
Extensions to Description Logics

Franz Baader

Ralf Küsters

Frank Wolter

Abstract

This chapter considers, on the one hand, extensions of Description Logics by features not available in the basic framework, but considered important for using Description Logics as a modeling language. In particular, it addresses the extensions concerning: concrete domain constraints; modal, epistemic, and temporal operators; probabilities and fuzzy logic; and defaults.

On the other hand, it considers non-standard inference problems for Description Logics, i.e., inference problems that—unlike subsumption or instance checking—are not available in all systems, but have turned out to be useful in applications. In particular, it addresses the non-standard inference problems: least common subsumer and most specific concept; unification and matching of concepts; and rewriting.

6.1 Introduction

Chapter 2 introduces the language \mathcal{ALCN} as a prototypical Description Logic, defines the most important reasoning tasks (like subsumption, instance checking, etc.), and shows how these tasks can be realized with the help of tableau-based algorithms. For many applications, the expressive power of \mathcal{ALCN} is not sufficient to express the relevant terminological knowledge of the application domain. Some of the most important extensions of \mathcal{ALCN} by concept and role constructs have already been briefly introduced in Chapter 2; these and other extensions have then been treated in more detail in Chapter 5. All these extensions are “classical” in the sense that their semantics can easily be defined within the model-theoretic framework introduced in Chapter 2. Although combinations of these constructs may lead to very expressive DLs (the unrestricted combination even to undecidable ones), all the DLs obtained this way can only be used to represent time-independent, objective, and certain knowledge. In addition, they do not allow for “built-in data structures” like numerical domains.

The “nonclassical” language extensions considered in the first part of this chapter try to overcome some of these deficiencies. The extension by *concrete domains* allows us to integrate numerical and other domains in a schematic way into Description Logics. The extension of DLs by *modal operators* allows for the representation of time-dependent and subjective knowledge (e.g., knowledge about knowledge and belief of intelligent agents). DLs that can explicitly represent *time* have also been introduced outside the modal framework. The extension by *epistemic operators* provides a model-theoretic semantics for rules, it can be used to impose “local” closed world assumptions, and to integrate integrity constraints into DLs. In order to represent *vague and uncertain knowledge*, different approaches based on probabilistic, possibilistic, and fuzzy logics have been proposed. Finally, non-monotonic Description Logics are obtained by the integration of *defaults* into DLs.

When building and maintaining large DL knowledge bases, inference services like subsumption and satisfiability are very helpful, but in general not quite sufficient for an adequate support of the knowledge engineer. For this reason, some DLs systems (e.g., CLASSIC) provide their users with additional system services, which can formally be reconstructed as new types of inference problems. In the second part of this chapter we will motivate and introduce the most prominent of these “non-standard” inference problems, and try to give an intuition on how they can be solved.

6.2 Language extensions

The extensions introduced in this section are “nonclassical” in the sense that defining their semantics is not obvious and requires an extension of the model-theoretic framework considered until now; for many (but not all) of these extensions, non-classical logics (such as modal and non-monotonic logics) are employed to provide the right framework.

6.2.1 Concrete domains

A drawback that all Description Logics introduced until now share is that all the knowledge must be represented on the abstract logical level. In many applications, one would like to be able to refer to concrete domains and predefined predicates on these domains when defining concepts. An example for such a concrete domain could be the set of nonnegative integers, with predicates such as \geq (greater-or-equal) or $<$ (less-than). For example, assume that we want to give an adequate definition of the concept *Woman*. The first idea could be to use the concept description $\text{Human} \sqcap \text{Female}$ for this purpose. However, a newborn female baby would probably not be called a woman, and neither would a three-year old toddler. Thus, as an

additional property, one could require that a female human-being should be old enough (e.g., at least 18) to be called a woman. In order to express this property, one would like to introduce a new (functional) role `has-age`, and define `Woman` by an expression of the form $\text{Human} \sqcap \text{Female} \sqcap \exists \text{has-age}.\geq_{18}$. Here \geq_{18} stands for the unary predicate $\{n \mid n \geq 18\}$ of all nonnegative integers greater than or equal to 18.

Stating such properties directly with reference to a given numerical domain seems to be easier and more natural than encoding them somehow into abstract concept expressions. In addition, such a direct representation makes it possible to use existing reasoners for the concrete domain. For example, we could have also decided to introduce a new atomic concept `AtLeast18` to express the property of being at least 18 years old. However, if for some reason we also need the property of being at least 21 years old, we must make sure that the appropriate subsumption relationship between `AtLeast18` and `AtLeast21` is asserted as well. While this could still be done by adding appropriate inclusion axioms, it does not appear to be an elegant solution, and it would still not take care of other relationships, e.g., the fact that $\text{AtLeast18} \sqcap \text{AtMost16}$ is unsatisfiable. In contrast, an appropriate reasoner for intervals of nonnegative integers would automatically take care of these relationships.

The need for such a language extension was already evident to the designers of early DL systems such as `MESON` [Edelmann and Owsnicki, 1986; Patel-Schneider *et al.*, 1990], `K-REP` [Mays *et al.*, 1988; 1991a], and `CLASSIC` [Brachman *et al.*, 1991; Borgida and Patel-Schneider, 1994]: in addition to abstract individuals, these systems also allow one to refer to “concrete” individuals such as numbers and strings. Both the `CLASSIC` and the `K-REP` reasoner can deal correctly with intervals, whereas in `MESON` the user had to supply the adequate relationships between the concrete predicates in a separate hierarchy. All these approaches are, however, ad hoc in the sense that they are restricted to a specific collection of concrete objects.

In contrast, Baader and Hanschke [1991a] propose a scheme for integrating (almost) arbitrary concrete domains into Description Logics. This extension was designed such that

- it still has a formal declarative semantics that is very close to the usual semantics employed for DLs;
- it is possible to combine the tableau-based algorithms available for DLs with existing reasoning algorithms in the concrete domain in order to obtain the appropriate algorithms for the extension;
- it provides a scheme for extending DLs by various concrete domains rather than constructing a single ad hoc extension for a specific concrete domain.

In the following, we will first introduce the original proposal by Baader and

Hanschke, and then describe two extensions of this proposal [Hanschke, 1992; Haarslev *et al.*, 1999].

6.2.1.1 The family of Description Logics $\mathcal{ALC}(\mathcal{D})$

Before we can define the members of this family of DLs, we must formalize the notion of a concrete domain.

Definition 6.1 A *concrete domain* \mathcal{D} consists of a set $\Delta^{\mathcal{D}}$, the domain of \mathcal{D} , and a set $pred(\mathcal{D})$, the predicate names of \mathcal{D} . Each predicate name $P \in pred(\mathcal{D})$ is associated with an arity n , and an n -ary predicate $P^{\mathcal{D}} \subseteq (\Delta^{\mathcal{D}})^n$. ■

Let us illustrate this definition by examples of interesting concrete domains. Let us start with some numerical ones:

- The concrete domain \mathcal{N} , which we have employed in our introductory example, has the set \mathbf{N} of all nonnegative integers as its domain, and $pred(\mathcal{N})$ consists of the binary predicate names $<, \leq, \geq, >$ as well as the unary predicate names $<_n, \leq_n, \geq_n, >_n$ for $n \in \mathbf{N}$, which are interpreted by predicates on \mathbf{N} in the obvious way.
- The concrete domain \mathcal{R} has the set \mathbf{R} of all real numbers as its domain, and the predicates of \mathcal{R} are given by formulae that are built by first-order means (i.e., by using Boolean connectives and quantifiers) from equalities and inequalities between integer polynomials in several indeterminates. For example, $x + z^2 = y$ is an equality between the polynomials $p(x, z) = x + z^2$ and $q(y) = y$; and $x > y$ is an inequality between very simple polynomials. From these equalities and inequalities one can for instance build the formulae $\exists z.(x + z^2 = y)$ and $\exists z.(x + z^2 = y) \vee (x > y)$. The first formula yields a predicate name of arity 2 (since it has two free variables), and it is easy to see that the associated predicate is $\{(r, s) \mid r \text{ and } s \text{ are real numbers and } r \leq s\}$. Consequently, the predicate associated to the second formula is $\{(r, s) \mid r \text{ and } s \text{ are real numbers}\} = \mathbf{R} \times \mathbf{R}$.
- The concrete domain \mathcal{Z} is defined just like \mathcal{R} , with the only difference that $\Delta^{\mathcal{Z}}$ is the set of all integers instead of all real numbers.

In addition to numerical domains, Definition 6.1 also captures more abstract domains:

- A given (fixed) relational database DB can be seen as a concrete domain \mathcal{DB} , whose domain is the set of atomic values occurring in DB, and whose predicates are the relations that can be defined over DB using a query language (such as SQL).

- One can also consider Allen’s interval calculus [Allen, 1983] as concrete domain \mathcal{IC} . Here $\Delta^{\mathcal{IC}}$ consists of time intervals, and the predicates are built from Allen’s basic interval relations (such as *before*, *after*, ...) with the help of Boolean connectives.
- Instead of time intervals one can also consider spatial regions (e.g., in $\mathbb{R} \times \mathbb{R}$), and use Boolean combinations of the basic RCC-8 relations as predicates [Randell *et al.*, 1992; Bennett, 1997].

Although syntax and semantics of DLs extended by concrete domains could be defined with the general notion of a concrete domain introduced in Definition 6.1, the requirement that the extended language should still have decidable reasoning problems adds some additional restrictions.

To be able to compute the negation normal form of concepts in the extended language, we must require that the set of predicate names of the concrete domain is *closed under negation*, i.e., if P is an n -ary predicate name in $\text{pred}(\mathcal{D})$ then there has to exist a predicate name Q in $\text{pred}(\mathcal{D})$ such that $Q^{\mathcal{D}} = (\Delta^{\mathcal{D}})^n \setminus P^{\mathcal{D}}$. We will refer to this predicate name by \overline{P} . In addition, we need a unary predicate name that denotes the predicate $\Delta^{\mathcal{D}}$. The domain \mathcal{N} from above satisfies these two properties since, e.g., $\overline{<_n} = \geq_n$ and $(\geq_0)^{\mathcal{N}} = \mathbf{N}$.

Let us now clarify what kind of reasoning mechanisms are required in the concrete domain. Let P_1, \dots, P_k be k (not necessarily different) predicate names in $\text{pred}(\mathcal{D})$ of arities n_1, \dots, n_k . We consider the conjunction

$$\bigwedge_{i=1}^k P_i(\underline{x}^{(i)}).$$

Here $\underline{x}^{(i)}$ stands for an n_i -tuple $(x_1^{(i)}, \dots, x_{n_i}^{(i)})$ of variables. It is important to note that neither all variables in one tuple nor those in different tuples are assumed to be distinct. Such a conjunction is said to be *satisfiable* iff there exists an assignment of elements of $\Delta^{\mathcal{D}}$ to the variables such that the conjunction becomes true in \mathcal{D} . We will call the problem of deciding satisfiability of finite conjunctions of this form the *satisfiability problem* for \mathcal{D} .

Definition 6.2 The concrete domain \mathcal{D} is called *admissible* iff (i) the set of its predicate names is closed under negation and contains a name $\top_{\mathcal{D}}$ for $\Delta^{\mathcal{D}}$, and (ii) the satisfiability problem for \mathcal{D} is decidable. ■

With the exception of \mathcal{Z} , all the concrete domains introduced above are admissible. For example, decidability of the satisfiability problem for \mathcal{R} is a consequence of Tarski’s decidability result for real arithmetic [Tarski, 1951; Collins,

1975]. In contrast, undecidability of the satisfiability problem for \mathcal{Z} is a consequence of the undecidability of Hilbert's 10th problem [Matiyasevich, 1971; Davis, 1973].

In the following, we will take the language \mathcal{ALC} as the (prototypical) starting point of our extension.¹ In the following, let \mathcal{D} be an arbitrary (but fixed) concrete domain. The interface between \mathcal{ALC} and the concrete domain is inspired by the agreement construct between chains of functional roles (see Chapter 2, Subsection 2.4.3). With this construct one can, for example, express the concept of all women whose father and husband are of the same age by the expression $\text{Woman} \sqcap \text{has-father} \circ \text{has-age} \doteq \text{has-husband} \circ \text{has-age}$. However, one cannot express that the husband is even older than the father. This becomes possible if we take the concrete domain \mathcal{N} . Then we can simply write

$$\text{Woman} \sqcap \exists(\text{has-father} \circ \text{has-age}, \text{has-husband} \circ \text{has-age}). <$$

More generally, our extension, called $\mathcal{ALC}(\mathcal{D})$, will allow to state that a tuple of chains of functional roles satisfies a (not necessarily binary) predicate, which is provided by the concrete domain in question.

Thus, $\mathcal{ALC}(\mathcal{D})$ extends \mathcal{ALC} in two respects. First, the set of role names is now assumed to be partitioned into a set of functional roles and a set of ordinary roles. Both types of roles are allowed to occur in value restrictions and in the existential quantification construct. In addition, there is a new constructor, called *existential predicate restriction*, which is defined by adding to the syntax rules for \mathcal{ALC} the rule

$$C, D \longrightarrow \exists(u_1, \dots, u_n).P,$$

where P is an n -ary predicate of \mathcal{D} and u_1, \dots, u_n are chains of functional roles. When considering $\mathcal{ALC}(\mathcal{D})$ -ABoxes, one must distinguish between names for abstract and for concrete individuals. Concrete predicates $P \in \text{pred}(\mathcal{D})$ give rise to additional ABox assertions of the form $P(x_1, \dots, x_n)$, where x_1, \dots, x_n are names for concrete individuals.

Definition 6.3 An *interpretation* \mathcal{I} for $\mathcal{ALC}(\mathcal{D})$ consists of a set $\Delta^{\mathcal{I}}$, the abstract domain of the interpretation, and an interpretation function. The abstract domain and the given concrete domain must be disjoint, i.e., $\Delta^{\mathcal{D}} \cap \Delta^{\mathcal{I}} = \emptyset$. As before, the interpretation function associates with each concept name a subset of $\Delta^{\mathcal{I}}$ and with each ordinary role name a binary relation on $\Delta^{\mathcal{I}}$. The new feature is that the functional roles are now interpreted by partial functions from $\Delta^{\mathcal{I}}$ into $\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{D}}$. If $u = f_1 \circ \dots \circ f_n$ is a chain of functional roles, then $u^{\mathcal{I}}$ denotes the composition $f_1^{\mathcal{I}} \circ \dots \circ f_n^{\mathcal{I}}$ of the partial functions $f_1^{\mathcal{I}}, \dots, f_n^{\mathcal{I}}$.

