{i,j} = [y_i/y_j]\mathcal{A}$ is obtained from \mathcal{A} by replacing each occurrence of y_i by y_j .	

FIGURE 2. Transformation rules of the satisfiability algorithm.

Consequent to the definition of the transformation rules the following lemma is valid:

Lemma 7 (Soundness). Assume that \mathcal{S}' is obtained from the finite set of ABoxes \mathcal{S} by application of a transformation rule. Then \mathcal{S} is consistent iff \mathcal{S}' is consistent.

The second important property of the set of transformation rules is that the transformation process always terminates:

Lemma 8 (Termination [Baasat99, Don et al. 97]). *Let C_0 be an $ALCN$ -concept description. There cannot be an infinite sequence of rule applications*

$$\{\{C_0(x_0)\}\} \rightarrow S_1 \rightarrow S_2 \rightarrow \dots$$

Lemma 9. *Let A be an ABox contained in S_i for some $i \geq 1$. Then:*

- For every individual $x \neq x_0$ occurring in A , there is a unique sequence R_1, \dots, R_l ($l \geq 1$) of role names and a unique sequence x_1, \dots, x_{l-1} of individual names such that $\{R_l(x_0, x_1), R_l(x_1, x_2), \dots, R_l(x_{l-1}, x)\} \subseteq A$. In this case, we say that x occurs on level l in A .
- If $C(x) \in A$ for an individual name x on level l , then the maximal role depth of C (i.e., the maximal nesting of constructors involving roles) is bounded by the maximal role depth of C_0 minus l . Consequently, the level of any individual in A is bounded by the maximal role depth of C_0 .
- If $C(x) \in A$, then C is a subdescription of C_0 . Consequently, the number of different concept assertions on x is bounded by the size of C_0 .
- The number of different role successors of x in A (i.e., individuals y such that $R(x, y) \in A$ for a role name R) is bounded by the sum of the numbers occurring in at least restrictions in C_0 plus the number of different existential restrictions in C_0 .

Starting with $\{\{C_0(x_0)\}\}$, we thus obtain after a finite number of rule applications a set of ABoxes \hat{S} , to which no more rules apply. An ABox A is called complete iff none of the transformation rules applies to it. Consistency of a set of complete ABoxes can be determined by looking for clashes. The ABox A contains a clash iff one of the following three situations occurs:

- $\{1(x)\} \subseteq A$ for some individual name x ;
- $\{A(x), \neg A(x)\} \subseteq A$ for some individual name x and some concept name A ;
- $\{\{\leq nR(x)\} \cup \{R(x, y_i) \mid 1 \leq i \leq n+1\} \vee \{y_i \neq y_j \mid 1 \leq i < j \leq n+1\}\} \subseteq A$ for individual names x, y_1, \dots, y_{n+1} , a nonnegative integer n , and a role name R .

Obviously, an ABox that contains a clash cannot be consistent. Hence, if all the ABoxes in \hat{S} contain a clash, then \hat{S} is inconsistent, and thus by the soundness lemma $\{C_0(x_0)\}$ is inconsistent as well. Consequently, C_0 is unsatisfiable. If, however, one of the complete ABoxes in \hat{S} is clash-free, then \hat{S} is consistent. By soundness of the rules, this implies consistency of $\{C_0(x_0)\}$, and thus satisfiability of C_0 .

Lemma 10 (Completeness). *Any complete and clash-free ABox A has a model.*

This lemma can be proved by defining the canonical interpretation I_A induced by A :

- the domain Δ^{I_A} of I_A consists of all the individual names occurring in A ;
- for all atomic concepts A we define $A^{I_A} = \{x \mid A(x) \in A\}$;

- for all atomic roles R we define $R^{I_A} = \{(x, y) \mid R(x, y) \in A\}$.

I_A satisfies all the role assertions in A , by definition, and, by induction on the structure of concept descriptions, it is easy to show that it satisfies the concept assertions as well. The inequality assertions are satisfied since $x \neq y \in A$ only if x, y are different individual names.

The facts stated in Lemma 9 imply that the canonical interpretation has the shape of a finite tree whose depth is linearly bounded by the size of C_0 and whose branching factor is bounded by the sum of the numbers occurring in at least restrictions in C_0 plus the number of different existential restrictions in C_0 . Consequently, $ALCN$ has the finite tree model property, i.e., any satisfiable concept C_0 is satisfiable in a finite interpretation I that has the shape of a tree whose root belongs to C_0 .

Theorem 2. *It is decidable whether or not an $ALCN$ -concept is satisfiable.*

Theorem 3. *Satisfiability of $ALCN$ -concept descriptions is PSPACE-complete.*

2.3.4. Extension to the consistency problem for ABoxes. Algorithm that decides consistency of $ALCN$ -ABoxes can be constructed as an extension of described tableau-based satisfiability algorithm. Let A be an $ALCN$ -ABox. To test A for consistency, we first add inequality assertions $a \neq b$ for every pair of distinct individual names a, b occurring in A . Let A_0 be the ABox obtained in this way. The consistency algorithm applies the rules of Figure 2 to the singleton set $\{A_0\}$. Soundness and completeness of the rule set can be shown as before.

Termination can be enabled by requiring that generating rules \rightarrow_3 and \rightarrow_{\geq} may only be applied if none of the other rules are applicable.

Following a similar idea, the consistency problem for $ALCN$ -ABoxes can be reduced to satisfiability of $ALCN$ -concept descriptions [Hol96]. Roughly speaking, this reduction works as follows: In a preprocessing step, one applies the transformation rules only to old individuals (i.e., individuals present in the original ABox). Subsequently, one can forget about the role assertions, i.e., for each individual name in the preprocessed ABox, the satisfiability algorithm is applied to the conjunction of its concept assertions.

Theorem 4. *Consistency of $ALCN$ -ABoxes is PSPACE-complete.*

2.3.5. Extension to general inclusion axioms. In the above subsections, we have considered the satisfiability problem for concept descriptions and the consistency problem for ABoxes without an underlying TBox. In fact, for acyclic TBoxes one can simply expand the definitions. Expansion is, however, no longer possible if general inclusion axioms of the form $C \sqsubseteq D$, where C and D may be complex descriptions, are allowed. Instead of considering finitely many such axioms $C_1 \sqsubseteq D_1, \dots, C_n \sqsubseteq D_n$, it is sufficient to consider the single axiom $T \sqsubseteq \hat{C}$, where

$$\hat{C} = (\neg C_1 \cup D_1) \cap \dots \cap (\neg C_n \cup D_n).$$

The axiom $T \sqsubseteq \hat{C}$ simply claims that any individual must belong to the concept \hat{C} . The tableau algorithm introduced above can easily be modified in such a manner

that it takes the following axiom into account: all individuals (both the original individuals and the ones newly generated by the \rightarrow_3 - and \rightarrow_{\geq} -rule) are simply asserted to belong to \hat{C} . However, this may produce nonterminating algorithm.

Termination can be regained by detecting cyclic computations, and then blocking the application of generating rules: the application of the rules \rightarrow_3 to an individual x is blocked by an individual y in an ABox \mathcal{A} iff $\{D \mid D(x) \in \mathcal{A}\} \subseteq \{D' \mid D'(y) \in \mathcal{A}\}$. The main idea underlying blocking is that the blocked individual x can use the role successors of y instead of generating new ones.

To avoid cyclic blocking (of x by y and vice versa), we consider an enumeration of all individual names, and define that an individual x may only be blocked by individuals y that occur before x in this enumeration. This notion of blocking, together with some other technical assumptions, enables soundness, completeness as well as termination of algorithm [Buc et al. 93, Baaz96]. Thus, consistency of $ALCN$ -ABoxes with respect to general inclusion axioms is decidable. Since an algorithm may generate role paths of exponential length before blocking, it is no longer in PSPACE. In fact, even for the language ALC , satisfiability with respect to a single general inclusion axiom is known to be EXPTIME [DonMas00]. The tableau-based algorithm sketched above is a NEXPTIME algorithm. However, using the translation technique mentioned at the beginning of this section, it can be shown [DeG95] that $ALCN$ -ABoxes and general inclusion axioms can be translated into PDL(Propositional Dynamic Logic), which satisfiability can be decided in exponential time.

Theorem 5. *Consistency of $ALCN$ -ABoxes with respect to general inclusion axioms is EXPTIME-complete.*

2.3.6. Extension to other language constructors. The tableau-based algorithms for checking concept satisfiability and ABox consistency can also be employed for languages with other concept and/or role constructors. Each new constructor requires a new rule, and this rule can usually be obtained by simply considering the semantics of the constructor. Soundness of such a rule is often very easy to show. Completeness and termination are more difficult to control, since they must also take into account interactions between different rules. As it was shown above, termination can sometimes only be obtained if the application of rules is restricted by an appropriate strategy. Of course, one may only impose such a strategy if one can show that it does not perturb completeness.

2.3.7. Reasoning with respect to terminologies. As it was said before, terminologies (TBoxes) are sets of concept definitions (i.e., equalities of the form $A \equiv C$ where A is atomic) such that every atomic concept occurs at most once as a left-hand side.

Acyclic terminologies. As shown in Section 2.2, reasoning with respect to acyclic terminologies can be reduced to reasoning without terminologies by expanding the TBox, followed by replacing name symbols by their definitions in the terminology. Unfortunately, this increases the complexity of reasoning, since the expanded TBox may be exponentially larger than the original one [Neb90].

For more expressive languages, the presence of acyclic TBoxes does not necessarily increase the complexity of the subsumption problem. For example, subsumption of concept descriptions in the language ALC is PSPACE-complete, and so is subsumption with respect to acyclic terminologies. Of course, in order to obtain a PSPACE-algorithm for subsumption in ALC with respect to acyclic TBoxes, one cannot first expand the TBox completely since this might need exponential space. The main idea is that one uses a tableau-based algorithm like the one described, with the difference that it receives concept descriptions containing name symbols as input. Expansion is then done by the following rule: if the tableau-based algorithm encounters an assertion of the form $A(x)$, where A is a name occurring on the left-hand side of a definition $A \equiv C$ at this stage. It is not difficult to show that this yields a PSPACE-algorithm for satisfiability (and thus also for subsumption) of concepts with respect to acyclic TBoxes in ALC [Lut99].

There are, however, extensions of ALC for which this technique is not proper. One such example is the language $ACCF$, i.e., ALC extended by functional roles as well as agreements and disagreements on chains of functional roles (see Section 2.4 below). Satisfiability of concepts is PSPACE-complete for this language [HolNut90], but satisfiability of concepts with respect to acyclic terminologies is NEXPTIME-complete [Lut99].

Cyclic terminologies. For cyclic terminologies, expansion would not terminate. If we use descriptive semantics, then cyclic terminologies are a special case of terminologies with general inclusion axioms. Thus, the tableau-based algorithm for handling general inclusion axioms previously introduced can also be used for cyclic $ALCN$ -TBoxes with descriptive semantics.

For less expressive DLs, more efficient algorithms can, however, be obtained with the help of techniques based on finite automata.

2.4. Language extensions. In Section 2.1 we have introduced the language $ALCN$ as a Description Logic prototype. For many applications, the expressive power of $ALCN$ is not sufficient. For this reason, various other language constructors have been introduced in the literature and are employed by systems. In [Baa et al. 02] these language extensions were roughly classified into two categories, "classical" and "nonclassical" extensions. Intuitively, a classical extension is one whose semantics can easily be defined within the model-theoretic framework introduced in Section 2.1, whereas defining the semantics of a nonclassical constructor is more problematic and requires an extension of the model-theoretic framework. Hereafter, the most important classical extensions of Description Logics will be briefly introduced.

2.4.1. Role constructors. Since roles are interpreted as binary relations, it is quite natural to employ the usual operations on binary relations (such as Boolean operators, composition, inverse, and transitive closure) as role-forming constructors.

Definition 1 (Role constructors). Every role name is a role description (atomic role), and if R , S are role descriptions, then $R \sqcap S$ (intersection), $R \sqcup S$ (union), $\neg R$

(complement), $R \circ S$ (composition), R^+ (transitive closure), R^- (inverse), $\text{id}(C)$ (role identity) are also role descriptions.

Given an interpretation \mathcal{I} is extended to (complex) role descriptions as follows:

- (i) $(R \cap S)^{\mathcal{I}} = R^{\mathcal{I}} \cap S^{\mathcal{I}}$, $(R \cup S)^{\mathcal{I}} = R^{\mathcal{I}} \cup S^{\mathcal{I}}$, $(\neg R)^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \setminus R^{\mathcal{I}}$;
- (ii) $(R \circ S)^{\mathcal{I}} = \{(a, c) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (\exists b)(a, b) \in R^{\mathcal{I}} \wedge (b, c) \in S^{\mathcal{I}}\}$;
- (iii) $(R^+)^{\mathcal{I}} = \bigcup_{i \geq 1} (R^{\mathcal{I}})^i$, i.e., $(R^+)^{\mathcal{I}}$ is the transitive closure of $(R^{\mathcal{I}})$;
- (iv) $(R^-)^{\mathcal{I}} = \{(b, a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a, b) \in R^{\mathcal{I}}\}$;
- (v) $\text{id}(C)^{\mathcal{I}} = \{(a, a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid a \in C^{\mathcal{I}}\}$, i.e., each instance of concept to itself.

For example, the union of the roles `hasSon` and `hasDaughter` can be used to define the role `hasChild`, and the transitive closure of `hasChild` expresses the role `hasOffspring`. The inverse of `hasChild` yields the role `hasParent`.

