Description Logics for Ontologies

Zusammenfassung der wissenschaftlichen Arbeiten

eingereicht bei der
Fakultät Informatik
der
Technischen Universität Dresden

anstelle einer Habilitationsschrift

von Dr. rer. nat. Ulrike Sattler
aus München

April 2003

Die eingereichten Arbeiten sind in den *References* mit $*$ gekennzeichnet. Im Appendix B ist der Beitrag der Autorin an gemeinschaftlichen Arbeiten beschrieben.

Contents

Abstract

Description Logics (DLs) are a family of knowledge representation formalisms designed for the representation of *terminological* knowledge. A DL knowledge base consists (at least) of a set of concept definitions, namely of those concepts that are relevant for the specific application. Standard inference services provided by DL-based knowledge representation systems include tests whether each defined concept is satisfiable and the computation of the subsumption hierarchy of the defined concepts, i.e., of the specialisation relation between the defined concepts.

Besides the well-defined semantics of DLs, these inference services make DLs suitable candidates for *ontology* languages, which have become of increasing importance due to the amount of information available electronically and the vision of the semantic web. For a variety of DLs, decision procedures, tight complexity bounds, and practical inference algorithms for the corresponding inference problems are known. It is clear that, to be of use as an ontology language, a description logic has to provide adequate expressive power, and we are thus concerned with the well-known trade-off between complexity and expressiveness.

After a brief introduction to ontologies, we introduce the basic description logic $\mathcal{ALC}$ and describe how DLs can be used as ontology languages. Next, we sketch the relationship between DLs and other formalisms such as first order and modal logic and data base conceptual models. To give a broader view of DLs, some standard expressive means in DLs are mentioned as well as their modal logic counterparts and their effect on the complexity of the inference problems. In Section 3, we give an intuitive explanation of standard reasoning techniques employed for DLs and discuss their respective advantages: tableau-based algorithms turned out to be well-suited for implementations, whereas automata-based algorithms yield elegant upper complexity bounds for EXPTIME logics. In many cases, first a DL was proven to be in EXPTIME using automata before a tableau-based algorithm was designed and implemented.

Having thus introduced description logics and how they can be used as ontology languages, in Section 4, we describe how we have designed the rather successful DLs $\mathcal{SHIQ}$ and $\mathcal{RIQ}$: we first observe that $\mathcal{ALC}$ lacks the expressive power to describe aggregated objects using a transitive part-whole relation, and then extend $\mathcal{ALC}$ with a new constructor that overcomes this expressive shortcoming while still allowing for a practical, tableau-based inference algorithm. Step by step, we further extend the resulting DLs with new constructors that were chosen according to the same design goal: to overcome expressive shortcomings while allowing for practical inference algorithms. For each extension, we describe how we have modified the tableau algorithm to take into account the new constructor.

In Section 5, we are concerned with *hybrid* logics, i.e., description and modal logics that allow to refer to single individuals using *nominals*. Nominals are a rather special constructor since they destroy a nice model theoretic property that most DLs enjoy, namely the tree model property. Despite this effect, we were able to show, for two example hybrid logics, that automata on trees can still be used to decide satisfiability of hybrid DLs and thus provide tight upper complexity bounds. To this purpose, we use a certain abstraction technique from (non)-tree models to tree structures. This technique turns out to be applicable also for tableau algorithms: we have used it to devise a tableau algorithm for the extension of (a restriction of) $\mathcal{SHIQ}$ with nominals.

1 Ontologies

A well-known attempt to define what constitutes an ontology is due to Gruber [1993]:

an ontology is an explicit specification of a conceptualisation,

where “a conceptualisation” means an abstract model of some aspect of the world. This was later elaborated to “a formal specification of a shared conceptualisation” [Borst *et al.* 1997]. In this abstract model, relevant concepts of the aspect in question are defined, including a description of the vital properties of their instances. For example, if we are concerned with transportation means, relevant concepts are “bicycle” and “power unit”, and the description of bicycles contains a statement such as “a bicycle is powered by a human through pedalling”.

In the last decade, ontologies became rather popular through applications like the Semantic Web [Berners-Lee 1999; Berners-Lee *et al.* 2001], enterprise knowledge management systems [Uschold *et al.* 1998], and medical terminology systems [Stevens *et al.* 2002; Rector & Horrocks 1997; Spackman 2000] and through the growing amount of data available electronically.

An ontology is *built* (possibly by a group of) domain experts—and it will *evolve* over time in any application that changes over time. Moreover, it is advisable to *integrate* existing ontologies if a larger aspect of the world is to be covered—instead of building a new one from scratch. Finally, if an ontology is *deployed*, knowledge is shared using the concepts defined in the ontology, e.g., concrete objects are described using the vocabulary defined in an ontology. Each of these tasks is rather complex: e.g. building and evolution involves a huge amount of creativity, integration requires knowledge in a large aspect. Moreover, all four ontology engineering tasks might involve co-operation, which increases the risk of misunderstanding, redundancy, etc. Thus if an ontology is to be engineered by more than one user, the use of an *unambiguous* language makes it easier to agree upon a specification and thus decreases the risk of misunderstandings.

The increasing importance of ontologies and their processing in computers has led to the development of ontology editors such as OntoEdit, Protégé, Rice, and OilEd [Sure *et al.* 2002; Protégé 2003; Cornet 2003; Bechhofer *et al.* 2001]. Due to the above mentioned complexity of ontology engineering tasks, it is highly desirable that these editors support the user in the design, evolution, integration, and deployment of ontologies through corresponding, intelligent system services. This automatic processing of ontologies has further implications on the ontology language. In addition to the reduction of misunderstandings, an unambiguous language enables the precise definition of the behaviour of system services, i.e., in the design phase of an ontology-based system, we can *define* what a system service is supposed to do on a given input. This enables the investigation of the soundness, completeness, and termination of the algorithms underlying the system services.

Moreover, if the un-ambiguousness of the ontology language is due to its semantics being defined via a translation into a logic, this yields further possibilities: we can base system services on logical reasoning problems, and thus investigate these problems and the reasoning algorithms designed to solve them in a rigorous way. That is, we can investigate the decidability and computational complexity of the corresponding logical problems. The latter is a pre-requisite to design *optimal* reasoning algorithms—or to prove that a given algorithm cannot be optimised substantially. The former is important if a reasoning problem underlying a system service is undecidable: in this case, one has to decide whether to live with heuristic, approximative, or partial algorithms, or modify the ontology language in order to re-gain decidability. Both the investigation of the decidability and computational complexity give insight into the sources of complexity, and thus might allow to devise a certain fragment of the ontology language for which the corresponding algorithms show more desirable properties or better performance.

Finally, using a logical formalism allows us to profit from the huge amount of work in computer science and logic. There exists a large variety of complexity results for logical problems which can be used instead of re-proving similar results again; as an example for such results by translation, see [Schild 1991; De Giacomo 1995]. We can also use a variety of results in model theory to learn more about the expressive power [Baader 1996a] of an ontology language [Borgida 1996; Vardi 1997].

Recently, the ontology editors Rice and OilEd have been developed which provide system services based on logical reasoning problems to support the user in the above mentioned ontology engineering tasks [Cornet 2003; Bechhofer *et al.* 2001]. For example, in the design phase, the computation of the taxonomy[1] taking into account the description of the concepts defined so far might help to detect modelling flaws in an early design phase: missing or unintended specialisation links are signs of incomplete or erroneous concept descriptions. In all tasks, singling out synonymous concepts helps reducing

[1]The specialisation hierarchy of the concepts defined so-far.

redundancy and thus misunderstandings. Finally, checking the consistency of concept description also helps detecting modelling flaws.

However, since ontology engineering is still a rather new field, various other such services will need to be devised to provide optimal support. For example, one would like to ask the editor to propose a new concept description as (the most specific) generalisation of a given set of instances; one would like to find a concept description that follows a certain "pattern" of a concept; or one would like to see a user-friendly approximation of a concept description, for instance in a frame-based notation. These services are provided by so-called *non-standard inference services* such the least common subsumer and most specific concept, matching of concepts, or computing the approximation of a concept expressed in a more expressive logic in a less expressive logic [Küsters 2001; Baader *et al.* 2001; Baader & Turhan 2002; Brandt *et al.* 2002].

The ontology editors Rice and OilEd are based on description logics, which are described in the next section, together with their usage as ontology languages.

2 Introduction to Description Logics

Description logics (DLs) [Baader *et al.* 2003; Baader & Sattler 2001; Calvanese *et al.* 1999b] are a family of logic-based knowledge representation formalisms designed to represent and reason about the knowledge of an application domain in a structured and well-understood way. They are descendants of semantic networks and frames, and are equipped with a formal, Tarski-style semantics.

The basic notions in DLs are *concepts* (unary predicates) and *roles* (binary relations), and a specific DL is mainly characterised by the *constructors* it provides to form complex concepts and roles from atomic ones. In this introduction, we only illustrate some typical constructors by example. The following concept describes "A cooler that is connected to something which (i) is a reactor, (ii) has a part that is a stirrer, and (iii) whose functionality is to stir or to cool (or both)":

$$\begin{aligned}\mathsf{Cooler} \sqcap \exists \mathsf{connectedTo}.(\mathsf{Reactor} \sqcap \exists \mathsf{hasPart}.\mathsf{Stirrer} \sqcap \\ \forall \mathsf{functionality}.(\mathsf{Cooling} \sqcup \mathsf{Stirring}))\end{aligned} \tag{1}$$

This concept employs the Boolean constructors *conjunction* ($\sqcap$) and *disjunction* ($\sqcup$) on concepts, which are interpreted as set intersection and union, respectively, as well as the *existential restriction* ($\exists r.C$) and the *value restriction* constructor ($\forall r.C$). For an object x to be instance of $\exists r.C$, there has to exist an object, say y, which belongs to C and is related via r to x. Similarly, x is an instance of $\forall r.C$ if all objects related to x via r are instances of C.

In addition to such a set of constructors, DLs are usually equipped with a terminological component, often called a *TBox*. In its simplest form, a TBox can be used to introduce names (abbreviations) for complex concepts. For example, we can introduce the abbreviation CooledStirringReactor for the concept in Equation 1 from above. More expressive TBox formalisms allow the statement of *general concepts inclusion axioms* (GCIs) such as

$$\exists \mathsf{hasPart}.\mathsf{Stirrer} \mathrel{\dot\sqsubseteq} \mathsf{Reactor} \sqcap \exists \mathsf{functionality}.\mathsf{Stirring}, \tag{2}$$

which says that only stirring reactors can have stirrers. In ontology applications, both kinds of assertions are useful: we can introduce names for all the relevant concepts of the application domain and, additionally, use GCIs to describe the background knowledge of the application, i.e., to constrain the set of possible models of the TBox.

Description logic systems provide their users with various reasoning capabilities that deduce implicit knowledge from the one explicitly stated in the TBox. The *subsumption* algorithm determines subconcept-superconcept relationships: a concept C is subsumed by a concept D w.r.t. a TBox if, in each model of the TBox, each instance of C is also an instance of D, i.e., each model of the TBox interprets C as a subset of the interpretation of D. Such an algorithm can be used to compute the *taxonomy* of a TBox, i.e., the subsumption hierarchy of all those concepts introduced in the TBox. The *satisfiability* algorithm tests whether a given concept can ever be instantiated.

Unsurprisingly, the higher the expressive power of a DL is, the more complex are the subsumption and the satisfiability problem. To use a DL for a certain application, it has to provide enough expressive power to describe the relevant properties of the objects in this application. On the other hand, the system services should be "practical" in that they run in realistic time and space. Thus, we are confronted with the well-known trade-off between expressivity and complexity, as in many other areas of computer science.

In the last decade, a lot of work was devoted to investigate DLs w.r.t. their expressive power and computational complexity. It turned out that the first DL systems were based on undecidable logics [Schmidt-Schauss 1989; Patel-Schneider 1989]. As a reaction, the expressive power was restricted severely, thus yielding DLs with polynomial reasoning problems

[Donini *et al.* 1991b; Patel-Schneider *et al.* 1991]. Then, in parallel with the discovery of the close relation between description and modal logics [Schild 1991; De Giacomo & Lenzerini 1994a], PSPACE-complete DLs were identified [Schmidt-Schauß & Smolka 1991], and a tableau-based reasoning algorithm was implemented for such a DL [Baader *et al.* 1994]. After certain optimisations, it turned out that this implementation behaves much better than the high worst-case complexity of the underlying reasoning problem suggests. This motivated the implementation of tableau-based reasoning algorithms for EXPTIME-complete DLs [Horrocks 1998b; Haarslev & Möller 2001]. Again, these implementations proved to be amenable to optimisations and behave surprisingly well in practise. This fostered the design and investigation of other EXPTIME-complete DLs together with tableau-based, "practicable" reasoning algorithms. Today, industrial strength DL systems are being developed for very expressive DLs with system services being based on highly optimised tableau algorithms and with applications like the Semantic Web or knowledge representation and integration in bio-informatics.

2.1 Basic notions in description logics

In this section, we define the basic description logic $\mathcal{ALC}$, TBox formalisms, and reasoning problems.

Definition 1 *Let* $\mathbf{C}$ *and* $\mathbf{R}$ *be disjoint sets of* concept *and* role names*. The set of* $\mathcal{ALC}$ -concepts *is the smallest set such that*

- *each concept name* $A \in \mathbf{C}$ *is an* $\mathcal{ALC}$*-concept and*
- *if* C *and* D *are* $\mathcal{ALC}$*-concepts and* r *is a role name, then* $\neg C$, $C \sqcap D$, $C \sqcup D$, $\exists r.C$, *and* $\forall r.C$ *are also* $\mathcal{ALC}$*-concepts. Concepts of the form* $\exists r.C$ *are called* existential restrictions*, whereas concepts* $\forall r.C$ *are called* universal restrictions.

A general concept inclusion axiom (GCI) *is of the form* $C \dot{\sqsubseteq} D$ *for* C, D $\mathcal{ALC}$*-concepts. A* TBox *is a finite set of GCIs.*

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *consists of a non-empty set* $\Delta^{\mathcal{I}}$*, the* interpretation domain*, and a mapping* $\cdot^{\mathcal{I}}$ *which associates, with each concept name* A*, a set* $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ *and, with each role name* r*, a binary relation*

$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. *The interpretation of complex concepts is defined as follows:*

$$\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}, \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}}, \\
\neg C^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}, \\
(\exists R.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \textit{there exists an } e \in \Delta^{\mathcal{I}} \textit{ with } \langle d, e\rangle \in R^{\mathcal{I}} \textit{ and } e \in C^{\mathcal{I}}\}, \\
(\forall R.C)^{\mathcal{I}} &= \{d \in \Delta^{\mathcal{I}} \mid \textit{for all } e \in \Delta^{\mathcal{I}}\textit{, if } \langle d, e\rangle \in R^{\mathcal{I}}\textit{, then } e \in C^{\mathcal{I}}\}.
\end{aligned}$$

An interpretation $\mathcal{I}$ satisfies *a GCI $C \dot{\sqsubseteq} D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; $\mathcal{I}$* satisfies *a TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies all GCIs in $\mathcal{T}$—in this case, $\mathcal{I}$ is called* a model of $\mathcal{T}$*. An element $d \in C^{\mathcal{I}}$ is called an* instance *of C and, if $\langle d, e\rangle \in r^{\mathcal{I}}$, then e is called an r*-successor *of d.*

A concept C is satisfiable *w.r.t. a TBox $\mathcal{T}$ if there is a model $\mathcal{I}$ of $\mathcal{T}$ with $C^{\mathcal{I}} \neq \emptyset$. A concept C is* subsumed *by a concept D w.r.t. $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) if, for each model $\mathcal{I}$ of $\mathcal{T}$, $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Two concepts are* equivalent *if they mutually subsume each other.*

As usual, we use $\top$ as an abbreviation for $A \sqcup \neg A$, $\bot$ for $\neg\top$, $C \Rightarrow D$ for $\neg C \sqcup D$, and $C \Leftrightarrow D$ for $(C \Rightarrow D) \sqcap (D \Rightarrow C)$. Moreover, we use $C \doteq D$ as an abbreviation for $C \dot{\sqsubseteq} D$ and $D \dot{\sqsubseteq} C$.