¹ All the definitions would, of course, also work for any other concept description language. The approach for combining the reasoning algorithms will work for many other languages, but not for all of them.

The semantics of the usual \mathcal{ALC} -constructors is defined as before. In particular, this means that complex concept descriptions are always interpreted as subsets of the abstract domain $\Delta^{\mathcal{I}}$. The existential predicate restriction is interpreted as follows:

$$(\exists(u_1, \dots, u_n).P)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{there exist } r_1, \dots, r_n \in \Delta^{\mathcal{D}} \text{ such that } u_1^{\mathcal{I}}(x) = r_1, \dots, u_n^{\mathcal{I}}(x) = r_n \text{ and } (r_1, \dots, r_n) \in P^{\mathcal{D}}\}.$$

■

Above, we have already seen two examples of concepts of $\mathcal{ALC}(\mathcal{N})$. The following $\mathcal{ALC}(\mathcal{R})$ -concepts describe rectangles and squares in the $\mathbb{R} \times \mathbb{R}$:

$$\begin{aligned} \text{Rectangle} &= \exists(x, y, b, h).\text{rectangle-cond}, \\ \text{Square} &= \text{Rectangle} \sqcap \exists(b, h).\text{equal}, \end{aligned}$$

where the concrete predicates `rectangle-cond` and `equal` are defined as $\text{equal}(x, y) \Leftrightarrow x = y$ and $\text{rectangle-cond}(x, y, b, h) \Leftrightarrow b > 0 \wedge h > 0$. In `rectangle-cond`, the first two arguments are assumed to express the x - and y - coordinate of the lower left corner of the rectangle, whereas the third and fourth argument express the breadth and height of the rectangle. We leave it to the reader to define the concept “pairs of rectangles” where the first component is a square that is contained in the second component.

A tableau-based algorithm for deciding consistency of $\mathcal{ALC}(\mathcal{D})$ -ABoxes for admissible \mathcal{D} was introduced in [Baader and Hanschke, 1991b]. The algorithm has an additional rule that treats existential predicate restrictions according to their semantics. The main new feature is that, in addition to the usual “abstract” clashes, there may be concrete ones, i.e., one must test whether the given combination of concrete predicate assertions is non-contradictory. This is the reason why we must require that the satisfiability problem for \mathcal{D} is decidable. As described in [Baader and Hanschke, 1991b], the algorithm is not in PSPACE. Using techniques similar to the ones employed for \mathcal{ALC} it can be shown, however, that the algorithm can be modified such that it needs only polynomial space [Lutz, 1999b], provided that the satisfiability procedure for \mathcal{D} is in PSPACE. In the presence of acyclic TBoxes, reasoning in $\mathcal{ALC}(\mathcal{D})$ may become NEXPTIME-hard even for rather simple concrete domains with a polynomial satisfiability problem [Lutz, 2001b].

This technique of combining a tableau-based algorithm for the description logics with a satisfiability procedure for the concrete domain can be extended to more expressive DLs (e.g., \mathcal{ALCN} and \mathcal{ALCN} with agreements and disagreements). However, this is not true for arbitrary DLs with tableau-based decision procedures. For example, the technique does not work if the tableau-based algorithm requires some sort of blocking (see Chapter 2, Subsection 2.3.2.4) to ensure termination. Tech-

nically, the problem is that concrete predicates can be used to state properties concerning different individuals in the ABox, and that blocking, which is concerned only with the properties of a single individual, cannot take this into account. The main idea underlying an undecidability proof for such a logic is that elements of the concrete domain (e.g., \mathcal{R}) can encode configurations of a Turing machine and that one can define a concrete predicate stating that one configuration is a direct successor of the other. Finally, the DL must provide some means of representing sequences of configurations of arbitrary length, which is usually the case for DLs requiring blocking. More concretely, it was shown in [Baader and Hanschke, 1992] (by reduction from Post's correspondence problem) that satisfiability of concepts becomes undecidable if transitive closure (of a single functional role) is added to $\mathcal{ALC}(\mathcal{R})$. Post's correspondence problem can also be used to show undecidability of $\mathcal{ALC}(\mathcal{R})$ with general inclusion axioms, although one cannot use exactly the same reduction as for transitive closure (see [Haarslev *et al.*, 1998] for a similar reduction). A notable exception to the rule of thumb that concrete domains together with general inclusion axioms lead to undecidability has recently been shown by Lutz [2001a], who combines \mathcal{ALC} with the concrete domain of rational numbers with equality and inequality predicates.

6.2.1.2 Predicate restrictions on role chains

The role chains occurring in predicate restrictions of $\mathcal{ALC}(\mathcal{D})$ are restricted to chains of functional roles. In [Hanschke, 1992] this restriction was removed. To be more precise, the syntax rules for \mathcal{ALC} are extended by the two rules

$$C, D \longrightarrow \exists(u_1, \dots, u_n).P \mid \forall(u_1, \dots, u_n).P,$$

where P is an n -ary predicate of \mathcal{D} and u_1, \dots, u_n are chains of (not necessarily functional) roles.

In this setting, ordinary roles are also allowed to have fillers in the concrete domain, i.e., both functional and ordinary roles are interpreted as subsets of $\Delta^{\mathcal{I}} \times (\Delta^{\mathcal{I}} \cup \Delta^{\mathcal{D}})$. Of course, functional roles must still be interpreted as partial functions. The extension of the predicate restrictions is defined as

$$\begin{aligned} (\exists(u_1, \dots, u_n).P)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \text{there exist } r_1, \dots, r_n \in \Delta^{\mathcal{D}} \text{ such that} \\ &\quad (x, r_1) \in u_1^{\mathcal{I}}, \dots, (x, r_n) \in u_n^{\mathcal{I}} \text{ and } (r_1, \dots, r_n) \in P^{\mathcal{D}}\}, \\ (\forall(u_1, \dots, u_n).P)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \text{for all } r_1, \dots, r_n: (x, r_1) \in u_1^{\mathcal{I}}, \dots, (x, r_n) \in u_n^{\mathcal{I}} \\ &\quad \text{implies } (r_1, \dots, r_n) \in P^{\mathcal{D}}\}. \end{aligned}$$

Using the universal predicate restriction one can, for example, define the concept of parents all of whose children are younger than 4 by the description

$$\text{Parent} \sqcap \forall \text{has-child} \circ \text{has-age} . \leq_4 .$$

Hanschke [1992] shows that an extension of the DL we have just introduced still has a decidable ABox consistency problem, provided that the concrete domain \mathcal{D} is admissible.

6.2.1.3 Predicate restrictions defining roles

In [Haarslev *et al.*, 1998; 1999], $\mathcal{ALC}(\mathcal{D})$ was extended in a different direction: predicate restrictions can now also be used to define new roles. To be more precise, if P is a predicate of \mathcal{D} of arity $n + m$ and $u_1, \dots, u_n, v_1, \dots, v_m$ are chains of functional roles, then

$$\exists(u_1, \dots, u_n)(v_1, \dots, v_m).P$$

is a complex role. These complex roles may be used both in value restrictions and in the existential quantification construct. The semantics of complex roles is defined as

$$\begin{aligned} (\exists(u_1, \dots, u_n)(v_1, \dots, v_m).P)^{\mathcal{I}} = \\ \{(x, y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid \text{there exist } r_1, \dots, r_n, s_1, \dots, s_m \in \Delta^{\mathcal{D}} \text{ such that} \\ u_1^{\mathcal{I}}(x) = r_1, \dots, u_n^{\mathcal{I}}(x) = r_n, v_1^{\mathcal{I}}(y) = s_1, \dots, v_m^{\mathcal{I}}(y) = s_m \\ \text{and } (r_1, \dots, r_n, s_1, \dots, s_m) \in P^{\mathcal{D}}\}. \end{aligned}$$

For example, the complex role $\exists(\text{has-age})(\text{has-age}).>$ consists of all pairs of individuals having an age such that the first is older than the second.

Unfortunately, it has turned out that the full logic obtained by this extension has an undecidable satisfiability problem [Haarslev *et al.*, 1998]. To overcome this problem, Haarslev *et al.* [1999] define syntactic restrictions on concepts such that the restricted language (i) is closed under negation, and (ii) has a decidable ABox consistency problem. Consequently, the subsumption and the instance problem are also decidable. The complexity of reasoning in this DL is investigated in [Lutz, 2001b]. Similar to the case of acyclic TBoxes, rather simple concrete domains can already make reasoning NEXPTIME-hard.

An approach for integrating arithmetic reasoning into Description Logics that considerably differs from the concrete domain approach described above was proposed by Ohlbach and Koehler [1999].

6.2.2 Modal extensions

Although the DLs discussed so far provide a wide choice of constructors, usually they are intended to represent only static knowledge and are not able to express various dynamic aspects such as time-dependence, beliefs of different agents, obligations, etc. For example, in every standard description language we can define a concept

“good car” as, say, a car with an air-conditioner:

$$\text{GoodCar} \equiv \text{Car} \sqcap \exists \text{part.Airconditioner}. \quad (6.1)$$

However, we have no means to represent the subtler knowledge that only John believes (6.1) to be the case, while Mary does not think so:

$$[\text{John believes}](6.1) \wedge \neg[\text{Mary believes}](6.1).$$

Nor can we express the fact that (6.1) holds now, but in the future the notion of a good car may change (since, for instance, all cars will have air conditioners):

$$(6.1) \wedge \langle \text{eventually} \rangle \neg(6.1).$$

A way to bridge this gap seems quite clear and will be discussed in this and the next section: one can simply combine a DL with a suitable modal language treating belief, temporal, deontic or some other intensional operators. However, there are a number of parameters that determine the design of a modal extension of a given DL.

(I) First, modal operators can be applied to different kinds of well-formed expressions of the DL.

One may apply them only to conceptual and assertional axioms thereby forming new axioms of the form:

$$[\text{John believes}](\text{GoodCar} \equiv \text{Car} \sqcap \exists \text{part.Airconditioner}),$$

$$[\text{Mary believes}] \langle \text{eventually} \rangle (\text{Rich}(\text{JOHN})).$$

Modal operators may also be applied to concepts in order to form new ones:

$$[\text{John believes}] \text{expensive}$$

i.e., the concept of all objects John believes to be expensive, or

$$\text{HumanBeing} \sqcap \exists \text{child}. [\text{Mary believes}] \langle \text{eventually} \rangle \text{GoodStudent}$$

i.e., the concept of all human beings with a child that Mary believes to be eventually a good student. By allowing applications of modal operators to both concepts and axioms we obtain expressions of the form

$$[\text{John believes}](\text{GoodCar} \equiv [\text{Mary believes}]\text{GoodCar})$$

i.e., John believes that a car is good if and only if Mary thinks so.

Finally, one can supplement the options above with modal operators applicable to roles. For example, using the temporal operator $[\text{always}]$ (in future) and the role

loves, we can form the new role $[always]loves$ (which is understood as a relation between objects x and y that holds if and only if x will always love y) to say

$$(\exists[always]loves.Woman)(JOHN)$$

i.e., John will always love the very same woman (but perhaps not only her), which is not the same as $([always]\exists loves.Woman)(JOHN)$.

(II) All these languages are interpreted with the help of the possible worlds semantics, in which the accessibility relations between worlds (or points in time, ...) treat the modal operators, and the worlds themselves are DL interpretations.

The properties of the modal operators are determined by the conditions we impose on the corresponding accessibility relations. For example, by imposing no condition at all we obtain what is known as the minimal normal modal logic \mathbf{K} —although of definite theoretical interest, it does not have the properties required to model operators like $[agent\ A\ knows]$, $\langle eventually \rangle$, etc. In the *temporal* case, depending on the application domain we may assume time to be linear and discrete (for example, the usual strict ordering of the natural numbers), or branching, or dense, etc. (see [Gabbay *et al.*, 1994; van Benthem, 1996]). Moreover, we have the possibility to work with intervals instead of points in time (see Section 6.2.4). In *epistemic logic*, transitivity of the accessibility relation for agent A 's knowledge means what is called *positive introspection* (A knows what she knows), euclideaness corresponds to *negative introspection* (A knows what she does not know), and reflexivity means that everything known by A is true; see Section 6.2.3 for a formulation of these principles in terms of Description Logics. For more information and further references consult [Fagin *et al.*, 1995; Meyer and van der Hoek, 1995].

(III) When connecting worlds—that is, ordinary interpretations of the pure description language—by accessibility relations, we are facing the problem of connecting their objects. Depending on the particular application, we may assume worlds to have arbitrary domains (the *varying domain assumption*), or we may assume that the domain of a world accessible from a world w contains the domain of w (the *expanding domain assumption*), or that all the worlds share the same domain (the *constant domain assumption*); see [van Benthem, 1996] for a discussion in the context of first-order temporal logic. Consider, for instance, the following axioms:

$$\begin{aligned} &\neg[agent\ A\ knows](Unicorn \equiv \perp), \\ &([agent\ A\ knows]\neg Unicorn) \equiv \top. \end{aligned}$$

The former means that agent A does not know that unicorns do not exist, while according to the latter, for every existing object, A knows that it is not a unicorn. Such a situation can be modeled under the expanding domain assumption, but these two formulas cannot be simultaneously satisfied in a model with constant domains.

(IV) Finally, one should take into account the difference between *global* (or *rigid*) and *local* (or *flexible*) symbols. In our context, the former are the symbols which have the same extension in every world in the model under consideration, while the latter are those whose interpretation is not fixed. Again the choice between these depends on the application domain: if the knowledge base is talking about employees of a company then the name *John Smith* should probably denote the same person no matter what world we consider, while *President of the company* may refer to different persons in different worlds. For a more detailed discussion consult, e.g., [Fitting, 1993; Kripke, 1980].

To describe the syntax and semantics more precisely we briefly introduce the modal extension $\mathcal{L}_{\mathcal{ALC}}^n$ of \mathcal{ALC} with n unary modal operators \Box_1, \dots, \Box_n , and their duals $\Diamond_1, \dots, \Diamond_n$.

Definition 6.4 (Concepts, roles, axioms) *Concepts* and *roles* of $\mathcal{L}_{\mathcal{ALC}}^n$ are defined inductively as follows: all concept names are concepts, and if C, D are concepts, R is a role, and \Diamond_i is a modal operator, then $C \sqcap D$, $\neg C$, $\Diamond_i C$, and $\exists R.C$ are concepts.¹ All role names are roles, and if R is a role, then $\Box_i R$ and $\Diamond_i R$ are roles.