Example 5. The following $\mathcal{ALC}T_{reg}$ TBox \mathcal{T}_{file} models a file-system constituted by file-system elements:

```

FSelem  $\sqsubseteq$  FName.String
FSelem  $\sqsubseteq$  Directory  $\sqcup$  File
Directory  $\sqsubseteq$  ~File
Directory  $\sqsubseteq$  Vchild.FSelem
File  $\sqsubseteq$  Vchild.1
Root  $\sqsubseteq$  Directory
Root  $\sqsubseteq$  Vchild.1
Root  $\sqsubseteq$  Vchild.1

```

The axioms in \mathcal{T}_{file} imply that in a model every object connected by a chain of role child to an instance of `Root` is an instance of `FSelem`. Formally,

$$\mathcal{T}_{file} \models \exists(\text{child})^+. \text{Root} \sqsubseteq \text{FSelem}$$

It is shown that the complexity of satisfiability and subsumption of concepts in the language $\mathcal{ALC}N^+$ (also called $\mathcal{ALC}N\mathcal{R}$ in the literature and which extends $\mathcal{ALC}N$ by intersection of roles) are still PSPACE-complete [Don et al. 97, Tob01]. Decidability of the extension of $\mathcal{ALC}N$ by the three Boolean operators and the inverse operator is a direct consequence of the fact that concepts of the extended language can be expressed in \mathcal{C}^2 , i.e., first-order predicate logic with two variables and counting quantifiers, which is known to be decidable in NEXPTIME [Grä et al. 97, Pac97]. It is also shown [LutSat00] that \mathcal{ALC} extended by role complement is EXPTIME-complete, whereas \mathcal{ALC} extended by role intersection and atomic role complement is NEXPTIME-complete.

For \mathcal{ALC}_{trans} (which extends \mathcal{ALC} by transitive-closure, composition, and union of roles) subsumption and satisfiability problem have been shown to be decidable [Baas91] and EXPTIME-complete [Frislad79, Pra79, Pra80]. The extension of \mathcal{ALC}_{trans} by the inverse constructor corresponds to converse PDL [Frislad79], which can also be shown to be decidable in deterministic exponential time [Pra85]. \mathcal{ALC}_{trans} extended by inverse and number restrictions does not have the finite

model property. Nevertheless, this DL still has an EXPTIME-complete subsumption and satisfiability problem.

2.4.2. Expressive number restrictions. First, we will consider the so-called qualified number restrictions, where the number restrictions are concerned with role-fillers belonging to a certain concept.

Example 6. Given the role `hasChild`, the simple number restrictions introduced above can only state that the number of all children is within certain limits, such as in the concept $\geq 2\text{hasChild} \sqcap \leq 5\text{hasChild}$. Qualified number restrictions can also express that there are at least 2 sons and at most 5 daughters:

$$\geq 2\text{hasChild.Male} \sqcap \leq 5\text{hasChild.Female}$$

Adding qualified number restrictions to \mathcal{ALC} leaves the important inference problems (like subsumption and satisfiability of concepts, and consistency of ABoxes) decidable: the worst-case complexity is still PSPACE-complete. The language is decidable if general sets of inclusion axioms are allowed [Buc et al. 93].

The second group of extensions are those which allow for complex role expressions inside number restrictions. The extension of $\mathcal{ALC}N$ by number restrictions involving composition has a decidable satisfiability and subsumption problem. On the other hand, if any number restrictions involving composition, union and inverse, or number restrictions involving composition and intersection are added, then satisfiability and subsumption become undecidable [Baasat96, Baasat99]. For \mathcal{ALC}_{trans} the extension by number restrictions involving compositions already undecidable [Baasat99].

Third, if the explicit numbers n in number restrictions are replaced by variables a that stand for arbitrary nonnegative integers, the expressive power of language can further be increased by introducing explicit quantification of the numeric variables.

It is shown that $\mathcal{ALC}N$ extended by such symbolic number restrictions with universal and existential quantification of numerical variables has an undecidable satisfiability and subsumption problem. If one restricts this language to existential quantification of numerical variables and negation on atomic concepts, then satisfiability becomes decidable, but subsumption remains undecidable.

2.4.3. Role-value-maps. Role-value-maps are a family of very expressive concept constructors, which were, however, available in the original KL-One-system.

Definition 2 (Role-value-maps). A role chain is a composition $R_0 \circ \dots \circ R_n$ of role names. If R_i, S are role chains, then $R \sqsubseteq S$ and $R = S$ are concepts.

A given interpretation \mathcal{I} is extended to role-value-maps as follows:

- (i) $(R \sqsubseteq S)^{\mathcal{I}} = \{(a \in \Delta^{\mathcal{I}} \mid (\forall b)((a, b) \in R^{\mathcal{I}} \rightarrow (a, b) \in S^{\mathcal{I}}))\}$
- (ii) $(R = S)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid (\forall b)((a, b) \in R^{\mathcal{I}} \leftrightarrow (a, b) \in S^{\mathcal{I}})\}$

Example 7. The concept

$$\text{Person} \sqcap (\text{hasChild} \circ \text{hasFriend} \sqsubseteq \text{knows})$$

describes the persons knowing all friends of their children, and

$$\text{Person} \sqcap (\text{marriedTo} \circ \text{likesToEat} = \text{likesToEat})$$

describes persons having the same favorite foods as their spouse.

Unfortunately, in the presence of role-value-maps, the subsumption problem is undecidable, even if the language allows only for conjunction and value restriction as additional constructors.

Solution to this problem is restricting the attention to role chains of functional roles, also called attributes or features in the literature. An interpretation \mathcal{I} interprets the role R as a functional role iff $\{(a, b), (a, c)\} \subset R^{\mathcal{I}}$ implies $b = c$. In the following, it will be assumed that the set of role names is partitioned into the set of functional roles and the set of ordinary roles. Any interpretation must interpret the functional roles as such. Functional roles will be denoted with small letters f, g , possibly with index.

Definition 3 (Agreements). If f, g are role chains of functional roles, then $f \doteq g$ and $f \neq g$ are concepts (agreement and disagreement).

A given interpretation \mathcal{I} is extended to agreements and disagreements as follows:

- (i) $(f \doteq g)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid (\exists b)(\exists c)(a, b) \in f^{\mathcal{I}} \wedge (a, c) \in g^{\mathcal{I}}\}$
- (ii) $(f \neq g)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid (\exists b_1, b_2)(b_1 \neq b_2 \wedge (a, b_1) \in f^{\mathcal{I}} \wedge (a, b_2) \in g^{\mathcal{I}})\}$

In the literature, the agreement constructor is sometimes also called the "same-as" constructor. Since f, g are the role chains between the functional roles, there can be at most one role filler for a with respect to the respective role chain. The semantics of agreements and disagreements requires these role fillers to exist (and be equal or distinct) for a to belong to the concept.

Example 8. Roles such as hasMother , hasFather and hasLastName with their usual interpretation are functional roles, whereas hasParent and hasChild are not. The concept

$$\text{Person} \sqcap (\text{hasLastName} \doteq \text{hasMother} \circ \text{hasLastName}) \sqcap (\text{hasLastName} \neq \text{hasFather} \circ \text{hasLastName})$$

describes persons whose last name coincides with the last name of their mother, but not with the last name of their father.

The restriction to functional roles makes reasoning in \mathcal{ALC} extended by agreements and disagreements decidable [HolNutt90]. However, if general inclusion axioms (or transitive closure of functional roles or cyclic definitions) are allowed, then agreements and disagreements between chains of functional roles again cause subsumption to become undecidable.

2.4.4. Functional restrictions (\mathcal{F}). Functional restrictions are the simplest form of number restrictions considered in description logics, and allow for specifying local functionality of roles, i.e., that instances of certain concepts have unique role-fillers for a given role. By adding functional restrictions on atomic roles and their inverse

to \mathcal{ALCFI}_{reg} we obtain the description logic \mathcal{ALCFI}_{reg} . Functional restrictions has a form $\leq 1Q$, where Q is a basic role, i.e., either an atomic role or the inverse of an atomic role. Such a functional restriction is interpreted as follows:

$$(\leq 1Q)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a, b) \in Q^{\mathcal{I}}\}| \leq 1\}$$

Reasoning in \mathcal{ALCFI}_{reg} is ExpTime-complete. Also, \mathcal{ALCFI}_{reg} has the tree model property, which states that if a \mathcal{ALCFI}_{reg} -concept C is satisfiable then it is satisfied in an interpretation which has the structure of a (possibly infinite) tree with bounded branching degree. This makes the space for using of techniques based on automata on infinite trees.

2.4.5. Qualified number restrictions (\mathcal{Q}). Qualified number restrictions is the most general form of number restrictions, and allow for specifying arbitrary cardinality constraints on roles with role-fillers belonging to a certain concept. In particular we will consider qualified number restrictions on basic roles, i.e., atomic roles and their inverse. By adding such constructs to \mathcal{ALCFI}_{reg} we obtain the description logic \mathcal{ACQFI}_{reg} .

Qualified number restrictions has a form $\leq nQC$ and $\geq nQC$, where n is a nonnegative integer, Q is a basic role, and C is an \mathcal{ALCFI}_{reg} -concept. Such constructs are interpreted as follows:

$$\begin{aligned} (\leq nQC)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a, b) \in Q^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}| \leq n\} \\ (\geq nQC)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a, b) \in Q^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}| \geq n\} \end{aligned}$$

Reasoning in \mathcal{ACQFI}_{reg} is still ExpTime-complete.

2.4.6. Relations of arbitrary arity. A limitation of traditional description logics is that only binary relationships between instances of concepts can be represented, which is a quite limitation in a process of modeling relationships among more than two objects in some real world situations. Such relationships can be described by making use of relations of arbitrary arity instead of (binary) roles.

Let us consider the description logic \mathcal{DLR} , which represents a natural generalization of traditional description logics towards n -ary relations. The basic elements of \mathcal{DLR} are atomic relations and atomic concepts, denoted by P and A , respectively. Arbitrary relations, of given arity between 2 and n_{max} , and arbitrary concepts are formed according to the following syntax

$$\begin{aligned} R &\rightarrow T_n \mid P \mid (i/n : C) \mid \neg R \mid R_1 \sqcap R_2 \\ C &\rightarrow T_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[i]R \mid \leq k[i]R \end{aligned}$$

where i denotes a component of a relation, i.e., an integer between 1 and n_{max} , n denotes the arity of a relation, i.e., an integer between 2 and n_{max} , and k denotes a nonnegative integer.

For \mathcal{DLR} interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \mathcal{I})$ is introduced as follows:

$$\begin{aligned} T_n^{\mathcal{I}} &\subseteq (\Delta^{\mathcal{I}})^n \\ P^{\mathcal{I}} &\subseteq T_n^{\mathcal{I}} \end{aligned}$$

$$(\neg R)^T = T_n^T \setminus R^T$$

$$(R_1 \cap R_2)^T = R_1^T \cap R_2^T$$

$$(i/n : C)^T = \{(a_1, \dots, a_n) \in T_n^T \mid a_i \in C^T\}$$

$$T_1^T = \Delta^T$$

$$A^T \subseteq \Delta^T$$

$$(-C)^T = \Delta^T \setminus C^T$$

$$(C_1 \cap C_2)^T = C_1^T \cap C_2^T$$

$$(\exists i)(R)^T = \{d \in \Delta^T \mid (\exists (a_1, \dots, a_n) \in R^T d_i = d)\}$$

$$\leq k(i)(R)^T = \{d \in \Delta^T \mid | \{(a_1, \dots, a_n) \in R^T \mid d_i = d\} | \leq k\}$$

Theorem 6. Logical implication in \mathcal{DLR} is ExpTime-complete .

\mathcal{DLR} can be extended to include regular expressions built over projections of relations on two of their components, thus obtaining \mathcal{DLR}_{reg} (decidability is also ExpTime-complete).

\mathcal{DLR} and \mathcal{DLR}_{reg} are generalizations of \mathcal{ALCQI} and \mathcal{ALCQI}_{reg} , and they can be extended by Boolean constructs on roles and role inclusion axioms. Obtained languages have ExpTime-complete logical implication.

Reasoning in \mathcal{SHIQ} , which is \mathcal{ALCQI} extended with roles that are transitive, and with role inclusion axioms on arbitrary roles (direct, inverse, and transitive), is still ExpTime-complete .

3. Description logics with modal operators

3.1. Preliminaries. We begin by defining the modal concept description language \mathcal{ALCM} and its semantics.

The primitive symbols of \mathcal{ALCM} are:

- concept names C_0, C_1, \dots ;
- role names R_0, R_1, \dots ; and
- object names a_0, a_1, \dots .

Starting from these we can form compound concepts and formulas using the following constructs. Suppose R is a role name and C, D are concepts (for the basis of our inductive definition we assume concept names to be atomic concepts). Then \top , $C \sqcap D$, $\neg C$, $\exists R.C$, and $\forall C$ (or CUD , CSD for a strict linear order) are concepts.

Atomic formulas are expressions of the form $\top, C = D, a : C$, and aRb , where a, b are object names. If φ and ψ are formulas then so are $\varphi \wedge \psi$, $\neg \varphi$, and $\forall \varphi$ (or $\forall \lambda \varphi$, $\psi \Delta \varphi$ for a strict linear order).

The pure description part of this language is \mathcal{ALC} . By adding the constructs for the formation of the union $R \cup S$, composition $R \circ S$, transitive reflexive closure R^+ and test $C_i^?$, we can extend it to C_i , and to CI (CIQ) by adding still inversion R^- (inversion R^- and number restrictions $\geq n B.C$, where B is a role name or its converse). The corresponding modal description language is denoted then by C_M , CI_M and CIQ_M .

A model of \mathcal{ALCM} based on a frame $\mathfrak{F} = (W, \triangleleft)$ is a pair $\mathfrak{M} = (\mathfrak{F}, I)$ in which I is a function associating with each $w \in W$ a structure

$$I(w) = \left\langle \Delta^{I,w}, R_0^{I,w}, \dots, C_0^{I,w}, \dots, a_0^{I,w}, \dots \right\rangle,$$

where $\Delta^{I,w}$ is a nonempty set of objects, the domain of w , $R_i^{I,w}$ are binary relations on $\Delta^{I,w}$, $C_i^{I,w}$ subsets of $\Delta^{I,w}$, and $a_i^{I,w}$ are objects in $\Delta^{I,w}$ such that $a_i^{I,w} = a_i^{I,v}$ for any $v, w \in W$.

One can distinguish between three types of models: those with constant, expanding, and varying domains. In models with constant domains $\Delta^{I,w} = \Delta^{I,v}$ for all $v, w \in W$. In models with expanding domains $\Delta^{I,v} \subseteq \Delta^{I,w}$ whenever $v \triangleleft w$. And models with varying domains are just arbitrary models.