Some remarks are in order here. Firstly, in $\mathcal{ALC}$, the two reasoning problems satisfiability and subsumption can be mutually reduced to each other: C is satisfiable w.r.t. $\mathcal{T}$ iff C is *not* subsumed by $\bot$ w.r.t. $\mathcal{T}$. And $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is *not* satisfiable w.r.t. $\mathcal{T}$.

Secondly, it can be shown that satisfiability (and thus subsumption) w.r.t. a general TBox is EXPTIME-complete [Schild 1991], whereas these problems become PSPACE-complete when considered w.r.t. the empty TBox [Schmidt-Schauß & Smolka 1991].

2.2 Description logics as ontology languages

As mentioned above, ontologies are employed in a variety of applications, and engineering ontologies is a complex task that requires optimal support from powerful system services. We have argued that a *logic-based* language enables the design of provably correct and optimal such services. Now *description logics* are such a class of logic-based knowledge representation languages. Moreover, they come with a knowledge base formalism which makes them good candidates for ontology languages: an ontology can be suitably formalised in a TBox. As indicated above, to formalise an ontology, we divide the TBox into the following two disjoint parts.

Background Knowledge GCIs of the form $C \dot{\sqsubseteq} D$, for C and D complex concepts, can be used to formalise background knowledge of the application domain and thus to constrain the set of models.

For example, we can express that the concepts `Device` and `Connection` are disjoint by

$$\texttt{Device} \dot{\sqsubseteq} \neg\texttt{Connection}$$

and use the GCI 2 from above to ensure that only stirring reactors can have stirrers.

Definitiorial Part For each concept relevant in the application domain, we can introduce a concept name A and a *concept definition* $A \dot{\sqsubseteq} C$ or $A \doteq C$ describing necessary or necessary and sufficient conditions for individuals to be an instance of A. We say that A is a *primitively defined* or a *defined* concept.

For example, we can (primitively) define connections as being material things having some input and some output, and then define a hose as a flexible connection:

$$\begin{array}{rcl}\texttt{Connection} & \dot{\sqsubseteq} & \texttt{MThing} \sqcap \exists\texttt{hasComp.Output} \sqcap \exists\texttt{hasComp.Input} \\ \texttt{Hose} & \doteq & \texttt{Connection} \sqcap \texttt{Flexible}\end{array}$$

Thus we have given necessary conditions for an object to be a connection, and necessary and sufficient conditions for an object to be hose.

Thus the (standard) system services provided by DL-based knowledge representation systems can be based on the logical problems satisfiability and subsumption:

- each concept defined in a TBox $\mathcal{T}$ is tested for satisfiability w.r.t. $\mathcal{T}$. Unsatisfiable concepts are returned to the user or, e.g., marked red.
- the computation of the *taxonomy* of a TBox $\mathcal{T}$: for each pair A_1, A_2 of concepts defined in the definitorial part of $\mathcal{T}$, we test whether $A_1 \dot{\sqsubseteq}_{\mathcal{T}} A_2$ and $A_2 \dot{\sqsubseteq}_{\mathcal{T}} A_1$. A taxonomy is the partial order of the defined concepts w.r.t. $\dot{\sqsubseteq}_{\mathcal{T}}$, and is often presented as the corresponding Hasse-diagram.

As mentioned above, unsatisfiable defined concepts and unintended or missing subsumption relationships are signs of modelling flaws, and thus these system services can be used to support the engineering of ontologies: in the design phase and when modifying or integrating an ontology, we can repeatedly use both system services to ensure that the TBox is consistent, that it reflects our intuition, and that it does not contain unintended redundancies, i.e., defined concepts that are equivalent. Unsurprisingly, it turned out that, in applications where the knowledge engineer is no description logic expert, ontology engineering requires more support [Sattler 1998; Molitor 2000]. For example, the domain expert wants to see automatically generated suggestions for a new concept definition as a generalisation of a set of example instances. This observation lead to the investigation of the afore mentioned *non-standard inferences* in description logics [Küsters 2001; Baader *et al.* 2001; Baader & Turhan 2002; Brandt *et al.* 2002].

State-of-the-art DL-based systems such as FaCT and Racer [Horrocks 1998b; Haarslev & Möller 2001] provide the above *standard* system services such as deciding the satisfiability and the computation of the taxonomy, and are based on a DL, called $\mathcal{SHIQ}$ [Horrocks *et al.* 1999], which is an extension of $\mathcal{ALC}$ with a variety of interesting expressive means [Sattler 2000]; $\mathcal{SHIQ}$ is discussed in detail in Section 4. Despite providing additional expressive means, reasoning w.r.t. TBoxes for $\mathcal{SHIQ}$ is of the the same worst-case complexity as for $\mathcal{ALC}$, namely ExpTime-complete [Tobies 2001a]. This high complexity implies that, in the worst-case, the computation might take far too much time. However, the algorithms in these DL-based systems proved to be amenable to a wide range of optimisations, as a consequence of which these systems behave surprisingly well in many realistic applications [Horrocks 1997; 1998b; Haarslev & Möller 2001; Horrocks & Sattler 2002].

The suitability of DLs as ontology languages has been highlighted by their role as the foundation for several web ontology languages, including OIL [Fensel *et al.* 2001], DAML+OIL [Horrocks & Patel-Schneider 2001; Horrocks *et al.* 2002], and OWL, a newly emerging ontology language standard being developed by the W3C Web-Ontology Working Group.[2] All of these languages have a syntax based on RDF Schema, but the basis for their design is a combination of the DLs $\mathcal{SHIQ}$ (mentioned above) and $\mathcal{SHOQ}(D)$ [Horrocks & Sattler 2001]. Both are DLs that were designed with the goal to find a good compromise between expressiveness and the complexity of reasoning.

[2]`http://www.w3.org/2001/sw/WebOnt/`

2.3 Relationship with other formalisms

Even though they were developed independently, description logics are closely related to other logical formalisms, most importantly to modal logics and other non-classical logics. In this section, we will briefly sketch these relationships; for a more detailed discussion, see Chapter 4 of [Baader *et al.* 2003].

First order predicate logic It is not too hard to see that each $\mathcal{ALC}$ concept C can be translated into a formula Φ_C in one free variable of the two-variable fragment of first order logic such that $a \in C^{\mathcal{I}}$ iff $\mathcal{I} \models \Phi_C(a)$, provided that we view interpretations both as DL and predicate logic interpretations, that a concept name is viewed as a unary predicate symbol, and a role name as a binary predicate symbol. The following two functions t_x and t_y realise this translation:

$$\begin{array}{rcl@{\qquad}rcl}
t_x(A) & = & A(x), & t_y(A) & = & A(y),\\
t_x(C \sqcap D) & = & t_x(C) \wedge t_x(D), & t_y(C \sqcap D) & = & t_y(C) \wedge t_y(D),\\
t_x(C \sqcup D) & = & t_x(C) \vee t_x(D), & t_y(C \sqcup D) & = & t_y(C) \vee t_y(D),\\
t_x(\exists R.C) & = & \exists y.R(x,y) \wedge t_y(C), & t_y(\exists R.C) & = & \exists x.R(y,x) \wedge t_x(C),\\
t_x(\forall R.C) & = & \forall y.R(x,y) \Rightarrow t_y(C), & t_y(\forall R.C) & = & \forall x.R(y,x) \Rightarrow t_x(C).
\end{array}$$

Then we can easily show that C is satisfiable w.r.t. a TBox $\mathcal{T} = \{C_i \dot{\sqsubseteq} D_i \mid 1 \leq i \leq n\}$ iff

$$t_x(C) \wedge \bigwedge_{i=1}^{n} \forall x.t_x(C_i) \Rightarrow t_x(D_i)$$

is satisfiable.

Thus $\mathcal{ALC}$ with general TBoxes can be viewed as a fragment of the two-variable fragment of first order logic, and hence satisfiability is in NExptime [Grädel *et al.* 1997]. For a detailed discussion of this relationship, see [Borgida 1996; Lutz *et al.* 2001].

Modal Logics The connection between description logics and the two-variable fragment of first order logic can be viewed as an immediate consequence of the close relationship between modal and description logics and the modal correspondence theory [van Benthem 1983]. $\mathcal{ALC}$ (without TBoxes) is a notational variant of the multi modal logic **K** [Halpern & Moses 1992], and a variety of description logic expressive means have close relatives in modal logic, which will be sketched in the next section. To see the connection between **K** and $\mathcal{ALC}$, it suffices to view *elements* of a DL interpretation domain as *worlds* in a Kripke structure, roles as modal parameters, universal restrictions as box formulae, and existential restrictions as diamond formulae. Then, for example, it can be easily seen that

$$A \sqcap \exists r.(C \sqcup \forall s.D) \text{ is equivalent to } A \wedge \langle r \rangle (C \vee [s]D)$$

for A, C, and D concept names and propositional variables, and r and s role names and modal parameters.

After the close relationship between modal and description logics was published [Schild 1991], it was successfully exploited, with dynamic logics and the μ-calculus being extremely useful. Firstly, complexity results and techniques were translated from modal logics to their description logic counterparts [De Giacomo & Lenzerini 1994b; 1994a; Schild 1994; Calvanese *et al.* 1999a]. Secondly, these DL counterparts were extended with DL specific expressive means such as number restrictions (defined in Section 2.4), which then gave raise to new complexity results for modal logics [De Giacomo & Lenzerini 1994b; 1994a; Schild 1994; Calvanese *et al.* 1999a]. Thirdly, DL reasoners can be used as modal logic theorem provers, and turned out to perform well on modal logic sets of benchmark formulae [Horrocks 1998a; Patel-Schneider & Horrocks 1999; Haarslev & Möller 2000]. Finally, automata-based reasoning techniques that were successfully employed in modal logic turned out to be equally efficient to devise tight upper complexity bounds for description logics [Calvanese *et al.* 1999a; Lutz & Sattler 2001; Sattler & Vardi 2001; Kupferman *et al.* 2002]. Please note that, despite this close relationship, a variety of expressive means and reasoning problems are only considered in description logics, for example concrete domains [Baader & Hanschke 1991; Lutz 2003], general and cyclic TBoxes with different fixpoint semantics [Nebel 1990; Baader 1996b], and the above mentioned non-standard inferences.

Finally, the close relationship between modal and description logics implies that DLs are closely related to the *guarded fragment* [Andréka *et al.* 1998; Grädel 1999], a fragment of first order logic generalising a variety of modal logics. This relationship motivated the design and investigation of n-ary DLs that are fragments of the guarded fragment [Lutz *et al.* 1999; Georgieva *et al.* 2003], and the usage of successful DL implementation and optimisation techniques for tableau algorithms for the guarded fragment [Hirsch & Tobies 2001; Hladik 2002].

Frame-based Systems Since DLs are descendants of frame-based systems, it is not surprising that these two families of formalisms are still closely related. For example, the DL based ontology editor Oiled [Bechhofer *et al.* 2001] uses a frame-based user interface rather similar to the one of Protégé, one of the most successful frame-based systems [Protégé 2003]. A detailed description of this relationship can be found in [Calvanese *et al.* 1994] and Section 4 of [Baader *et al.* 2003].

Conceptual Graphs Both Conceptual graphs and DLs were originally designed as "non-logical" knowledge representation formalisms, thus avoiding variables and formulae. Recently, it turned out that, indeed, these families of formalisms are closely related [Baader *et al.* 1999]: (a generalisation of) *simple* graphs can be translated into an existential description logic, and another class can be translated into the guarded fragment. Again, exploiting these relationships yields decidability and complexity results "for free".

Conceptual Models Description Logics can be used to reason about conceptual database models, e.g. Entity-Relationship (ER) diagrams for relational databases or UML schemas for object-oriented databases [Calvanese *et al.* 1994; 1998]. In general, one can translate a conceptual model (i.e., an ER diagram or a UML Schema) into a TBox such that an entity (a relationship, or a class) can be instantiated in a database conforming to the model iff its translation is satisfiable w.r.t. the resulting TBox. Analogously, implicit IS-A links and containment relations can be reduced to subsumption. This extremely useful application led to the implementation of Icom, a tool for intelligent conceptual modelling, which is based on the FaCT reasoner for $\mathcal{SHIQ}$ [Franconi & Ng 2000].

2.4 Standard expressive means in description logics

To give the reader an impression of what DLs are, we present a variety of expressive means that are commonly used in DLs, and discuss, if appropriate, their modal logic equivalent and their influence on the computational complexity.

TBoxes were introduced in Section 2.1, and it was mentioned in Section 2.2 that they are divided into a *background knowledge* part and a *definitorial* part. Some DLs only allow for the definitorial part and possibly require this part to be free of "definitorial cycles". In this case, fixing the interpretation of undefined concept names uniquely determines the interpretation of defined concepts—which is not the case in the presence of cyclic definitions [Nebel 1990; Baader 1996b]. Moreover, reasoning w.r.t. acyclic concept definitions can be reduced to pure concept reasoning: one can either use a (sub-optimal) technique, called *unfolding*, which reduces reasoning w.r.t. acyclic concept definition to pure concept reasoning [Nebel 1990], or use more direct techniques [Lutz 1999]. As a result of the latter, it turned out that, for a variety of logics, reasoning w.r.t. acyclic concept definitions is as complex as pure concept reasoning.

The modal logic counterpart to TBoxes is the *universal role*, a role that is interpreted as $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. It enables the *internalisation* of a TBox [Baader 1991; Schild 1991; Baader *et al.* 1993; Horrocks *et al.* 1999]: for u a universal role, C is satisfiable w.r.t. a TBox $\{C_i \dot{\sqsubseteq} D_i \mid 1 \leq i \leq n\}$ iff the concept

$$\exists u.C \sqcap \bigsqcap_{i=1}^{n} \forall u.(C_i \Rightarrow D_i)$$

is satisfiable. Moreover, we can translate a concept C involving a universal role into a concept C' and a (general) TBox $\mathcal{T}_C$ without the universal role such that C is satisfiable iff C' is satisfiable w.r.t. $\mathcal{T}_C$ [Lutz 2002]. This translation yields another proof of $\mathcal{ALC}$ with general TBoxes being Exptime-hard by a reduction of the (Exptime-hard) extension of the modal logic **K** with the universal role [Spaan 1993a].

