

2. Description Logics and Standard Inferences

In order to define concepts in a DL knowledge base, one starts with a set N_C of concept names (unary predicates) and a set N_R of role names (binary predicates), and defines more complex *concept descriptions* using the *concept constructors* provided by the concept description language of the particular system. In this paper, we consider the DL \mathcal{ALCN} and some of its sublanguages. *Concept descriptions* of \mathcal{ALCN} are built using the constructors shown in the first part of Table 1. In this table, r stands for a role name, n for a nonnegative integer, A for a concept name, and C, D for arbitrary concept descriptions.

A *concept definition* $A \equiv C$ (as shown in the second part of Table 1) assigns a concept name A to a complex description C . A finite set of such definitions is called a *TBox* if and only if it is unambiguous, i.e., each name has at most one definition. The concept names occurring on the left-hand side of a concept definition are called *defined* concepts, and the others *primitive*. In many cases, one restricts the attention to *acyclic* TBoxes, where the definition of a defined concept A cannot (directly or indirectly) refer to A itself.

A (concept or role) *assertion* is of the form shown in the last part of Table 1. Here, a, b belong to an additional set N_I of individual names. A finite set of such assertions is called an *ABox*.

The *sublanguages* of \mathcal{ALCN} that will be considered in this paper are shown in Table 2. The first column explains the naming scheme for the members of the \mathcal{AL} -family.

The *semantics* of concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of \mathcal{I} is a non-empty set and the interpretation function $\cdot^{\mathcal{I}}$ maps each concept name $A \in N_C$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual name $a \in N_I$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined, as shown in the third column

Name	Syntax	Semantics
top-concept	\top	$\Delta^{\mathcal{I}}$
bottom-concept	\perp	\emptyset
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x, y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
at-least restriction	$\geq n r$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x, y) \in r^{\mathcal{I}}\} \geq n\}$
at-most restriction	$\leq n r$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x, y) \in r^{\mathcal{I}}\} \leq n\}$
concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$r(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

TABLE 1. Syntax and semantics of concept descriptions, definitions, and assertions

of Table 1. In the rows treating at-least and at-most number restrictions, $\#M$ denotes the cardinality of a set M .

The interpretation \mathcal{I} is a *model* of the TBox \mathcal{T} if it satisfies all its concept definitions, i.e., $A^{\mathcal{I}} = C^{\mathcal{I}}$ for all $A \equiv C$ in \mathcal{T} , and it is a *model* of the ABox \mathcal{A} if it satisfies all its assertions, i.e., $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all concept assertions $C(a)$ in \mathcal{A} and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ for all role assertions $r(a, b)$ in \mathcal{A} .

Based on this semantics, we can now formally introduce the standard inference problems in description logics.

Definition 2.1. Let \mathcal{A} be an ABox, \mathcal{T} a TBox, C, D concept descriptions, and a an individual name.

- C is *satisfiable* w.r.t. \mathcal{T} if there is a model \mathcal{I} of \mathcal{T} such that $C^{\mathcal{I}} \neq \emptyset$.
- D *subsumes* C w.r.t. \mathcal{T} ($C \sqsubseteq_{\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models \mathcal{I} of \mathcal{T} .

Symbol	Syntax	\mathcal{ALC}	$\mathcal{AL}\mathcal{EN}$	$\mathcal{AL}\mathcal{E}$	$\mathcal{AL}\mathcal{N}$	\mathcal{EL}	\mathcal{FL}_0
\mathcal{AL}	\top	x	x	x	x	x	x
	\perp	x	x	x	x		
	\sqcap	x	x	x	x	x	x
	$\neg A$	x	x	x	x		
	$\forall r.C$	x	x	x	x		x
\mathcal{C}	$\neg C$	x					
\mathcal{E}	$\exists r.C$	x	x	x		x	
\mathcal{U}	$C \sqcup D$	x					
\mathcal{N}	$(\leq nr), (\geq nr)$		x		x		

TABLE 2. The relevant sublanguages of \mathcal{ALCN}

- \mathcal{A} is *consistent* w.r.t. \mathcal{T} if there is a model \mathcal{I} of \mathcal{T} that is also a model of \mathcal{A} .
- a is an *instance* of C in \mathcal{A} w.r.t. \mathcal{T} ($\mathcal{A}, \mathcal{T} \models C(a)$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for all models \mathcal{I} of \mathcal{T} and \mathcal{A} .