Given a model \mathfrak{M} and a world w in it, we define the value $C^{I,w}$ of a concept C in w and the truth-relation $(\mathfrak{M}, w) \models \varphi$ (or simply $w \models \varphi$, if \mathfrak{M} is understood) by taking:

$$\begin{aligned} \top^{I,w} &= \Delta, \text{ and } C^{I,w} = C_i^{I,w}, \text{ for } C = C_i; \\ (C \sqcap D)^{I,w} &= C^{I,w} \cap D^{I,w}; \quad (-C)^{I,w} = \Delta \setminus C^{I,w}; \\ x \in (CSD)^{I,w} &\text{ iff } \exists v \triangleright w \text{ s.t. } x \in C^{I,v}; \\ x \in (\exists RC)^{I,w} &\text{ iff } \exists y \in C^{I,w} \text{ s.t. } x R^{I,w} y; \end{aligned}$$

$$\begin{aligned} w \models C &= D &\text{ iff } C^{I,w} &= D^{I,w}; \\ w \models a : C &&\text{ iff } a^{I,w} &\in C^{I,w}; \\ w \models aRb &&\text{ iff } a^{I,w} R^{I,w} b^{I,w}; \\ w \models \forall \varphi &&\text{ iff } \exists v \triangleright w \text{ s.t. } v \models \varphi; \\ w \models \varphi \wedge \psi &&\text{ iff } w \models \varphi \text{ and } w \models \psi; \\ w \models \neg \varphi &&\text{ iff } w \not\models \varphi. \end{aligned}$$

If $\mathfrak{F} = (W, \triangleleft)$ is a strict linear order with modal operators U and S , then we have $x \in (CUUD)^{I,w}$ iff there is $u \triangleright w$ such that $x \in D^{I,u}$ and $x \in C^{I,v}$ for all $v \in (u, w)$; $x \in (CSD)^{I,w}$ iff there is $u < w$ such that $x \in D^{I,u}$ and $x \in C^{I,v}$ for all $v \in (u, w)$; $w \models \psi U \chi$ iff there is $u \triangleright w$ such that $u \models \chi$ and $v \models \psi$ for all $v \in (u, w)$; and $w \models \psi S \chi$ iff there is $u < w$ such that $u \models \chi$ and $v \models \psi$ for all $v \in (u, w)$.

A formula φ is *satisfiable* in a class of models \mathcal{M} if there is a model $\mathfrak{M} \in \mathcal{M}$ and a world w in \mathfrak{M} such that $w \models \varphi$. We will use special names for certain classes of models with one accessibility relation. Namely,

- \mathcal{K} the class of all models;
- $\mathcal{S5}$ the class of models based on frames with the universal relations, i.e., $u \triangleleft v$ for all u and v ;
- $\mathcal{KD45}$ the class of transitive, serial ($\forall u \exists v u \triangleleft v$) and Euclidean ($u \triangleleft v \wedge u \triangleleft w \rightarrow v \triangleleft w$) models;
- $\mathcal{S4}$ the class of all quasi-ordered models;
- $\mathcal{K4}$ the class of transitive models;

GL the class of transitive Noetherian models

(i.e., containing no infinite ascending chains); and
 \mathcal{N} the class of models based on $(\mathbb{N}, <)$.

We are in a position now to present known decidability and complexity results concerning formula-satisfiability problems [MosZakh99, Mos2000].

Theorem 7. (1) *The formula-satisfiability problem, when we adopt expanding domain assumption, for the language ACC_M in each of the classes \mathcal{K} , \mathcal{N} , \mathcal{GL} , $S4$, and $K4$ is NEXPTIME-hard.*

(2) *The formula-satisfiability problem for the language ACC_M and CIQ_M in the class \mathcal{K} is NEXPTIME-complete (no matter whether the models have constant or expanding domains).*

(3) *The formula-satisfiability problem for the language ACC_M and CIQ_M in the class $S5$ is NEXPTIME-complete.*

(4) *The formula-satisfiability problem for the language ACC_M and CIQ_M in the class \mathcal{N} is EXPSpace-complete.*

For these logics, tableau algorithms were developed [Lutz et al. 01, Lutz et al. 02]. Further on, we will continue with presenting of one temporal extension of description logics [Arta et al. 01, Arta et al. 02], as a special case of modal extension of description logics.

3.2. The Temporal Description Logic. Here, we adopt the *snapshot* representation of abstract temporal databases (and temporal conceptual models), see for example [ChoSa98]. The flow of time $\mathcal{T} = \langle \mathcal{T}_n, < \rangle$, where \mathcal{T}_n is a set of time points (or chronons) and $<$ a binary precedence relation on \mathcal{T}_n , is assumed to be isomorphic to $(\mathbb{Z}, <)$. Thus, a temporal database can be regarded as a map from time points in \mathcal{T} to standard (relational) databases with the same domain of attributes and the same interpretation of constants.

As a language of temporal database conceptual schemas we use a natural combination of the propositional linear temporal logic with *Since* and *Until* [SisC85, Gab et al. 94] and the (non-temporal) description logic \mathcal{DLR} [Cal et al. 98]. The resulting *temporal description logic* will be denoted by $\mathcal{DLR}_{\mathcal{L}_{\mathcal{D}}}$.

The basic syntactical types of $\mathcal{DLR}_{\mathcal{L}_{\mathcal{D}}}$ are *entities* (i.e., unary predicates, also known as *concepts*) and *nary relations* of arity ≥ 2 . Starting from a set EN of *atomic entities* and a set RN of *atomic relations* we define inductively (complex) entity and relation expressions as is shown in the upper part of Fig. 3, where the binary constructs $(\cap, \sqcup, \mathcal{U}, \mathcal{S})$ are applied to relations of the same arity, i, j, k, n are natural numbers, $i \leq n$, and j does not exceed the arity of R .

A *temporal conceptual database schema* (or a *knowledge base*) is a finite set Σ of $\mathcal{DLR}_{\mathcal{L}_{\mathcal{D}}}$ -formulas. Atomic formulas are formulas of the form $E_1 \sqsubseteq E_2$ and $R_1 \sqsubseteq R_2$, with R_1 and R_2 being relations of the same arity. If φ and ψ are $\mathcal{DLR}_{\mathcal{L}_{\mathcal{D}}}$ -formulas, then so are $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \mathcal{U} \psi$, $\varphi \mathcal{S} \psi$. $E_1 \sqsubseteq E_2$ is used as an abbreviation for $(E_1 \sqsubseteq E_2) \wedge (E_2 \sqsubseteq E_1)$, for both entities and relations. Temporal conceptual database schemas will serve as constraints for temporal databases.

$$\begin{aligned}
R &\rightarrow T_n[RN] \neg R[R_1 \sqcap R_2] R_1 \sqcup R_2 [t/n : E] \\
&\quad \Diamond^+ R[\Diamond^- R[\Box^+ R[\Box^- R] \oplus R] \oplus R[R_1 R_2] R_1 S R_2] \\
E &\rightarrow T_n[EN] \neg E[E_1 \sqcap E_2] E_1 \sqcup E_2 [\exists^{\geq k} t_j] R \\
&\quad \Diamond^+ E[\Diamond^- E[\Box^+ E[\Box^- E] \oplus E] \oplus E[E_1 \mathcal{U} E_2] E_1 S E_2] \\
&\quad \subseteq (\Delta^I)^n \\
&\quad \subseteq (T_n)^{f(t)} \\
&\quad \subseteq (\neg R)^{f(t)} \setminus R^{f(t)} \\
&\quad = (R_1 \sqcap R_2)^{f(t)} \\
&\quad = (t/n : E)^{f(t)} \\
&\quad = (R_1 \mathcal{U} R_2)^{f(t)} \\
&\quad = \{(d_1, \dots, d_n) \in (T_n)^{f(t)} \mid d_i \in E^{f(t)}\} \\
&\quad = \{(d_1, \dots, d_n) \in (T_n)^{f(t)} \mid \exists v > t. \langle \langle d_1, \dots, d_n \rangle \in R_1^{f(v)} \rangle \\
&\quad \quad \wedge \forall w \in \langle t, v \rangle. \langle \langle d_1, \dots, d_n \rangle \in R_2^{f(w)} \rangle\} \\
&\quad = \{(d_1, \dots, d_n) \in (T_n)^{f(t)} \mid \exists v < t. \langle \langle d_1, \dots, d_n \rangle \in R_2^{f(v)} \rangle \\
&\quad \quad \wedge \forall w \in \langle v, t \rangle. \langle \langle d_1, \dots, d_n \rangle \in R_1^{f(w)} \rangle\} \\
&\quad = \{(\Diamond^+ R)^{f(t)}\} \\
&\quad = \{(d_1, \dots, d_n) \in (T_n)^{f(t)} \mid \exists v > t. \langle d_1, \dots, d_n \rangle \in R^{f(v)}\} \\
&\quad = \{(\oplus R)^{f(t)}\} \\
&\quad = \{(d_1, \dots, d_n) \in (T_n)^{f(t)} \mid \langle d_1, \dots, d_n \rangle \in R^{f(t+1)}\} \\
&\quad = \{(\Diamond^- R)^{f(t)}\} \\
&\quad = \{(d_1, \dots, d_n) \in (T_n)^{f(t)} \mid \exists v < t. \langle d_1, \dots, d_n \rangle \in R^{f(v)}\} \\
&\quad = \{(\ominus R)^{f(t)}\} \\
&\quad = \{(d_1, \dots, d_n) \in (T_n)^{f(t)} \mid \langle d_1, \dots, d_n \rangle \in R^{f(t-1)}\} \\
&\quad = \Delta^I \\
&\quad \subseteq T^{f(t)} \\
&\quad = EN^{f(t)} \\
&\quad = (\neg E)^{f(t)} \setminus E^{f(t)} \\
&\quad = (E_1 \sqcap E_2)^{f(t)} \\
&\quad = (E_1 \mathcal{U} E_2)^{f(t)} \\
&\quad = \{d \in T^{f(t)} \mid \exists v > t. \langle d \in E_2^{f(v)} \wedge \forall w \in \langle t, v \rangle. d \in E_1^{f(w)} \rangle\} \\
&\quad = \{d \in T^{f(t)} \mid \exists v < t. \langle d \in E_2^{f(v)} \wedge \forall w \in \langle v, t \rangle. d \in E_1^{f(w)} \rangle\}
\end{aligned}$$

FIGURE 3. Syntax and semantics of $\mathcal{DLR}_{\mathcal{L}_{\mathcal{D}}}$.

The language of $\mathcal{DLR}_{\mathcal{L}_{\mathcal{D}}}$ is interpreted in *temporal models* over \mathcal{T} , which are triples of the form $I = \langle \mathcal{T}, \Delta^I, \cdot^I \rangle$, where Δ^I is nonempty set of objects (the domain of I) and \cdot^I an *interpretation function* such that, for every $t \in \mathcal{T}$, every entity E , and every n -ary relation R , we have $E^I(t) \subseteq \Delta^I$ and $R^I(t) \subseteq (\Delta^I)^n$. The semantics of entity and relation expressions is defined in the lower part of Fig. 3, where $\langle u, v \rangle = \{w \in \mathcal{T} \mid u < w < v\}$ and the operators \Box^+ (always in the future) and \Box^- (always in the past) are the duals of \Diamond^+ (some time in the future) and \Diamond^- (some time in the past), respectively, i.e., $\Box^+ E \equiv \neg \Diamond^+ \neg E$ and $\Box^- E \equiv \neg \Diamond^- \neg E$, for both entities and relations. For entities, the temporal operators \Diamond^+ and \Diamond^- (at the next moment), and their past counterparts can be defined via \mathcal{U} and \mathcal{S} : $\Diamond^+ E \equiv \mathcal{U} E$, $\Diamond^- E \equiv \mathcal{S} E$, etc. However, this is not possible for relations of arity > 1 , since T_n —the top nary relation—can be interpreted by different subsets of the n -ary

cross product $T \times \dots \times T$ at different time points.³ The operators \Diamond^* (at some moment) and its dual \Box^* (at all moments) can be defined for both entities and relations as $\Diamond^* E \equiv E \cup \Diamond^+ E \cup \Diamond^- E$ and $\Box^* E \equiv E \cup \Box^+ E \cup \Box^- E$, respectively.

The nontemporal fragment of \mathcal{DLRus} coincides with \mathcal{DLR} . For both entity and relation expressions all the Boolean constructs are available. The selection expression i/n : E denotes an n -ary relation whose i -th argument ($i \leq n$) is of type E ; if it is clear from the context, we omit n and write $(i : E)$. The projection over the i th argument of the relation R (which coincides with $\exists^{[1]}[i]R$). It is also possible to use the named attribute version of the model by replacing argument position numbers with role names.

Given a formula φ , an interpretation I , and a time point $t \in T$, the truthrelation $I, t \models \varphi$ (φ holds in I at moment t) is defined inductively as follows:

$$\begin{aligned} I, t \models E_1 \subseteq E_2 & \text{ iff } E_1^{(t)} \subseteq E_2^{(t)} \\ I, t \models R_1 \subseteq R_2 & \text{ iff } R_1^{(t)} \subseteq R_2^{(t)} \\ I, t \models \varphi \wedge \psi & \text{ iff } I, t \models \varphi \text{ and } I, t \models \psi \\ I, t \models \neg \varphi & \text{ iff } I, t \not\models \varphi \\ I, t \models \forall x \psi & \text{ iff } \forall v > t, (I, v \models \psi \wedge \forall w \in (t, v) I, w \models \varphi) \\ I, t \models \exists x \psi & \text{ iff } \exists v < t, (I, v \models \psi \wedge \forall w \in (t, v) I, w \models \varphi) \end{aligned}$$

A formula φ is called *satisfiable* if there is a temporal model I such that $I, t \models \varphi$, for some time point t . A conceptual schema Σ is *satisfiable* if the conjunction $\bigwedge \Sigma$ of all formulas in Σ is satisfiable (we write $I, t \models \Sigma$ instead of $I, t \models \bigwedge \Sigma$); in this case I is called a *model* of Σ . We say that Σ is *globally satisfiable* if there is I such that $I, t \models \Sigma$ for every t ($I, t \models \Sigma$, in symbols). An entity E (or relation R) is *satisfiable* if there is I such that $E^{(t)} \neq \emptyset$ (respectively, $R^{(t)} \neq \emptyset$), for some time point t . Finally, we say that Σ (*globally*) *implies* φ and write $\Sigma \models \varphi$ if we have $I \models \varphi$ whenever $I \models \Sigma$.