Number Restrictions are an expressive means rather popular in DLs: they are present in almost all implemented DL systems. They are of the form $(\geqslant nr.C)$ (atleast restriction) or $(\leqslant nr.C)$ (atmost restriction), for n a non-negative integer, r a role, and C a (possibly complex) concept. Their interpretation is defined as follows:

$$\begin{array}{rcl}(\geqslant nr.C)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \#\{e \in C^{\mathcal{I}} \mid (d,e) \in r^{\mathcal{I}}\} \geq n\},\\ (\leqslant nr.C)^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \#\{e \in C^{\mathcal{I}} \mid (d,e) \in r^{\mathcal{I}}\} \leq n\},\end{array}$$

where $\#M$ denotes the cardinality of a set M. They can be used, e.g., to described pipes as those connections having exactly one input and one output (we use $(= nr.C)$ as an abbreviation for $(\geqslant nr.C) \sqcap (\leqslant nr.C)$):

$$\texttt{Connection} \sqcap (= 1\texttt{hasComponent.Input}) \sqcap (= 1\texttt{hasComponent.Output})$$

In their simpler form, number restrictions only allow for the concept $\top$ in the place of C above. A further restriction only allows for 2 in atleast restrictions and 1 in atmost restrictions. Finally, *features* are role names that are to be interpreted as partial functions—they can be viewed as a "globalised" version of a simple form of number restrictions. Number restrictions rarely seem to have effects on the complexity of DLs: for a variety of logics, extending them with number restrictions does not change their complexity, even if such an extension yields the loss of the *finite model property* (see Section 4.4 for a more detailed discussion). For example, when extended with number restrictions, $\mathcal{ALC}$ remains in PSPACE [Tobies 2001b], $\mathcal{ALC}$ with TBoxes remains in EXPTIME, even if further extended with other expressive means such as inverse roles (see below) [Tobies 2001a; Kupferman *et al.* 2002]. These observations hold regardless of the coding of numbers in number restrictions, i.e, they are independent of whether numbers are coded in *unary* (i.e., $|(\geqslant nr.C)| = |(\leqslant nr.C)| = n + 1 + |C|$, for $|C|$ the length of a concept) or in *binary* (i.e., $|(\geqslant nr.C)| = |(\leqslant nr.C)| = \log(n) + 1 + |C|$).

Number restrictions are known in modal logics as *graded modalities* [Fine 1972; van der Hoek & De Rijke 1995], whereas features play an important role in dynamic logic: they are syntactic variants of *deterministic programs* [Ben-Ari *et al.* 1982].

Nominals are, in their simplest form, special concept names that are to be interpreted as singleton sets. For example, the following concept describes those objects that are part of the oil platform Brent Spar

$$\exists\texttt{partOf}.\texttt{BrentSpar},$$

where `BrentSpar` is a nominal, i.e., has only one instance.

In description logics, a weak form of nominals, ABoxes ("A" for assertional), are widely known and used: they provide a second knowledge base component to describe individuals and their interrelationship while possibly referring to the background knowledge and concepts defined in a TBox. Given a set of individual names $\mathbf{I}$, *assertions* are of the form

$$a : C \text{ and } \langle a, b\rangle \in r$$

for $a, b \in \mathbf{I}$, C a concept, and r a role. An *ABox* is a finite set of assertions. An interpretation $\mathcal{I}$ associates, additionally, with each $a \in \mathbf{I}$, some $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$, and $\mathcal{I}$ satisfies an assertion of the form

$$\begin{array}{rl} a : C & \text{if } a^{\mathcal{I}} \in C^{\mathcal{I}} \text{ and} \\ \langle a, b\rangle : r & \text{if } \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in r^{\mathcal{I}}. \end{array}$$

An interpretation satisfying all assertions in an ABox is a *model* of the ABox, and an ABox that has a model is called *consistent.*

Some DLs employ the *unique name assumption*, i.e., they require interpretations $\mathcal{I}$ to map different individual names to different elements of the interpretation domain. Not employing this assumption seems to be the more general way since we can impose, on demand, difference of individuals by adding assertions of the form $a{:}A$ and $b{:}\neg A$.[3]

In most description logics, we thus have a clear distinction between the terminological part of a knowledge base and the assertional one: we can use concepts defined in the TBox to describe individuals, but we cannot use individuals (from the ABox) to define concepts. With nominals, such a distinction does not exist.

Using nominals, it is straightforward to express ABox assertions: using a nominal N_a for each individual name a and a new role name u, an ABox is translated into a conjunction over all its assertions, $a{:}C$ is translated into $\exists u.(N_a \sqcap C)$, and $\langle a, b\rangle{:}r$ is translated into $\exists u.(N_a \sqcap \exists r.N_b)$. Then it can be easily seen that an ABox is consistent iff its translation is satisfiable. This reduction is also possible in the presence of a TBox and a role hierarchy (see below).

In many cases, consistency of an ABox is of the same complexity as concept satisfiability [Schaerf 1994]; examples are $\mathcal{ALC}$ without and with TBoxes [Baader & Hollunder 1991; De Giacomo & Lenzerini 1996] and $\mathcal{SHIQ}$ [Tobies 2001a]. Moreover, the tableau algorithm for $\mathcal{SHIQ}$ can be extended to decide consistency of ABoxes, see [Horrocks *et al.* 2000b]. In contrast, extending a description logic with nominals often increases its complexity. For example, $\mathcal{ALC}$ with inverse roles (see below) is PSPACE-complete, but becomes EXPTIME-complete when extended with nominals [Areces *et al.* 1999]. If, additionally, number restrictions are present, the complexity leaps from EXPTIME-completeness to NEXPTIME-completeness [Tobies 2001a]. A reason for this increase in complexity might be that nominals destroy the *tree model property* [Vardi 1997]: a logic enjoys the tree model property if every satisfiable concept/formula has a model whose relational structure forms a tree. For example, for nominals N_1 and N_2, the concept $N_1 \sqcap \exists r.(N_2 \sqcap \exists r.N_1)$ only has models with a cycle of length two.

Nominals originate in *hybrid* logic [Prior 1967; Areces *et al.* 1999; 2000], and are known in description logics as an elegant and powerful generalisation of ABoxes; they are discussed in more detail in Section 5.

Inverse Roles In various applications, one wants to use both "directions" of a role, e.g., one wants to use both `hasPart` and `isPartOf` [Sattler 2000].

[3]Such a reduction is also possible the other way round, i.e., from "with unique name assumption" to "without", by substituting individual names. However, this involves a non-deterministic step.

To model these roles adequately, i.e., to ensure that $\langle x,y\rangle \in \texttt{hasPart}^{\mathcal{I}}$ iff $\langle y,x\rangle \in \texttt{isPartof}^{\mathcal{I}}$, some description logics provide *inverse roles*: for r a role name, r^- is an inverse role, which is interpreted as $(r^-)^{\mathcal{I}} = \{\langle y,x\rangle \mid \langle x,y\rangle \in r^-\}$.

A variety of DLs can be extended with inverse roles without affecting their computational complexity: examples are $\mathcal{ALC}$ with or without TBoxes and possibly with number restrictions [Calvanese *et al.* 1999b; Tobies 2001a]. However, there are counter-examples such as $\mathcal{ALC}$ with concrete domains, which becomes NEXPTIME-complete when extended with inverse roles [Lutz 2003] and $\mathcal{ALC}$ with nominals and without TBoxes, which becomes EXPTIME-complete when extended with inverse roles [Areces *et al.* 1999]. Inverse roles are closely related to the tense logic "past" modality [Rescher & Urquhart 1971; Spaan 1993b; Vardi 1998] and are syntactic variants of converse programs in dynamic logics and the μ-calculus [Streett 1982; Vardi 1985; 1998].

Transitive Roles are special role names $r \in \mathbf{R}_+ \subseteq \mathbf{R}$ that are to be interpreted as *transitive relations* [Sattler 1996]. Transitive roles can be used to model transitive relations such as `isAncestorOf` or `isPartOf` [Sattler 2000]. Another way to extend DLs with transitivity is to allow for the *transitive closure operator* on roles, i.e., to allow for roles r^* in the place of roles [Baader 1991; De Giacomo 1995], where r^* is to be interpreted as the transitive closure of $r^{\mathcal{I}}$. We will discuss the expressiveness of transitive roles in more detail in Section 4.1.

Adding transitive roles to $\mathcal{ALC}$ without TBoxes yields a DL whose reasoning problems are still PSPACE-complete [Sattler 1996], whereas adding the transitive closure operator on roles yields an EXPTIME-complete logic [Fischer & Ladner 1979]. Transitive roles are notational variants of transitive accessibility relations in modal logics [Halpern & Moses 1992], whereas a transitive closure operator is also present in the dynamic logic PDL [Fischer & Ladner 1979], which is a notational variant of $\mathcal{ALC}$ with regular role expressions [Schild 1991; Baader 1991].

Boolean Operator on Roles So far, we considered DLs with full Boolean operators on concepts, but no Boolean operators on roles. In DLs, Boolean operators on roles are mostly restricted to intersection [Donini *et al.* 1991a], or to union and difference [De Giacomo & Lenzerini 1996; Calvanese *et al.* 1995]. They are interpreted in the obvious way, i.e., $(r \sqcap s)^{\mathcal{I}} = r^{\mathcal{I}} \cap s^{\mathcal{I}}$, etc., and are an interesting expressive means. For example, role negation allows to express the so-called *window* operator from modal logic [Gargov *et al.* 1987]. The window operator can be viewed as the dual[4] of universal restrictions: an instance of $\forall$`connectedTo.Pipe` is connected *only* to pipes, whereas an instance of the concept $\forall$`Pipe.connectedTo` using the window operator is connected to *all* pipes. It can be easily seen that the concept $\forall\neg$`connectedTo.`$\neg$`Pipe` using role negation is equivalent to $\forall$`Pipe.connectedTo`. For a complete description of the (mostly dramatic) effects of adding Boolean operators on roles to the computational complexity of $\mathcal{ALC}$, see [Lutz & Sattler 2001].

In dynamic logic, union of programs is present in all logics allowing for regular programs [Fischer & Ladner 1979], and Boolean operators on modalities are discussed, e.g., in [Gargov *et al.* 1987].

Role Hierarchies Another expressive means on roles are *role hierarchies*, which are finite sets of *role inclusion axioms*: a role inclusion axiom is an expression of the form $r \dot\sqsubseteq s$, for r and s role names. In case the underlying DL allows for inverse roles, r and/or s can also be inverse roles. An interpretation $\mathcal{I}$ *satisfies* a role hierarchy $\mathcal{R}$ iff $r^\mathcal{I} \subseteq s^\mathcal{I}$ for each $r \dot\sqsubseteq s$ in $\mathcal{R}$. Such an interpretation is called a *model* of $\mathcal{R}$. Satisfiability and subsumption w.r.t. role hierarchies are defined in the obvious way.

Role hierarchies can be used, for example, to introduce a sub-role `hasComponent` of `hasPart`. In the presence of inverse roles, role hierarchies can be used to enforce symmetric roles using $r^- \dot\sqsubseteq r$ and $r \dot\sqsubseteq r^-$.

It should be noted that role hierarchies provide a weak form of role intersection: replacing each role expression $r_1 \sqcap r_2$ with a new role name $r_{1,2}$ and adding $r_{1,2} \dot\sqsubseteq r_1$ and $r_{1,2} \dot\sqsubseteq r_2$ to the role hierarchy yields a "weakened" form of intersection since $r_{1,2}^\mathcal{I} \subseteq r_1^\mathcal{I} \cap r_2^\mathcal{I}$. Moreover, for s a transitive role, the role inclusion axioms $r \dot\sqsubseteq s$ yields a weakened form of the transitive closure: s is interpreted as *some* transitive role containing r, whereas r^* is interpreted as *the smallest* transitive role containing r. The latter observation implies that pure concept satisfiability of $\mathcal{ALC}$, when extended with both transitive roles and role inclusion axioms, becomes EXPTIME-hard [Sattler 1996]. In DLs, role inclusion axioms are widely known since they add some form of expressive power that allows for adequate modelling of various applications whereas, in this simple form, they do not seem to have modal logic counterparts.

[4]Dual is here not used in the strict, logical sense.

General Role Inclusion Axioms (g-RIAs) are a generalisation of the above role inclusion axioms to the form $r_1 \ldots r_m \dot\sqsubseteq s_1 \ldots s_n$ for r_i, s_j role names [Horrocks & Sattler 2003]. A model of such an axiom satisfies

$$r_1^\mathcal{I} \circ \ldots \circ r_m^\mathcal{I} \subseteq s_1^\mathcal{I} \circ \ldots \circ s_n^\mathcal{I},$$

where $\circ$ denotes the composition of binary relations. *Role value maps*, i.e., concepts of the form $r_1 \ldots r_m \Rightarrow s_1 \ldots s_n$ with the semantics

$$(r_1 \ldots r_m \Rightarrow s_1 \ldots s_n)^\mathcal{I} = \{x \mid \forall y.\langle x, y\rangle \in r_1^\mathcal{I} \circ \ldots \circ r_m^\mathcal{I} \Rightarrow \langle x, y\rangle \in s_1^\mathcal{I} \circ \ldots \circ s_n^\mathcal{I}\},$$

can be viewed as a "local" form of g-RIAs. Both constructors have dramatic effects on the decidability of a description logic: it was shown in [Schmidt-Schauss 1989] that subsumption of the very weak DL allowing only for conjunction and universal restrictions becomes undecidable when extended with role value maps. DLs with g-RIAs are closely related to *grammar logics* [Demri 2001; Baldoni 1998; Baldoni *et al.* 1998], i.e., the multi modal logic **K** with the accessibility relations being constrained by a grammar: a production rule of the form $s_1 \ldots s_n \rightarrow r_1 \ldots r_m$ can be viewed as a notational variant of the g-RIA $r_1 \ldots r_m \dot\sqsubseteq s_1 \ldots s_n$, and thus enforces models to interpret $r_1 \ldots r_m$ as a sub-relation of $s_1 \ldots s_n$. It is well-known that each context-free grammar can be transformed into an equivalent one in Chomsky normal form, and that there are context-free grammars such that multi modal **K** becomes undecidable if the accessibility relations are constrained by these grammars [Baldoni 1998; Baldoni *et al.* 1998]. Transferring this to DLs yields that satisfiability of $\mathcal{ALC}$-concepts w.r.t. g-RIAs of the form $r_1 r_2 \dot\sqsubseteq s$ is undecidable.

Fixpoint Operators are expressive means that are not first order definable, and they are known in DLs in at least three forms: a restricted form includes the transitive closure operator on roles [Baader 1991; De Giacomo & Lenzerini 1996] (see above) and an operator that allows to enforce that a role is interpreted as a well-founded relation [Calvanese *et al.* 1995]. Secondly, general least and greatest fixpoints operators in DLs [Calvanese *et al.* 1999a] are notational variants of the fixpoint operators in the μ-calculus [Kozen 1982]. Thirdly, *cyclic* concept definitions such as

$$\begin{array}{rcl} \texttt{Device} & \doteq & \texttt{TechThing} \sqcap \neg\texttt{Connection} \sqcap \forall\texttt{connectedTo.Connection} \\ \texttt{Connection} & \doteq & \texttt{TechThing} \sqcap \neg\texttt{Device} \sqcap \forall\texttt{connectedTo.Device} \end{array}$$

can be read with least or greatest *fixpoint semantics* [Nebel 1990; Baader 1996b]: in contrast to the *descriptive* semantics, which takes into account *all* fixpoints of such GCIs, one might chose to take into account only the least or the greatest fixpoints. A discussion on the consequences of such a choice can be found in [Nebel 1990].

Adding least and greatest fixpoints operators to $\mathcal{ALC}$ yields a notational variant of the μ-calculus and thus brings its complexity from PSPACE- to EXPTIME-completeness [Streett & Emerson 1989], where it remains even if the logic is further extended with either inverse roles and features, number restrictions, nominals, or n-ary relations [Vardi 1998; Kupferman *et al.* 2002; Sattler & Vardi 2001; Calvanese *et al.* 1999a].

3 Introduction to DL Reasoning Techniques

In this section, we sketch two prominent reasoning techniques employed to decide concept satisfiability (and thus subsumption) w.r.t. TBoxes, and discuss their advantages and disadvantages.

3.1 Tableau algorithms

For several expressive DLs, there exist efficient tableau-based implementations that decide satisfiability of concepts w.r.t. a TBox [Horrocks 1998b; Haarslev & Möller 2001]. For an extensive survey of tableau algorithms for description logics, see, e.g., [Baader & Sattler 2001]. In the following, we will give an intuitive description of description logic tableau algorithms. In general, they work on trees whose nodes stand for individuals of an interpretation, and the input is assumed to be in negation normal form.[5] Nodes are labelled with sets of concepts, namely those they are assumed to be an instance of. Edges between nodes are labelled with role names or sets of role names, namely those that hold between the corresponding individuals.