Let C and D be concepts, R a role, and a, b object names. Then expressions of the form $C \equiv D$, $R(a, b)$, and $C(a)$ are axioms. If φ and ψ are axioms then so are $\Diamond_i \varphi$, $\neg \varphi$, and $\varphi \wedge \psi$. ■

We remind the reader that models of a propositional modal language are based on Kripke frames, i.e., structures of the form $\mathfrak{F} = \langle W, \triangleleft_1, \dots, \triangleleft_n \rangle$ in which each \triangleleft_i is a binary (accessibility) relation on the set of worlds W . What is going on inside the worlds is of no importance in the propositional framework (see, e.g., [Chagrov and Zakharyashev, 1997] for more information on propositional modal logics). Models of $\mathcal{L}_{\mathcal{ALC}}^n$ are also constructed on Kripke frames; however, in this case their worlds come equipped with interpretations of \mathcal{ALC} .

Definition 6.5 (model) A *model* of $\mathcal{L}_{\mathcal{ALC}}^n$ based on a frame $\mathfrak{F} = \langle W, \triangleleft_1, \dots, \triangleleft_n \rangle$ is a pair $\mathfrak{M} = \langle \mathfrak{F}, I \rangle$ in which I is a function associating with each $w \in W$ an \mathcal{ALC} -interpretation

$$I(w) = \langle \Delta^{I,w}, I^{I,w} \rangle.$$

\mathfrak{M} has constant domain iff $\Delta^{I(v)} = \Delta^{I(w)}$, for all $v, w \in W$. \mathfrak{M} has expanding domains iff $\Delta^{I(v)} \subseteq \Delta^{I(w)}$ whenever $v \triangleleft_i w$, for some i . ■

Definition 6.6 For a model $\mathfrak{M} = \langle \mathfrak{F}, I \rangle$ and a world w in it, the *extensions* $C^{I,w}$

¹ Note that value restrictions (the modal box operators \Box_i) need not explicitly be included here since they can be expressed using negation and existential restrictions (the modal diamond operators \Diamond_i).

and $R^{I,w}$, and the *satisfaction relation* $w \models \varphi$ (φ an axiom) are defined inductively. The interesting new steps of the definition are:

- (i) $x \in (\diamond_i C)^{I,w}$ iff $\exists v. v \triangleright_i w$ and $x \in C^{I,v}$;
- (ii) $(x, y) \in (\diamond_i R)^{I,w}$ iff $\exists v. v \triangleright_i w$ and $(x, y) \in R^{I,v}$;
- (iii) $w \models \diamond_i \varphi$ iff $\exists v. v \triangleright_i w$ and $v \models \varphi$.

An axiom φ (a concept C) is *satisfiable* in a class of models \mathcal{M} if there is a model $\mathfrak{M} \in \mathcal{M}$ and a world w in \mathfrak{M} such that $w \models \varphi$ ($C^{I,w} \neq \emptyset$). ■

Given a class of frames \mathcal{K} , the satisfiability problems for axioms and concepts in \mathcal{K} are the most important reasoning tasks; others are reducible to them (see [Wolter and Zakharyashev, 1998; 1999b]). Notice that the satisfiability problem for concepts is reducible to that for axioms since $\neg(C \equiv \perp)$ is satisfiable iff C is satisfiable. Also, the satisfiability problem for models with expanding or varying domain is reducible to that for models with constant domain (see [Wolter and Zakharyashev, 1998]).

We are now going to survey briefly the state of the art in the field. We will restrict ourselves first to modal description logics which are not temporal logics. The latter will be considered in Section 6.2.4. Chronologically, the first investigations of modal description logics are [Laux, 1994; Gräber *et al.*, 1995; Baader and Laux, 1995; Baader and Ohlbach, 1993; 1995]. The papers [Laux, 1994; Gräber *et al.*, 1995] construct multi-agent epistemic description logics in which the belief operators apply only to axioms; the accessibility relations are transitive, serial, and euclidean. The decidability of the satisfiability problem for axioms follows immediately from the decidability of both, the propositional fragment of the logic and \mathcal{ALC} , because in languages without modalized concepts and roles there is no interaction between the modal operators and role quantification (see [Finger and Gabbay, 1992]). Baader and Laux [1995] introduce a DL in which modal operators can be applied to both axioms and concepts (but not to roles); it is interpreted in models with arbitrary accessibility relations under the expanding domain assumption. The decidability of the satisfiability problem for axioms is proved by constructing a complete tableau calculus. This tableau calculus was modified and extended for checking satisfiability in models with constant domain in [Lutz *et al.*, 2002]. It decides satisfiability in constant domain models in NEXPTIME, which matches the lower bound established in [Mosurovic and Zakharyashev, 1999] (see also [Gabbay *et al.*, 2002]).

The papers [Wolter and Zakharyashev, 1998; 1999a; 1999c; 1999b; Wolter, 2000; Mosurovic and Zakharyashev, 1999] investigate the decision problem for various families of modal description logics in detail. For example, in [Wolter and Zakharyashev, 1999c; 1999b] it is shown that the satisfiability problem for arbitrary axioms (possibly containing modalized roles) is decidable in the class of all frames

and in the class of polymodal **S5**-frames—frames in which all accessibility relations are equivalence relations—based on constant, expanding, and varying domains. It becomes undecidable, however, if common knowledge epistemic operators (in the sense of [Fagin *et al.*, 1995]) are added to the language or if the class of frames consists of the flow of time $\langle \mathbb{N}, < \rangle$. In [Wolter and Zakharyashev, 1999a; 1998] it is shown that for expressive modal languages—like logics with common knowledge operators or Propositional Dynamic Logics—the satisfiability problem for axioms becomes decidable when modalized roles are not included. Wolter [2000] shows that the satisfiability problem for concepts interpreted in frames with global (i.e., world-independent) roles is decidable for expressive modal logics based on \mathcal{ALC} while the satisfiability problem for axioms is undecidable for them. However, even the complexity of the satisfiability problem for concepts becomes non-elementary for these logics [Gabbay *et al.*, 2002]. In fact, for various decidable modal description logics only computationally non-elementary decision procedures are known and the precise complexity has not yet been determined (consult [Gabbay *et al.*, 2002] for further results).

The papers [Baader and Ohlbach, 1993; 1995] introduce a multi-dimensional description language that is even more expressive than $\mathcal{L}_{\mathcal{ALC}}^n$ (but without object names). Roughly, in this approach each dimension (object, time, belief, etc.) is represented by a set D_i (of objects, moments of time, possible worlds, etc.), concepts are interpreted as subsets of the cartesian product $\prod_{i=1}^n D_i$, and roles of dimension i as binary relations between n -tuples that may differ only in the i th coordinate. One can quantify over both, roles and concepts, *in any dimension*. Thus, in contrast to $\mathcal{L}_{\mathcal{ALC}}^n$ arbitrarily many dimensions are considered and no dimension is labelled as the “modal” or “ \mathcal{ALC} ”-one. This language has turned out to be extremely expressive. The satisfiability problem for the full language is known to be undecidable and even for natural fragments no sound and complete reasoning procedures have appeared. Baader and Ohlbach [1995] provide only a sound satisfiability checking algorithm for such a fragment.

6.2.3 Epistemic operators

The systems CLASSIC and LOOM provide their users with the possibility to include *procedural rules* into knowledge bases (see also Chapter 2, Section 2.2.5). Such rules take the form

$$C \Rightarrow D,$$

where C and D are concepts. The meaning of a procedural rule is different from the meaning of an inclusion axiom: while $C \sqsubseteq D$ represents conceptual knowledge and says that—no matter what is known about individuals—the concept D subsumes

C , the rule $C \Rightarrow D$ represents the incidental fact that “if an individual is known to be an instance of C , then we can conclude that it is an instance of D ”. Consider the following example: suppose a knowledge base Φ consists of

$$\text{GreatLogician} \sqsubseteq \text{Professor}, \quad \neg\text{Professor}(a).$$

Obviously we can derive $\neg\text{GreatLogician}(a)$ from Φ . In this representation we assume a conceptual relation between the terms ‘professor’ and ‘great logician’. More appropriate, however, seems to be the weaker claim that people who are known to be great logicians are professors: let Φ' be the knowledge base which results from Φ when $\text{GreatLogician} \sqsubseteq \text{Professor}$ is replaced with

$$\text{GreatLogician} \Rightarrow \text{Professor}.$$

The assertion $\neg\text{GreatLogician}(a)$ turns out to be not derivable from Φ' . The *procedural* explanation for this phenomenon is this: in the knowledge base Φ' we do not find an individual belonging to the concept **GreatLogician**. Therefore the rule **GreatLogician** \Rightarrow **Professor** does not “fire” and nothing new about the world is derivable by using it. However, Description Logic is aiming at an extensional semantics for frame-based systems, hence it would be desirable to have a precise model-theoretic explanation of the behavior of procedural rules as well.

It turns out that adding an *epistemic operator* together with a possible worlds semantics interpreting it provides us with the required models. Integrating the operator **K**—‘the knowledge base knows that’—into \mathcal{ALC} will allow us to rephrase the rule **GreatLogician** \Rightarrow **Professor** by the inclusion axiom **KGreatLogician** \sqsubseteq **Professor**, which says that all objects that are *known* to be great logicians are professors. Actually, it will turn out that extensions of Description Logics by means of epistemic operators are useful in other contexts as well. We postpone their discussion until we have introduced some technical prerequisites. We will follow [Donini *et al.*, 1992b; 1998a], where the extension of \mathcal{ALC} by epistemic operators was introduced and investigated.

Formulated in terms of Section 6.2.2, we consider the language $\mathcal{L}_{\mathcal{ALC}}^1$ in which the modal operator \Box_1 (now denoted by **K**) can be applied to concepts and roles but not to axioms. Following [Donini *et al.*, 1998a] we call this language \mathcal{ALCK} . The following principles are assumed to govern the epistemic operator (we formulate them here for **K** applied to concepts; the formulation for roles is similar):

- **KC** \sqsubseteq C (only true facts are known: if an object is known to be an instance of C , then it is an instance of C);
- **KC** \sqsubseteq **KKC** (positive introspection: if it is known that an object is an instance of C , then this is known);

- $\neg\mathbf{KC} \sqsubseteq \mathbf{K}\neg\mathbf{KC}$ (negative introspection: if it is not known whether an object is an instance of C , then this is known).

These principles are valid in all models based on a Kripke frame $\mathfrak{F} = \langle W, \triangleleft \rangle$ iff \mathfrak{F} is an **S5**-frame, or, equivalently, if \triangleleft is the universal relation on W , i.e., $\triangleleft = W \times W$. So, we consider frames of the form $\langle W, W \times W \rangle$ only.

We assume also that:

- it is known which object an object name denotes (so, object names are assumed to be global (or rigid) designators),
- the set of existing objects Δ is known and countably infinite (so, we adopt the constant domain assumption).

These assumptions together allow us to simplify the possible worlds semantics considerably: we can identify the set of worlds W with a set of interpretations \mathcal{M} (all having the same countably infinite domain Δ and the same interpretation of the object names) and the accessibility relation is implicitly given as the universal relation on \mathcal{M} . Hence, we call any set of interpretations \mathcal{M} satisfying these constraints a *model* (for \mathcal{ALCK}) and can define the *extensions* $C^{\mathcal{I},\mathcal{M}}$ and $R^{\mathcal{I},\mathcal{M}}$ of a concept C and a role R in an interpretation \mathcal{I} in \mathcal{M} as follows:

$$\begin{aligned}
A^{\mathcal{I},\mathcal{M}} &= A^{\mathcal{I}} \text{ for atomic concepts } A \\
P^{\mathcal{I},\mathcal{M}} &= P^{\mathcal{I}} \text{ for atomic roles } P \\
(\neg C)^{\mathcal{I},\mathcal{M}} &= \Delta \setminus C^{\mathcal{I},\mathcal{M}} \\
(C_1 \sqcap C_2)^{\mathcal{I},\mathcal{M}} &= C_1^{\mathcal{I},\mathcal{M}} \cap C_2^{\mathcal{I},\mathcal{M}} \\
(\exists R.C)^{\mathcal{I},\mathcal{M}} &= \{a \in \Delta \mid \exists b. (a, b) \in R^{\mathcal{I},\mathcal{M}} \wedge b \in C^{\mathcal{I},\mathcal{M}}\} \\
(\mathbf{KC})^{\mathcal{I},\mathcal{M}} &= \bigcap_{\mathcal{J} \in \mathcal{M}} C^{\mathcal{J},\mathcal{M}} (= \{a \in \Delta \mid \forall \mathcal{J} \in \mathcal{M}. a \in C^{\mathcal{J},\mathcal{M}}\}) \\
(\mathbf{KR})^{\mathcal{I},\mathcal{M}} &= \bigcap_{\mathcal{J} \in \mathcal{M}} R^{\mathcal{J},\mathcal{M}} (= \{(a, b) \in \Delta \mid \forall \mathcal{J} \in \mathcal{M}. (a, b) \in R^{\mathcal{J},\mathcal{M}}\})
\end{aligned}$$

So, \mathbf{KC} comprises the set of all objects that are instances of C in every world regarded as possible.

An \mathcal{ALCK} -*knowledge base* Φ consists of a set of inclusion axioms and ABox assertions whose concepts and roles are in \mathcal{ALCK} . A model \mathcal{M} *satisfies* Φ (is a Φ -model) iff all inclusion and membership assertions of Φ are true in every $\mathcal{I} \in \mathcal{M}$.