Note that an entity E is satisfiable iff $\neg(E \subseteq \perp)$ is satisfiable. An n -ary relation R is satisfiable iff $\neg(\exists^{[1]}[i]R \subseteq \perp)$ is satisfiable for some $i \leq n$. A conceptual schema Σ is globally satisfiable iff $\Box^*(\bigwedge \Sigma)$ is satisfiable. And $\Sigma \models \varphi$ iff $\Box^*(\bigwedge \Sigma) \wedge \neg \varphi$ is not satisfiable. Thus, all reasoning tasks connected with the notions introduced above reduce to satisfiability of formulas.

The logic \mathcal{DLRus} can be regarded as a rather expressive fragment of the first-order temporal logic $L(\text{since}, \text{until})$; cf. [ChosSaa98, Hod et al. 2000].

3.3. Temporal queries. One more important reasoning task is known as the problem of query containment (see, e.g., [ChosSaa98, Chos94, Abi et al. 96] for a survey and a discussion about temporal queries). A *non-recursive Datalog query* (i.e., a disjunction of conjunctive queries or SP-queries) over a \mathcal{DLRus} schema Σ is an

³For instance, we may have $\langle d_1, d_2 \rangle \in (\Diamond^+ R)^{(t)}$ because $\langle d_1, d_2 \rangle \in (\Diamond^+ R)^{(t+(+2))}$, but $\langle d_1, d_2 \rangle \notin (T_2)^{(t+(+1))}$.

expression of the form

$$Q(\vec{x}) : - \bigvee_j Q_j(\vec{x}, \vec{y}_j, \vec{c}_j),$$

where each Q_j is a conjunction of atoms

$$Q_j(\vec{x}, \vec{y}_j, \vec{c}_j) \equiv \bigwedge_i P_j^i(\vec{x}_i, \vec{y}_i, \vec{c}_i),$$

P_j^i are \mathcal{DLRus} entity or relation expressions, \vec{x}_i, \vec{y}_i , and \vec{c}_i are sequences of distinguished variables, existential variables, and constants, respectively, the number of which is in agreement with the arity of P_j^i . The variables \vec{x} in the head are the union of all the distinguished variables in each Q_j ; the existential variables are used to make coreferences in the query, and constants are fixed values. The arity of Q is the number of variables in \vec{x} .

It is to be noted that we allow entities and relations in the query to occur in the conceptual schema Σ . This approach is similar to that of [Cal et al. 98], where atoms in a query can be constrained by means of schema formulas.

The semantics of queries is defined as follows. Let I be a temporal model and t a time point in T such that I satisfies Σ at t , i.e., $I, t \models \Sigma$. The snapshot interpretation

$$I(t) = \langle \Delta^t, \{E^{(t)} \mid E \in EN\}, \{R^{(t)} \mid R \in RN\} \rangle$$

can be regarded as a usual first-order structure (i.e., a snapshot nontemporal database at time t conforming in a sense to the conceptual schema), and so the whole I as a first-order temporal model (with constant domain Δ^t in which some values of the query constants are specified). The *evaluation* of a query Q of arity n under the constraints Σ in the model I at moment t is the set

$$\text{ans}(Q, I(t)) = \left\{ \vec{d} \in (\Delta^t)^n \mid I, t \models \bigvee_j \exists \vec{y}_j Q_j(\vec{d}, \vec{y}_j, \vec{c}_j) \right\}$$

Given two queries (of the same arity) Q_1 and Q_2 over Σ , we say that Q_1 is *contained* in Q_2 under the constraints Σ and write $\Sigma \models Q_1 \subseteq Q_2$ if, for every temporal model I and every time point t , we have $\text{ans}(Q_1, I(t)) \subseteq \text{ans}(Q_2, I(t))$ whenever $I, t \models \Sigma$. Note that the *query satisfiability problem*—given a query Q over a schema Σ , to determine whether there are I and t such that $I, t \models \Sigma$ and $\text{ans}(Q, I(t)) \neq \emptyset$ —is reducible to query containment: Q is satisfiable iff $\Sigma \models Q(\vec{x}) \subseteq P(\vec{x}) \wedge \neg P(\vec{x})$, where P is a \mathcal{DLRus} relation of the same arity as Q .

3.4. Conceptual Schema and Query Examples. As an example, let us consider the following conceptual schema Σ , where we introduce a shortcut for global atomic formulas $E_1 \sqsubseteq^* E_2 \equiv \Box^*(E_1 \subseteq E_2)$, for both entities and relations:

Works for $\sqsubseteq^* \text{emp}/2$: Employee $\sqcap \text{act}/2$: Project
 Manages $\sqsubseteq^* \text{man}/2$: TopManager $\sqcap \text{pi}/2$: Project
 Employee $\sqsubseteq^* \exists^{[1]}[\text{from}] \text{PaySlipNumber}$
 $\sqcap \exists^{[1]}[\text{from}] \text{PaySlipNumber} \sqcap \text{to}/2$: Integer

$$\top \sqsubseteq \exists^1 \text{to}[(\text{PaySlipNumber} \sqcap \text{from}/2 : \text{Integer})]$$

$$\text{Managerv} \sqsubseteq \text{Employee} \sqcap (\text{AreaManager} \sqcup \text{TopManager})$$

$$\text{AreaManager} \sqsubseteq \text{Manager} \sqcap \neg \text{TopManager}$$

$$\text{TopManager} \sqsubseteq \text{Manager} \sqcap \neg \text{AreaManager}$$

$$\text{Project} \sqsubseteq \exists^1 \text{act}[\text{Works-for} \sqcap \exists^1 \text{prj}[\text{Manages}]]$$

$$\text{Employee} \sqcap \neg(\exists^1 \text{emp}[\text{Works-for}]) \sqsubseteq \text{Manager}$$

$$\text{Managerv} \sqsubseteq \neg(\exists^1 \text{emp}[\text{Works-for}]) \sqcap (\text{QualifiedS} (\text{Employee} \sqcap \neg \text{Manager}))$$

The theory introduces *Works-for* as a binary relation between Projects and employees, and *Manages* as a binary relation between managers and projects. Employees have exactly one pay slip number and one salary each, which are represented as binary relations (with from and to roles) with an integer domain; moreover, a pay slip number uniquely identifies an employee (it acts as a key). It is stated that managers are employees, and are partitioned into area managers and top managers. Top Managers participate exactly once in the relation *Manages*, i.e., every top manager manages exactly one project. Projects participate at least once to the relation *Works-for* and exactly once in the relation *Manages*. Finally, employees not working for a project are exactly the managers, and managers should be qualified, i.e., should have passed a period of being employees. The meaning of the above conceptual schema (with the exception of the last two formulas) is illustrated by the left-hand part of the diagram in Fig. 4.

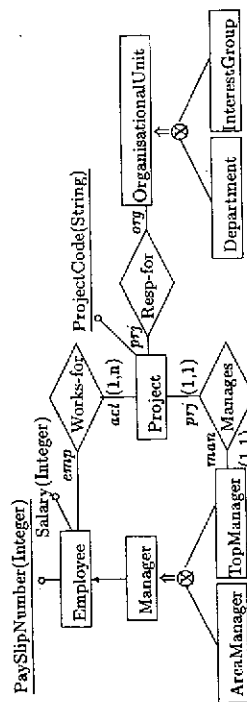


FIGURE 4. The example EER diagram.

The conceptual schema Σ globally logically implies that, for every project, there is at least one employee who is not a manager, and that a top manager worked in a project before managing some (possibly different) project:

$$\Sigma \models \text{Project} \sqsubseteq \exists^1 \text{act}[(\text{Works-for} \sqcap \text{emp} : \neg \text{Manager})]$$

$$\Sigma \models \text{TopManager} \sqsubseteq \Diamond \neg \exists^1 \text{emp}[(\text{Works-for} \sqcap \text{act} : \text{Project})]$$

Note also that if we add to Σ the formula

$$\text{Employee} \sqsubseteq \exists^1 \text{emp}[\text{Works-for}]$$

saying that every employee should work for at least one project, then all the entities and the relations mentioned in the conceptual schema are interpreted as the empty set in every model of Σ , i.e., they are not satisfiable relative to Σ .

The expressivity of the query language can be understood with the following examples:

"Find all people who have worked for only one project"

$$Q(x) : - (\exists^1 \text{emp}[(\Diamond^* \text{Works-for})])(x)$$

"Find all managers whose terminal project has code prj342"

$$Q(x) : - \text{Manager}(x) \wedge \text{Manages}(x, \text{prj342}) \wedge (\Box^+ \neg \text{Manages})(x, y)$$

"Find all projecthoppers—people who never spent more than two consecutive years in a project"

$$Q(x) : - (\Box^* \neg \exists^1 \text{emp}[(\text{Works-for} \sqcap \Diamond^* \text{Works-for})])(x)$$

"Find all people who did not work between two projects"

$$Q(x) : - (\Diamond^* \neg \exists^1 \text{emp}[(\text{Works-for})(x) \wedge (\exists^1 \text{emp}[\text{Works-for}](x) \wedge (\Diamond^* \exists^1 \text{emp}[\text{Works-for}](x))])$$

We now consider the problem of query containment under constraints, where the constraints are expressed by the above exemplified schema Σ . Consider the following queries

$$Q_1(x, y) : - \neg \text{AreaManager}(x) \wedge \text{Manages}(x, z) \wedge \text{Project}(z) \wedge$$

$$\text{Resp-for}(y, z) \wedge \text{Department}(y)$$

$$Q_2(x, y) : - (\Diamond^* \exists^1 \text{prj}[\text{Works-for}](x) \wedge \text{Manages}(x, z) \wedge$$

$$\text{Resp-for}(y, z) \wedge \neg \text{InterestGroup}(y))$$

It is not difficult to see that these two queries are equivalent under the constraints in Σ , i.e., $\Sigma \models Q_1 \subseteq Q_2$ and $\Sigma \models Q_2 \subseteq Q_1$.

3.5. Decidability and complexity. In this section we only summarise the computational behaviour of $\mathcal{DLR}_{\mathcal{L}_{\mathcal{S}}}$ and its fragments over the flow of time $(\mathbb{Z}, <)$. Unfortunately, full $\mathcal{DLR}_{\mathcal{L}_{\mathcal{S}}}$, even restricted to atomic formulas, turns out to be undecidable.

Theorem 8. *The global satisfiability problem for $\mathcal{DLR}_{\mathcal{L}_{\mathcal{S}}}$ conceptual schemas containing only atomic formulas is undecidable.*

Proof. The proof is by reduction of the well-known undecidable tiling problem [Rob71]: given a finite set of square tiles of fixed orientation and with coloured edges, decide whether it can tile the grid $\mathbb{Z} \times \mathbb{N}$. Suppose $T = \{t_1, \dots, t_k\}$ is a

set of tiles with colours $left(T_i)$, $right(T_i)$, and $down(T_i)$. Consider the following schema Σ , where D_1, \dots, D_k are concepts and R is a binary relation:

$$\begin{aligned} R &\doteq \square^+ R, & R &\doteq \diamond^+ R, & \top &\doteq \exists R.T, \\ D_i &\subseteq \neg D_j, & \top &\doteq D_1 \cup \dots \cup D_k, & \text{for } i \neq j, \\ D_i &\subseteq \bigcup_{\text{right}(T_i)=left(T_j)} \text{VAR } D_j, & \text{for } i \leq k, \\ D_i &\subseteq \bigcup_{up(T_i)=down(T_j)} \oplus D_j, & \text{for } i \leq k. \end{aligned}$$

(Here $\exists R.C \doteq \exists z \top [(R \top \exists z/2 : C), \forall R.C \doteq \neg \exists R.\neg C]$). It is readily checked that Σ is globally satisfiable iff T tiles $\mathbb{Z} \times \mathbb{N}$. \square

The main technical reason for undecidability is the possibility of temporalising binary relations. The proof uses a very small fragment of $\mathcal{DLR}_{\text{KUS}}$: even \mathcal{ALC} with \square^+ or one global role is enough to get undecidability. This gives us some grounds to conjecture that already the basic temporal EER model with just snapshot relations is undecidable.

The fragment $\mathcal{DLR}_{\text{KUS}}$, in which the temporal operators can be applied only to entities and formulas, exhibits a much better computational behaviour. In this case we have the following hierarchy:

Theorem 9. *Let the flow of time be $(\mathbb{Z}_n, <)$. Then*

- (1) *the problem of logical implication in $\mathcal{DLR}_{\text{KUS}}$ involving only atomic formulas is EXPTIME-complete;*
- (2) *the formula satisfiability problem (and so the problem of logical implication) in $\mathcal{DLR}_{\text{KUS}}$ is EXPSpace-complete;*
- (3) *the querycontainment problem for nonrecursive Datalog queries under $\mathcal{DLR}_{\text{KUS}}$ -constraints is decidable in 2EXPTIME and is EXPSpace-hard.*

In the remainder of the section we sketch a proof of this theorem. To make it more transparent, we confine ourselves to considering only the future fragment $\mathcal{DLR}_{\text{KUS}}^+$ of $\mathcal{DLR}_{\text{KUS}}$. (From now on \square stands for \square^+ and \bigcirc for \oplus .) The main technical tool in the proof is the method of quasimodels developed in [WolZak198, WolZak199]. The idea behind the notion of a quasimodel is to represent the state of the (in general, infinite) domain of a temporal model at a each moment of time by finitely many types of the domain objects at this moment (modulo a given finite set of formulas); the evolution of types in time is described by special functions called runs.