Intuitively, to decide the satisfiability of a concept C, a tableau algorithm starts with an instance x_0 of C, i.e., a tree consisting of a root node x_0 only with C as its node label (written $\mathcal{L}(x_0) = \{C\}$). Then the algorithm breaks down concepts in node labels syntactically, thus inferring new constraints on the model of C to be built, and possibly generating new individuals, i.e., new nodes. Roughly speaking, if $(D \sqcap E) \in \mathcal{L}(y)$ has already been inferred, it adds D and E to $\mathcal{L}(y)$. For $\exists r.F \in \mathcal{L}(y)$, it generates a new r-successor node of y, say z, and sets $\mathcal{L}(z) = \{F\}$. If a node y has some r-successor z

[5]A concept is in negation normal form if negation occurs in front of concept names only. It can be easily seen that each concept can be transformed into an equivalent one in negation normal form by pushing negation inwards, using de Morgan's rules and the duality between existential and universal restrictions.

and it finds $\forall r.G \in \mathcal{L}(y)$, then G is added to $\mathcal{L}(z)$. Finally, in the presence of a TBox $\mathcal{T}$, it adds, for each GCI $C_i \dot{\sqsubseteq} D_i \in \mathcal{T}$, and for each node y, the (negation normal form of the) concept $(\neg C_i \sqcup D_i)$ to $\mathcal{L}(y)$. Now, for logics with disjunctions, various tableau algorithms non-deterministically choose whether to add D or E to $\mathcal{L}(y)$ for $(D \sqcup E) \in \mathcal{L}(y)$. The answer behaviour is as follows: if this "completion" can be carried out exhaustively without encountering a node with both a concept and its negation in its label—a so-called *clash*, then the algorithm answers that the input concept was satisfiable, and unsatisfiable, otherwise.

Since we are talking about decision procedures, *termination* is an important issue. Even though tableau algorithms for some DLs terminate "automatically", this is not the case for more expressive ones. For example, consider the algorithm sketched above on the input concept A and TBox $\{A \dot{\sqsubseteq} \exists r.A\}$: it would create an infinite r-chain of nodes with labels $\{A, \exists r.A\}$. To guarantee termination, the tableau algorithm needs to be stopped explicitly. Intuitively, the processing of an element z is stopped if all "relevant" concepts in the label of z are also present in the label of an "older" element z'. In this case, z' is said to *block* z. The definition of "relevant" has to be chosen carefully since it is crucial for the correctness of the algorithm [Baader & Sattler 1999; Horrocks *et al.* 1999; 2000a] and for the efficiency of the implementation [Horrocks & Sattler 2002; Hladik 2002].

Correctness of DL tableau algorithms are often proved as follows: first, termination is proved by, roughly spoken, showing that the algorithm builds a (tree) structure of bounded size in a monotonic manner. Soundness is proved by constructing a model (or an abstraction of a model) of the input concept (and TBox) in case that the algorithm stops without having generated a clash. If a blocking situation has occurred, this often yields either a finite, cyclic model or an "unravelled" infinite tree model (or an abstraction thereof). Completeness can be proved by using a model of the input concept and TBox to steer the application of the non-deterministic rules and proving that no clash occurs using this control.

In the tableau sketched above and in various other tableau algorithms, disjunction is treated non-deterministically. When implementing tableau algorithms or designing an optimal tableau algorithm for a logic that is complete for a deterministic complexity class, this non-determinism has to be circumvented. A natural solution is back-tracking which, unfortunately, is often not sufficient to design optimal tableau algorithms. For example, to the best of our knowledge, the algorithm in [Donini & Massacci 2000] is the only known worst-case optimal tableau algorithm for an Exp-

TIME-complete description logic, and employs a rather intricate technique to circumvent non-determinism due to disjunction. In contrast, the tableau algorithm implemented in state-of-the-art DL systems such as FaCT and Racer uses (highly optimised) back-tracking and is 2NEXPTIME even though the underlying logic is EXPTIME-complete [Horrocks *et al.* 1999; Tobies 2001a]. Despite this sub-optimality, these tableau algorithms allow for a set of well-known efficient optimisations, so that they perform much better in practise than their worst-case complexity suggests; see [Horrocks 1998b; Horrocks *et al.* 2000a; Haarslev & Möller 2000; 2001; Horrocks & Sattler 2002] for descriptions of these optimisations. An interesting open question is whether an implementation of a worst-case optimal algorithm would behave better in practise—so far, only implementations of worst-case sub-optimal algorithms exist.

In summary, tableau algorithms are used in state-of-the-art implementations, and many well-understood optimisations are available. They have proven to perform well on realistic inputs. However, they involve special techniques to ensure termination and avoid non-determinism, and are thus rarely worst-case optimal for logics complete for deterministic complexity classes.

3.2 Automata-based algorithms

For several expressive description and modal logics, there exist optimal automata-based algorithms that decide satisfiability (and thus subsumption) of concepts, possibly w.r.t. a TBox or a universal role [Vardi & Wolper 1986; Streett & Emerson 1989; Vardi 1998; Calvanese *et al.* 1999a; Lutz & Sattler 2001; Sattler & Vardi 2001; Kupferman *et al.* 2002]: for a concept C and a TBox $\mathcal{T}$, we define an automaton $\mathcal{A}_{C,\mathcal{T}}$ which accepts exactly the (abstractions of) models of C and $\mathcal{T}$. Thus, the satisfiability problem is reduced to the emptiness problem of automata. In most cases, abstractions of models are finite or infinite trees—depending on the logic. Thus the target automata are automata on finite or infinite trees. Moreover, we can use deterministic, non-deterministic, or, as a generalisation, *alternating* automata, where the latter class of automata allows for rather elegant translations of many logics. In many cases, the emptiness test for non-deterministic automata is polynomial and the translation yields an automaton of size exponential in the input concept and TBox. In contrast, the translation into an alternating automaton usually yields an automaton of polynomial size (see, for example, [Vardi 1998; Calvanese *et al.* 1999a])—however, testing emptiness of (two-way and one-way) alternating automata is EXPTIME-complete [Kupferman & Vardi 1998; Vardi 1998]. In both cases, this reduction yields a worst-case optimal algorithm for EXPTIME-complete logics.

One advantage of this approach is that standard abstraction techniques such as *unravelling* [Thomas 1992] yield, in general, abstractions of models that are *infinite* trees. Using automata on infinite trees, we can directly work with these standard, infinite abstractions. This is a clear advantage for logics lacking the finite model property, and where it would be tedious to invent *finite* abstractions. In tableau algorithms, we had to work with *finite* representations of possibly infinite models to ensure termination, and used blocking to ensure termination. In contrast, for automata on infinite trees, termination is no issue since input trees are, by definition, infinite structures.

Another advantage of non-deterministic automata is that non-determinism due to disjunctions can be translated into non-deterministic transitions. For alternating automata, we can also translate "universal" quantification—e.g. due to conjunction—into the transition function. In the following, we sketch the construction of an alternating automaton for an $\mathcal{ALC}$-concept C and a TBox $\mathcal{T}$. We use a state q_D for each sub-concept D of C or $\mathcal{T}$. Nodes of input trees are labelled with sets of concept names and stand for elements of a model. The transition function δ takes a state (the current one) and a label (the one from the current node), and returns a positive Boolean formulae, where each atom can be read as "send a copy of the automaton in state q to the jth neighbour" or "send a copy of the automaton in state q to the current node". A tree is accepted by the automaton if there exists a "successful" run of the automaton on the tree. Such a run is, basically, a repeated sending of copies to nodes, each of which "satisfies" the corresponding Boolean formulae of the transition function. Let us start with the easiest case: for X a concept name, $\delta(q_X, \sigma) = \mathsf{true}$ if $X \in \sigma$, and false otherwise. Next, for conjunctions, $\delta(q_{D \sqcap E}, \sigma) = (0, q_D) \wedge (0, q_E)$, which sends, when being in state $q_{D \sqcap E}$ and reading a node label σ, one copy of the automaton in state q_D *and* one copy in state q_D to the same node ("0" stands for "to the same node"). Similarly, $\delta(q_{D \sqcup E}, \sigma) = (0, q_D) \vee (0, q_E)$, which sends one copy of the automaton in state q_D *or* one copy in state q_D to the same node. We assume the sub-concepts of C and $\mathcal{T}$ that are existential restrictions to be ordered linearly, and reserve the j-th successor of a node for the j-th existential restriction $\exists r.E$. Then

$$\delta(q_{\exists r.E}, \sigma) = (j, q_E) \wedge ((j, q_{\neg C_1}) \vee (j, q_{D_1})) \wedge \ldots \wedge ((j, q_{\neg C_m}) \vee (j, q_{D_m})),$$

which sends one copy of the automaton in state q_E to the j-th successor and, for each GCI $C_i \dot{\sqsubseteq} D_i$ in $\mathcal{T}$, either one copy in state $q_{\neg C_i}$ or one copy in state q_{D_i} to the j-th successor. The translation of universal restrictions is slightly more complicated since it depends on whether a node has a j-th successor, and thus omitted here. In summary, the description logic translates in a natural way into an automaton.

The main drawback of automata lies in the fact that their complexity is exponential not only in the worst case, but in every case: either the automaton $\mathcal{A}_C$ is exponential in $|C|$ or, in the case of alternating automata, is polynomial but is translated into a non-deterministic automaton $\mathcal{A}'_C$ of exponential size to decide its emptiness [Kupferman & Vardi 1998; Vardi 1998]. Therefore, a naive implementation is doomed to failure and, to the best of our knowledge, no optimised implementation is yet available. However, a first attempt to gain insight into such an implementation has been carried out successfully [Pan *et al.* 2002].

Correctness of automata-based algorithms is often proved as follows. Firstly, it is shown that each satisfiable concept (and TBox) has a tree model (or a tree abstraction of a model) and, in case tree abstractions are used, that each abstraction corresponds to a model. Then one shows that the automaton constructed for a concept C and a TBox $\mathcal{T}$ accepts exactly the (abstractions of) tree models of C w.r.t. $\mathcal{T}$.

In summary, automata-based approaches often allow for a very elegant and natural translation of a logic and provide EXPTIME upper complexity bounds and are thus optimal for EXPTIME-hard logics. Equally important, they handle infinite structures and non-determinism implicitly.

Thus, the advantages of tableau- and automata-based algorithms are rather complementary. In the past, it occurred quite often that first, a description logic was proven to be in EXPTIME using automata, and then a tableau-based algorithm for this logic was developed for implementation purposes. In this case, fixing tight complexity bounds for a DL is not only of theoretic interest, but also of a practical one: so far, no tableau-based decision procedure for NEXPTIME-complete DLs was implemented, but for a variety for EXPTIME-complete DLs. Thus proving a certain DL to be in EXPTIME might justify an attempt to design a "practical" tableau algorithm.

3.3 Other reasoning techniques

For certain DLs that are *not* propositionally closed such as the one used in the system CLASSIC [Patel-Schneider *et al.* 1991], one can use a reasoning technique called *structural subsumption*: roughly speaking, to decide the subsumption between two concepts C and D, both concepts are transformed into a certain normal form C' and D', and then subsumption can be decided by a syntactic comparison of C' and D', see Section 2.3.1 of [Baader *et al.* 2003]. This technique yields a polynomial time decision procedure for a sub-Boolean fragment of $\mathcal{ALC}$ with number restrictions, but seems to be applicable only to DLs without disjunction and existential restrictions.

Finally, the successful resolution-based theorem prover SPASS was modified into a decision procedure for expressive modal and description logics, then called MSPASS [Hustadt & Schmidt 2000]. Interestingly, it is well-suited for DLs extending $\mathcal{ALC}$ with Boolean operators on roles and can be extended to n-ary description logics [Georgieva *et al.* 2003].

4 DLs with Expressive Operators on Roles

In various applications such as engineering and medicine, *aggregated objects* play a central role, where we call an object aggregated if it is composed of various parts, which again can be composite, etc. It is natural to describe an aggregated object by means of its parts and, vice versa, to describe parts by means of the aggregate they belong to. For example, the following statements describe a control rod and a reactor core by means of their parts and wholes:

$$\begin{aligned}
\texttt{ControlRod} &\;\dot{\sqsubseteq}\; \texttt{Device} \sqcap \exists\texttt{partOf}.\texttt{ReactorCore}\\
\texttt{ReactorCore} &\;\dot{\sqsubseteq}\; \texttt{Device} \sqcap \exists\texttt{hasPart}.\texttt{ControlRod} \sqcap \exists\texttt{partOf}.\texttt{NuclReactor}
\end{aligned}$$

In contrast to, for example, the relation `likes`, the part-whole relation has a variety of properties; for a complete collection of these properties, we refer to [Simons 1987]. Most importantly, the general part-whole relation is a strict partial order, i.e., it is *transitive* and *asymmetric* (and hence irreflexive). That is, if x `partOf` y and y `partOf` z, then x `partOf` z, and if x `partOf` y, then not y `partOf` x. Moreover, an aggregated object has at least two parts where none is a part of the other. Next, we might consider to assume that two objects consisting of the same parts are identical. As a last example, we might assume the existence of atoms, i.e., indivisible objects of which all other objects are composed. This is equivalent to assuming that `hasPart` is well-founded and thus to excluding infinite chains x_0 `hasPart` x_1 `hasPart` $x_2 \ldots$.

Besides the properties mentioned above, it might be useful to distinguish various sub-relations of the part-whole relation such as, for example, the relation between a *component* and its *composite* (e.g. between a motor and the car the motor is in), the relation between *matter* and an *object* containing this matter (e.g. between metal and a car), or the relation between a *member* and a *collection* it belongs to (e.g. between a tree and the forest this tree belongs to). These sub-part-whole relations are subject to several investigations and discussions; see, for example, [Winston *et al.* 1987; Gerstl & Pribbenow 1995; Pribbenow 1995]. However, various questions concerning part-whole relations are still open, for example concerning the relationship between these sub-relations and their interaction.

4.1 Adding transitivity

Coming back to representing aggregated objects in ontologies using DLs, we observe that the DL $\mathcal{ALC}$ provides no means to express that a relation is *transitive*. For example, in $\mathcal{ALC}$, the concept

$$\mathtt{Device} \sqcap \exists \mathtt{hasPart}.(\mathtt{ReactorCore} \sqcap \exists \mathtt{hasPart}.\mathtt{ControlRod})$$

is *not* subsumed by

$$\mathtt{Device} \sqcap \exists \mathtt{hasPart}.\mathtt{ControlRod},$$

although the first concept is a specialisation of the second one under the assumption that `hasPart` is interpreted as a transitive relation.

Thus the correct modelling of aggregated objects asks for the extension of $\mathcal{ALC}$ with some form of transitivity. As mentioned in Section 2.4, there are at least two such possible extensions. One might argue that modelling aggregated objects requires a *direct*, non-transitive part-whole relation `hasDirectPart`. In this case, we need a DL with a transitive closure operator to also provide its transitive super-role `hasPart`. However, the way in which aggregated objects are decomposed strongly depends on the individual taste, aims, and circumstances. Continuing the previous example, we introduce the following GCIs concerning controlled reactors:

$$\mathtt{CReactor} \ \dot{\sqsubseteq} \ \mathtt{Reactor} \sqcap \exists \mathtt{hasDirectPart}.\mathtt{ControlRod}$$

as well as

$$\begin{aligned} \mathtt{CReactor} \ &\dot{\sqsubseteq} \ \mathtt{Reactor} \sqcap \exists \mathtt{hasDirectPart}.\mathtt{ControlledRCore} \text{ and} \\ \mathtt{ControlledRCore} \ &\dot{\sqsubseteq} \ \mathtt{RCore} \sqcap \exists \mathtt{hasDirectPart}.\mathtt{ControlRod} \end{aligned}$$

However, using all three GCIs yields models where a reactor has a control rod as a direct part, and where the control rod is also a direct part of the reactor's core—which clearly clashes with our intuition of *direct* part-whole relations. Thus the the transitive closure of roles is only required in case that all decompositions of aggregated objects are unique, and we have preferred to add transitive roles to $\mathcal{ALC}$.