So far, we have introduced a rather simple version of the epistemic extensions of \mathcal{ALC} discussed in Section 6.2.2. In the present section, however, we are not interested in the satisfiability of epistemic knowledge bases, but in a relation \models between knowledge bases and assertions such that $\Phi \models \varphi$ iff a knowledge base knows φ under the assumption that “all the knowledge base knows is Φ ”. For example, if Φ is empty (the knowledge base knows nothing), then $\neg\mathbf{KC}(a)$ as well as $\neg\mathbf{K}\neg C(a)$

should be derivable, since the knowledge base does not know whether a is an instance of C or not. On the semantic level this means that we are not interested in arbitrary models satisfying Φ but only in those Φ -models that refute as many \mathcal{ALC} -assertions as possible. In other words, we consider Φ -models only with as many worlds as possible (corresponding to the intuition that more worlds are regarded as possible if less is known). For example, if Φ is empty, then the intended models comprise *all* interpretations (with a fixed domain and interpretation of the object names), since all interpretations are regarded as possible by an empty knowledge base. Here are the precise definitions:

Definition 6.7 An *epistemic model* for Φ is a *maximal* non-empty set of interpretations \mathcal{M} satisfying Φ . The knowledge base Φ *logically implies* an assertion φ , written $\Phi \models \varphi$, if every epistemic model \mathcal{M} for Φ satisfies φ . ■

Consequently, \models is a *non-monotonic* consequence relation: while $\emptyset \models (\neg \mathbf{K}C \wedge \neg \mathbf{K}\neg C)(a)$, we have $C(a) \models \mathbf{K}C(a)$. On the propositional level, this type of reasoning is known as *ground non-monotonic S5* (see [Donini *et al.*, 1995; 1997c; Nardi and Rosati, 1995]).

Reasoning with arbitrary \mathcal{ALCK} -knowledge bases has not been investigated. In fact, all applications considered in the literature require only very small fragments of \mathcal{ALCK} . In what follows, we shall briefly introduce two such fragments and some of their applications.

6.2.3.1 \mathcal{ALCK} as a query language

We first confine ourselves to knowledge bases that are ordinary \mathcal{ALC} -ABoxes. Hence, the epistemic operator \mathbf{K} can be used only in queries. Recall that concept languages can be applied as query languages in a straightforward manner: the answer set of a query consisting of a concept C to a knowledge base Φ comprises the set of individuals a with $\Phi \models C(a)$. Queries with epistemic operators enable us to extract the knowledge which the knowledge base has about its own knowledge. Consider, for example, the knowledge base $\Phi = \{\exists \text{friend.Male}(\text{SUSAN})\}$, which contains incomplete information about Susan. Applications of \mathbf{K} to different concepts and roles in $\exists \text{friend.Male}$ enable us to form a variety of different queries:

- $\exists \text{friend.Male}$; clearly, the answer to this query is $\{\text{SUSAN}\}$.
- $\exists \text{friend.KMale}$; the answer set is empty, since no *known* male is a friend of Susan.
- $\exists \mathbf{K}\text{friend.Male}$; the answer set is empty since we do not find a male individual that is *known* to be a friend of Susan.
- $\mathbf{K}\exists \text{friend.Male}$; the answer set is $\{\text{SUSAN}\}$ since the knowledge base *knows* that Susan has a friend who is male.

Observe that, for $\Phi' = \Phi \cup \{\text{friend}(\text{SUSAN}, \text{BOB}), \text{Male}(\text{BOB})\}$, the answer set would consist of SUSAN in all four cases. We refer the reader to [Donini *et al.*, 1992b; 1998a] for more examples.

Epistemic queries can also be used to formulate *integrity constraints*. Recall that integrity constraints can be viewed as epistemic sentences that state what a knowledge base must know about the world [Reiter, 1990]. For example, suppose that we want to rule out those knowledge bases that are uncertain about whether a given course is a course for undergraduates or graduates. This can be expressed using the query

$$\neg \mathbf{K}\text{Course} \sqcup (\mathbf{K}\text{Undergraduate} \sqcup \mathbf{K}\text{Graduate}). \quad (6.2)$$

A knowledge base satisfies the integrity constraint iff it logically implies the assertion (6.2)(*a*), for every object name *a* appearing in it. Observe, by the way, that the query $\neg \text{Course} \sqcup (\text{Undergraduate} \sqcup \text{Graduate})$ has a different meaning: while $\emptyset \models (6.2)(a)$, for all *a* (corresponding to the intention), $\emptyset \not\models (\neg \text{Course} \sqcup (\text{Undergraduate} \sqcup \text{Graduate}))(a)$. We refer the reader to [Levesque, 1984; Lifschitz, 1991; Reiter, 1990] for a discussion of the use of epistemic queries in general.

What is the computational complexity of querying \mathcal{ALC} -ABoxes by means of \mathcal{ALCK} -concepts? The following result is proved in [Donini *et al.*, 1992b; 1998a]:

Theorem 6.8 *There is an algorithm for deciding, given an \mathcal{ALC} -ABox Σ , an object name *a*, and an \mathcal{ALCK} -concept *C*, whether $\Sigma \models C(a)$. More precisely, the problem $\Sigma \models C(a)$ is PSPACE-complete (w.r.t. the size of *C* and Σ).*

Recall that querying \mathcal{ALC} -ABoxes with \mathcal{ALC} -concepts is PSPACE-complete as well [Hollunder, 1996]. Thus, the additional epistemic operators in queries do not cause any increase of the computational complexity.

6.2.3.2 Semantics for procedural rules

To capture the meaning of procedural rules as discussed above (and in Chapter 2, Section 2.2.5), we must admit assertions of the form $\mathbf{K}C \sqsubseteq D$ in the knowledge base. A *rule ABox* consists of an \mathcal{ALC} -ABox and a set of sentences of the form

$$\mathbf{K}C \sqsubseteq D,$$

where *C*, *D* are \mathcal{ALC} -concepts and *C* is not equivalent to \top (the reason for this technical condition will be discussed below).

Fortunately, the additional inclusion axioms again do not lead to any increase of the complexity [Donini *et al.*, 1992b; 1998a].

Theorem 6.9 *There is an algorithm for deciding, given a rule \mathcal{ALC} -ABox Σ , an*

object name a , and an \mathcal{ALCK} -concept C , whether $\Sigma \models C(a)$. More precisely, the problem $\Sigma \models C(a)$ is PSPACE-complete (w.r.t. the size of C and Σ).

Observe that this result does not extend to the language with inclusion axioms of the form $\mathbf{KC} \sqsubseteq D$, where C is equivalent to \top . In this case \mathbf{KC} would be equivalent to \top as well, and so $\mathbf{KC} \sqsubseteq D$ would be equivalent to $D \equiv \top$. However, for knowledge bases with axioms of this type instance checking is known to be EXPTIME-complete [Schild, 1994]. Notice that in applications a rule of the form $\top \Rightarrow C$ does not make sense.

6.2.3.3 An extension of \mathcal{ALCK}

The non-monotonic logic $MKNF$ is an expressive extension of ground non-monotonic $\mathbf{S5}$, which can simulate in a natural manner Default Logic, Autoepistemic Logic, and Circumscription (see [Lifschitz, 1994]). This is achieved by adding to classical logic not only the operator \mathbf{K} (of ground non-monotonic $\mathbf{S5}$) but also a second epistemic operator \mathbf{A} , which is interpreted in terms of autoepistemic assumption. The papers [Donini *et al.*, 1997b; Rosati, 1998] study the corresponding bimodal extension of \mathcal{ALC} by means of \mathbf{K} and \mathbf{A} , called \mathcal{ALCB} in what follows.

We first consider the two operators \mathbf{K} and \mathbf{A} separately: the consequence relation \models for assertions containing \mathbf{K} only is still the one introduced above. On the other side, for assertions containing \mathbf{A} (‘it is assumed that’) only we are interested in a consequence relation \models^{AE} such that $\Phi \models^{AE} \varphi^1$ iff φ belongs to every stable expansion of Φ , i.e., iff φ belongs to every reasonable theory² about the world which a rational agent who assumes only the assertions in Φ can have. In particular, it is assumed that agents are capable of introspection. Consider, for example, an agent assuming precisely $\Phi = \{\mathbf{AC} \equiv \top\}$ (‘the set of all objects I assume to be in C comprises all existing objects’). We still assume that agents know which objects exist (the constant domain assumption). Hence Φ can be rephrased as ‘I assume that all objects belong to C ’. Now, according to the autoepistemic approach such an agent cannot have a coherent theory about the world because if she would have one then she should assume as well that $C \equiv \top$ from the very beginning.

From the ‘possible worlds’ viewpoint the relation \models^{AE} can be captured as follows. Firstly, the extension of \mathcal{ALC} by \mathbf{A} is interpreted in pairs $(\mathcal{I}, \mathcal{M})$ in precisely the same manner as \mathcal{ALCK} . However, now we allow that the actual world \mathcal{I} is not in \mathcal{M} —corresponding to the idea that assumptions (in contrast to known assertions) are not always true. Thus we may have $(\mathbf{AC})^{\mathcal{I}, \mathcal{M}} = \top$ but $C^{\mathcal{I}, \mathcal{M}} \neq \top$, which is not possible for \mathbf{K} . The intended models are called AE-models in what follows.

¹ AE indicates that autoepistemic propositional logic in the sense of [Moore, 1985] is extended here to \mathcal{ALC} .

² In terms of propositional logic a theory T is called reasonable iff the following conditions hold: (0) T is closed under classical reasoning, (1) if $P \in T$, then $\mathbf{AP} \in T$, (2) if $P \notin T$, then $\neg \mathbf{AP} \in T$.

Definition 6.10 An *AE-model* for a set of assertions Φ is a set of interpretations \mathcal{M} that satisfies Φ and such that, for every interpretation $\mathcal{I} \notin \mathcal{M}$, Φ is refuted in $(\mathcal{I}, \mathcal{M})$. Now put $\Phi \models^{AE} \varphi$ iff φ is satisfied in all AE-models for Φ . ■

So, we do not maximize the set of possible worlds, but we exclude the case that Φ is true in an actual world that is not regarded possible (i.e., is not a member of \mathcal{M}). The consequence relation \models^{AE} is also non-monotonic since $\emptyset \models^{AE} \neg \mathbf{A}C(a)$ but $C(a) \models \mathbf{A}C(a)$. Observe that \models and \models^{AE} are different: while $\mathbf{A}C \equiv \top$ has no AE-models, $\mathbf{K}C \equiv \top$ has the epistemic model consisting of all interpretations in which $C \equiv \top$.

How to interpret the combined language \mathcal{ALCB} and define a consequence relation? Following Lifschitz [1994], the intended models (called \mathcal{ALCB} -models) are defined as follows.

Definition 6.11 The *\mathcal{ALCB} -models* for a set of \mathcal{ALCB} -assertions Φ are those models \mathcal{M} satisfying Φ and the following maximality condition: if a non-empty set of new worlds \mathcal{N} is added to \mathcal{M} , \mathbf{K} is interpreted in the model $\mathcal{M} \cup \mathcal{N}$, and \mathbf{A} is interpreted in the old model \mathcal{M} , then Φ is refuted in some interpretation from \mathcal{N} . Now Φ *logically implies* φ , in symbols $\Phi \models \varphi$, iff φ is satisfied in every \mathcal{ALCB} -model satisfying Φ . ■

Thus, roughly speaking, we still maximize the set of worlds, but now we require that any larger set of possible worlds contains a world at which Φ is refuted under the interpretation of \mathbf{A} by means of the original set of possible worlds. But this corresponds, for the operator \mathbf{A} , to the definition of AE-models. Clearly, the new consequence relation is a conservative extension of the one defined for \mathcal{ALCK} above (and of \models^{AE} as well). Hence using the same symbol for both does not cause any ambiguity.

The new logic is considerably more expressive than \mathcal{ALCK} . Donini *et al.* [1997b] show that Default Logic can be embedded into \mathcal{ALCB} more naturally than into \mathcal{ALCK} . They also consider the formalization of integrity constraints *in* knowledge bases, which cannot be expressed in \mathcal{ALCK} , and they discuss how role and concept closure can be formalized in \mathcal{ALCB} . Here we confine ourselves to a brief discussion of the formalization of integrity constraints in \mathcal{ALCB} . Above we have seen that the query (6.2) can be used to express the constraint that every course known to the knowledge base should be known to be for undergraduates or graduates. Sometimes it is more useful not to formalize integrity constraints as queries, but as part of the knowledge base (see [Donini *et al.*, 1997b]). However, the addition of constraints should not change the content of the knowledge base, but just force the knowledge base to be inconsistent iff the constraint is violated. How can this be achieved in

\mathcal{ALCK} ? The naive idea is to add the assertion $(6.2) \equiv \top$ to the knowledge base in order to express the constraint. Unfortunately, this does not work: consider the knowledge base Φ consisting of $\text{Course}(a)$, which does not satisfy the integrity constraint. However, the knowledge base obtained from Φ by adding $(6.2) \equiv \top$ does not tell us that the constraint is violated in Φ since the extended knowledge base is still consistent: the set \mathcal{M} consisting of all interpretations \mathcal{I} (with a fixed domain and interpretation of a) satisfying $a^{\mathcal{I}} \in \text{Course}^{\mathcal{I}} \cap \text{Graduate}^{\mathcal{I}}$ is an epistemic model for the extended knowledge base. In fact, there is no way to formulate the required constraint within \mathcal{ALCK} . On the other hand, by adding the \mathcal{ALCB} -assertion

$$\mathbf{K}\text{Course} \sqsubseteq \mathbf{A}\text{Graduate} \sqcup \mathbf{A}\text{Undergraduate}$$

to Φ , we obtain a knowledge base without \mathcal{ALCB} -models, as required. Note, for example, that the model \mathcal{M} introduced above is not an \mathcal{ALCB} -model for this knowledge base because any set of worlds $\mathcal{N} = \{\mathcal{I}\}$ with $\mathcal{I} \notin \mathcal{M}$ and $a^{\mathcal{I}} \in \text{Course}^{\mathcal{I}}$ refutes the maximality condition.

Donini *et al.* [1997b] present a number of decidability results for reasoning with \mathcal{ALCB} knowledge bases.

6.2.4 Temporal extensions

Temporal extensions are a special form of modal extensions of description logics. However, because of the intended interpretation in flows of time they have a specific flavour, which is slightly different from general modal logic. Chronologically, the first example of a “modalized” description logic was the temporal description logic of Schmiedel [1990]. The papers [Bettini, 1997; Artale and Franconi, 1994; 1998] introduce and investigate variants of Schmiedel’s formalism. The papers mentioned so far employ an *interval-based* approach to the semantics of temporal operators. *Point-based* temporal description logics have been introduced by Schild [1993] and further investigated by Wolter and Zakharyashev [1999e].

For simplicity, let us first consider propositional temporal logic and then see how it can be extended to temporal description logic. In what follows we assume that a *flow of time* $\mathfrak{T} = \langle T, < \rangle$ consists of a set of points in time T and a precedence relation $<$ between points in time which is assumed to be a strict linear order. This corresponds to the intuition that, for any two moments $t_1, t_2 \in T$, either t_1 precedes t_2 , t_2 precedes t_1 , or t_1 equals t_2 .