Suppose that Γ consists of a finite set $f(\Gamma)$ of $\mathcal{DLR}_{\text{KUS}}^+$ -formulas and a finite set $c(\Gamma)$ of concepts, $f(\Gamma)$ is closed under sub-formulas, $c(\Gamma)$ under subconcepts, both are closed under (single) negation, and each concept occurring in $f(\Gamma)$ belongs to $c(\Gamma)$. A *concept type* for Γ is a subset t of $c(\Gamma)$ such that

$$\begin{aligned} C \cap D &\in t \text{ iff } C, D \in t, \text{ for all } C \cap D \in c(\Gamma); \\ \neg C &\in t \text{ iff } C \notin t, \text{ for all } C \in c(\Gamma). \end{aligned}$$

A *formula type* for Γ is a subset Φ of $f(\Gamma)$ such that

$$\begin{aligned} \psi \wedge \chi &\in \Phi \text{ iff } \psi, \chi \in \Phi, \text{ for all } \psi \wedge \chi \in f(\Gamma); \\ \neg \psi &\in \Phi \text{ iff } \psi \notin \Phi, \text{ for all } \psi \in f(\Gamma). \end{aligned}$$

A pair (T, Φ) , where T is a set of concept types and Φ a formula type for Γ , is called a *quasistate candidate* for Γ . We say that the quasistate candidate $\mathcal{C} = (T, \Phi)$ is a *quasistate* for Γ if the following (non-temporal) \mathcal{DLR} -formula ac

$$\left(\bigcup_{t \in T} c(t) \doteq \top \right) \wedge \bigwedge_{t \in T} \neg(c(t) \doteq \perp) \wedge \Phi$$

is satisfiable. Here $c(t)$ denotes the conjunction of all concepts in t , concepts of the form CUD are regarded as atomic concepts $Acud$, and formulas of the form $\forall x(\psi)$ in Φ are regarded as atomic formulas $A\forall x\psi = \top$.

Consider now a sequence of quasistates $Q = (Q(n) : n \in \mathbb{Z})$, where $Q(n) = (T_n, \Phi_n)$. A run in Q is a sequence $r = (r(n) : n \in \mathbb{Z})$ such that

1. $r(n) \in T_n$ for every $n \in \mathbb{Z}$;
2. for every $CUD \in c(\Gamma)$ and every $n \in \mathbb{Z}$, we have $CUD \in r(n)$ iff there is $l > n$ such that $D \in r(l)$ and $C \in r(k)$ for all $k \in (n, l)$;
3. for every $n \in \mathbb{Z}$ and every $t \in T_n$ there is a run r in Q such that $r(n) = t$;
4. for every $\forall x(\psi) \in f(\Gamma)$ and every $n \in \mathbb{Z}$, we have $\forall x(\psi) \in \Phi_n$ iff there is $l > n$ such that $\chi \in \Phi_l$ and $\psi \in \Phi_k$ for all $k \in (n, l)$.

Given a $\mathcal{DLR}_{\text{KUS}}^+$ -formula φ , we denote by $d(\varphi)$ the closure under (single) negation of the set of subformulas and subconcepts of φ .

Theorem 10. *$\mathcal{DLR}_{\text{KUS}}^+$ -formula φ is satisfiable iff there is a quasimodel for $d(\varphi)$ such that $\varphi \in \Phi_0$.*

Proof. Suppose φ is satisfied in a model I with domain Δ . For every $n \in \mathbb{Z}$, define $Q(n) = (T_n, \Phi_n)$ by taking $T_n = \{r^n(x) : x \in \Delta\}$, $\Phi_n = \{\psi \in d(\varphi) : I, n \models \psi\}$, where $r^n(x) = \{C \in d(\varphi) : x \in C^I(n)\}$. It is easy to see that $(Q(n) : n \in \mathbb{Z})$ is a quasimodel for φ . (Note that the sequence $(r^n(x) : n \in \mathbb{Z})$ is a run through $r^n(x)$, for every $n \in \mathbb{Z}$ and every $x \in \Delta$). To show the converse we require the following lemma.

Lemma 11. *For any cardinal $\kappa \geq \aleph_0$ and any quasistate \mathcal{C} for φ , the formula ac is satisfied in a (non-temporal) \mathcal{DLR} -model J in which $|x|^J| = \kappa$ for all x in the domain Δ of J , where $|x|^J = \{y \in \Delta : \forall C \in d(\varphi)(x \in C^J \Leftrightarrow y \in C^J)\}$.*

Proof. As \mathcal{DLR} is a fragment of first-order logic, we have a countable \mathcal{DLR} -model I satisfying ac . Define J as the disjoint union of κ copies of I , more precisely, let

$$\begin{aligned} P_i^J &= \{(x_0, \xi), \dots, (x_n, \xi) : (x_0, \dots, x_n) \in P_i^I, \xi < \kappa\}, \\ (T_n)^J &= \{(x_0, \xi), \dots, (x_n, \xi) : (x_0, \dots, x_n) \in (T_n)^I, \xi < \kappa\}. \end{aligned}$$

It is easy to see that J is as required. \square

Suppose now that $\varphi \in \Phi_0$, for a quasimodel Q . Let κ be a cardinal exceeding the cardinality of the set Ω of all runs in Q and N_0 , and let $\Delta = \{(r, \xi) : r \in \Omega, \xi < \kappa\}$.

Note that $\{(\tau, \xi) \in \Delta : r(n) = t\} = \kappa$, for every $n \in \mathbb{Z}$ and every $t \in T_n$. By Lemma 11, for every $n \in \mathbb{Z}$ there is a \mathcal{DLR} -model $J(n)$ with domain Δ satisfying $\alpha_Q(n)$ and such that $\{C \in \mathcal{C}(\varphi) : (r, \xi) \in C^{J(n)}\} = r(n)$, for all $r \in \Omega$ and $\xi < \kappa$. It is easy to see that the temporal \mathcal{DLR} -model $I = (\mathbb{Z}, \Delta, J(n))$ defined by taking $I(n) = J(n)$, for every $n \in \mathbb{Z}$, satisfies φ at moment 0. \square

Thus, the satisfiability problem for \mathcal{DLR}_U -formulas reduces to checking satisfiability in quasimodels. Consider now a \mathcal{DLR}_U -schema Σ and two queries

$$Q_i(\vec{x}) : - \bigvee_j Q_j(\vec{x}, \vec{y}_j, \vec{\omega}_j), \quad i = 1, 2.$$

Denote by $\mathcal{C}(\Sigma, Q_1, Q_2)$ the closure under (single) negation of the set of all formulas and concepts occurring in Σ , Q_1 and Q_2 . Given a formula or a concept χ , denote by $\bar{\chi}$ the result of replacing all subformulas (subconcepts) in χ of the form $\chi_1 \mathcal{U} \chi_2$ with $A_{\chi_1 \mathcal{U} \chi_2} = \top$ (respectively, $A_{\chi_1 \mathcal{U} \chi_2}$). Thus, $\bar{\chi}$ is a \mathcal{DLR} formula or concept, and the Q_i are non-temporal \mathcal{DLR} -queries.

Theorem 11. Q_1 is not contained in Q_2 relative to Σ iff there is a quasimodel Q for $\mathcal{C}(\Sigma, Q_1, Q_2)$ such that \bar{Q}_1 is not contained in \bar{Q}_2 relative to $\Sigma \cup \{\bar{\alpha}_Q(0)\}$.

Proof. (\Rightarrow) Without loss of generality we may assume that we have a model I such that $I(0) \models \Sigma$ and $\text{ans}(Q_1, I(0)) \not\subseteq \text{ans}(Q_2, I(0))$. Construct a quasimodel Q for $\mathcal{C}(\Sigma, Q_1, Q_2)$ as in the proof of Theorem 10. To show that \bar{Q}_1 is not contained in \bar{Q}_2 relative to $\Sigma \cup \{\bar{\alpha}_Q(0)\}$, it is enough to extend the (non-temporal) model $I(0)$ to the new surrogate atoms of the form $A_{\chi_1 \mathcal{U} \chi_2}$ and $A_{\chi_1 \mathcal{U} \chi_2}$ in accordance with their behaviour in I at time point 0:

$$A_{C_1 \mathcal{U} C_2}^{I(0)} = (C_1 \mathcal{U} C_2)^{I(0)} \quad \text{and} \quad A_{\chi_1 \mathcal{U} \chi_2}^{I(0)} = \begin{cases} \top, & \text{if } I(0) \models \chi_1 \mathcal{U} \chi_2 \\ \perp, & \text{otherwise.} \end{cases}$$

(\Leftarrow) is also proved similarly to Theorem 10. The only difference is that now we select $J(0)$ such that $J(0) \models \Sigma \wedge \{\bar{\alpha}_Q(0)\}$ and $\text{ans}(\bar{Q}_1, J(0)) \not\subseteq \text{ans}(\bar{Q}_2, J(0))$. \square

So, the query-containment problem for \mathcal{DLR}_U reduces to satisfiability in quasimodels and the query-containment problem for (non-temporal) \mathcal{DLR} . The latter problem was shown to be decidable in 2ExpTime time in [Cal et al. 98]. But how to check satisfiability in quasimodels? First of all, we need a procedure deciding whether a quasistate candidate is a quasistate for a given set of formulas and concepts. The following proposition can be proved using the reduction in [Cal et al. 98].

Proposition 1. (i) Given a \mathcal{DLR}_U -formula φ , it is decidable in NExpTime whether a quasistate candidate for $\mathcal{C}(\varphi)$ is a quasistate.

(ii) Given a \mathcal{DLR}_U -schema Σ and queries Q_1, Q_2 , it is decidable in 2ExpTime whether \bar{Q}_1 is contained in \bar{Q}_2 relative to $\Sigma \cup \{\bar{\alpha}_C\}$ for a quasistate candidate \mathcal{C} for $\mathcal{C}(\Sigma, Q_1, Q_2)$.

Now, given a set Γ as defined above, we have at most $O(2^{2^{|\Gamma|}})$ distinct quasistates for Γ . The problem then is whether they can be properly arranged to form a quasimodel for Γ . As we have no past temporal operators, it is enough to consider the flow of time $(\mathbb{N}, <)$ and quasimodels of the form $Q = (Q(n) : n \in \mathbb{N})$.

Let Q be a sequence of quasistates $Q(i) = (T_i, \Phi_i)$, $i \in \mathbb{N}$, and τ a sequence of elements from T_i such that $\tau(i) \in T_i$. Say that τ realises $CUD \in r(n)$ in m steps if there is $l \leq m$ such that $D \in r(n+l)$ and $C \in r(n+k)$ for all $k \in (0, l)$. A formula $\psi \mathcal{U} \chi \in \Phi_n$ is realised in m steps if there is $l \leq m$ such that $\chi \in \Phi_{n+l}$ and $\psi \in \Phi_{n+k}$ for all $k \in (0, l)$. We also say that a pair t, t' of concept types is suitable if for every $CUD \in \Gamma$, $CUD \in t$ iff either $D \in t'$ or $C \in t'$ and $CUD \in t'$.

Suppose Q_1 and Q_2 are finite sequences of quasistates for Γ of length l_1 and l_2 , respectively, and let $Q = Q_1 * Q_2^*$ (i.e., $Q = Q_1 * Q_2 * Q_2 * \dots$) with $Q(n) = (T_n, \Phi_n)$. One can check that Q is a quasimodel for Γ if the following conditions hold:

- (a) for every $i \leq l_1 + l_2$ and every $t' \in T_{i+1}$, there is $t \in T_i$ such that the pair t, t' is suitable;
- (b) for every $i \leq l_1 + 1$ and every $t_i \in T_i$, all concepts of the form $CUD \in t_i$ are realised in $t_i + l_2 - i$ steps in some sequence $t_i, t_{i+1}, \dots, t_{l_1+l_2}$ in which $t_{i+2} \notin T_{i+2}$ and every pair of adjacent elements is suitable;
- (c) for every $i \leq l_1 + l_2$, and every $\psi \mathcal{U} \chi \in \Gamma$, $\psi \mathcal{U} \chi \in \Phi_i$ iff either $\chi \in \Phi_{i+1}$ or $\psi \in \Phi_{i+1}$ and $\psi \mathcal{U} \chi \in \Phi_{i+1}$;
- (d) for every $i \leq l_1 + 1$, all formulas of the form $\psi \mathcal{U} \chi \in \Phi_i$ are realised in $l_1 + l_2 - i$ steps.

Moreover, given a quasimodel for Γ , one can always extract from it a subquasimodel $Q = Q_1 * Q_2^*$ which satisfies (a)-(d) above, all quasistates in Q_i are distinct and $|Q_2| = O(2^{2^{|\Gamma|}})$.

Using this observation together with Proposition 1 one can construct an ExpSpace formula-satisfiability checking algorithm and a 2ExpTime query-containment checking algorithm.

A proof of ExpSpace-hardness of the formula-satisfiability problem we show for a much weaker logic \mathcal{ALCU} .

The primitive symbols of \mathcal{ALCU} are: concept names C_0, C_1, \dots and role names R_0, R_1, \dots . Starting from these we can form compound concepts and formulas using the following constructs. Suppose R is a role name and \bar{C}, D are concepts (for the basis of our inductive definition we assume concept names to be atomic concepts). Then $\top, C \sqcap D, \neg C, \exists R.C$, and CUD are concepts. Atomic formulas are expressions of the form \top , and $C = D$. If φ and ψ are formulas then so are $\varphi \wedge \psi$, $\neg \varphi$, and $\psi \mathcal{U} \varphi$.⁴

Lemma 12. Let the flow of time be $N = (\mathbb{N}, <)$.⁵ Then the formula satisfiability problem in \mathcal{ALCU} is ExpSpace-hard.

⁴Language \mathcal{ALCU} is a fragment of \mathcal{DLR}_U since $\exists R.C$ is abbreviation for $\exists x [1][R][x] (C)$.

⁵The class of models based on $(\mathbb{N}, <)$.

In the proof we will use abbreviations (having in mind that we do not use S)
 $\Box^* \psi = \psi \wedge \Box \psi$, $\Diamond^* \psi = \psi \vee \Diamond \psi$ for formula ψ , and for a concept C abbreviations
 $\Box^* C = C \cap \Box C$, $\Diamond^* C = C \cup \Diamond C$; $\text{VR}_i C = \neg \exists R_i . \neg C$.