By $\mathcal{S}$, we refer to the description logic $\mathcal{ALC}$ extended with transitive roles, i.e., where the set of of role names contains a subset of transitive role names $\mathbf{R}_+ \subseteq \mathbf{R}$.[6] Obviously, $\mathcal{S}$ provides the means to represent the general part-whole relation as a transitive relation by asserting that `partOf` is a transitive role. Additionally, since $\mathcal{S}$ has a tree model property, all satisfiable concepts and TBoxes have a model in which `partOf` is interpreted as a strict partial ordering.

Tableau algorithm for $\mathcal{S}$ A naive extension of the tableau algorithm for $\mathcal{ALC}$ sketched in Section 3.1 to transitive roles does not necessarily terminate, even without TBoxes: assume the algorithm is started with the concept

$$C_0 := C \sqcap \exists r.C \sqcap \forall r.(\exists r.C)$$

for r a transitive role. After some rule applications, the algorithm has generated three nodes, x, y, and z where y is an r-successor of x, z is an r-successor of y, and

$$\begin{array}{rcl} C_0, \forall r.(\exists r.C) & \in & \mathcal{L}(x) \\ \exists r.C & \in & \mathcal{L}(y) \\ C & \in & \mathcal{L}(z). \end{array}$$

Since r is a transitive role, we could make z an r-successor of x, but this would destroy the tree structure that turned out to be quite useful. Instead, we do something which has the same effect: we add $\forall r.(\exists r.C)$ to $\mathcal{L}(y)$. More precisely, if $\forall r.C \in \mathcal{L}(x)$ and x has an r-successor y, we add both C and $\forall r.C$ to y's label. In our example, this and the fact that z is an r-successor of y leads to $\exists r.C \in \mathcal{L}(z)$. It can easily be seen that the repeated application of this modification builds an infinite r-chain, and thus leads to non-termination. To re-gain termination without corrupting soundness or completeness of the algorithm, we use the blocking technique mentioned in Section 3.1: we stop generating new successors of a node z in case there is another node z' with $\mathcal{L}(z) \subseteq \mathcal{L}(z')$. In this case, we say that z' *blocks* z, and we can build a model by "merging" z and z' (and all other nodes which

[6]The logic $\mathcal{S}$ has previously been called $\mathcal{ALC}_{R^+}$, but this becomes too cumbersome when adding letters to represent additional features.

z' blocks), thus building a finite, possibly cyclic model [Horrocks & Sattler 1999].

4.2 Further adding inverse roles

When modelling aggregated objects using $\mathcal{S}$ and using role names `partOf` and `hasPart`, we might end up with an inadequate representation: for example, extending the TBox in the beginning of Section 4 with

$$\texttt{NuclReactor} \sqcap \exists\texttt{hasPart}.\texttt{Faulty} \mathrel{\dot{\sqsubseteq}} \texttt{Dangerous},$$

we would assume that

$$\texttt{ControlRod} \sqcap \texttt{Faulty} \text{ is subsumed by } \exists\texttt{partOf}.\texttt{Dangerous}$$

w.r.t. to this TBox—which is only the case if `partOf` was *the inverse* of `hasPart`, i.e., if $\langle x, y\rangle \in \texttt{hasPart}^{\mathcal{I}}$ iff $\langle y, x\rangle \in \texttt{partOf}^{\mathcal{I}}$. Thus we can choose between the following three options: we

1. use either `partOf` or `hasPart`, but not both,
2. use `partOf` and `hasPart` and live with the fact that our model is not adequate in the above sense and we thus might lose inferences, or
3. extend $\mathcal{S}$ with inverse roles, see Section 2.4.

The third options yields a description logic called $\mathcal{SI}$, which allows to describe both objects by means of the wholes they belong to and by means of the parts they have. Substituting `hasPart` by $\texttt{partOf}^-$ in the last example yields a TBox with respect to which $\texttt{ControlRod} \sqcap \texttt{Faulty}$ is indeed subsumed by $\exists\texttt{partOf}.\texttt{Dangerous}$.

Tableau algorithm for $\mathcal{SI}$ Intuitively, we can extend the tableau algorithm for $\mathcal{S}$ as follows to yield a decision procedure for satisfiability of $\mathcal{SI}$-concepts [Horrocks & Sattler 1999; Horrocks *et al.* 1999]: if $\forall r.C \in \mathcal{L}(w)$, in addition to adding C to the label of r-successors, we also add C to *r-predecessors.*[7] For example, for the concept $\exists r^-.(C \sqcap \forall r.B) \in \mathcal{L}(x)$, we would first create an r^--successor y of x with $C \sqcap \forall r.B \in \mathcal{L}(y)$. For $\forall r.B \in \mathcal{L}(y)$, we would then add B to $\mathcal{L}(x)$ since x is an r-predecessor of y (because y is an r^--successor of x). Moreover, the blocking condition has to be more

[7] Here, r-predecessors might seem to be defined the wrong way round, but this notation turned out to be useful.

strict: for z' to block z, they must have identical labels, i.e., $\mathcal{L}(z) = \mathcal{L}(z')$. Finally, blocking becomes necessarily "dynamic": in the presence of inverse roles, node labels influence each other up and down the completion tree. Thus the label of a node x blocking some node y further down the tree can change due to some of its other successors, the node labels of x and y become different, and we must "unblock" them.

This tableau algorithm decides satisfiability (and thus subsumption) of $\mathcal{SI}$-concepts w.r.t. TBoxes. Moreover, we were able to prove that, in the absence of a TBox and employing a certain strategy and a more intricate blocking condition, it uses polynomial space only. This is one example for the fact that the definition of the blocking condition is not only crucial for the correctness of the algorithm, but also for its complexity. As a consequence, $\mathcal{ALC}$ without TBoxes and with transitive and inverse roles is of the same complexity as pure $\mathcal{ALC}$, namely PSPACE-complete [Horrocks *et al.* 1999].

4.3 Further adding role inclusion axioms

If we want to use, beside the general part-whole relation, certain sub-part-whole relations such as "is a component of" or "is an ingredient of", we can use role hierarchies [Horrocks & Gough 1997], as defined in Section 2.4. The extension of $\mathcal{SI}$ with role hierarchies is called $\mathcal{SHI}$.

Adding role hierarchies to $\mathcal{SI}$ has mainly two consequences: firstly, we can introduce (possibly transitive—depending on the additional relation) role names such as `hasComponent` or `hasIngredient` and add role inclusion axioms

$$\begin{aligned} \texttt{hasComponent} &\;\dot{\sqsubseteq}\; \texttt{hasPart} \text{ and} \\ \texttt{hasIngredient} &\;\dot{\sqsubseteq}\; \texttt{hasPart}. \end{aligned}$$

This turns out to be quite useful in various applications since it allows for a concise and natural description not only of aggregated objects.

Secondly, $\mathcal{SHI}$ (as well as $\mathcal{SH}$ and $\mathcal{SHIQ}$) has the expressive power for the *internalisation* of TBoxes [Baader 1991; Horrocks & Sattler 1999]. This technique polynomially reduces reasoning w.r.t. a general TBox to pure concept reasoning as follows. We introduce a new transitive role name $u \in \mathbf{R}_+$ and specify that u is a super-role of all roles and their respective inverses. This implies that, in connected models, u behaves like a universal role, i.e., u relates all elements of the interpretation domain; cf. Section 2.4. Since each satisfiable $\mathcal{SHI}$ concept is satisfiable in a connected model, it can be shown that a concept C is satisfiable w.r.t. $\{C_i \;\dot{\sqsubseteq}\; D_i \mid 1 \leq i \leq n\}$ iff $\exists u.C \sqcap \forall u. \bigsqcap_{1 \leq i \leq n} (C_i \Rightarrow D_i)$ is satisfiable.

Tableau algorithm for $\mathcal{SHI}$ The tableau algorithm for $\mathcal{SI}$ and TBoxes can be extended to $\mathcal{SHI}$ as follows [Horrocks & Sattler 1999]. Basically, it involves an adaption of the notion of an "r-successor" to take into account role hierarchies: if y is an r-successor of x and $r \mathrel{\dot\sqsubseteq} s$ is in the role hierarchy, then y is also an s-successor of x. An analogous adaption for predecessors is also required in the presence of inverse roles, and transitive roles require a further, rather complex adaption of the propagation of universal restrictions $\forall r.C$. Moreover, the correctness proof of the tableau algorithm becomes more complex since the tree structure the algorithm works on does no longer correspond to the relational structure that is to be built in case that the algorithm answers "satisfiable": this is already the case in the presence of transitive roles, but becomes more notable if, additionally, role hierarchies are taken into account. For $\mathcal{SI}$, the tree structure was only missing "transitively implied" edges. For $\mathcal{SHI}$, these as well as those edges implied by the role hierarchy are possibly missing.

4.4 Further adding number restrictions

In general, when describing the relevant concepts of an application domain, it seems to be natural to describe an object by restricting the number of objects it is related to via a certain relation. For example, the following are concept definitions for pipes and forks:

$$\begin{array}{lcl}\texttt{Pipe} & \doteq & \texttt{Connection} \sqcap (= 1\ \texttt{partOf}^-\ \texttt{Input}) \sqcap (= 1\ \texttt{partOf}^-\ \texttt{Output}) \\ \texttt{Fork} & \doteq & \texttt{Connection} \sqcap (= 1\ \texttt{partOf}^-\ \texttt{Input}) \sqcap (\geq 2\ \texttt{partOf}^-\ \texttt{Output})\end{array}$$

Before adding number restriction to $\mathcal{SHI}$, we have to define *simple* roles since only simple roles are allowed in number restrictions—without that restriction, satisfiability of $\mathcal{SHI}$ extended with number restrictions is undecidable [Horrocks *et al.* 1999].

A (possibly inverse) role is called *simple* if it is neither transitive nor has a transitive sub-role.

$\mathcal{SHIQ}$ is obtained from $\mathcal{SHI}$ by allowing, additionally, for concepts of the form $(\geqslant nR.C)$ and $(\leqslant nR.C)$ for n a non-negative integer, R a simple role, and C a $\mathcal{SHIQ}$-concept. The semantics of number restrictions is given in Section 2.4.

In contrast to $\mathcal{SHI}$, $\mathcal{SHIQ}$ lacks the finite model property. That is, there are concepts that are satisfiable, but only in *infinite* models. For example, for r a transitive role and $s \mathrel{\dot\sqsubseteq} r$, each model of the following concept contains an infinite, acyclic r-chain:

$$\neg A \sqcap \exists s.A \sqcap \forall r.((\exists s.A) \sqcap (\geqslant 1 s^-.\top)).$$

As mentioned in Section 2.2, state-of-the-art DL reasoners FaCT and Racer implement tableau algorithms for $\mathcal{SHIQ}$ [Horrocks 1998b; Haarslev & Möller 2001]. Thus, $\mathcal{SHIQ}$ forms the logical basis of ontology editors Rice and Oiled [Cornet 2003; Bechhofer *et al.* 2001], and of the intelligent conceptual modelling tool Icom [Franconi & Ng 2000].

Tableau algorithm for $\mathcal{SHIQ}$ It is not difficult to see that, in the presence of number restrictions, we have to add two new rules to our tableau algorithm:

1. if $(\geqslant nr.C) \in \mathcal{L}(x)$ and x has less than n r-neighbours with C in their label, then generate these missing r-neighbours and set their labels to $\{C\}$.
2. if $(\leqslant nr.C) \in \mathcal{L}(x)$ and x has more than n r-neighbours with C in their label, then merge some of them, so that only n remain.

However, this is not sufficient. Firstly, such a naive extension might easily yield a "yo-yo" effect: for example, if applied to a node x with $(\geqslant 3r.(C \sqcap D)), (\leqslant 2r.C) \in \mathcal{L}(x)$, the above tableau algorithm would generate three r-successors y_i with $C \sqcap D \in \mathcal{L}(y_i)$, break down the conjunctions $C \sqcap D \in \mathcal{L}(y_i)$, and then notice that there are too many r-successors y_i of x with $C \in \mathcal{L}(y_i)$ for $(\leqslant 2r.C) \in \mathcal{L}(x)$. Thus two of them would be merged into a single one. Now there are not enough r-successors for $(\geqslant 3r.(C \sqcap D))$, so one would be generated, and so on, thus leading to non-termination. To regain termination, we can use, for example, an explicit inequality relation $\neq$ that prevents nodes that were introduced for one $(\geqslant nr.C)$ from being merged again later. Moreover, we extend the notion of a "clash" to cases where $(\leqslant nr.C) \in \mathcal{L}(x)$ and x has more than n $\neq$-distinct r-successors with C in their labels.

Secondly, consider the concept

$$C := (\geqslant 3r.B) \sqcap (\leqslant 1r.A) \sqcap (\leqslant 1r.\neg A).$$

So far, for $C \in \mathcal{L}(x)$, the tableau algorithm would generate three r-successors y_i of x with $\{B\} = \mathcal{L}(y_i)$, and stop with the answer "C is satisfiable". However, the concept C is obviously unsatisfiable: the algorithm's unsoundness is due to its ignorance of which of the y_i are instances of A and which are instances of $\neg A$. To overcome this problem, we add a third rule

3. if $(\leqslant nr.C) \in \mathcal{L}(x)$ and y is an r-neighbour of x, then non-deterministically add C or $\neg C$ to $\mathcal{L}(y)$.

Thirdly, one also needs to modify the blocking condition—otherwise, the algorithm would still be unsound. Roughly spoken, the $\mathcal{SHIQ}$ blocking condition involves two pairs of subsequent nodes whose labels must coincide pairwise. Together, these three modifications indeed yield a decision procedure for the satisfiability of $\mathcal{SHIQ}$ [Horrocks *et al.* 1999].

Interestingly, the first proposal of the $\mathcal{SHIQ}$ blocking condition was so strict that it delayed blocking severely, thus enlarging the search space for a model dramatically and degrading the performance of FaCT. Investigating the soundness and completeness proof of the $\mathcal{SHIQ}$ tableau algorithm more closely, we were able to devise an new blocking condition which still ensures soundness, completeness, and termination, but was less strict [Horrocks & Sattler 2002]. Intuitively, node labels had only to be equal for "relevant concepts" in the respective nodes, a fact that made the formulation of the new blocking condition (and its testing in FaCT) rather intricate. However, an empirical evaluation of the new tableau algorithm in FaCT showed that this more intricate but less strict blocking condition pays off: it improves performance up to two orders of magnitude.

Concerning worst-case complexity, both the original and the optimised $\mathcal{SHIQ}$ tableau algorithm are far from being optimal: in the worst case, they run in 2NExptime, whereas satisfiability of $\mathcal{SHIQ}$-concepts is known to be in ExpTime, even with numbers in number restrictions coded in binary [Tobies 2001a]. Despite this worst-case sub-optimality, its implementation in the FaCT and Racer systems behave surprisingly well in practise [Horrocks & Sattler 2002; Haarslev & Möller 2001]. However, the worst-case complexity implies that there exist rather small example inputs for which these systems need so much time that they are practically not terminating [Berardi *et al.* 2001]. To investigate the nature of input concepts and TBoxes on which these system perform well and those where they fail is part of future work.