How to define a satisfiability relation \models between entities in a flow of time and formulas? There exist (at least) two different possibilities to select the entities at which formulas are evaluated: points in time and intervals. While in the first case we are considering a relation $t \models \varphi$ between time-points t and formulas φ , in the second case we have a relation $[u, v] \models \varphi$ between intervals $[u, v] = \{z \in T \mid u \leq z \leq v\}$,

where $u \leq v$, in \mathfrak{T} and formulas φ . Denote by \mathfrak{T}^* the set of all intervals in \mathfrak{T} . Both, point- and interval-based temporal logics, are special instances of modal logics: in the former the worlds of Kripke frames are interpreted as time-points while in the latter they are interpreted as intervals. Point- as well as interval-based temporal models are easily extended to temporal \mathcal{ALC} -models:

Definition 6.12 A *point-based temporal \mathcal{ALC} -model* $\mathfrak{M} = (\mathfrak{T}, I)$ consists of a flow of time \mathfrak{T} and a function I which associates with every $t \in T$ an interpretation

$$I(t) = \langle \Delta^{I,t}, \cdot^{I,t} \rangle.$$

An *interval-based temporal \mathcal{ALC} -model* $\mathfrak{M} = \langle \mathfrak{T}, I \rangle$ consists of a flow of time \mathfrak{T} and a function I which associates with every interval $i \in \mathfrak{T}^*$ an interpretation

$$I(i) = \langle \Delta^{I,i}, \cdot^{I,i} \rangle. \quad \blacksquare$$

We can now evaluate \mathcal{ALC} -concepts and axioms in point- and interval-based temporal models. For example,

- $(\mathfrak{M}, t) \models \text{Alive}(a)$ iff $a^{I,t} \in \text{Alive}^{I,t}$, i.e., a is alive at moment t ,
- $(\mathfrak{M}, i) \models \text{Sleep}(a)$ iff $a^{I,i} \in \text{Sleep}^{I,i}$, i.e., a is sleeping in the interval i .

We now add temporal operators and quantifiers to \mathcal{ALC} , which enable us to relate different moments and intervals to each other.

For the point-based approach we have discussed appropriate operators already: we can form the language $\mathcal{L}_{\mathcal{ALC}}^1$ and interpret the operator $\Box = \Box_1$ as ‘*always in the future*’. Thus, $t \models \Box(C \equiv D)$ iff $t' \models C \equiv D$ for all $t' > t$, (always in the future of t , C and D are interpreted as the same set), and $x \in (\Diamond C)^{I,t}$ iff there exists $t' > t$ such that $x \in C^{I,t'}$ (eventually x is an instance of C). Often, however, more expressive temporal operators are required. The operator \mathcal{U} (until), for example, is a binary temporal operator with the following truth-conditions, for all concepts C , D and axioms φ, ψ :

- (i) $x \in (CUD)^{I,t}$ iff there exists $t' > t$ such that $x \in D^{I,t'}$ and, for all t'' with $t < t'' < t'$, $x \in C^{I,t''}$,
- (ii) $t \models \varphi \mathcal{U} \psi$ iff there exists $t' > t$ such that $t' \models \psi$ and, for all t'' with $t < t'' < t'$, $t'' \models \varphi$.

In this language we can define a mortal as, say, a living being that is alive until it dies:

$$\text{Mortal} \equiv \text{LivingBeing} \Box (\text{LivingBeing} \mathcal{U} \Box \neg \text{LivingBeing}).$$

This language, interpreted in the flow of time $\langle \mathbb{N}, < \rangle$, was first considered by Schild [1993], who showed that the satisfiability problem for concepts (without

modalized or global roles) is decidable. Wolter [2000] proves the decidability for concepts with global roles (but without modalized roles). However, the complexity of the decision problem for this language is non-elementary [Gabbay *et al.*, 2002]. Wolter and Zakharyashev [1999e] prove that even for axioms the satisfiability problem is decidable, provided that they do not contain modalized or global roles. Tableau calculi (running in double-exponential time) for the case of expanding and constant domains were developed in [Sturm and Wolter, 2002; Lutz *et al.*, 2001b]. The satisfiability problem for axioms in the full language with the flow of time $\langle \mathbb{N}, < \rangle$ is undecidable.

For the interval-based approach we find both languages that extend \mathcal{ALC} by means of temporal operators which are interpreted by accessibility relations between intervals [Bettini, 1997] and languages that allow for explicit quantification over intervals [Schmiedel, 1990; Artale and Franconi, 1994; 1998].

We start the discussion with the temporal operators approach. Bettini [1997] extends the propositional interval-based temporal logic of [Halpern and Shoham, 1991] to \mathcal{ALC} (and weaker description logics). Thus, given a concept C , we can now form new concepts like $\langle \textit{starts} \rangle C$ and $\langle \textit{finishes} \rangle C$. They are interpreted in interval-based models $\langle \mathfrak{I}, I \rangle$ as follows:

- $x \in (\langle \textit{starts} \rangle C)^{I,[u,v]}$ iff $\exists t \in T. u \leq t < v \wedge x \in C^{I,[u,t]}$
 (x is an instance of $\langle \textit{starts} \rangle C$ in the interval $[u, v]$ iff x is an instance of C in some interval starting $[u, v]$),
- $x \in (\langle \textit{finishes} \rangle C)^{I,[u,v]}$ iff $\exists t \in T. u < t \leq v \wedge x \in C^{I,[t,v]}$.

In other words, the modal operators $\langle \textit{starts} \rangle$ and $\langle \textit{finishes} \rangle$ are interpreted in the standard “possible worlds manner” by means of the accessibility relations ‘*starts*’ and ‘*finishes*’, respectively, where $(i, j) \in \textit{starts}$ iff j starts i and $(i, j) \in \textit{finishes}$ if j finishes i . By adding the converse operators of $\langle \textit{starts} \rangle$ and $\langle \textit{finishes} \rangle$ to the language, we obtain a language that can express all the thirteen Allen relations between intervals [Allen, 1983]. Here is a definition of *Mortal* in this language:

$$\textit{Mortal} \equiv \textit{LivingBeing} \sqcap \langle \textit{after} \rangle \neg \textit{LivingBeing}.$$

Unfortunately, for the full language based on \mathcal{ALC} the satisfiability problem for concepts is undecidable in all interesting flows of time. This follows from the fact that propositional interval-based temporal logic is undecidable already in $\langle \mathbb{R}, < \rangle$, $\langle \mathbb{Q}, < \rangle$, $\langle \mathbb{N}, < \rangle$, etc. (see [Halpern and Shoham, 1991]). However, there are numerous open decision problems when description logics weaker than \mathcal{ALC} and different notions of intervals are considered (see [Bettini, 1997; Artale and Franconi, 2000; 2001]).

Now, let us consider interval-based temporal extensions of description logics that

allow for explicit quantification over intervals. Schmiedel [1990] develops an expressive formalism in which we have two quantifiers $\Box(i)$ ¹ (‘for all intervals i ’) and $\Diamond(i)$ (‘there exists an interval i ’), where i is a variable ranging over intervals. The language does not contain negation so that the quantifiers are not mutually definable. The quantifiers are relativized (alias bounded or guarded) by so called *time nets*, which can, for example, be some relations like *starts* or *finishes* between intervals (metric and granularity constraints are admitted as well). An operator @ specifies the interval at which a concept applies to an object and \sharp denotes a reference interval. The following concept can be regarded as a definition of the concept **Mortal** in Schmiedel’s language:

$$\text{LivingBeing} \sqcap (\Diamond(i)(\text{after } i \sharp)(\neg \text{LivingBeing} @ i)).$$

Here (after $i \sharp$) is the time net which relativizes the quantifier $\Diamond(i)$ by means of the constraint expressing that i must be after the reference interval denoted by \sharp . According to this definition, an object x is an instance of **Mortal** at the reference interval \sharp iff x is living at \sharp and there exists an interval i that is after \sharp , and at which x is not living.

Schmiedel [1990] does not address computational problems for his language. However, it is not difficult to see that, in the presence of negation, this language is more expressive than the one of Bettini [1997] considered above—and thus subsumption is undecidable for all interesting flows of time. The decision problem for the language without negation appears to be open.

A brief remark concerning the relation between interval-based temporal logic with and without explicit quantification over intervals is in order. Of course, explicit quantification provides more expressive power. Using the temporal operators introduced above, it is not possible to represent relations between more than two intervals because reference to a fixed reference interval is impossible. On the other hand, variable-free languages are much closer in spirit to pure description logics and therefore seem to be more natural candidates for temporalizations of description logics; we refer the reader to [Artale and Franconi, 2000; 2001] for a detailed discussion.

The papers [Artale and Franconi, 1994; 1998] present a number of languages weaker than Schmiedel’s with a decidable subsumption problem. Among others, they define a temporal extension of a description logic extending \mathcal{ALC} with functional roles. They show decidability of concept subsumption and PSPACE-completeness of satisfiability w.r.t. an empty KB in an unbounded and dense flow of time. The main reason for the decidability is that the language does not admit universal quantification over intervals and that the constructors of the underlying

¹ Here and in what follows we use the notation of [Artale and Franconi, 1998].

description logic cannot be applied to the temporalized part of the language. In particular, the negation of the underlying DL cannot be used to define the universal quantifier by means of the existential one. The authors show by means of a number of examples that their formalism still has enough expressive power to represent non-trivial actions and plans.

An interesting feature of the subsumption algorithm presented by Artale and Franconi [1998] is that it consists of two parts: firstly, a normalization procedure is employed to reduce the subsumption problem for the temporalized DL to that problem for the pure DL, which can then be solved with known algorithms [Hollunder and Nutt, 1990]

For a more detailed survey of the state of art in temporal description logic we refer the reader to [Artale and Franconi, 2000; 2001], where one can also find an introduction to the work of Weida and Litman [1992], who propose a loose hybrid integration between description logics and constraint networks with the aim of reasoning about plans.

6.2.5 Representing uncertain and vague knowledge

Description Logics whose semantics is based on classical first-order logic cannot express vague or uncertain knowledge. To overcome this deficiency, approaches for integrating probabilistic logic and fuzzy logic into Description Logics have been proposed. Although both types of approaches assign numerical values to entries in the knowledge base, they are quite different, not only from a technical point of view, but also w.r.t. the basic phenomena they are trying to model. We talk about uncertainty if we deal with propositions that are either true or false, but due to a lack of information we do not know for certain which is the case. This gives rise to statements about the probability with which a proposition is assumed to be true. In contrast, vagueness means that the propositions themselves are only true to a certain degree. This vagueness is not caused by incomplete knowledge; it is due to the fact that fuzzy notions, i.e., notions without crisp boundaries (e.g., tall person) are modeled.

In the following, we will restrict our attention to the probabilistic extensions of DLs introduced in [Heinsohn, 1994; Jaeger, 1994; Koller *et al.*, 1997; Yelland, 2000] and the fuzzy extensions of DLs introduced in [Yen, 1991; Tresp and Molitor, 1998; Straccia, 1998; 2001]. The possibilistic extension by Hollunder [1994b] can be viewed as lying between these two approaches: possibilistic logic is mainly used to model uncertainty, but its formal semantics is defined in terms of fuzzy sets of interpretations.

6.2.5.1 Probabilistic extensions

Let us first concentrate on how to extend the terminological (TBox) formalism. In classical Description Logics, one has very restricted means of expressing (and testing for) relationships between concepts. Given two concepts C and D , subsumption tells us whether C is contained in D , and the satisfiability test (applied to $C \sqcap D$) tells us whether C and D are disjoint. Relationships that are in-between (e.g., 90% of all C s are D s) can neither be expressed nor be derived.

This deficiency is overcome in [Heinsohn, 1994; Jaeger, 1994] by allowing for *probabilistic terminological axioms* of the form¹

$$P(C|D) = p,$$

where C, D are concept descriptions and $0 < p < 1$ is a real number. Such an axiom states that the conditional probability for an object known to be in D to belong to C is p . A given *finite* interpretation \mathcal{I} satisfies $P(C|D) = p$ iff

$$\frac{|(C \sqcap D)^{\mathcal{I}}|}{|D^{\mathcal{I}}|} = p.$$

More generally, the formal semantics of the extended language is defined in terms of probability measures on the set of all concept descriptions (modulo equivalence).

Given a knowledge base \mathcal{P} consisting of probabilistic terminological axioms, the main *inference task* is then to derive optimal bounds for additional conditional probabilities. Intuitively,

$$\mathcal{P} \models P(C|D) \in [p, q]$$

iff in all probability measures satisfying \mathcal{P} the conditional probability $P(C|D)$ belongs to the interval $[p, q]$. Given \mathcal{P}, C, D , one is interested in finding the maximal p and minimal q such that $\mathcal{P} \models P(C|D) \in [p, q]$ is true.

Heinsohn [1994] introduces local inference rules that can be used to derive bounds for conditional probabilities, but these rules are not complete, that is, in general they are not sufficient to derive the optimal bounds.

Jaeger [1994] only describes a naive method for computing optimal bounds. A more sophisticated version of that method reduces the inference problem to a linear optimization problem. In the following, we will sketch the main idea underlying this reduction. Assume that C_1, \dots, C_m are the concept descriptions occurring in \mathcal{P} and $P(C|D)$, and consider all conjunctions $D_1 \sqcap \dots \sqcap D_m$, where D_i is either C_i or $\neg C_i$. Let \mathfrak{A} be the set of those conjunctions that are satisfiable. Given a probability measure on all concept descriptions, the values of this measure on C_1, \dots, C_m is uniquely determined by the values on \mathfrak{A} . To be more precise, its value for C_i can

¹ Actually, Heinsohn uses a different notation and allows for more expressive axioms stating that $P(C|D)$ belongs to an interval $[p_l, p_u]$, where $0 \leq p_l \leq p_u \leq 1$.

be obtained as the sum of the values for those elements of \mathfrak{A} that are subsumed by C_i (i.e., the ones where C_i occurs positively). The idea is to introduce a numerical variable x_t (ranging over the real interval $(0,1)$) for each element $t \in \mathfrak{A}$. For example, if C_1, C_2 are two concept names, then \mathfrak{A} consists of the four elements $t_0 = \neg C_1 \sqcap \neg C_2$, $t_1 = \neg C_1 \sqcap C_2$, $t_2 = C_1 \sqcap \neg C_2$, and $t_3 = C_1 \sqcap C_2$, for which we introduce the variables x_0, x_1, x_2, x_3 , respectively. Thus, the probability associated with $C_1 \sqcap C_2$ is x_3 and the one for C_2 is $x_1 + x_3$. Consequently, the probabilistic terminological axiom $P(C_1|C_2) = 0.7$ can be represented by the (linear) constraint $x_3 = 0.7(x_1 + x_3)$.