Proof. We will show here the lower bound for the satisfaction problem in \mathcal{N} by reducing to it the n -CORRIDOR tiling problem, n given in binary, which is known to be EXPSpace-complete [Boasson]. Namely, for a set $T = \{t_1, \dots, t_s\}$ of tiles and $n < \omega$, we construct an ALC_{ij} -formula φ of length $O(n^2 + s^2)$ such that φ is satisfied iff T tiles $2^n \times m$ rectangle, for some $m < \omega$ in such a way that sides of this rectangle are, say, white.

To encode the 2^n column, we define 2^n concepts B_j , $0 \leq j < 2^n$, using n concept names C_0, \dots, C_{n-1} and a role name R . Let ψ be the conjunction of the following formulas:

$$\begin{aligned} \exists R.T &= T, \quad \neg(\neg C_0 \cap \dots \cap \neg C_{n-1}) = \perp, \\ \bigwedge_{i=0}^{n-1} \left(\bigwedge_{j=0}^{i-1} C_j \rightarrow (C_i \rightarrow \text{VR}_i \neg C_i) \cap (\neg C_i \rightarrow \text{VR}_i C_i) = T \right), \\ \bigwedge_{i=0}^{n-1} \left(\bigwedge_{j=0}^{i-1} \neg C_j \rightarrow (C_i \rightarrow \text{VR}_i C_i) \cap (\neg C_i \rightarrow \text{VR}_i \neg C_i) = T \right). \end{aligned}$$

For any $j \in \{0, \dots, 2^n - 1\}$ written in binary as (d_{n-1}, \dots, d_0) , we put $B_j = C_0^{d_0} \cap \dots \cap C_{n-1}^{d_{n-1}}$, where C^d is C if $d = 1$ and $\neg C$ otherwise. If ψ_1 is satisfied in an ALC -model, then the sets B_j in this model are nonempty, pairwise disjoint and cover the domain of the model.

Let B, Q_0, \dots, Q_{n-1} be $(n+1)$ new concept names. We use them to encode 2^n sets $[j]$ of worlds containing all w_j for which $w_j \models Q_0^d \cap \dots \cap Q_{n-1}^{d_{n-1}} = T$ and (d_{n-1}, \dots, d_0) is binary representations of j . B will coincide with B_j in the all worlds from $[j]$. This is ensured by the formula ψ_2 :

$$\begin{aligned} \bigwedge_{i=0}^{n-1} (\Diamond^* C_i = \Box^* C_i) \wedge \left(B = \bigcap_{i=0}^{n-1} ((C_i \cap Q_i) \cup (\neg C_i \cap \neg Q_i)) \right) \wedge \Box^* (\Diamond^* B = T) \\ \wedge \Box^* \left(\bigwedge_{i=0}^{n-1} ((Q_i = T) \vee (Q_i = \perp)) \right) \wedge (Z = T) \wedge \Box^* (Z = \neg Q_0 \cap \dots \cap \neg Q_{n-1}). \end{aligned}$$

Let $w_0^d < w_1^d < \dots$ be the ordering of worlds in $[0]$. Worlds $w \in [j]$ such that $w_0^d < w < w_{i+1}^d$ will be denoted by w_j^i . For example, we may have:

$$w_0^0 < w_0^1 < w_1^0 < w_1^1 < w_2^0 < w_2^1 < w_3^0 < w_3^1 < w_4^0 < w_4^1 < w_5^0 < w_5^1 < w_6^0 < w_6^1 < w_7^0 < w_7^1 < w_8^0 < w_8^1 < w_9^0 < w_9^1 < w_{10}^0 < w_{10}^1 < w_{11}^0 < w_{11}^1 < w_{12}^0 < w_{12}^1 < w_{13}^0 < w_{13}^1 < w_{14}^0 < w_{14}^1 < w_{15}^0 < w_{15}^1 < w_{16}^0 < w_{16}^1 < w_{17}^0 < w_{17}^1 < w_{18}^0 < w_{18}^1 < w_{19}^0 < w_{19}^1 < w_{20}^0 < w_{20}^1 < w_{21}^0 < w_{21}^1 < w_{22}^0 < w_{22}^1 < w_{23}^0 < w_{23}^1 < w_{24}^0 < w_{24}^1 < w_{25}^0 < w_{25}^1 < w_{26}^0 < w_{26}^1 < w_{27}^0 < w_{27}^1 < w_{28}^0 < w_{28}^1 < w_{29}^0 < w_{29}^1 < w_{30}^0 < w_{30}^1 < w_{31}^0 < w_{31}^1 < w_{32}^0 < w_{32}^1 < w_{33}^0 < w_{33}^1 < w_{34}^0 < w_{34}^1 < w_{35}^0 < w_{35}^1 < w_{36}^0 < w_{36}^1 < w_{37}^0 < w_{37}^1 < w_{38}^0 < w_{38}^1 < w_{39}^0 < w_{39}^1 < w_{40}^0 < w_{40}^1 < w_{41}^0 < w_{41}^1 < w_{42}^0 < w_{42}^1 < w_{43}^0 < w_{43}^1 < w_{44}^0 < w_{44}^1 < w_{45}^0 < w_{45}^1 < w_{46}^0 < w_{46}^1 < w_{47}^0 < w_{47}^1 < w_{48}^0 < w_{48}^1 < w_{49}^0 < w_{49}^1 < w_{50}^0 < w_{50}^1 < w_{51}^0 < w_{51}^1 < w_{52}^0 < w_{52}^1 < w_{53}^0 < w_{53}^1 < w_{54}^0 < w_{54}^1 < w_{55}^0 < w_{55}^1 < w_{56}^0 < w_{56}^1 < w_{57}^0 < w_{57}^1 < w_{58}^0 < w_{58}^1 < w_{59}^0 < w_{59}^1 < w_{60}^0 < w_{60}^1 < w_{61}^0 < w_{61}^1 < w_{62}^0 < w_{62}^1 < w_{63}^0 < w_{63}^1 < w_{64}^0 < w_{64}^1 < w_{65}^0 < w_{65}^1 < w_{66}^0 < w_{66}^1 < w_{67}^0 < w_{67}^1 < w_{68}^0 < w_{68}^1 < w_{69}^0 < w_{69}^1 < w_{70}^0 < w_{70}^1 < w_{71}^0 < w_{71}^1 < w_{72}^0 < w_{72}^1 < w_{73}^0 < w_{73}^1 < w_{74}^0 < w_{74}^1 < w_{75}^0 < w_{75}^1 < w_{76}^0 < w_{76}^1 < w_{77}^0 < w_{77}^1 < w_{78}^0 < w_{78}^1 < w_{79}^0 < w_{79}^1 < w_{80}^0 < w_{80}^1 < w_{81}^0 < w_{81}^1 < w_{82}^0 < w_{82}^1 < w_{83}^0 < w_{83}^1 < w_{84}^0 < w_{84}^1 < w_{85}^0 < w_{85}^1 < w_{86}^0 < w_{86}^1 < w_{87}^0 < w_{87}^1 < w_{88}^0 < w_{88}^1 < w_{89}^0 < w_{89}^1 < w_{90}^0 < w_{90}^1 < w_{91}^0 < w_{91}^1 < w_{92}^0 < w_{92}^1 < w_{93}^0 < w_{93}^1 < w_{94}^0 < w_{94}^1 < w_{95}^0 < w_{95}^1 < w_{96}^0 < w_{96}^1 < w_{97}^0 < w_{97}^1 < w_{98}^0 < w_{98}^1 < w_{99}^0 < w_{99}^1 < w_{100}^0 < w_{100}^1 < w_{101}^0 < w_{101}^1 < w_{102}^0 < w_{102}^1 < w_{103}^0 < w_{103}^1 < w_{104}^0 < w_{104}^1 < w_{105}^0 < w_{105}^1 < w_{106}^0 < w_{106}^1 < w_{107}^0 < w_{107}^1 < w_{108}^0 < w_{108}^1 < w_{109}^0 < w_{109}^1 < w_{110}^0 < w_{110}^1 < w_{111}^0 < w_{111}^1 < w_{112}^0 < w_{112}^1 < w_{113}^0 < w_{113}^1 < w_{114}^0 < w_{114}^1 < w_{115}^0 < w_{115}^1 < w_{116}^0 < w_{116}^1 < w_{117}^0 < w_{117}^1 < w_{118}^0 < w_{118}^1 < w_{119}^0 < w_{119}^1 < w_{120}^0 < w_{120}^1 < w_{121}^0 < w_{121}^1 < w_{122}^0 < w_{122}^1 < w_{123}^0 < w_{123}^1 < w_{124}^0 < w_{124}^1 < w_{125}^0 < w_{125}^1 < w_{126}^0 < w_{126}^1 < w_{127}^0 < w_{127}^1 < w_{128}^0 < w_{128}^1 < w_{129}^0 < w_{129}^1 < w_{130}^0 < w_{130}^1 < w_{131}^0 < w_{131}^1 < w_{132}^0 < w_{132}^1 < w_{133}^0 < w_{133}^1 < w_{134}^0 < w_{134}^1 < w_{135}^0 < w_{135}^1 < w_{136}^0 < w_{136}^1 < w_{137}^0 < w_{137}^1 < w_{138}^0 < w_{138}^1 < w_{139}^0 < w_{139}^1 < w_{140}^0 < w_{140}^1 < w_{141}^0 < w_{141}^1 < w_{142}^0 < w_{142}^1 < w_{143}^0 < w_{143}^1 < w_{144}^0 < w_{144}^1 < w_{145}^0 < w_{145}^1 < w_{146}^0 < w_{146}^1 < w_{147}^0 < w_{147}^1 < w_{148}^0 < w_{148}^1 < w_{149}^0 < w_{149}^1 < w_{150}^0 < w_{150}^1 < w_{151}^0 < w_{151}^1 < w_{152}^0 < w_{152}^1 < w_{153}^0 < w_{153}^1 < w_{154}^0 < w_{154}^1 < w_{155}^0 < w_{155}^1 < w_{156}^0 < w_{156}^1 < w_{157}^0 < w_{157}^1 < w_{158}^0 < w_{158}^1 < w_{159}^0 < w_{159}^1 < w_{160}^0 < w_{160}^1 < w_{161}^0 < w_{161}^1 < w_{162}^0 < w_{162}^1 < w_{163}^0 < w_{163}^1 < w_{164}^0 < w_{164}^1 < w_{165}^0 < w_{165}^1 < w_{166}^0 < w_{166}^1 < w_{167}^0 < w_{167}^1 < w_{168}^0 < w_{168}^1 < w_{169}^0 < w_{169}^1 < w_{170}^0 < w_{170}^1 < w_{171}^0 < w_{171}^1 < w_{172}^0 < w_{172}^1 < w_{173}^0 < w_{173}^1 < w_{174}^0 < w_{174}^1 < w_{175}^0 < w_{175}^1 < w_{176}^0 < w_{176}^1 < w_{177}^0 < w_{177}^1 < w_{178}^0 < w_{178}^1 < w_{179}^0 < w_{179}^1 < w_{180}^0 < w_{180}^1 < w_{181}^0 < w_{181}^1 < w_{182}^0 < w_{182}^1 < w_{183}^0 < w_{183}^1 < w_{184}^0 < w_{184}^1 < w_{185}^0 < w_{185}^1 < w_{186}^0 < w_{186}^1 < w_{187}^0 < w_{187}^1 < w_{188}^0 < w_{188}^1 < w_{189}^0 < w_{189}^1 < w_{190}^0 < w_{190}^1 < w_{191}^0 < w_{191}^1 < w_{192}^0 < w_{192}^1 < w_{193}^0 < w_{193}^1 < w_{194}^0 < w_{194}^1 < w_{195}^0 < w_{195}^1 < w_{196}^0 < w_{196}^1 < w_{197}^0 < w_{197}^1 < w_{198}^0 < w_{198}^1 < w_{199}^0 < w_{199}^1 < w_{200}^0 < w_{200}^1 < w_{201}^0 < w_{201}^1 < w_{202}^0 < w_{202}^1 < w_{203}^0 < w_{203}^1 < w_{204}^0 < w_{204}^1 < w_{205}^0 < w_{205}^1 < w_{206}^0 < w_{206}^1 < w_{207}^0 < w_{207}^1 < w_{208}^0 < w_{208}^1 < w_{209}^0 < w_{209}^1 < w_{210}^0 < w_{210}^1 < w_{211}^0 < w_{211}^1 < w_{212}^0 < w_{212}^1 < w_{213}^0 < w_{213}^1 < w_{214}^0 < w_{214}^1 < w_{215}^0 < w_{215}^1 < w_{216}^0 < w_{216}^1 < w_{217}^0 < w_{217}^1 < w_{218}^0 < w_{218}^1 < w_{219}^0 < w_{219}^1 < w_{220}^0 < w_{220}^1 < w_{221}^0 < w_{221}^1 < w_{222}^0 < w_{222}^1 < w_{223}^0 < w_{223}^1 < w_{224}^0 < w_{224}^1 < w_{225}^0 < w_{225}^1 < w_{226}^0 < w_{226}^1 < w_{227}^0 < w_{227}^1 < w_{228}^0 < w_{228}^1 < w_{229}^0 < w_{229}^1 < w_{230}^0 < w_{230}^1 < w_{231}^0 < w_{231}^1 < w_{232}^0 < w_{232}^1 < w_{233}^0 < w_{233}^1 < w_{234}^0 < w_{234}^1 < w_{235}^0 < w_{235}^1 < w_{236}^0 < w_{236}^1 < w_{237}^0 < w_{237}^1 < w_{238}^0 < w_{238}^1 < w_{239}^0 < w_{239}^1 < w_{240}^0 < w_{240}^1 < w_{241}^0 < w_{241}^1 < w_{242}^0 < w_{242}^1 < w_{243}^0 < w_{243}^1 < w_{244}^0 < w_{244}^1 < w_{245}^0 < w_{245}^1 < w_{246}^0 < w_{246}^1 < w_{247}^0 < w_{247}^1 < w_{248}^0 < w_{248}^1 < w_{249}^0 < w_{249}^1 < w_{250}^0 < w_{250}^1 < w_{251}^0 < w_{251}^1 < w_{252}^0 < w_{252}^1 < w_{253}^0 < w_{253}^1 < w_{254}^0 < w_{254}^1 < w_{255}^0 < w_{255}^1 < w_{256}^0 < w_{256}^1 < w_{257}^0 < w_{257}^1 < w_{258}^0 < w_{258}^1 < w_{259}^0 < w_{259}^1 < w_{260}^0 < w_{260}^1 < w_{261}^0 < w_{261}^1 < w_{262}^0 < w_{262}^1 < w_{263}^0 < w_{263}^1 < w_{264}^0 < w_{264}^1 < w_{265}^0 < w_{265}^1 < w_{266}^0 < w_{266}^1 < w_{267}^0 < w_{267}^1 < w_{268}^0 < w_{268}^1 < w_{269}^0 < w_{269}^1 < w_{270}^0 < w_{270}^1 < w_{271}^0 < w_{271}^1 < w_{272}^0 < w_{272}^1 < w_{273}^0 < w_{273}^1 < w_{274}^0 < w_{274}^1 < w_{275}^0 < w_{275}^1 < w_{276}^0 < w_{276}^1 < w_{277}^0 < w_{277}^1 < w_{278}^0 < w_{278}^1 < w_{279}^0 < w_{279}^1 < w_{280}^0 < w_{280}^1 < w_{281}^0 < w_{281}^1 < w_{282}^0 < w_{282}^1 < w_{283}^0 < w_{283}^1 < w_{284}^0 < w_{284}^1 < w_{285}^0 < w_{285}^1 < w_{286}^0 < w_{286}^1 < w_{287}^0 < w_{287}^1 < w_{288}^0 < w_{288}^1 < w_{289}^0 < w_{289}^1 < w_{290}^0 < w_{290}^1 < w_{291}^0 < w_{291}^1 < w_{292}^0 < w_{292}^1 < w_{293}^0 < w_{293}^1 < w_{294}^0 < w_{294}^1 < w_{295}^0 < w_{295}^1 < w_{296}^0 < w_{296}^1 < w_{297}^0 < w_{297}^1 < w_{298}^0 < w_{298}^1 < w_{299}^0 < w_{299}^1 < w_{300}^0 < w_{300}^1 < w_{301}^0 < w_{301}^1 < w_{302}^0 < w_{302}^1 < w_{303}^0 < w_{303}^1 < w_{304}^0 < w_{304}^1 < w_{305}^0 < w_{305}^1 < w_{306}^0 < w_{306}^1 < w_{307}^0 < w_{307}^1 < w_{308}^0 < w_{308}^1 < w_{309}^0 < w_{309}^1 < w_{310}^0 < w_{310}^1 < w_{311}^0 < w_{311}^1 < w_{312}^0 < w_{312}^1 < w_{313}^0 < w_{313}^1 < w_{314}^0 < w_{314}^1 < w_{315}^0 < w_{315}^1 < w_{316}^0 < w_{316}^1 < w_{317}^0 < w_{317}^1 < w_{318}^0 < w_{318}^1 < w_{319}^0 < w_{319}^1 < w_{320}^0 < w_{320}^1 < w_{321}^0 < w_{321}^1 < w_{322}^0 < w_{322}^1 < w_{323}^0 < w_{323}^1 < w_{324}^0 < w_{324}^1 < w_{325}^0 < w_{325}^1 < w_{326}^0 < w_{326}^1 < w_{327}^0 < w_{327}^1 < w_{328}^0 < w_{328}^1 < w_{329}^0 < w_{329}^1 < w_{330}^0 < w_{330}^1 < w_{331}^0 < w_{331}^1 < w_{332}^0 < w_{332}^1 < w_{333}^0 < w_{333}^1 < w_{334}^0 < w_{334}^1 < w_{335}^0 < w_{335}^1 < w_{336}^0 < w_{336}^1 < w_{337}^0 < w_{337}^1 < w_{338}^0 < w_{338}^1 < w_{339}^0 < w_{339}^1 < w_{340}^0 < w_{340}^1 < w_{341}^0 < w_{341}^1 < w_{342}^0 < w_{342}^1 < w_{343}^0 < w_{343}^1 < w_{344}^0 < w_{344}^1 < w_{345}^0 < w_{345}^1 < w_{346}^0 < w_{346}^1 < w_{347}^0 < w_{347}^1 < w_{348}^0 < w_{348}^1 < w_{349}^0 < w_{349}^1 < w_{350}^0 < w_{350}^1 < w_{351}^0 < w_{351}^1 < w_{352}^0 < w_{352}^1 < w_{353}^0 < w_{353}^1 < w_{354}^0 < w_{354}^1 < w_{355}^0 < w_{355}^1 < w_{356}^0 < w_{356}^1 < w_{357}^0 < w_{357}^1 < w_{358}^0 < w_{358}^1 < w_{359}^0 < w_{359}^1 < w_{360}^0 < w_{360}^1 < w_{361}^0 < w_{361}^1 < w_{362}^0 < w_{362}^1 < w_{363}^0 < w_{363}^1 < w_{364}^0 < w_{364}^1 < w_{365}^0 < w_{365}^1 < w_{366}^0 < w_{366}^1 < w_{367}^0 < w_{367}^1 < w_{368}^0 < w_{368}^1 < w_{369}^0 < w_{369}^1 < w_{370}^0 < w_{370}^1 < w_{371}^0 < w_{371}^1 < w_{372}^0 < w_{372}^1 < w_{373}^0 < w_{373}^1 < w_{374}^0 < w_{374}^1 < w_{375}^0 < w_{375}^1 < w_{376}^0 < w_{376}^1 < w_{377}^0 < w_{377}^1 < w_{378}^0 < w_{378}^1 < w_{379}^0 < w_{379}^1 < w_{380}^0 < w_{380}^1 < w_{381}^0 < w_{381}^1 < w_{382}^0 < w_{382}^1 < w_{383}^0 < w_{383}^1 < w_{384}^0 < w_{384}^1 < w_{385}^0 < w_{385}^1 < w_{386}^0 < w_{386}^1 < w_{387}^0 < w_{387}^1 < w_{388}^0 < w_{388}^1 < w_{389}^0 < w_{389}^1 < w_{390}^0 < w_{390}^1 < w_{391}^0 < w_{391}^1 < w_{392}^0 < w_{392}^1 < w_{393}^0 < w_{393}^1 < w_{394}^0 < w_{394}^1 < w_{395}^0 < w_{395}^1 < w_{396}^0 < w_{396}^1 < w_{397}^0 < w_{397}^1 < w_{398}^0 < w_{398}^1 < w_{399}^0 < w_{399}^1 < w_{400}^0 < w_{400}^1 < w_{401}^0 < w_{401}^1 < w_{402}^0 < w_{402}^1 < w_{403}^0 < w_{403}^1 < w_{404}^0 < w_{404}^1 < w_{405}^0 < w_{405}^1 < w_{406}^0 < w_{406}^1 < w_{407}^0 < w_{407}^1 < w_{408}^0 < w_{408}^1 < w_{409}^0 < w_{409}^1 < w_{410}^0 < w_{410}^1 < w_{411}^0 < w_{411}^1 < w_{412}^0 < w_{412}^1 < w_{413}^0 < w_{413}^1 < w_{414}^0 < w_{414}^1 < w_{415}^0 < w_{415}^1 < w_{416}^0 < w_{416}^1 < w_{417}^0 < w_{417}^1 < w_{418}^0 < w_{418}^1 < w_{419}^0 < w_{419}^1 < w_{420}^0 < w_{420}^1 < w_{421}^0 < w_{421}^1 < w_{422}^0 < w_{422}^1 < w_{423}^0 < w_{423}^1 < w_{424}^0 < w_{424}^1 < w_{425}^0 < w_{425}^1 < w_{426}^0 < w_{426}^1 < w_{427}^0 < w_{427}^1 < w_{428}^0 < w_{428}^1 < w_{429}^0 < w_{429}^1 < w_{430}^0 < w_{430}^1 < w_{431}^0 < w_{431}^1 < w_{432}^0 < w_{432}^1 < w_{433}^0 < w_{433}^1 < w_{434}^0 < w_{434}^1 < w_{435}^0 < w_{435}^1 < w_{436}^0 < w_{436}^1 < w_{437}^0 < w_{437}^1 < w_{438}^0 < w_{438}^1 < w_{439}^0 < w_{439}^1 < w_{440}^0 < w_{440}^1 < w_{441}^0 < w_{441}^1 < w_{442}^0 < w_{442}^1 < w_{443}^0 < w_{443}^1 < w_{444}^0 < w_{444}^1 < w_{445}^0 < w_{445}^1 < w_{446}^0 < w_{446}^1 < w_{447}^0 < w_{447}^1 < w_{448}^0 < w_{448}^1 < w_{449}^0 < w_{449}^1 < w_{450}^0 < w_{450}^1 < w_{451}^0 < w_{451}^1 < w_{452}^0 < w_{452}^1 < w_{453}^0 < w_{453}^1 < w_{454}^0 < w_{454}^1 < w_{455}^0 < w_{455}^1 < w_{456}^0 < w_{456}^1 < w_{457}^0 < w_{457}^1 < w_{458}^0 < w_{458}^1 < w_{459}^0 < w_{459}^1 < w_{460}^0 < w_{460}^1 < w_{461}^0 < w_{461}^1 < w_{462}^0 < w_{462}^1 < w_{463}^0 < w_{463}^1 < w_{464}^0 < w_{464}^1 < w_{465}^0 < w_{465}^1 < w_{466}^0 < w_{466}^1 < w_{467}^0 < w_{467}^1 < w_{468}^0 < w_{468}^1 < w_{469}^0 < w_{469}^1 < w_{470}^0 < w_{470}^1 < w_{471}^0 < w_{471}^1 < w_{472}^0 < w_{472}^1 < w_{473}^0 < w_{473}^1 < w_{474}^0 < w_{474}^1 < w_{475}^0 < w_{475}^1 < w_{476}^0 < w_{476}^1 < w_{477}^0 < w_{477}^1 < w_{478}^0 < w_{478}^1 < w_{479}^0 < w_{479}^1 < w_{480}^0 < w_{480}^1 < w_{481}^0 < w_{481}^1 < w_{482}^0 < w_{482}^1 < w_{483}^0 < w_{483}^1 < w_{484}^0 < w_{484}^1 < w_{485}^0 < w_{485}^1 < w_{486}^0 < w_{486}^1 < w_{487}^0 < w_{487}^1 < w_{488}^0 < w_{488}^1 < w_{489}^0 < w_{489}^1 < w_{490}^0 < w_{490}^1 < w_{491}^0 < w_{491}^1 < w_{492}^0 < w_{492}^1 < w_{493}^0 < w_{493}^1 < w_{494}^0 < w_{494}^1 < w_{495}^0 < w_{495}^1 < w_{496}^0 < w_{496}^1 < w_{497}^0 < w_{4$$