4.5 Further adding more expressive role inclusion axioms

Although $\mathcal{SHIQ}$ is rather expressive, there is a common phenomenon that $\mathcal{SHIQ}$ is not able to express, and that would be useful for many applications, especially for those involving aggregated objects. This phenomenon is often coined *propagation of properties*: for example, one wants to express that a fracture located in the shaft of the femur (which is a division of the femur) is a fracture located in the femur. Or one might want to express that the owner of a thing also owns the parts of this thing. The importance of this expressive means is illustrated by the fact that the Grail DL [Horrocks *et al.* 1996; Rector *et al.* 1997], which was designed for medical terminologies, is able to express these propagations (although it is quite weak in other respects). In two other medical terminology applications, rather complex workarounds to represent propagations can be found: SEP-triplets[8] in [Schulz & Hahn 2001] and right-identities in [Spackman 2000]. Finally, the CycL knowledge representation language provides the `transfersThro` statement for similar propagations [Lenat & Guha 1989]. So far and to the best of our knowledge, none of these systems were proven to handle these propagations in a sound and complete way.

It is rather straightforward to extend $\mathcal{SHIQ}$ to allow for the propagation of properties: obviously, it suffices to extend role hierarchies to the general role inclusion axioms (g-RIAs, see Section 2.4) of the form $r \circ s \dot{\sqsubseteq} t$ and to require that a model satisfies $r^{\mathcal{I}} \circ s^{\mathcal{I}} \subseteq t^{\mathcal{I}}$ for each $r \circ s \dot{\sqsubseteq} t$ in a role hierarchy (where $\circ$ is standard composition of binary relations). For the first example, one would introduce an axiom $\mathtt{hasLocation} \circ \mathtt{divisionOf} \dot{\sqsubseteq} \mathtt{hasLocation}$ and, indeed, w.r.t. this axiom,

$$\mathtt{Fracture} \sqcap \exists \mathtt{hasLocation}.(\mathtt{Shaft} \sqcap \exists \mathtt{isDivisionOf}.\mathtt{Femur})$$

is subsumed by

$$\mathtt{Fracture} \sqcap \exists \mathtt{hasLocation}.\mathtt{Femur}.$$

For the second example, one would introduce an axiom $\mathtt{owns} \circ \mathtt{hasPart} \sqsubseteq \mathtt{owns}$ and, w.r.t. this axiom,

$$\exists \mathtt{owns}.(\mathtt{Bicycle} \sqcap \exists \mathtt{hasPart}.\mathtt{SuspensionFork})$$

is subsumed by

$$\exists \mathtt{owns}.\mathtt{SuspensionFork}.$$

As mentioned in Section 2.4, results in grammar and description logics imply that extending $\mathcal{ALC}$ with role inclusion axioms of the form $r \circ s \dot{\sqsubseteq} t$ yields a logic for which satisfiability and subsumption are undecidable [Baldoni 1998; Baldoni *et al.* 1998; Wessel 2001]. However, for expressing propagation of properties, we only need axioms of the form $r \circ s \dot{\sqsubseteq} s$ or $r \circ s \dot{\sqsubseteq} r$ [Horrocks *et al.* 1996; Rector 2002]. Unfortunately, extending $\mathcal{SHIQ}$ with this restricted form of axioms still yields an undecidable logic [Horrocks & Sattler 2003]. This is a rather surprising result since, roughly spoken, the former undecidability is due to the close relationship between axioms $r \circ s \dot{\sqsubseteq} t$ and context-free production rules $t \rightarrow rs$ in grammars. A set of axioms $r \circ s \dot{\sqsubseteq} t$ can be viewed as a context-free grammar in Chomsky normal

[8] *SEP-triplets* are used both to compensate for the absence of transitive roles in $\mathcal{ALC}$, and to express the propagation of properties across a distinguished "part-of" role.

form, which then can be used to prove undecidability via a reduction of the intersection problem for context-free grammars to the satisfiability of concepts. Hence the restriction to axioms of the form $r \circ s \dot{\sqsubseteq} s$ or $r \circ s \dot{\sqsubseteq} r$ is seemingly severe, but still yields an undecidable logic.

One way to re-gain decidability would be to restrict the underlying logic $\mathcal{SHIQ}$. Since we have argued that, especially for the representation of aggregated objects, the concept- and role-forming operators of $\mathcal{SHIQ}$ are crucial, we have chosen a different approach, namely to further restrict the role inclusion axioms: further restricting role hierarchies to not contain "affecting cycles" of length greater than one finally yields a decidable logic. Roughly speaking, "affecting" is the transitive closure of the relation "directly affecting", and r directly affects s if $r \circ s \dot{\sqsubseteq} s$, $s \circ r \dot{\sqsubseteq} s$, or $r \dot{\sqsubseteq} s$ is contained in the role hierarchy. A role hierarchy containing no "affecting" cycles of length greater than one is called *acyclic*, and the extension of $\mathcal{SHIQ}$ with acyclic role hierarchies is called $\mathcal{RIQ}$.

Thus in $\mathcal{RIQ}$, we can model the propagation of properties as mentioned above, and the restriction to acyclicity does not seem to be too severe since non-trivial cycles seem to indicate modelling flaws [Rector 2002].

Tableau algorithm for $\mathcal{RIQ}$ The tableau algorithm for $\mathcal{RIQ}$ [Horrocks & Sattler 2003] involves two pre-processing steps that transform the role hierarchy into a more explicit and manageable structure. Firstly, acyclic role hierarchies are unfolded in a similar way as acyclic TBoxes can be unfolded [Nebel 1990], thus making all implicit implications explicit. As a result of this unfolding, we obtain, for each role name r, a regular expression τ_r on role names. Secondly, we construct, for each τ_r, a non-deterministic finite automaton $\mathcal{A}_r$ which accepts $L(\tau_r)$.

Then, in the tableau rules, we add three rules

1. if $\forall r.C \in \mathcal{L}(x)$, then we add $\forall \mathcal{A}_r.C$.

2. if $\forall \mathcal{A}.C \in \mathcal{L}(x)$ and x has an s-successor y, then we add $\forall \mathcal{A}'.C$ to $\mathcal{L}(y)$ for each automaton $\mathcal{A}'$ that is the result of $\mathcal{A}$ reading s, i.e., $\mathcal{A}'$ is obtained from $\mathcal{A}$ by simply changing the initial state to a state that is reachable from $\mathcal{A}$'s initial state by an s transition.

3. if $\forall \mathcal{A}.C \in \mathcal{L}(x)$ and $\varepsilon \in L(\mathcal{A})$, then add C to $\mathcal{L}(x)$.

The pre-processing together with these three rules yield a decision procedure for $\mathcal{RIQ}$.

Unfortunately, the first pre-processing step, i.e., the unfolding of the role hierarchy, may yield an exponential blow-up—similar to the one for acyclic TBoxes [Nebel 1990]. The second pre-processing step is less complex: since we use non-deterministic automata, $\mathcal{A}_r$ is polynomial in τ_r and can be constructed in polynomial time. Thus the $\mathcal{RIQ}$ tableau algorithm is of the same complexity as the one for $\mathcal{SHIQ}$, provided that the input role hierarchy is already unfolded. So far, we have defined a property on role hierarchies that implies that the unfolding is polynomial, i.e., that avoids this blow-up. Currently, we are investigating whether this blow-up can always be avoided.

This tableau algorithm for $\mathcal{RIQ}$ is implemented successfully in FaCT. The additional overhead introduced by using automata in tableau rules seems to pay off since it does not degrade the performance of FaCT, yields a more comprehensible algorithm, and can draw additional inferences like the one from the examples above [Horrocks & Sattler 2003].

5 DLs with Nominals

Nominals were introduced in Section 2.4 and provide interesting expressiveness for ontology languages. For example, the web ontology language OWL proposed by the W3C Web-Ontology Working Group is based upon the extension of $\mathcal{SHIQ}$ with nominals. Coming back to the Brent Spar example, we can define a concept BSP for those devices that are part of Brent Spar by

$$\texttt{BSP} \doteq \texttt{Device} \sqcap \exists \texttt{partOf}.\texttt{BrentSpar},$$

and the following two concepts D_1 and D_2:

$$\begin{aligned} D_1 &\doteq \texttt{BSP} \sqcap \forall \texttt{partOf}.(\texttt{BrentSpar} \Rightarrow \forall \texttt{hasPart}.\texttt{Harmless}) \\ D_2 &\doteq \forall \texttt{connectedTo}.(\texttt{BSP} \Rightarrow \texttt{Harmless}) \end{aligned}$$

Now, the subsumption relationship between D_1 and D_2 w.r.t. the definition of BSP depends on the number of instances of BrentSpar:

- if BrentSpar is a concept name, $D_1 \sqcap \neg D_2$ is satisfiable, i.e., D_1 is *not* subsumed by D_2. However, in each model of $D_1 \sqcap \neg D_2$, there are at least two instances of BrentSpar. As a consequence,

- if BrentSpar is a nominal, then $D_1 \sqcap \neg D_2$ is *not* satisfiable, i.e., D_1 *is* subsumed by D_2 w.r.t. the definition of BSP.

Hence, using nominals, we can not only model individuals adequately, but also infer additional subsumption relationships. Moreover, it turned out that hybrid logics are well-suited for the internalisation of deduction in proof systems [Blackburn 2000].

Thus nominals are an interesting expressive means—unfortunately, they often have a dramatic effect on the computational complexity. As mentioned in Section 2.4, adding nominals to

- $\mathcal{ALC}$ with inverse roles and without TBoxes increases the complexity from PSPACE-completeness to EXPTIME-completeness [Areces *et al.* 1999];
- $\mathcal{ALC}$ with inverse roles, number restrictions, and TBoxes increases the complexity from EXPTIME-completeness to NEXPTIME-completeness [Tobies 2001a];

One reason for this effect is surely the loss of the tree model property [Vardi 1997]. This property is enjoyed by most description and modal logics, and seems to be one reason for their "robust" decidability; for example, even the extension of many description or modal logics with expressive means as powerful as fixpoint operators preserves their decidability—possibly because it also preserves the tree model property. However, for many logics with nominals, it is possible to define, instead of *the* tree model property, an appropriate, less strict tree model property. As sketched in Section 3.2, we can abstract from non-tree models to tree abstractions of models, and then use these tree abstractions in (automata or tableau-based) reasoning algorithms. In the following, we will show three examples of logics with nominals where this is possible. For a better readability, we use DL terms, e.g., roles instead of programs or modal parameters, even though the μ-calculus and information logics are described in different terms.

5.1 The hybrid μ-calculus

The μ-calculus [Kozen 1982; Streett & Emerson 1989] was originally designed to describe the behaviour of programs, but turned out to be a useful super-logic of various other EXPTIME logics: besides $\mathcal{SH}$, many expressive description logics [De Giacomo & Lenzerini 1994a; 1994b], the propositional dynamic logic (PDL) [Fischer & Ladner 1979], and various temporal logics [Emerson 1990] are fragments of the μ-calculus. It can easily be seen that the μ-calculus provides the expressive power to *internalise* TBoxes (see the TBox paragraph of Section 2.4).

The μ-calculus is the extension of $\mathcal{ALC}$ (or multi modal $\mathbf{K}$, see Section 2.3) with least and greatest fixpoint operators. For example, the least fixpoint concept $\mu X.(P \sqcup \exists r.X)$ is satisfied by all those individuals that have an r path to an instance of P, and thus equivalent to $\exists r^*.P$. The greatest fixpoint concept $\nu X.\exists r.X$ is satisfied by all those individuals that are on an infinite r path.

In [Vardi 1998], it was shown that the μ-calculus can be extended with inverse roles and features (axioms of the form $\top \doteq (\leqslant 1 r.\top)$) while remaining in ExpTime. Despite its high expressive power, the μ-calculus does not provide nominals and, since it has the tree model property, it cannot express nominals. In [Sattler & Vardi 2001], we have shown that the *hybrid* μ-calculus, i.e., the extension of the μ-calculus with inverse roles and nominals, remains in ExpTime. Since the only known decision procedures for the μ-calculus are automata-based, we also used automata for the hybrid μ-calculus.

To overcome the loss of the tree model property, we devise *a* suitable tree model property for this logic, i.e., we define suitable, tree-shaped abstractions of non-tree models. Another difficulty to overcome was due to the "global" character of nominals: since a nominal N can be reached in one step from each individual in a model using $\exists r.(N \sqcap \ldots)$, the information about which concepts a nominal is an instance of and how nominals are inter-related becomes "global". Moreover, all nodes that have a nominal N as an r-successor can be reached from N in one step using $\forall r^-.D$. Now automata can only work locally, i.e., on one node, its successors, and, if the automaton is two-way, its predecessors. We "localise" the "global" information concerning nominals in a *guess*: a piece of information that is passed from node to node without being changed, and to which we apply a sort of "book-keeping" to ensure that the local information conforms with the global guess. Employing this technique, we can define, for a hybrid μ-calculus concept φ, an alternating two-way automaton on infinite trees $\mathcal{A}_\varphi$ that accepts exactly the tree abstractions of φ's models. Thus we have reduced the satisfiability problem of the hybrid μ-calculus to the emptiness problem of automata: each tree accepted by $\mathcal{A}_\varphi$ can be "folded" into a model of φ and, vice versa, each model of φ can be unravelled into a tree structure that is accepted by $\mathcal{A}_\varphi$.

As a consequence, any logic that is a fragment of the μ-calculus can be extended with nominals while remaining in ExpTime.

5.2 Information logics

Information logics are designed to model and reason about *information systems* [Pawlak 1981; Orłowska 1998]. Most information logics combine expressive means whose combination are known to make reasoning rather difficult such as a form of intersection of roles, the universal role (see Section 2.4), and nominals. Among the class of information logics, the logic SIM introduced in [Konikowska 1997] plays a special role since it provides various expressive ingredients: it provides Boolean operators on roles and nominals at the level of concepts and at the level role expressions. This highly expressive logic was designed to represent and reason about relevant properties of *similarity relations* induced by a set of *attribute names* $\mathcal{A}$: we only consider role names r_A for $A \subseteq \mathcal{A}$ a set of attribute names, and require interpretations $\mathcal{I}$ to

- interpret each role r_A as a reflexive symmetric binary relation $r_A^{\mathcal{I}}$,
- satisfy $r_\emptyset = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and
- satisfy $r_{a \cup b}^{\mathcal{I}} = r_a^{\mathcal{I}} \cap r_b^{\mathcal{I}}$.

Like other logics with nominals, SIM lacks *the* tree model property, but it is possible to define tree abstractions of (non-tree) models. Thus, for a SIM concept φ, one can define a Büchi tree automaton $\mathcal{A}_\varphi$ that accepts all tree abstractions of models of φ, and thus reduce the satisfiability problem for SIM to the emptiness problem for Büchi tree automata. Using a certain normal form for modal expressions and a similar localisation of global information as for the hybrid μ-calculus, it is possible to define $\mathcal{A}_\varphi$ such that its size is only exponential in the length of φ. Since emptiness of Büchi tree automata can be decided in polynomial time in the number of the automaton's states [Vardi & Wolper 1986], satisfiability of SIM concept is in ExpTime [Demri & Sattler 2002]. It is ExpTime-complete since it contains multi modal logic **K** with the universal modality, which is known to be ExpTime-hard [Spaan 1993a].

5.3 Nominals for DLs with rich roles

In [Horrocks & Sattler 2001], we introduce the description logic $\mathcal{SHOQ}(D)$, i.e., $\mathcal{SHIQ}$ with nominals and *concrete datatypes*, but without inverse roles. Concrete datatypes are a rather trivial restriction of concrete domains (see [Lutz 2003]) and are merely useful syntactic sugar. In contrast, nominals provide interesting, additional expressive power.