We have to find the maximal and minimal values that $P(C|D)$ attains on the set of values (x_0, \dots, x_n) satisfying the linear constraints induced by \mathcal{P} . The value of the function $P(C|D)$ (in terms of the variables x_t) is given by

$$\frac{\sum\{x_t \mid t \in \mathfrak{A} \wedge t \sqsubseteq C \sqcap D\}}{\sum\{x_t \mid t \in \mathfrak{A} \wedge t \sqsubseteq D\}}.$$

By a simple transformation, this *fractional optimization problem* can be transformed into a linear optimization problem [Amarger *et al.*, 1991].

Jaeger [1994] also extends the assertional formalism by allowing for *probabilistic assertions* of the form

$$P(C(a)) = p,$$

where C is a concept description, a an individual name, and p a real number between 0 and 1. It should be noted that this kind of probabilistic statement is quite different from the one introduced by the terminological formalism. Whereas probabilistic terminological axioms state *statistical information*, which is usually obtained by observing a large number of objects, probabilistic assertions express a *degree of belief* in assertions for specific individuals. The formal semantics of probabilistic assertions is again defined with the help of probability measures on the set of all concept descriptions, one for each individual name. Intuitively, the measure for a tells us for each concept C how likely it is (believed to be) that a belongs to C .

Given a knowledge base \mathcal{P} consisting of probabilistic terminological axioms and assertions, the main *inference task* is now to derive optimal bounds for additional probabilistic assertions. However, if the probabilistic terminological axioms are supposed to have an impact on this inference problem, the semantics as sketched until now is not sufficient. In fact, until now there is no connection between the probability measure used for the terminological part and the measures for the assertional part. Intuitively, one wants that the measures for the assertional part “most closely resemble” the measure for the terminological part, while not violating the probabilistic assertions. Jaeger [1994] uses *cross entropy minimization* in order to give a formal meaning to this intuition. Until now, there is no algorithm for comput-

ing optimal bounds for $P(C(a))$, given a knowledge base consisting of probabilistic terminological axioms and assertions.

The work reported in [Koller *et al.*, 1997], which is restricted to the terminological component, has a focus that is quite different from the one in [Heinsohn, 1994; Jaeger, 1994]. In the latter work, the probabilistic terminological axioms provide constraints on the set of admissible probability measures. However, these constraints may still be satisfied by a large set of distributions, and hence the optimal interval entailed for the probabilities of interest can be fairly large. In contrast, Koller *et al.* [1997] present a framework for the specification of a unique probability distribution on the set of all concept descriptions (modulo equivalence). Since there are infinitely many such descriptions, providing such a (finite) specification is a nontrivial task. The basic idea is to specify a distribution on concepts of role-depth 0, and then to specify how to extend a distribution on concepts of role-depth n to one on concepts of role-depth $n + 1$. Koller *et al.* [1997] employ Bayesian networks as the basic representation language for the required probabilistic specifications. The probability $P(C)$ of a concept description C can then be computed by using inference algorithms developed for Bayesian networks. The complexity of this computation is linear in the length of C . Under certain restrictions on the Bayesian networks used in the specification, it is polynomial in the size of that specification.

Yelland [2000] also combines Bayesian networks and Description Logics. In contrast to [Koller *et al.*, 1997], this work extends Bayesian networks by Description Logic features rather than the other way round. The Description Logic used in [Yelland, 2000] is rather inexpressive, but this allows the author to avoid restrictions on the network that had to be imposed by Koller *et al.* [1997].

6.2.5.2 Fuzzy extensions

The concepts in Description Logics are interpreted as crisp sets, i.e., an individual either belongs to the set or not. However, many “real-life” concepts are vague in the sense that they do not have precisely defined membership criteria. Consider, for example, the concept of a tall person. It does not make sense to fix an exact boundary such that persons of height larger than this boundary are tall and others are not. In fact, what about a person whose height is 1 millimeter below the boundary? It is more sensible to say that an individual belongs to the concept “tall person” only to a certain degree $n \in [0, 1]$, which depends on the height of the individual. This is exactly what fuzzy logic allows one to do.

The main idea underlying the fuzzy extensions of Description Logics proposed in [Yen, 1991; Tresp and Molitor, 1998; Straccia, 1998; 2001] is to leave the syntax as it is, but to use fuzzy logic for defining the semantics. Thus, an interpretation now assigns fuzzy sets to concepts and roles, i.e., concept names A are in-

terpreted by membership degree functions of the form $A^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow [0, 1]$, and role names R by membership degree functions of the form $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \rightarrow [0, 1]$. The interpretation of the Boolean operators and the quantifiers must then be extended from $\{0, 1\}$ to the interval $[0, 1]$. Fuzzy logics provides different options for such an extension. In [Yen, 1991; Tresp and Molitor, 1998; Straccia, 1998; 2001], the usual interpretation of conjunction as minimum, disjunction as maximum, negation as $\lambda x.(1 - x)$, universal quantifier as infimum, and existential quantifier as supremum is considered. For example,

$$(\forall R.C)^{\mathcal{I}}(d) = \inf\{\max\{1 - R^{\mathcal{I}}(d, e), C^{\mathcal{I}}(d, e)\} \mid e \in \Delta^{\mathcal{I}}\},$$

since $\forall R.C$ corresponds to the formula $\forall x.(\neg R(x, y) \vee C(y))$.

Tresp and Molitor [1998] also propose an extension of the syntax by so-called manipulators, which are unary operators that can be applied to concepts. Examples of manipulators could be “mostly”, “more or less”, or “very”. For example, if Tall is a concept (standing for the fuzzy set of all tall persons), then VeryTall, which is obtained by applying the manipulator Very to the concept Tall, is a new concept (standing for the fuzzy set of all very tall persons). Intuitively, the manipulators modify the membership degree functions of the concepts they are applied to appropriately. In our example, the membership function for VeryTall should have its largest values at larger heights than the membership function for Tall. Formally, the semantics of a manipulators is defined by a function that maps membership degree functions to membership degree functions. The manipulators considered in [Tresp and Molitor, 1998] are, however, of a very restricted form.

Lets us now consider what kind of inference problems are of interest in this context. Yen [1991] considers crisp subsumption of fuzzy concepts, i.e., given two concepts C, D defined in the fuzzy DL, he is interested in the question whether $C^{\mathcal{I}}(d) \leq D^{\mathcal{I}}(d)$ for all fuzzy interpretations \mathcal{I} and $d \in \Delta^{\mathcal{I}}$. Thus, the subsumption relationship itself is not fuzzified. He describes a structural subsumption algorithm for a rather small fuzzy DL, which is almost identical to the subsumption algorithm for the corresponding classical DL. In contrast, Tresp and Molitor [1998] are interested in determining fuzzy subsumption between fuzzy concepts, i.e., given concepts C, D , they want to know to which degree C is a subset of D . In [Straccia, 1998; 2001] and [Molitor and Tresp, 2000], also ABoxes are considered, where the ABox assertions are equipped with a degree. In this context one wants to find out to which degree other assertions follow from the ABox.

Both [Straccia, 1998; 2001] and [Tresp and Molitor, 1998] contain complete algorithms for solving these inference problems in the respective fuzzy extension of \mathcal{ALC} . Although both algorithms are extensions of the usual tableau-based algorithm for \mathcal{ALC} , they differ considerably. For example, the algorithm in [Tresp and Molitor, 1998] introduces numerical variables for the degrees, and produces a lin-

ear optimization problem, which must be solved in place of the usual clash test. In contrast, Straccia deals with the membership degrees within his tableau-based algorithm.

6.2.6 Extensions by default rules

In Description Logics, inclusion axioms of the form $C \sqsubseteq D$ are interpreted as universal statement, i.e., *all* instances of C also belong to D . The same is true for inferred subsumption relationships. In commonsense reasoning, however, one often wants to state and infer relationships that are only “normally” true, but may have exceptions. The most prominent example from the non-monotonic reasoning community is the statement that all birds fly; but of course penguins and other non-flying birds are exceptions. Allowing for such *default* statements has a strong impact both on the semantics and the reasoning capabilities of Description Logics. Instead of basing the semantics on classical first-order logic, one must employ a non-monotonic logic [Ginsberg, 1987]. In fact, conclusions drawn from a given knowledge base with defaults may ultimately turn out to be false when additional knowledge is added, and thus must be withdrawn.

Since most of the classical Description Logics can be seen as fragments of first-order predicate logic, an obvious approach for extending DLs by non-monotonic reasoning capabilities is to take one of the well-known non-monotonic logics, and restrict the first-order version of this logic to the DL in question. This approach was employed in [Baader and Hollunder, 1995a], where Reiter’s default logic [Reiter, 1980] is integrated into DLs. In addition to terminological axioms in the TBox and assertions in the ABox, Baader and Hollunder allow for *terminological defaults* of the form

$$\frac{C(x) : D(x)}{E(x)},$$

where C, D, E are concept descriptions (viewed as first-order formulae with one free variable x). Intuitively, such a default rule can be applied to an ABox individual a , i.e., $E(a)$ is added to the current set of beliefs, if its prerequisite $C(a)$ is already believed for this individual and its justification $D(a)$ is consistent with the set of beliefs. Formally, the consequences of a *terminological default theory* (consisting of a TBox, ABox, and a set of terminological defaults) are defined with reference to the notion of an *extension*, which is a set of deductively closed first-order formulae defined by a fixpoint construction (see [Reiter, 1980], p.89). In general, a default theory may have more than one extension, or even no extension. Depending on whether one wants to employ *skeptical* or *credulous* reasoning, an assertion $F(a)$ is a *consequence of a default theory* iff it is in all extensions or if it is in at least one extension of the theory.

It should be noted that in this setting the application of default rules is restricted to individuals explicitly present in the ABox.¹ For example, assume that the ABox consists of the fact that Tom has a child that is a doctor, i.e., $\mathcal{A} = \{(\exists\text{has-child.Doctor})(\text{TOM})\}$, and that by default we assume that doctors are usually rich:

$$\frac{\text{Doctor}(x) : \text{Rich}(x)}{\text{Rich}(x)}.$$

Intuitively, one might expect that $(\exists\text{has-child.Rich})(\text{TOM})$ is a default consequence of this terminological default theory. However, since the ABox does not contain a name for Tom’s child, the default cannot be applied to this “implicit” individual, and thus one cannot conclude that Tom has a rich child by default. Baader and Hollunder [1995a] give two reasons that justify restricting the application of defaults to explicit individuals. From a semantic point of view, adapting Reiter’s treatment of implicit individuals via Skolemization is quite unsatisfactory, since semantically equivalent (but syntactically different) ABoxes may lead to different default consequences. From the algorithmic point of view, the application of defaults to implicit individuals is problematic since it may lead to an undecidable default consequence relation, even though the employed DL is decidable. In contrast, the restriction of default application to explicit individuals ensures that reasoning in terminological default theories stays decidable whenever reasoning in the underlying DL is decidable.

A major drawback, which terminological default logic inherits from general default logic, is that it does not take precedence of more specific defaults over more general ones into account. For example, assume that we have a default that says that doctors are usually rich, and another one that says that general practitioners are usually not rich, and that classification shows that general practitioners are doctors. Intuitively, for any general practitioner the more specific second default should be preferred, which means that there should be only one default extension in which the general practitioner is not rich. However, in default logic the second default has no priority over the first one, which means that one also gets a second extension where the general practitioner is rich. This behaviour has already been criticized in the general context of default logic, but it is all the more problematic in the terminological case where the emphasis lies on the hierarchical organization of concepts. To overcome this problem, Baader and Hollunder [1995b] first define a prioritized version of Reiter’s default logic, where priorities are given by an arbitrary partial order on defaults. In the terminological case, the priority is induced by the subsumption relationship between prerequisites of defaults. A similar approach is

¹ This agrees with the semantics given to (monotonic) rules in DLs (see Subsection 6.2.3 and Chapter 2, Subsection 2.2.5).

proposed in [Straccia, 1993], with the main difference that in that paper the defaults also influence the priority order. In addition, Straccia also allows for defaults of the form

$$\frac{A(x) \wedge r(x, y) : C(y)}{C(y)},$$

where A is an atomic concept, r a role name, and C a concept description. Such a default can, for example, be used to say that usually a child of a doctor is again a doctor.

A quite different proposal for how to treat defaults in Description Logics can be found in [Quantz and Royer, 1992]. There, preference semantics [Shoham, 1987] is employed to define the semantics of default assertions $C \rightsquigarrow D$, which are intuitively interpreted as saying: “whenever an object is an instance of C , it is also an instance of D , unless this is in conflict with other knowledge”. Though on this intuitive level the meaning of the default $C \rightsquigarrow D$ coincides with that of the terminological default $C(x) : D(x)/D(x)$, the formal semantics (and thus also the default consequences) differ significantly. The semantics proposed by Quantz and Royer is based on a preference relation on models, which tries to minimize the exceptions to defaults while maximizing the number of defaults that have been fired. In contrast to the work mentioned above, Quantz and Royer restrict reasoning with defaults not only to the derivation of concept assertions of the form $C(a)$. They also consider *default subsumption* between concepts. However, default subsumption is reduced to reasoning about individuals. The subsumption relationship $C \sqsubseteq D$ follows by default from the knowledge base iff the knowledge base extended by $C(a)$ implies $D(a)$ by default, where a is a new individual name. Designing reasoning methods for such a model-based approach to non-monotonic reasoning is rather hard. Quantz and Royer only provide some ideas for how to obtain a sound but incomplete procedure.

Default subsumption is also considered in [Padgham and Zhang, 1993], where non-monotonic inheritance networks [Horty *et al.*, 1987] are extended in the direction of DLs, though the DL employed is of a very limited expressive power.

6.3 Non-standard inference problems

All DL systems provide their users with standard inference services like computing the subsumption hierarchy and testing ABox consistency. In some applications it has turned out, however, that these services are not quite sufficient for providing an optimal support when building and maintaining large DL knowledge bases. For this reason, some DL systems (e.g., CLASSIC) provide their users with additional system services, which can formally be reconstructed as new types of inference problems.