$$\psi_{10} = \square^* \left(\bigwedge_{j=1}^n (W \cap A \subseteq T_j \rightarrow W \cap A^u \subseteq \bigsqcup_{\text{up}(j)=\text{down}(i)} T_i) \right).$$

One can show that $\varphi = \psi_1 \wedge \dots \wedge \psi_{10}$ is as required. \square

It follows, in particular, that the query-containment problem is ExpSpace-hard as well. It is an open problem, however, whether there exists an ExpSpace algorithm deciding this problem.

Finally, we show $\text{ExpTime-completeness}$ of the logical implication for atomic formulas in \mathcal{DLR}_U by means of a polynomial reduction of \mathcal{DLR}_U^* to the logic \mathcal{DLR}_{reg} of [Cal et al. 98]. For our purposes, it is enough to know that \mathcal{DLR}_{reg} allows one to form the transitive closure R^* of every binary relation R , and that the satisfiability problem in \mathcal{DLR}_{reg} is in ExpTime . To simplify the presentation, we reduce here the fragment $\mathcal{DLR}_{\square\bigcirc}$ of \mathcal{DLR}_U with the temporal operators \square and \bigcirc only (the reader should not have problems to extend this reduction to the language with \sqcup).

Fix a binary relation R and define a translation $*$ from $\mathcal{DLR}_{\square\bigcirc}^*$ to \mathcal{DLR}_{reg} as follows: $P^* = P$ for every atom P of \mathcal{DLR} , $(\bigcirc C)^* = \text{VR}.C^*$ and $(\square C)^* = \text{VR}.C^*$; $*$ commutes with the remaining constructs, and $(P_1 \sqsubseteq P_2)^* = P_1^* \sqsubseteq P_2^*$.

Lemma 13. Suppose that $\Gamma \cup \{\varphi\}$ is a set of atomic $\mathcal{DLR}_{\square\bigcirc}^*$ -formulas and that R does not occur in $\Gamma \cup \{\varphi\}$. Then $\Gamma \models \varphi$ iff φ^* is a logical consequence of the following set Ξ of \mathcal{DLR}_{reg} -formulas

$$\Gamma^*, \Xi \models R.T \models \top, \Xi \models R^-.T \models \top,$$

where $\Xi \models R^-.C \models \Xi \models [2](R \cap 1/2 : C)$.

Proof. Suppose $\Gamma \not\models \varphi$. Then there is a model I such that $I, 0 \models \varphi$, but $I, n \not\models \Gamma$ for all $n \in \mathbb{Z}$. Define a \mathcal{DLR} -model $J = \langle \Delta, P_1^j, \dots, R^j \rangle$ by taking $\Delta' = \Delta^j \times \mathbb{Z}$,

$$\langle \langle x_1, n_1 \rangle, \dots, \langle x_i, n_i \rangle \rangle \in P_i^j \text{ iff } n_i = n_j, \text{ for } i, j \leq i, \text{ and } \{x_1, \dots, x_n\} \in P_i^{f(n)}$$

$$\langle \langle x_1, n_1 \rangle, \langle x_2, n_2 \rangle \rangle \in R^j \text{ iff } x_1 = x_2 \text{ and } n_2 = n_1 + 1.$$

It is readily checked that $J \models \Xi$ and $J \not\models \varphi^*$.