The tableau algorithm for $\mathcal{SHOQ}(D)$ [Horrocks & Sattler 2001] is, roughly spoken, obtained by modifying the one for $\mathcal{SHIQ}$ as follows: for each nominal N, we have one "distinguished" node x_N with $N \in \mathcal{L}(x)$, and all other nodes y with $N \in \mathcal{L}(y)$ are blocked by x_N. Moreover, we make sure that $\mathcal{L}(y) \subseteq \mathcal{L}(x_N)$, for each un-distinguished node y with $N \in \mathcal{L}(y)$. In this way, if we do not take into account incoming edges of distinguished nodes, we can still work on a forest, i.e., a set of trees. This technique is similar to the one employed for the localisation of global information introduced for the hybrid μ-calculus [Sattler & Vardi 2001] described above. Again, we can prove soundness, completeness, and termination of this algorithm, and thus show that it decides satisfiability and subsumption of $\mathcal{SHOQ}(D)$-concepts w.r.t. TBoxes.

In the remainder of this section, we explain why we have chosen to add nominals to the restriction of $\mathcal{SHIQ}$ without inverse roles, and not to full $\mathcal{SHIQ}$. In the following, we use $\mathcal{SHIQO}$ for the extension of $\mathcal{SHIQ}$ with nominals. Firstly, as a consequence of the NExptime-hardness of $\mathcal{ALC}$ with inverse roles, number restrictions, and TBoxes [Tobies 2001a], $\mathcal{SHIQO}$ is NExptime-hard—in contrast to $\mathcal{SHIQ}$ which is Exptime-complete. Secondly and equally important, all attempts to designing a "goal-directed" decision procedure for $\mathcal{SHIQO}$ failed: by "goal-directed", we refer to any algorithm that decides satisfiability in a more sophisticated way than guessing a model of exponential size and then performing model checking.

As mentioned in Section 4.2, in the presence of inverse roles, blocking is dynamic, i.e., established blocks are possibly broken. And if x blocks y, then x will play the role of y when constructing a model from a clash-free completion tree. So in a completion tree or forest, blocking yields one form of "identity" between nodes. Now, in the presence of nominals, a second kind of identity has to be considered: consider, for example, the concept

$$\begin{aligned} C := \; & \exists r_1.\exists r_2.\ldots\exists r_n.C_1 \sqcap \\ & \exists r_1.\exists r_2.\ldots\exists r_n.C_2 \sqcap \\ & \forall r_1.\forall r_2.\ldots\forall r_n.N \sqcap \\ & \forall r_1.((\leqslant 1 r_1^-.\top) \sqcap \forall r_2.((\leqslant 1 r_2^-.\top) \sqcap \forall r_3.\ldots\forall r_{n-1}.(\leqslant 1 r_n^-.\top))\cdots). \end{aligned}$$

A tableau algorithm would start with $C \in \mathcal{L}(x)$, then build $r_1.\ldots.r_i$-successors y_i of x with $C_1 \in \mathcal{L}(y_n)$ and $r_1.\ldots.r_i$-successors z_i of x with $C_2 \in \mathcal{L}(z_n)$. When applying the $\forall$-rules, the algorithm learns that y_n and z_n denote the same object. But then, due to the other atmost restrictions, also y_{n-1} and z_{n-1} must denote the same object, and so on, until y_1 and z_1. This "zipper" behaviour yields the second form of identities that a tableau algorithm would need to take into account. Now suppose y_i blocks some node w which is an r_i-successor of v. Thus the first identity yields that y_i and w also represent the same object, and the atmost restriction in $\mathcal{L}(y_i)$ implies that y_{i-1} and v represent the same object. To capture this identity, one would need to merge the node labels of y_i and w and of y_{i-1} and v (or do something similar). However, the block between y_i and w may be broken later—which would mean that this label merging and its effects had to be undone. It is an interesting question whether the implementation of such an algorithm can decide the satisfiability of any non-trivial input: its design and empirical evaluation are part of future work.

6 Summary and Outlook

As argued above, to be of use in realistic applications, a logic-based ontology language has to provide adequate expressive power and useful system services. Thus we are concerned with the well-known trade-off between expressive power and computational complexity, and designing useful ontology languages consists, to a considerable amount, in finding a "good" compromise for this trade-off. In the last years, it turned out that such a good compromise needs two sorts of theoretical results: on the one hand, tight worst-case complexity bounds for the reasoning problems these system services are based upon give insight into the intrinsic difficulty of reasoning for a certain ontology language. On the other hand and equally important, practical reasoning algorithms are to be developed and their amenability to optimisation techniques is to be evaluated. It is commonly agreed that being below a certain complexity is necessary for practicability, but that different logics can behave quite differently even though they are of the same worst-case complexity. Interestingly, the complexity class that is believed to be practical changed during the last decade from polynomial time (with the development of Classic [Patel-Schneider *et al.* 1991]) over polynomial space (with the development of Kris [Baader *et al.* 1994]) to exponential time (with the development of FaCT and Racer [Horrocks 1998b; Haarslev & Möller 2001]).

In this thesis, we have described the development of practical reasoning algorithms for a family of description logics, starting from $\mathcal{ALC}$ and, when they turned out to be amenable to optimisations but still lacking expressive power for certain applications, continuing with more expressive logics. This development led to the description logic $\mathcal{RIQ}$, which is the extension of $\mathcal{ALC}$ (a syntactic variant of multi-modal $\mathbf{K}$) with inverse roles, number restrictions, transitive roles, role hierarchies, and general role inclusions of the form $r \circ s \rightarrow s$ and $r \circ s \rightarrow r$. $\mathcal{RIQ}$ has been implemented successfully in the DL system FaCT and proved to behave well in a realistic, medical terminology application: the tableau-based reasoning algorithms performs well and draws useful inferences that none of its predecessors were able to draw.

Moreover, we have investigated the computational complexity of and developed tableau algorithms for *hybrid* description logics, i.e., those providing nominals. This is an expressive means that seems to be required in various applications, but is rarely provided by DLs, one reason being that its presence destroys the tree model property. This property is one reason for description and modal logics to be rather robustly decidable [Vardi 1997]. However, we were able to devise, for three hybrid logics, *a* tree model property, even though these logics lack *the* tree model property: in all three cases, it is possible to define tree-shaped abstractions of non-tree models. This allows to devise practical, tableau-based and/or optimal automata-based reasoning algorithms for these logics. The latter yields, for the hybrid μ-calculus and SIM, a tight EXPTIME upper complexity bound. Since the μ-calculus is already EXPTIME-complete, we can thus extend it with nominals "for free". This behaviour is not always the case: there are EXPTIME-complete logics that become NEXPTIME-complete when extended with a single nominal [Tobies 2001a]. Thus we can now better distinguish those DLs that can be extended with nominals without increasing their complexity from those where such an extension yields a more complex logic. For example, $\mathcal{SH}$ with number restrictions is a logic of the former kind, and the design of $\mathcal{SHOQ}(D)$ and its tableau algorithm was motivated by this insight.

These investigations have an interesting impact: currently, industrial strength systems are being developed for expressive, EXPTIME-complete fragments of $\mathcal{SHIQ}$[9] whereas, for a long time, the only industrial-strength DL system, CLASSIC [Patel-Schneider *et al.* 1991], was based on a polynomial time DL. Moreover, the proposal for an ontology language standard for the Semantic Web proposed by the W3C Web-Ontology Working Group,[10] OWL, is based on a description logic, namely the union of $\mathcal{SHIQ}$ and $\mathcal{SHOQ}(D)$.

Despite this success, there are still various interesting open questions concerning the logical foundations of DL-based systems, three of which are sketched here as parts of future work. Firstly, there is no clear explanation of why DL reasoners behave so well on various "natural" ontologies, and which properties of ontologies are responsible for a good performance or its degradation. So far, there are some hints that TBoxes from a graphical user interface such as a UML editor (see Section 2.3) are more likely to degrade the system's performance than those from textual or frame-based

[9]See `http://www.networkinference.com/`.

[10]`http://www.w3.org/2001/sw/WebOnt/`

editors [Berardi *et al.* 2001; Horrocks & Sattler 2002]. One could speculate that, using a graphical interface, one introduces more easily cyclic relationships and interdependencies than in a textual one. A deeper understanding of the complexity sources will enable a better estimation of the complexity of the input and thus enable the design of DL systems with a more stable performance.

Secondly, as mentioned above, no "goal-directed" algorithm is known that decides the satisfiability of $\mathcal{SHIQO}$, the combination of $\mathcal{SHIQ}$ and $\mathcal{SHOQ}(D)$, which is the DL underlying OWL. It is known that satisfiability of $\mathcal{SHIQO}$ is NExptime-hard, and it is conjectured that it is also in NExptime. Clearly, this high complexity implies that it is yet to be seen whether an implementation of such a goal-directed algorithm has any performance advantages over an incomplete reasoner or a first order theorem prover (for the translation of $\mathcal{SHIQO}$-concepts and TBoxes into first order logic). However, to answer this question, such an algorithm needs to be designed and its correctness needs to be proven. Finally, if this algorithm were automata-based, it would be interesting to see automata applied to a NExptime-hard problem—one would possibly learn more about automata for other NExptime-hard problems.

Thirdly, so far, we only described research concerned with reasoning in arbitrary models. As we have seen, $\mathcal{SHIQ}$ lacks the *finite model property*, i.e., some satisfiable concepts only admit infinite models. Now, if an application is only concerned with finite models such as many database applications—since most databases are finite—reasoners for arbitrary model reasoning can not provide the best possible system services. For example, they possibly classify too many entities as satisfiable (namely those that are satisfiable but finitely unsatisfiable), and might not detect all implicit containment relationships (namely those where all counterexamples depend on infinite models). Thus finite model reasoning is an interesting problem, for which first progress has already been made in DLs [Calvanese 1996; Lutz *et al.* 2003]. It has become clear that, due to the combinatoric nature of finite model reasoning, novel reasoning techniques have to be employed such as translating the input ontology into a set of linear in-equations over the non-negative integers. The empirical evaluation of the feasibility of this kind of decision procedures and their extension to other logics such as $\mathcal{SHIQ}$ are interesting open questions.

References

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

[Areces *et al.* 1999] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In *Annual Conference of the European Association for Computer Science Logic (CSL'99)*, vol. 1683 of *Lecture Notes in Computer Science*, pages 307–321. Springer-Verlag, 1999.

[Areces *et al.* 2000] C. Areces, P. Blackburn, and M. Marx. The computational complexity of hybrid temporal logics. *Logic Journal of the IGPL*, 8(5), 2000.

[Baader *et al.* 1993] F. Baader, H.-J. Bürckert, B. Nebel, W. Nutt, and G. Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. *Journal of Logic, Language and Information*, 2:1–18, 1993.

[Baader *et al.* 1994] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H. Profitlich. An empirical analysis of optimization techniques for terminological representation systems, or: Making KRIS get a move on. *Applied Artificial Intelligence*, 4:109–132, 1994.

[Baader *et al.* 1999] F. Baader, R. Molitor, and S. Tobies. Tractable and decidable fragments of conceptual graphs. In W. Cyre and W. Tepfenhart, eds., *Proc. of the 7th Int. Conference on Conceptual Structures (ICCS'99)*, vol. 1640 of *Lecture Notes in Computer Science*, pages 480–493. Springer-Verlag, 1999.

[Baader *et al.* 2001] F. Baader, S. Brandt, and R. Küsters. Matching under side conditions in description logics. In B. Nebel, ed., *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 213–218, Seattle, Washington, 2001. Morgan Kaufmann, Los Altos.

[Baader *et al.* 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.

[Baader 1991] F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proceedings of the*

Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91), Sydney, 1991.

[Baader 1996a] F. Baader. A formal definition for the expressive power of terminological knowledge representation languages. *Journal of Logic and Computation*, 6(1):33–54, 1996.

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

[Baader & Hanschke 1991] F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, 1991.

[Baader & Hollunder 1991] F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of the Workshop on Processing Declarative Knowledge, PDK-91*, vol. 567 of *Lecture Notes in Artificial Intelligence*, pages 67–86. Springer-Verlag, 1991.

[Baader & Sattler 1999] F. Baader and U. Sattler. Expressive number restrictions in description logics. *Journal of Logic and Computation*, 9(3):319–350, 1999.

[Baader & Sattler 2001] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001. An abridged version appeared in *Tableaux 2000*, volume 1847 of LNAI, 2000. Springer-Verlag.

[Baader & Turhan 2002] F. Baader and A.-Y. Turhan. On the problem of computing small representations of least common subsumers. In *Proc. of the 25th German Conference on Artificial Intelligence (KI 2002)*, vol. 2479 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2002.

[Baldoni *et al.* 1998] M. Baldoni, L. Giordano, and A. Martelli. A tableau calculus for multimodal logics and some (un)decidability results. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX-98)*, vol. 1397 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1998.

[Baldoni 1998] M. Baldoni. *Normal Multimodal Logics: Automatic Deduction and Logic Programming Extension.* PhD thesis, Dipartimento di Informatica, Università degli Studi di Torino, Italy, 1998.

[Bechhofer *et al.* 2001] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a reason-able ontology editor for the semantic web. In *Proceedings of the 2001 Description Logic Workshop (DL 2001)*, pages 1–9. CEUR (`http://ceur-ws.org/`), 2001.

[Ben-Ari *et al.* 1982] M. Ben-Ari, J. Halpern, and A. Pnueli. Deterministic propositional dynamic logic: finite models, complexity and completeness. *Journal of Computer and System Science*, 25:402–417, 1982.

[Berardi *et al.* 2001] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML Class Diagrams using Description Logic Based Systems. In *Proc. of the KI'2001 Workshop on Applications of Description Logics.* CEUR (`http://ceur-ws.org/`), 2001.

[Berners-Lee *et al.* 2001] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, 284(5):34–43, 2001.

[Berners-Lee 1999] T. Berners-Lee. *Weaving the Web.* Harpur, San Francisco, 1999.

[Blackburn 2000] P. Blackburn. Internalizing labelled deduction. *Journal of Logic and Computation*, 10(1):137–168, 2000.

[Borgida 1996] A. Borgida. On the relative expressive power of Description Logics and Predicate Calculus. *Artificial Intelligence Journal*, 82(1), 1996.

[Borst *et al.* 1997] P. Borst, H. Akkermans, and J. Top. Engineering ontologies. *International Journal of Human-Computer Studies*, 46:365–406, 1997.

[Brandt *et al.* 2002] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, eds., *Proceedings of the Eighth International Conference on the Principles of Knowledge Representation and Reasoning (KR-02)*, pages 203–214. Morgan Kaufmann, Los Altos, 2002.

[Calvanese *et al.* 1994] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In J. Doyle, E. Sandewall, and P. Torasso, eds., *Proceedings of the Fourth International*

Conference on the Principles of Knowledge Representation and Reasoning (KR-94), pages 109–120, Bonn, 1994. Morgan Kaufmann, Los Altos.

[Calvanese *et al.* 1995] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proceedings of the Fourth International Conference on Deductive and Object-Oriented Databases (DOOD-95)*, vol. 1013 of *Lecture Notes in Computer Science*, pages 229–246, 1995.

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

[Calvanese *et al.* 1999a] D. Calvanese, G. De Giacomo, and M. Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*. Morgan Kaufmann, Los Altos, 1999.

[Calvanese *et al.* 1999b] D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi. Reasoning in expressive description logics. In A. Robinson and A. Voronkov, eds., *Handbook of Automated Reasoning*. Elsevier Science Publishers (North-Holland), Amsterdam, 1999.

[Calvanese 1996] D. Calvanese. *Unrestricted and Finite Model Reasoning in Class-Based Representation Formalisms*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1996.

[Cornet 2003] R. Cornet. Rice ontology editor, 2003. Homepage at `http://www.b1g-systems.com/ronald/rice/`.

[De Giacomo 1995] G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Università degli Studi di Roma "La Sapienza", 1995.