First, the standard inferences can be applied *after* a new concept has been defined to find out whether the concept is non-contradictory or whether its place in the

taxonomy coincides with the intuition of the knowledge engineer; however, these inferences do not directly support the process of actually defining the new concept. To overcome this problem, the non-standard inference services of computing the *least common subsumer* and the *most specific concept* have been proposed.

Second, if a knowledge base is maintained by different knowledge engineers, one needs support for detecting multiple definitions of the same intuitive concept. Since different knowledge engineers might use different names for the “same” primitive concept, the standard equivalence test may not be adequate to check whether different descriptions refer to the same notion. The non-standard inference service *unification of concept descriptions* tackles this problem by allowing to replace concept names by appropriate concept descriptions before testing for equivalence. *Matching* is a special case of unification, which has, for example, been used for pruning irrelevant parts of large concept descriptions before displaying them to the user.

Third, and very abstractly speaking, *rewriting* of concept descriptions allows one to transform a given concept description C into a “better” description D , which satisfies certain optimality criteria (e.g., small size) and is in a certain relationship (e.g., equivalence or subsumption) with the original description C .

Before describing the different non-standard inferences in more detail, we start with some general remarks on how these new problems have until now been tackled in the literature. An overview of the state of the art in this field and detailed proofs of several of the results mentioned below can be found in [Küsters, 2001].

6.3.1 Techniques for solving non-standard inferences—a general remark

Approaches for solving the new inference problems are usually based on an appropriate characterization of subsumption, which can be used to obtain a structural subsumption algorithm. First, the concept descriptions are turned into a certain normal form, in which implicit facts have been made explicit. Second, the structure of the normal forms is compared appropriately. This is one of the reasons why most of the results on non-standard inferences are restricted to languages that can be treated by structural subsumption algorithms.

One can distinguish two kinds of normal forms proposed in the literature. In one approach, called *language-based* approach in the sequel, the normal form of a concept description is given in terms of certain finite or regular sets of words over the alphabet of all role names. Then, subsumption can be characterized via the inclusion of these sets (see Chapter 2, Section 2.3.3.2). The second approach, called *graph-based* in the following, turns concept descriptions into so-called description graphs. Here, subsumption of concept descriptions is characterized via the existence of certain homomorphisms between the corresponding description graphs.

The structural subsumption algorithm introduced in Chapter 2, Subsection 2.3.1, can be represented in this way (although this was not explicitly done in Chapter 2).

For the sublanguage \mathcal{ALN} of CLASSIC, the graph-based approach can be seen as special implementation of the language-based approach [Baader *et al.*, 1998a]. In general, however, either the language-based or the graph-based approach may turn out to be more appropriate, depending on the DL under consideration. On the one hand, the language-based approach is particularly useful for characterizing subsumption between cyclic concept descriptions, i.e., descriptions defined by means of cyclic terminologies in \mathcal{FL}_0 and \mathcal{ALN} [Baader, 1996b; Küsters, 1998]. On the other hand, the graph-based approach can be employed to handle full CLASSIC [Borgida and Patel-Schneider, 1994] as well as $\mathcal{AL\mathcal{E}}$ [Baader *et al.*, 1999b], which extends \mathcal{FL}_0 by primitive negation and existential restrictions. Although Borgida and Patel-Schneider did not explicitly characterize subsumption in terms of homomorphisms between description graphs, their subsumption algorithm does in fact check for the existence of an appropriate homomorphism.

The known approaches for solving non-standard inference problems are usually based on one of the two approaches for characterizing subsumption, depending on the DL of choice. In the sequel, we will give an idea of how to solve the inference problems by mainly looking at the language-based approach for the DL \mathcal{FL}_0 . We will also briefly comment on how to treat extensions of \mathcal{FL}_0 .

6.3.2 Least common subsumer and most specific concept

Intuitively, the least common subsumer of a given collection of concept descriptions is a description that represents the properties that all the elements of the collection have in common. More formally, it is the most specific concept description that subsumes the given descriptions:

Definition 6.13 Let \mathcal{L} be a description language. A concept description E of \mathcal{L} is the *least common subsumer* (lcs) of the concept descriptions C_1, \dots, C_n in \mathcal{L} ($\text{lcs}(C_1, \dots, C_n)$ for short) iff it satisfies

- (i) $C_i \sqsubseteq E$ for all $i = 1, \dots, n$, and
- (ii) E is the least \mathcal{L} -concept description satisfying (i), i.e., if E' is an \mathcal{L} -concept description satisfying $C_i \sqsubseteq E'$ for all $i = 1, \dots, n$, then $E \sqsubseteq E'$. ■

As an easy consequence of this definition, the lcs is unique up to equivalence. In fact, if E_1 and E_2 are both least common subsumers of the same collection of concepts, then $E_1 \sqsubseteq E_2$ (since E_2 satisfies (i) and E_1 is the least concept description satisfying (i)). The subsumption relationship $E_2 \sqsubseteq E_1$ can be derived analogously. It should be noted, however, that the lcs need not always exist. This can have

two different reasons: (a) there may be several subsumption incomparable minimal concept descriptions satisfying (i) of the definition; (b) there may be an infinite chain of more and more specific descriptions satisfying (i). It is easy to see, however, that for DLs allowing for conjunction of descriptions (a) cannot occur.

The lcs has first been introduced by Cohen *et al.* [1992] as a new inference task that is useful for a number of different reasons. First, finding the most specific concept that generalizes a set of examples is a common operation in inductive learning, called learning from examples. Cohen and Hirsh [1994a] as well as Frazier and Pitt [1994] investigate the learnability of sublanguages of CLASSIC with regard to the PAC learning model proposed by Valiant [1984]. The lcs-computation is used as a subprocedure in their learning algorithms. Experimental results concerning the learnability of concepts based on computing the lcs can be found in [Cohen and Hirsh, 1994b].

Another motivation for considering the lcs is to use it as an alternative to disjunction. The idea is to replace disjunctions like $C_1 \sqcup \dots \sqcup C_n$ by the lcs of C_1, \dots, C_n . In [Cohen *et al.*, 1992; Borgida and Etherington, 1989], this operation is called *knowledge base vivification*. Although, in general, the lcs is not equivalent to the corresponding disjunction, it is the best approximation of the disjunctive concept within the available DL. Using such an approximation is motivated by the fact that, in many cases, adding disjunction would increase the complexity of reasoning. Observe that, if the DL already allows for disjunction, we have $\text{lcs}(C_1, \dots, C_n) \equiv C_1 \sqcup \dots \sqcup C_n$. In particular, this means that, for such DLs, the lcs is not really of interest.

Finally, as proposed in [Baader and Küsters, 1998; Baader *et al.*, 1999b], the lcs operation can be used to support the “bottom-up” construction of DL knowledge bases. In contrast to the usual “top-down” approach, where the knowledge engineers first define the terminology of the application domain in the TBox and then uses this terminology when describing individuals in the ABox, the “bottom-up” approach proceeds as follows. The knowledge engineer first specifies some “typical” examples of a concept to be defined using individuals in the ABox. Then, in a second step, these individuals are generalized to their most specific concept, i.e., a concept description that (i) has all the individuals as instances, and (ii) is the most specific description satisfying property (i). Finally, the knowledge engineers inspects and possibly modifies the concept description obtained this way.

Let us now define the most specific concept of an ABox individual in more detail.

Definition 6.14 A concept description E in some description language \mathcal{L} is the *most specific concept* (msc) of the individuals a_1, \dots, a_n defined in an ABox \mathcal{A} ($\text{msc}(a_1, \dots, a_n)$ for short) iff

- (i) $\mathcal{A} \models E(a_i)$ for all $i = 1, \dots, n$, and

- (ii) E is the least concept satisfying (i), i.e., if E' is an \mathcal{L} -concept description satisfying $\mathcal{A} \models E'(a_i)$ for all $i = 1, \dots, n$, then $E \sqsubseteq E'$. ■

The task of computing the msc can be split into two subtasks: computing the most specific concept of a single individual, and computing the least common subsumer of a given finite number of concepts. In fact, it is easy to see that $msc(a_1, \dots, a_n) \equiv lcs(msc(a_1), \dots, msc(a_n))$.

6.3.2.1 Computing the lcs and the msc

We will now give an intuition on how to compute the lcs for the DL \mathcal{FL}_0 and an extension, and briefly comment on the problems that arise when considering the msc . As mentioned above, the first step towards an algorithm for computing the lcs is to characterize subsumption of concept descriptions. For the DL \mathcal{FL}_0 , we will present such a characterization using the language-based approach.

The normal form of \mathcal{FL}_0 -concept descriptions employed in the language-based approach is the so-called *concept-centered normal form* (CCNF), which has already been introduced in Chapter 2, Section 2.3.3.2. For example, using the equivalence $\forall R.(C \sqcap D) \equiv \forall R.C \sqcap \forall R.D$ as well as commutativity of concept conjunction, the \mathcal{FL}_0 -concept description $C = \forall R.(\forall S.A \sqcap \forall R.B) \sqcap \forall S.\forall S.A$ can be transformed into CCNF as follows:

$$\begin{aligned} C &\equiv \forall R.\forall S.A \sqcap \forall S.\forall S.A \sqcap \forall R.\forall R.B \\ &\equiv \forall\{RS, SS\}.A \sqcap \forall\{RR\}.B. \end{aligned}$$

Recall that $\forall\{RS, SS\}.A$ has been introduced in Chapter 2, Subsection 2.3.3.2 as an abbreviation for $\forall R.\forall S.A \sqcap \forall S.\forall S.A$. Similarly, $\forall\{RR\}.B$ abbreviates $\forall R.\forall R.B$.

In general, if N_C is a finite set of atomic concepts and N_R is a finite set of role names, then the CCNF of a concept C built using only these names is of the form

$$C \equiv \prod_{A \in N_C} \forall U_A.A,$$

where U_A is a finite set of words over the alphabet of role names, i.e., $U_A \subseteq N_R^*$. Note that $\forall \emptyset.A$ represents the universal concept \top , and $\forall\{\varepsilon\}.A$ for the empty word ε is equivalent to A .

If the CCNF of D is $\prod_{A \in N_C} \forall V_A.A$, then subsumption of C by D can be characterized as follows:

Proposition 6.15 $C \sqsubseteq D$ iff $V_A \subseteq U_A$ for all $A \in N_C$.

As an easy consequence, we obtain

Corollary 6.16 $lcs(C, D) \equiv \prod_{A \in N_C} \forall(U_A \cap V_A).A$.

By Proposition 6.15, this concept description obviously subsumes C and D . Moreover, $U_A \cap V_A$ is the largest set contained in both U_A and V_A , and thus $\prod_{A \in N_C} \forall(U_A \cap V_A).A$ is in fact the least concept subsuming both C and D .

As an example consider the concept C specified above and $D \equiv \forall\{RS, RR\}.A \sqcap \forall\{RR, SR\}.B$. Then, $\text{lcs}(C, D) \equiv \forall\{RS\}.A \sqcap \forall\{RR\}.B$.

For DLs extending \mathcal{FL}_0 by constructs that can express unsatisfiable concepts, like \perp , the language-based approach can still be applied. However, in order to characterize subsumption, we need to consider certain infinite regular languages instead of finite ones. The reason is that \perp is subsumed by an infinite number of concept descriptions. For example, although $\forall\{R, RSR\}.\perp \sqsubseteq \forall\{RR\}.\perp$, we do *not* have $V_\perp = \{RR\} \subseteq \{R, RSR\} =: U_\perp$. However, we know that $\forall\{R\}.\perp$ is subsumed by $\forall\{Rw\}.\perp$ for any word w of the alphabet N_R . Consequently, we must use $U_\perp \cdot N_R^* = \{vw \mid v \in U_\perp \text{ and } w \in N_R^*\}$ in place of U_\perp in the inclusion test. For this reason, the lcs must also be described in terms of possibly infinite regular languages. As a simple example, consider the concept descriptions $C \equiv \forall\{R, SR\}.\perp$ and $D \equiv \forall\{RS, S\}.\perp$. Then,

$$\begin{aligned} \text{lcs}(C, D) &\equiv \forall(\{R, SR\} \cdot N_R^* \cap \{RS, S\} \cdot N_R^*).\perp \\ &\equiv \forall(\{RS, SR\} \cdot N_R^*).\perp \\ &\equiv \forall\{RS, SR\}.\perp \end{aligned}$$

A detailed description of how to compute the lcs in \mathcal{ALN} , which extends \mathcal{FL}_0 by \perp , atomic complement, and number restrictions, is given in [Baader and Küsters, 1998]. Moreover, Baader and Küsters investigate cyclic \mathcal{ALN} -concept descriptions, which are defined in terms of cyclic terminologies with greatest fixpoint semantics. In this context, the languages U_A introduced above can be arbitrary regular languages (see also Chapter 2, Section 2.3.3.2).

Cyclic descriptions become necessary if one wants to guarantee the existence of the msc. Consider, for example, the ABox consisting only of the assertion $R(a, a)$. Then, we know that $\text{msc}(a) \sqsubseteq \forall R. \dots \forall R. (\leq 1 R)$ for arbitrarily deep nesting of value restrictions. Baader and Küsters show that there does not exist an acyclic \mathcal{ALN} -concept description presenting the msc of a . However, the msc of individuals described in \mathcal{ALN} -ABoxes can always be represented by a cyclic \mathcal{ALN} -concept description. In our example, $\text{msc}(a)$ can be represented by the concept A defined by $A \equiv (= 1 R) \sqcap \forall R.A$, if this definition is interpreted with greatest fixpoint semantics.

Using the graph-based approach, the lcs can be computed for the DL that extends \mathcal{FL}_0 by the same-as construct [Cohen and Hirsh, 1994a; Frazier and Pitt, 1994; Küsters and Borgida, 2001], for the language \mathcal{ALE} , which extends \mathcal{FL}_0 by full existential quantification as well as primitive negation [Baader *et al.*, 1999b], and for the language \mathcal{ALEN} , which extends \mathcal{ALE} by number restrictions [Küsters and Molitor,

2001b]. On the one hand, it is not clear how to handle these languages with the language-based approach. On the other hand, up to now the graph-based approach cannot deal with cyclic concept descriptions, which are needed for computing the msc. Consequently, for the extensions of \mathcal{FL}_0 treated with the help of the graph-based approach, the msc can currently only be approximated [Cohen and Hirsh, 1994b; Küsters and Molitor, 2001a].