Conversely, suppose that $J = \langle \Delta, P_1^j, \dots, R^j \rangle$ is a model such that $J \models \Xi$ but $J \not\models \varphi^*$. Let $\Sigma = \bigcup \{cl(x) : x \in \Gamma \cup \{\varphi\}\}$ and, for every $x \in \Delta$,

$$l(x) = \{C \in c(\Sigma) : x \in (C^*)^j\}.$$

Then the pair (T, Φ) , where $T = \{x \in \Delta : x \in \Delta\}$ and $\Phi = \{x \in f(\Sigma) : J \models x^*\}$, is a quasistate for Σ . Define a map Q by taking $Q(n) = (T, \Phi)$ for all $n \in \mathbb{Z}$. It is easy to see that Q is a quasimodel. Hence, by Theorem 10, we have a model I such that $I \models \Gamma$ but $I, 0 \not\models \varphi$. \square

4. Conclusion

DLs are a family of knowledge representation languages constructed for a wide area of application domains. This paper presents one type of expressive description logic \mathcal{DLR}_{US} , which has been modeled with an aim to overcome problems

of reasoning over conceptual schemas and queries in temporal databases. It is a special type of description logic extended with modal operators. \mathcal{DLR}_{US} is a \mathcal{DLR} description logic with a temporal dimension.

\mathcal{DLR} have been used in the area of non-temporal databases to characterize in a uniform way both conceptual modeling and queries [LevRous98, Cal et al. 98]. Some of interesting properties of \mathcal{DLR} logic are [AriaFra01]:

- allows the logical reconstruction and the extension of data and knowledge; representational tools;
- has an ability to completely define entities and relations as \mathcal{DLR} views over other entities and relation over conceptual schemas
- can express a large class of integrity constraints
- enables a view-based query answering.

Its combination with the propositional temporal logic, enabled with operators *Since* and *Until* [SisC85, Gah et al. 94] resulted in a \mathcal{DLR}_{US} . \mathcal{DLR}_{US} allowed using temporal operators to all syntactical terms of \mathcal{DLR} : entities, relations and schemas.

In this paper we presented the syntax and the semantics of \mathcal{DLR}_{US} as well as the solution of the query containment task problem. An example of conceptual schema and query is given. At the end we summarize the computational behavior of \mathcal{DLR}_{US} and its fragments over the flow of time.

References

- [Abi et al. 96] S. Abiteboul, L. Herr, J. Van den Bussche, *Temporal versus firstorder logic to query temporal databases*, in: *Proc. 15th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'96)*, 1996, pp. 49–57.
- [AriaFra01] A. Ariaie, E. Franconi, *A survey of temporal extensions of description logics*, *Ann. Math. Art. Intell.* 30(14) (2001).
- [Aria et al. 01] A. Ariaie, E. Franconi, M. Mosurović, F. Wolter, M. Zakharyashev, *The $\mathcal{DLR}(US)$ temporal description logic*, in: Stanford, D. McGuinness, P. Patel-Schneider, C. Goble, R. Moeller (eds), *Proc. Internat. Workshop on Description Logic (DL 2001)*, pp. 96–105.
- [Aria et al. 02] A. Ariaie, E. Franconi, F. Wolter, M. Zakharyashev, *A temporal description logic for reasoning over conceptual schemas and queries*, in: S. Flesca, S. Greco, N. Leone, G. Ianni (eds), *Proc. JELIA 02*, Lect. Notes Comput. Sci. 2424, Springer-Verlag, 2002, pp. 98–110.
- [Baas91] F. Baader, *Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles*, in: *Proc. 12th Int. Joint Conf. Artif. Intell. (IJCAI'91)*, 1991.
- [Baas95] F. Baader, A. Laus, *Terminological logics with modal operators*, in: *Proc. 14th Int. Joint Conf. Artif. Intell. (IJCAI'95)*, Morgan Kaufman, 808–814.
- [Baa et al. 02] F. Baader, D. L. McGuinness, D. N. Peter, F. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge Univ. Press, 2002.
- [Baas96] F. Baader, M. Buchheit, B. Hollunder, *Cardinality restrictions on concepts*, *Artif. Intell.* 88(12) (1996), 195–213.
- [BaaOh93] F. Baader and H. Ohlbach, *A multi-dimensional terminological knowledge representation language*, in: *Proc. 13th Int. Joint Conf. Artif. Intell.*, 1993, pp. 690–695.
- [Baas96] F. Baader, U. Sattler, *Number restrictions on complex roles in description logics: A preliminary report*, in: *Proc. 5th Int. Conf. Principles of Knowledge Representation and Reasoning (KR'96)*, 1996, pp. 328–338.
- [Baas99] F. Baader, U. Sattler, *Expressive number restrictions in description logics*, *J. Logic Comput.* 9(3) (1999), 319–350.

- [Boas96] van Boas Boas, *The convenience of things*, Technical Report CT-96-01, Institute for Logic, Language and Computation, University of Amsterdam.
- [BorPas94] R. J. Brachman, H. J. Levesque, *The tractability of subsumption in frame-based description languages*, in: *Proc. 4th Nat. Conf. Artif. Intell. (NAAI'94)*, 1994, pp. 34-37.
- [Boe96] A. Borgi, *On the relative expressiveness of description logics and predicate logics*, *Artif. Intell.* 82(1/2) (1996), 353-367.
- [Bra77] R. J. Brachman, *What's in a concept: Structural foundations for semantic networks*, *Int. J. Man-Machine Studies* 9(2) (1977), 127-152.
- [Bra79] R. J. Brachman, *Structured inheritance networks*, in: W. A. Woods and R. J. Brachman, editors, *Research in Natural Language Understanding*, Quarterly Progress Report No. 1, BBN Report No. 3742, pp. 36-78, Bolt, Beranek and Newman, Cambridge, Massachusetts, 1978.
- [BrasSch85] R. J. Brachman, J. G. Schmolze, *An overview of the KL-ONE knowledge representation system*, *Cognitive Sci.* 9(2) (1985), 171-216.
- [Buc et al. 93] M. Buchheit, F. M. Donini, A. Schaefer, *Decidable reasoning in terminological knowledge representation systems*, *J. Artif. Intell. Research* 1 (1993), 109-138.
- [Ca et al. 98] D. Calvanese, G. De Giacomo, M. Lenzerini, *On the decidability of query containment under constraints*, in: *Proc. 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pp. 149-158.
- [Choa94] J. Chomicki, *Temporal query languages: a survey*, in: *Proc. 1st Internat. Conf. Temporal Logic (ICTL'94)*, pp. 506-534 (1994).
- [ChoaSa98] J. Chomicki, G. Saake, editors, *Logics for Databases and Information Systems*, Kluwer, 1998.
- [DeG85] G. De Giacomo, *Decidability of Class-Based Knowledge Representation Formalisms*, PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza" (1985).
- [Don et al. 97] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, *The complexity of concept languages*, *Inform. Comput.* 134 (1997), 1-58.
- [DomMa90] F. M. Donini, F. Massacci, *EXPTIME tableaux for ALC*, *Artif. Intell.* 124(1) (2000), 87-138.
- [FisLad79] M. J. Fischer, R. E. Ladner, *Propositional dynamic logic of regular programs*, *J. Comput. System Sci.* 18 (1979), 194-211.
- [Fri93] M. Fitting, *Basic model logic*, in: *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pp. 365-448, Oxford Science Publications (1993).
- [Gab72] D. M. Gabbay, *Crucial interpolation theorem for modal logics*, in: *Conference in Mathematical Logic, London 70*, Lect. Notes Math. 235, Springer-Verlag, 1972.
- [Gab et al. 94] D. M. Gabbay, J. Hodkinson, M. Reynolds, *Temporal Logic: mathematical Foundations and Computational Aspects*, Oxford University Press, 1994.
- [Gör et al. 97] E. Gödel, M. Otto, F. Rosen, *Two-variable logic with counting is decidable*, in: *Proc. 14th IEEE Symp. Logic in Computer Science (LICS'97)*, pp. 306-317, IEEE Computer Society Press, 1997.
- [Hod et al. 2000] J. Hodkinson, F. Wolter, M. Zakharyashev, *Decidable fragments of first-order temporal logics*, *Ann. Pure Appl. Log.* 106 (2000), 85-134.
- [Hol96] B. Hollunder, *Consistency checking reduced to satisfiability of concepts in terminological systems*, *Ann. Math. Artif. Intell.* 18(2/4) (1996), 133-157.
- [HolNut90] B. Hollunder, W. Nutt, *Subsumption algorithms for concept languages*, Technical Report RR-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, 1990.
- [Lau94] A. Lauer, *Goals in multi-agent worlds: a terminological approach*, in: *Proc. 11th Europ. Conf. Artificial Intelligence*, pp. 299-303, Amsterdam, 1994.
- [Leh92] F. Lehmann, *Semantic Networks in Artificial Intelligence*, Pergamon Press, Oxford, 1992.
- [LevBra97] H. J. Levesque, R. J. Brachman, *Expressiveness and tractability in knowledge representation and reasoning*, *Comput. Intell.* 3 (1987), 78-93.
- [LevRous98] A. Y. Levy, M.-C. Rousset, *Combining horn rules and description logics in CALIN*, *Artif. Intell.* 104(1-2) (1998), 165-209.
- [Lutz99] C. Lutz, *Complexity of terminological reasoning revisited*, in: *Proc. 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, Lect. Notes Artif. Intell. 1703, pp. 181-200, Springer-Verlag, 1999.
- [LutzSt00] C. Lutz, U. Sattler, *The complexity of reasoning with boolean modal logic*, in: *Proc. Advances in Modal Logic (AIML 2000)*.
- [Lutz et al. 01] C. Lutz, H. Sturm, F. Wolter, M. Zakharyashev, *Tableaux for temporal description logic with constant domains*, in: *Automated Reasoning*, Lect. Notes. Artif. Intell. 2083, Springer-Verlag, 121-136 (2001).
- [Lutz et al. 02] C. Lutz, H. Sturm, F. Wolter, and M. Zakharyashev, *A tableau decision algorithm for modalized ALC with constant domains*, *Studia Logica* 72 (2002), 199-232.
- [Mor95] M. Mordechai, *On languages with two variables, λ . Math. Logik Grundlagen Math.* 21 (1975), 135-140.
- [MosMosurović] M. Mosurović, *Stožnost optimizacija s modalnim operatorima*, Doktorska disertacija, Univerzitet u Beogradu (2000).
- [MosZab99] M. Mosurović, M. Zakharyashev, *On the complexity of description logics with modal operators*, in: P. Kolaitis and G. Kolaitis, editors, *Proc. 3rd Panhellenic Logic Symp.*, pp. 166-171, Delphi (1999).
- [Neu90] R. Nebel, *Terminological reasoning is inherently intractable*, *Artif. Intell.* 43 (1990), 235-249.
- [Pac97] L. Pacholski, W. Sawast, L. Tendera, *Complexity of two-variable logic with counting*, in: *Proc. 14th IEEE Symp. Logic in Computer Science (LICS'97)*, pp. 318-327, IEEE Computer Society Press (1997).
- [Pra79] V. R. Pratt, *Models of program logic*, in: *Proc. 20th Annual Symp. Foundations of Computer Science (FOCS'79)*, pp. 115-122 (1979).
- [Pra89] V. R. Pratt, *A near-optimal method for reasoning about action*, *J. Comput. System Sci.* 20 (1980), 231-255.
- [Rob71] R. Robinson, *Undecidability and non-periodicity for tilings of the plane*, *Invent. Math.* 12 (1971), 177-209.
- [Sch93] K. Schüld, *A correspondence theory for terminological logics: Preliminary report*, in: *Proc. 15th Int. Joint Conf. on Artif. Intell. (IJCAI'93)*, pp. 468-471 (1993).
- [Sch93] K. Schüld, *Combining terminological logics with tense logic*, in: *Proc. 6th Portug. Conf. Artif. Intell.*, pp. 103-120, Porto (1993).
- [SchSm93] M. Schmidt-Schädl, G. Smolka, *Attributive concept descriptions with complements*, *Artif. Intell.* 48(1) (1991), 1-26.
- [Sch94] A. Schaefer, *Reasoning with individuals in concept languages*, *Data and Knowledge Engineering* 13(2) (1994), 141-176.
- [SisCis83] A. S. Sistla, E. M. Clarke, *The complexity of propositional linear temporal logics*, *J. Assoc. Comput. Mach.* 32(3) (1983), 733-749.
- [Smol88] G. Smolka, *A feature logic with subscripts*, ILL-OG report 83, IBM Deutschland, Stuttgart, West Germany (1988).
- [StasStu04] S. Staab, R. Studer, *Handbook on Ontologies*, International Handbooks on Information Systems, Springer-Verlag (2004).
- [Tob00] S. Thiele, *FSM-based reasoning for graded modal logics*, *J. Logic Comput.* 11(1) (2001), 85-106.
- [Var85] M. Y. Vardi, *The taming of converse: Reasoning about two-way computations*, in: R. Parikh, editor, *Proc. 4th Workshop on Logics of Programs*, Lect. Notes Comput. Sci. 197, pp. 413-424, Springer-Verlag (1985).
- [VarWol86] M. Y. Vardi, P. Wolper, *Automata-theoretic techniques for modal logics of programs*, *J. Comput. System Sci.* 32 (1986), 183-221, a preliminary version appeared in *Proc. 16th ACM SIGACT Symp. on Theory of Computing (STOC'84)*.

- [WolZakh98] F. Wolter, M. Zakharyashev, *Satisfiability problem in description logics with modal operators*, in: *Proc. 6th Internat. Conf. Principles of Knowledge Representation and Reasoning (KR'98)*, pp. 512-523, Trento, Italy (June 1998).
- [WolZakh99a] F. Wolter, M. Zakharyashev, *Modal description logics: Modalizing roles*, *Fundam. Inform.* 39(4) (1999), 411-438.
- [WolZakh99b] F. Wolter, M. Zakharyashev, *Multidimensional description logics*, in: *Proc. IJ-CAI'99*, pp. 104-109 (1999).
- [WolZakh99c] F. Wolter, M. Zakharyashev, *Temporalizing description logics*, in: D. Gabbay and M. de Rijke, editors, *Frontiers of Combining Systems*, Studies Press-Wiley (1999).