[De Giacomo & Lenzerini 1994a] G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics (extended abstract). In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*. AAAI Press, 1994.

[De Giacomo & Lenzerini 1994b] G. De Giacomo and M. Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with mu-calculus. In A. Cohn, ed., *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-94)*. John Wiley & Sons, 1994.

[De Giacomo & Lenzerini 1996] G. De Giacomo and M. Lenzerini. Tbox and Abox reasoning in expressive description logics. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 316–327. Morgan Kaufmann, Los Altos, 1996.

[Demri 2001] S. Demri. The complexity of regularity in grammar logics and related modal logics. *Journal of Logic and Computation*, 11(6), 2001.

∗[Demri & Sattler 2002] S. Demri and U. Sattler. Automata-theoretic decision procedures for information logics. *Fundamenta Informaticae*, 53(1):1–22, 2002.

[Donini *et al.* 1991a] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)*, Boston, MA, USA, 1991.

[Donini *et al.* 1991b] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 458–463, Sydney, 1991.

[Donini & Massacci 2000] F. M. Donini and F. Massacci. Exptime tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, 124(1):87–138, 2000.

[Emerson 1990] E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, pages 997–1072. Elsevier Science Publishers (North-Holland), Amsterdam, 1990.

[Fensel *et al.* 2001] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

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

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

[Franconi & Ng 2000] E. Franconi and G. Ng. The i.com tool for intelligent conceptual modelling. In *Working Notes of the ECAI2000 Workshop on Knowledge Representation Meets Databases (KRDB2000)*. CEUR (`http://ceur-ws.org/`), 2000.

[Gargov *et al.* 1987] G. Gargov, S. Passy, and T. Tinchev. Modal environment for Boolean speculations. In D. Skordev, ed., *Mathematical Logic and Applications*, pages 253–263. Plenum Publ. Co., New York, 1987.

[Georgieva *et al.* 2003] L. Georgieva, U. Hustadt, and R. A. Schmidt. Hyperresolution for guarded formulae. *Journal of Symbolic Logic*, 2003. To appear.

[Gerstl & Pribbenow 1995] P. Gerstl and S. Pribbenow. Midwinters, end games and bodyparts. *International Journal of Human-Computer Studies*, 43:847–864, 1995.

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

[Grädel 1999] E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64(4):1719–1742, 1999.

[Gruber 1993] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, eds., *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[Haarslev & Möller 2000] V. Haarslev and R. Möller. Consistency testing: The RACE experience. In R. Dyckhoff, ed., *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2000)*, vol. 1847 of *Lecture Notes in Artificial Intelligence*, pages 1–18. Springer-Verlag, 2000.

[Haarslev & Möller 2001] V. Haarslev and R. Möller. RACER system description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR-01)*, vol. 2083 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[Halpern & Moses 1992] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logic of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[Hirsch & Tobies 2001] C. Hirsch and S. Tobies. A tableau algorithm for the clique guarded fragment. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, eds., *Advances in Modal Logics Volume 3*, Stanford, 2001. CSLI Publications.

[Hladik 2002] J. Hladik. Implementation and optimisation of a tableau algorithm for the guarded frament. In U. Egly and C. G. Fermüller, eds., *Proc. of the Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2002)*, vol. 2381 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2002.

[Horrocks *et al.* 1996] I. Horrocks, A. Rector, and C. Goble. A description logic based schema for the classification of medical data. In *Working Notes of the ECAI-96 Workshop on Knowledge Representation Meets Databases (KRDB-96)*. CEUR (http://ceur-ws.org/), 1996.

∗[Horrocks *et al.* 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, eds., *Proceedings of the Sixth International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, vol. 1705 of *Lecture Notes in Artificial Intelligence*, pages 161–180. Springer-Verlag, 1999.

[Horrocks *et al.* 2000a] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–264, May 2000.

[Horrocks *et al.* 2000b] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic shiq. In D. MacAllester, ed., *Proceedings of the 17th Conference on Automated Deduction (CADE-17)*, vol. 1831 of *Lecture Notes in Computer Science*, Germany, 2000. Springer-Verlag.

[Horrocks *et al.* 2002] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. Reviewing the design of DAML+OIL: An ontology language for the semantic web. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-02)*, 2002.

[Horrocks 1997] I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.

[Horrocks 1998a] I. Horrocks. The FaCT system. In H. de Swart, ed., *Proceedings of the International Conference on Automated Reasoning with*

Analytic Tableaux and Related Methods (TABLEAUX-98), vol. 1397 of *Lecture Notes in Artificial Intelligence*, pages 307–312. Springer-Verlag, 1998.

[Horrocks 1998b] I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *Proceedings of the Sixth International Conference on the Principles of Knowledge Representation and Reasoning (KR-98)*. Morgan Kaufmann, Los Altos, 1998.

[Horrocks & Gough 1997] I. Horrocks and G. Gough. Description logics with transitive roles. In M.-C. Rousset, R. Brachmann, F. Donini, E. Franconi, I. Horrocks, and A. Levy, eds., *Proceedings of the 1997 Description Logic Workshop (DL'97)*, pages 25–28, 1997.

[Horrocks & Patel-Schneider 2001] I. Horrocks and P. Patel-Schneider. The generation of DAML+OIL. In *Proceedings of the 2001 Description Logic Workshop (DL 2001)*, pages 30–35. CEUR (`http://ceur-ws.org/`), volume 49, 2001.

$*$[Horrocks & Sattler 1999] I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3), 1999.

$*$[Horrocks & Sattler 2001] I. Horrocks and U. Sattler. Ontology reasoning in the $\mathcal{SHOQ}(D)$ description logic. In B. Nebel, ed., *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 199–204. Morgan Kaufmann, Los Altos, 2001.

$*$[Horrocks & Sattler 2002] I. Horrocks and U. Sattler. Optimised reasoning for $\mathcal{SHIQ}$. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI-2002)*, 2002.

$*$[Horrocks & Sattler 2003] I. Horrocks and U. Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*. Morgan Kaufmann, Los Altos, 2003. To appear. A long version has appeared as technical report LTCS 02-06 available at `http://lat.inf.tu-dresden.de/research/reports.html`.

[Hustadt & Schmidt 2000] U. Hustadt and R. A. Schmidt. MSPASS: Modal reasoning by translation and first-order resolution. In R. Dyckhoff, ed., *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2000)*, vol. 1847 of *Lecture Notes in Artificial Intelligence*, pages 67–71. Springer-Verlag, 2000.

[Konikowska 1997] B. Konikowska. A logic for reasoning about relative similarity. *Studia Logica*, 58(1):185–226, 1997.

[Kozen 1982] D. Kozen. Results on the propositional μ-calculus. In M. Nielsen and E. M. Schmidt, eds., *Automata, Languages and Programming, 9th Colloquium*, vol. 140 of *Lecture Notes in Computer Science*, pages 348–359. Springer-Verlag, 1982.

[Kupferman *et al.* 2002] O. Kupferman, U. Sattler, and M. Y. Vardi. The complexity of the graded mu-calculus. In *Proceedings of the 18th Conference on Automated Deduction (CADE-18)*, vol. 2392 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2002.

[Kupferman & Vardi 1998] O. Kupferman and M. Vardi. Weak alternating automata and tree automata emptiness. In *Proceedings of the Thirtieth ACM SIGACT Symposium on Theory of Computing (STOC-98)*, pages 224–233, 1998.

[Küsters 2001] R. Küsters. *Non-Standard Inferences in Description Logics*, vol. 2100 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[Lenat & Guha 1989] D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems*. Addison Wesley Publ. Co., Reading, Massachussetts, 1989.

[Lutz *et al.* 1999] C. Lutz, U. Sattler, and S. Tobies. A suggestion for an n-ary description logic. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, eds., *Proceedings of the 1999 Description Logic Workshop (DL'99)*, pages 81–85, Linköping, Sweden, 1999. CEUR (`http://ceur-ws.org/`).

[Lutz *et al.* 2001] C. Lutz, U. Sattler, and F. Wolter. Modal logics and the two-variable fragment. In *Annual Conference of the European Association for Computer Science Logic (CSL-01)*, vol. 2142 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.

[Lutz *et al.* 2003] C. Lutz, U. Sattler, and L. Tendera. The complexity of finite model reasoning in description logics. In *Proceedings of the 19th Conference on Automated Deduction (CADE-19)*, Lecture Notes in

Artificial Intelligence. Springer-Verlag, 2003. To appear. A long version has appeared as technical report LTCS 02-05 available at `http://lat.inf.tu-dresden.de/research/reports.html`.

[Lutz 1999] C. Lutz. Complexity of terminological reasoning revisited. In *Proceedings of the Sixth International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, Lecture Notes in Artificial Intelligence, pages 181–200. Springer-Verlag, 1999.

[Lutz 2002] C. Lutz. *The Complexity of Description Logics with Concrete Domains*. PhD thesis, RWTH Aachen, 2002.

[Lutz 2003] C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Volume 4*. World Scientific Publishing Co. Pte. Ltd., 2003.

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

[Molitor 2000] R. Molitor. *Unterstützung der Modellierung verfahrenstechnischer Prozesse durch Nicht-Standardinferenzen in Beschreibungslogiken*. PhD thesis, RWTH Aachen, 2000.

[Nebel 1990] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, vol. 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1990.

[Orłowska 1998] E. Orłowska, ed. *Incomplete Information: Rough Set Analysis*. Studies in Fuzziness and Soft Computing. Physica, Heidelberg, 1998.

[Pan *et al.* 2002] G. Pan, U. Sattler, and M. Y. Vardi. BDD-based decision procedures for K. In *Proceedings of the 18th Conference on Automated Deduction (CADE-18)*, vol. 2392 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2002.

[Patel-Schneider *et al.* 1991] P. Patel-Schneider, D. McGuinness, R. Brachman, L. Resnick, and A. Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rationale. *SIGART Bulletin*, 2(3):108–113, 1991.

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

[Patel-Schneider & Horrocks 1999] P. F. Patel-Schneider and I. Horrocks. DLP and FaCT. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX-99)*, vol. 1397 of *Lecture Notes in Artificial Intelligence*, pages 19–23. Springer-Verlag, 1999.

[Pawlak 1981] Z. Pawlak. Information systems theoretical foundations. *Information Systems*, 6(3):205–218, 1981.

[Pribbenow 1995] S. Pribbenow. Modeling physical objects: Reasoning about (different kinds of) parts. In *Time, Space, and Movement Workshop 95*, Bonas, France, 1995.

[Prior 1967] A. Prior. *Past, Present and Future.* Oxford University Press, 1967.

[Protégé 2003] Protégé. Homepage at `http://protege.stanford.edu/`, 2003.

[Rector *et al.* 1997] A. Rector, S. Bechhofer, C. A. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon. The GRAIL concept modelling language for medical terminology. *AI in Medicine*, 9:139–171, 1997.

[Rector 2002] A. Rector. Analysis of propagation along transitive roles: Formalisation of the galen experience with medical ontologies. In *Proceedings of the 2002 Description Logic Workshop (DL 2002).* CEUR (`http://ceur-ws.org/`), 2002.

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

[Rescher & Urquhart 1971] N. Rescher and A. Urquhart. *Temporal Logic.* Springer-Verlag, 1971.

[Sattler 1996] U. Sattler. A concept language extended with different kinds of transitive roles. In G. Görz and S. Hölldobler, eds., *20. Deutsche Jahrestagung für Künstliche Intelligenz*, vol. 1137 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 1996.

[Sattler 1998] U. Sattler. *Terminological knowledge representation systems in a process engineering application.* PhD thesis, RWTH Aachen, 1998.

∗[Sattler 2000] U. Sattler. Description logics for the representation of aggregated objects. In W. Horn, ed., *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000).* IOS Press, Amsterdam, 2000.

∗[Sattler & Vardi 2001] U. Sattler and M. Y. Vardi. The hybrid μ-calculus. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR-01)*, vol. 2083 of *Lecture Notes in Artificial Intelligence*, pages 76–91. Springer-Verlag, 2001.

[Schaerf 1994] A. Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.

[Schild 1991] K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, Sydney, 1991.

[Schild 1994] K. Schild. Terminological cycles and the propositional μ-calculus. In J. Doyle, E. Sandewall, and P. Torasso, eds., *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 509–520, Bonn, 1994. Morgan Kaufmann, Los Altos.

[Schmidt-Schauss 1989] M. Schmidt-Schauss. Subsumption in KL-ONE is undecidable. In *Proceedings of the First International Conference on the Principles of Knowledge Representation and Reasoning (KR-89)*, pages 421–431, Boston (USA), 1989.

[Schmidt-Schauß & Smolka 1991] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[Schulz & Hahn 2001] S. Schulz and U. Hahn. Parts, locations, and holes - formal reasoning about anatomical structures. In *Proc. of AIME 2001*, vol. 2101 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[Simons 1987] P. M. Simons. *Parts. A study in Ontology.* Oxford: Clarendon, 1987.

[Spaan 1993a] E. Spaan. *Complexity of Modal Logics.* PhD thesis, University of Amsterdam, 1993.

[Spaan 1993b] E. Spaan. The complexity of propositional tense logics. In M. de Rijke, ed., *Diamonds and Defaults*, pages 287–307. Kluwer Academic Publishers, 1993.

[Spackman 2000] K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *Journal of the American Medical Informatics Association*, 2000. Fall Symposium Special Issue.

[Stevens *et al.* 2002] R. Stevens, I. Horrocks, C. Goble, and S. Bechhofer. Building a bioinformatics ontology using OIL. *IEEE Information Technology in Biomedicine. special issue on Bioinformatics.*, 6(2):135–141, 2002.

[Streett 1982] R. S. Streett. Propositional dynamic logic of looping and converse is elementarily decidable. *Information and Computation*, 54:121–141, 1982.

[Streett & Emerson 1989] R. S. Streett and E. A. Emerson. An automata theoretic decision procedure for the propositional μ-calculus. *Information and Computation*, 81:249–264, 1989.

[Sure *et al.* 2002] Y. Sure, S. Staab, and J. Angele. OntoEdit: Guiding ontology development by methodology and inferencing. In *Proceedings of the Confederated International Conferences DOA, CoopIS and ODBASE 2002*, vol. 2519 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[Thomas 1992] W. Thomas. Automata on infinite objects. In J. van Leeuwen, ed., *Handbook of theoretical computer science*, vol. B. Elsevier Science Publishers (North-Holland), Amsterdam, 1992.

[Tobies 2001a] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001. electronically available at `http://www.bth.rwth-aachen.de/ediss/ediss.html`.

[Tobies 2001b] S. Tobies. PSPACE reasoning for graded modal logics. *Journal of Logic and Computation*, 11(1):85–106, 2001.

[Uschold *et al.* 1998] M. Uschold, M. King, S. Moralee, and Y. Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13, 1998.

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

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

[Vardi 1985] M. Y. Vardi. The taming of converse: Reasoning about two-way computations. In R. Parikh, ed., *Proc. of the 4th Workshop on Logics of Programs*, vol. 193 of *Lecture Notes in Computer Science*, pages 413–424. Springer-Verlag, 1985.

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

[Vardi 1998] M. Y. Vardi. Reasoning about the past with two-way automata. In *Proceedings of the 25th International Colloquium on Automata, Languages, and Programming*, vol. 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer-Verlag, 1998.

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

[Wessel 2001] M. Wessel. Obstacles on the way to qualitative spatial reasoning with description logics: Some undecidability results. In *Proceedings of the 2001 Description Logic Workshop (DL 2001)*. CEUR (`http://ceur-ws.org/`), 2001.

[Winston *et al.* 1987] M. Winston, R. Chaffin, and D. Herrmann. A taxonomy of part whole relations. *Cognitive Science*, 11:417–444, 1987.