6.3.3 Unification and matching

Unification and matching are non-standard inferences that allow us to replace certain concept names by concept descriptions before testing for equivalence or subsumption. This capability turns out to be useful when maintaining (large) knowledge bases. In this subsection, we will first introduce unification and matching and mention the main motivations for considering these new inference tasks. We will then review the results available in the literature, and give an intuition on how unification problems in the small language \mathcal{FL}_0 can be solved.

6.3.3.1 Unification

Unification of concepts has first been introduced by Baader and Narendran [1998], motivated by the following application problem. If several knowledge engineers are involved in defining new concepts, and if this knowledge acquisition process takes rather long (several years), it happens that the same (intuitive) concept is introduced several times, often with slightly differing descriptions. Testing for equivalence of concepts is not always sufficient to find out whether, for a given concept description, there already exists another concept description in the knowledge base describing the same notion. As an example, let us ask whether the following two \mathcal{FL}_0 -concept descriptions might denote the same (intuitive) concept?

$$\begin{aligned} & \forall \text{has-child}.\forall \text{has-child}.\text{Rich} \sqcap \forall \text{has-child}.\text{Rmr}, \\ & \text{Acr} \sqcap \forall \text{has-child}.\text{Acr} \sqcap \forall \text{has-child}.\forall \text{has-spouse}.\text{Rich}. \end{aligned}$$

The answer is yes, since replacing the concept name Rmr by the description $\text{Rich} \sqcap \forall \text{has-spouse}.\text{Rich}$ and Acr by $\forall \text{has-child}.\text{Rich}$ yields the descriptions

$$\begin{aligned} & \forall \text{has-child}.\forall \text{has-child}.\text{Rich} \sqcap \forall \text{has-child}.\text{(Rich} \sqcap \forall \text{has-spouse}.\text{Rich)}, \\ & \forall \text{has-child}.\text{Rich} \sqcap \forall \text{has-child}.\forall \text{has-child}.\text{Rich} \sqcap \forall \text{has-child}.\forall \text{has-spouse}.\text{Rich}, \end{aligned}$$

which are obviously equivalent. Thus, under the assumption that Rmr stands for “Rich and married rich” and Acr for “All children are rich”, we can conclude that both descriptions are meant to express the concept “All grandchildren are rich and all children are rich and married rich”.

A substitution of concept descriptions for concept names that makes two concept

descriptions C, D equivalent is called a unifier of C and D . Of course, before testing for unifiability, one must decide which of the concept names the unifier is allowed to replace. These names are then called *concept variables* to distinguish them from the usual concept names, which cannot be replaced. In the above example, the strange acronyms Acr and Rmr were considered to be variables, whereas Rich was treated as a (non-replaceable) concept name. Concept descriptions containing variables are called concept patterns. More precisely, \mathcal{FL}_0 -*concept patterns* are defined by means of the following syntax rules:

$$C, D \longrightarrow X \mid A \mid \forall R.C \mid C \sqcap D$$

where X stands for concept variables.

Now, a *substitution* in \mathcal{FL}_0 is a mapping from the concept variables into the set of \mathcal{FL}_0 -concept descriptions. An example is the substitution $\{\text{Rmr} \mapsto \text{Rich} \sqcap \forall \text{has-spouse.Rich}, \text{Acr} \mapsto \forall \text{has-child.Rich}\}$ used in our example. The application of a substitution can be extended from variables to \mathcal{FL}_0 -concept patterns in the usual way (as exemplified above).

Definition 6.17 Let C, D be \mathcal{FL}_0 -concept patterns. Then, a substitution σ is a *unifier* of the unification problem $C \equiv^? D$ iff $\sigma(C) \equiv \sigma(D)$. ■

Of course, it is not necessarily the case that concept descriptions that are unifiable in this way are really meant to represent the same notion. A unifiability test can, however, suggest to the knowledge engineer possible candidate descriptions.

6.3.3.2 Matching

Matching can be seen as a special case of unification, where one of the two expressions to be unified do not contain variables [Baader and Narendran, 1998; 2001]. Thus, a matching problem is of the form $C \equiv^? D$ where C is a concept description and D a concept pattern. A substitution σ is a *matcher* of this problem iff $C \equiv \sigma(D)$.

Borgida and McGuinness [1996] have introduced a different notion of matching, which we call *matching modulo subsumption* to distinguish it from *matching modulo equivalence*, as introduced above. A matching problem modulo subsumption is of the form $C \sqsubseteq^? D$, where C is a concept description and D is a concept pattern. Such a problem asks for a substitution σ such that $C \sqsubseteq \sigma(D)$.

Since σ is a solution of $C \sqsubseteq^? D$ iff σ solves $C \equiv^? C \sqcap D$, matching modulo subsumption can be reduced to matching modulo equivalence, and thus to unification. However, in the context of matching modulo subsumption, one is interested in finding “minimal” solutions of $C \sqsubseteq^? D$, i.e., σ should satisfy the property that there does not exist another substitution δ such that $C \sqsubseteq \delta(D) \sqsubset \sigma(D)$. In addition,

Baader *et al.* [1999a] introduce side conditions of the form $X \sqsubseteq E$ and $X \sqsubset E$, with X a variable and E a concept pattern, to further restrict possible substitutions for the variables occurring in the matching problem.

The original reason for introducing matching modulo equivalence was (i) to help filter out unimportant aspects of complicated concepts appearing in large knowledge bases, and (ii) to specify patterns for explaining proofs carried out by DL systems [McGuinness and Borgida, 1995]. For example, matching the concept pattern

$$D = \forall \text{research-interests}. X$$

against the description

$$C = \forall \text{pets}. \text{Cat} \sqcap \forall \text{research-interests}. \text{AI} \sqcap \forall \text{hobbies}. \text{Gardening}$$

yields the minimal matcher $\sigma = \{X \mapsto \text{AI}\}$, and thus finds the scientific interest described in the concept, filtering out the other aspects described by C .

Another motivation for matching as well as unification can be found in the area of integrating data or knowledge base schemata represented in some DL. An integrated schema can be viewed as the union of the local schemata along with some interschema assertions satisfying certain conditions. Finding such interschema assertions can be supported by solving matching or unification problems. Borgida and Küsters [2000] propose a formal framework for schema integration, and provide initial theoretical as well as experimental results concerning this application of unification and matching.

6.3.3.3 Results on matching and unification

As with computing the lcs, the algorithms for matching that can be found in the literature follow either the language-based or the graph-based approach. Matching modulo subsumption for a description language containing most of the constructs available in CLASSIC has been considered in [Borgida and McGuinness, 1996]. Borgida and McGuinness describe a polynomial-time matching algorithm, which follows the graph-based approach. However, this algorithm cannot be applied to arbitrary patterns, and it is not complete. Using the language-based approach, complete and polynomial-time algorithms for matching modulo equivalence and matching modulo subsumption in \mathcal{FL}_0 were presented in [Baader and Narendran, 1998; 2001]. This result was extended to the language \mathcal{ALN} by Baader *et al.* [1999a] and its extension \mathcal{ALN}_{reg} by the role constructors union, composition, and transitive closure by Küsters [2001]. Baader *et al.* [2001] consider matching under side conditions in more detail. Basically, subsumption conditions of the form $X \sqsubseteq E$ leave the complexity of matching in \mathcal{ALN} polynomial, whereas strict subsumption conditions $X \sqsubset E$ cause NP-hardness. Matching in \mathcal{ALE} based on the characterization of subsumption by homomorphism between graphs has been investigated in [Baader and

Küsters, 2000]. It is shown that matching modulo equivalence is NP-complete, and that appropriate matchers can be computed in exponential time. Finally, complete algorithms for matching in CLASSIC are provided by Küsters [2001].

For unification, the only results available until now are for the small DL \mathcal{FL}_0 and its extension \mathcal{FL}_{reg} by the role constructors union, composition, and transitive closure. In [Baader and Narendran, 1998; 2001] it is shown that deciding unifiability of \mathcal{FL}_0 -patterns is an EXPTIME-complete problem, and in [Baader and Küsters, 2001] this result is extended to \mathcal{FL}_{reg} . In the remainder of this subsection, we will try to give a flavor of how to solve unification problems in \mathcal{FL}_0 .

As an immediate consequence of Proposition 6.15, equivalence of \mathcal{FL}_0 -concept descriptions $C = \prod_{A \in N_C} \forall U_A.A$ and $D = \prod_{A \in N_C} \forall V_A.A$ in CCNF can be characterized as follows:

$$C \equiv D \quad \text{iff} \quad U_A = V_A \quad \text{for all } A \in N_C. \quad (6.3)$$

This fact can be used to turn \mathcal{FL}_0 -unification problems into certain formal language equations, which then can be solved using tree automata.

Let us illustrate this on the example from Subsection 6.3.3.1. There, we considered the unification problem¹

$$\forall\{cc\}.R \sqcap \forall\{c\}.X \equiv^? \forall\{\varepsilon, c\}.Y \sqcap \forall\{cs\}.R.$$

As an easy consequence of (6.3), a substitution σ of the form

$$\{X \mapsto \forall U_X.R, Y \mapsto \forall U_Y.R\},$$

where U_X, U_Y are sets of words over the alphabet $\{c, s\}$, is a unifier of this problem iff the assignment $X = U_X$ and $Y = U_Y$ solves the formal language equation

$$\{cc\} \cup \{c\} \cdot X = \{cs\} \cup \{\varepsilon, c\} \cdot Y.$$

For example, the unifier $\{X \mapsto R \sqcap \forall s.R, Y \mapsto \forall c.R\}$ corresponds to the solution $X = \{\varepsilon, s\}$, $Y = \{c\}$ of the above formal language equation. In general, unification problems correspond to systems of formal language equations of the form

$$S_0 \cup S_1 \cdot X_1 \cup \dots \cup S_n \cdot X_n = T_0 \cup T_1 \cdot X_1 \cup \dots \cup T_n \cdot X_n,$$

where the S_i, T_i are given finite sets of words and the X_i are variables ranging over finite sets of words. In [Baader and Narendran, 1998; 2001] it is shown that solvability of such a system of equations can be reduced (in exponential time) to the emptiness problem for automata on finite trees. This yields an EXPTIME-decision procedure for unification in \mathcal{FL}_0 . For unification in \mathcal{FL}_{reg} , the S_i, T_i are

¹ To increase readability, *has-spouse* is replaced by *s*, *has-child* by *c*, *Rich* by *R*, and *Rmr, Acr* by the variables *X, Y*. In addition, we have already transformed the patterns into their CCNF.

regular languages, and to test the equation for solvability one must employ automata working on infinite trees.

6.3.4 Concept rewriting

A general framework for rewriting concepts using terminologies has been proposed in Baader *et al.* [2000]. Assume that $\mathcal{L}_1, \mathcal{L}_2$, and \mathcal{L}_3 are three description languages, and let C be an \mathcal{L}_1 -concept description and \mathcal{T} an \mathcal{L}_2 -TBox. We are interested in rewriting (i.e., transforming) C into an \mathcal{L}_3 -concept description D such that C and D are in a certain relationship (e.g., equivalence, subsumption w.r.t. \mathcal{T}) and such that D satisfies certain optimality criteria (e.g., being of minimal size).

This very general framework has several interesting instances. In the following, we will discuss the three most promising ones.

The first instance is the *translation of concept descriptions* from one DL into another. Here, we assume that \mathcal{L}_1 and \mathcal{L}_3 are different description languages, and that the TBox \mathcal{T} is empty. By trying to rewrite an \mathcal{L}_1 -concept C into an *equivalent* \mathcal{L}_3 -concept D , one can find out whether C is expressible in \mathcal{L}_3 . In many cases, such an exact rewriting may not exist. In this case, one can try to approximate C by an \mathcal{L}_3 -concept from above (below), i.e., find a minimal (maximal) concept description D in \mathcal{L}_3 such $C \sqsubseteq D$ ($D \sqsubseteq C$). An inference service that can compute such rewritings could, for example, support the transfer of knowledge bases between different systems. First results in this direction for the case where \mathcal{L}_1 is \mathcal{ALC} and \mathcal{L}_3 is $\mathcal{AL}\mathcal{E}$ can be found in [Brandt *et al.*, 2001].

The second instance comes from the database area, where the problem of *rewriting queries using views* is a well-known research topic [Beeri *et al.*, 1997]. The aim is to optimize the runtime of queries by using cached views, which allows one to minimize the (more expensive) access to source relations. In the context of the above framework, views can be regarded as TBox definitions and queries as concept descriptions. Beeri *et al.* [1997] investigate the instance where $\mathcal{L}_1 = \mathcal{L}_2 = \mathcal{ALCN}\mathcal{R}$ and $\mathcal{L}_3 = \{\sqcap, \sqcup\}$. More precisely, they are interested in maximally contained total rewritings, i.e., D should be subsumed by C , contain only concept names defined in the TBox, and be a maximal concept (w.r.t. subsumption) satisfying these properties. They show that such a rewriting is computable (whenever it exists).

The third instance of the general framework, which was first proposed in [Baader and Molitor, 1999], tries to increase the readability of large concept descriptions by using concepts defined in a TBox. The motivation comes from the experiences made with non-standard inferences (like lcs, msc and matching) in applications. The concept descriptions produced by these services are usually unfolded (i.e., do not use defined names), and are thus often very large and hard to read and comprehend. Therefore, one is interested in automatically generat-

ing an equivalent concept description of minimal length that employs the concept names defined in the underlying terminology. Referring to the framework, one thus considers the case where $\mathcal{L} = \mathcal{L}_1 = \mathcal{L}_2 = \mathcal{L}_3$ and the TBox is non-empty. For a given concept description C and a TBox \mathcal{T} in \mathcal{L} one is interested in an \mathcal{L} -concept description D (containing concept names defined in \mathcal{T}) such that $C \equiv_{\mathcal{T}} D$ and the size of D is minimal. Rewriting in this sense has been investigated for the languages \mathcal{ALN} and $\mathcal{AL\mathcal{E}}$ [Baader and Molitor, 1999; Baader *et al.*, 2000]. Rewritings can be computed by a nondeterministic polynomial algorithm that uses an oracle for deciding subsumption. The corresponding decision problem (i.e., the question whether there exists a rewriting of size $\leq k$ for a given number k) is NP-hard for both languages.

Acknowledgement

We would like to thank Jochen Heinsohn and Manfred Jaeger for helpful discussions regarding the treatment of uncertain and vague knowledge and Riccardo Rosati regarding the treatment of epistemic operators.