In case the TBox \mathcal{T} is empty, we omit the appendage “w.r.t. \emptyset .” In particular, we say that D *subsumes* C and write this as $C \sqsubseteq D$. Two concept descriptions are *equivalent* ($C \equiv D$) if they subsume each other (w.r.t. the empty TBox), i.e., if $C \sqsubseteq D$ and $D \sqsubseteq C$. We write $C \sqsubset D$ to express that $C \sqsubseteq D$ but $D \not\sqsubseteq C$.

If the DL under consideration allows for full negation (\mathcal{C}), then subsumption and satisfiability are interreducible, and the same is true for the instance and the consistence problem. In addition, satisfiability (subsumption) can always be reduced to ABox-consistency (instance checking). This follows from the following equivalences:

- $C \sqsubseteq_{\mathcal{T}} D$ if and only if $C \sqcap \neg D$ is unsatisfiable w.r.t. \mathcal{T} ;
- C is unsatisfiable w.r.t. \mathcal{T} if and only if $C \sqsubseteq_{\mathcal{T}} \perp$;
- $\mathcal{A}, \mathcal{T} \models C(a)$ if and only if $\mathcal{A} \cup \{\neg C(a)\}$ is inconsistent w.r.t. \mathcal{T} ;

- \mathcal{A} is inconsistent w.r.t. \mathcal{T} if and only if $\mathcal{A}, \mathcal{T} \models \{\perp(a)\}$ where a is an arbitrary individual name;
- C is satisfiable w.r.t. \mathcal{T} if and only if $\{C(a)\}$ is consistent where a is an arbitrary individual name;
- $C \sqsubseteq_{\mathcal{T}} D$ if and only if $\{C(a)\}, \mathcal{T} \models D(a)$ where a is an arbitrary individual name.

If the TBox \mathcal{T} is acyclic, then reasoning w.r.t. \mathcal{T} can be reduced to reasoning w.r.t. the empty TBox by expanding concept definitions, i.e., by replacing defined concept by their definitions until all defined concepts have been replaced. This can, however, result in an exponential blow-up of the problem [82].

Most of the early research on reasoning in DLs concentrated on the subsumption problem for concept descriptions (i.e., w.r.t. the empty TBox). For the DLs introduced above, the worst-case complexity of this problem is well-investigated. Subsumption in \mathcal{ALCN} , \mathcal{ALC} , and \mathcal{ALEN} is PSPACE-complete, whereas subsumption in \mathcal{ALE} is NP-complete. The subsumption problem for \mathcal{ALN} , \mathcal{EL} , and \mathcal{FL}_0 is polynomial (see [49] for references and additional complexity results for other DLs).

In the presence of an acyclic TBox, the complexity of subsumption may increase, but not in all cases. For example, subsumption w.r.t. an acyclic TBox in \mathcal{FL}_0 is coNP-complete [82], but it remains polynomial in \mathcal{EL} [8] and PSPACE-complete in \mathcal{ALCN} [75]. Cyclic TBoxes may increase the complexity of the subsumption problem even further (for example, for \mathcal{FL}_0 to PSPACE [4, 68]), but again not in all cases (for example, for \mathcal{EL} , subsumption w.r.t. cyclic TBoxes remains polynomial [8]).

In most cases, the complexity of the instance problem is the same as the complexity of the subsumption problem (for example, in \mathcal{ALCN} [57] and \mathcal{EL} [7]), but in some cases it may be harder (for example, in \mathcal{ALE} , where it is PSPACE-complete [51]).

The original KLONE system [40] as well as its early successor systems (such as BACK [83], KREP [78], and LOOM [76]) employed so-called *structural subsumption algorithms*, which first

normalize the concept descriptions, and then recursively compare the syntactic structure of the normalized descriptions. These algorithms are usually very efficient (polynomial), but they have the disadvantage that they are complete only for very inexpressive DLs, i.e., for more expressive DLs they cannot detect all the existing subsumption relationships. The DL \mathcal{ALN} is an example of a DL where this structural approach yields a polynomial-time subsumption algorithm (see [27] for a sketch of such an algorithm and [38] for a detailed description of a structural subsumption algorithm for an extension of \mathcal{ALN}).

The syntactic characterization of subsumption in \mathcal{EL} and \mathcal{ALC} given in Section 4 can in principle also be used to obtain a structural subsumption algorithm for these DLs. It should be noted, however, that in the case of \mathcal{ALC} the normalization phase is not polynomial. For \mathcal{EL} , the normalization phase is void, but a naive top-down structural comparison would not result in a deterministic polynomial-time algorithm. To obtain a polynomial subsumption algorithm, one must use a dynamic programming approach, i.e., work bottom-up. Overall, structural subsumption does not seem to be the right tool for solving standard inferences for expressive DLs. However, as we will see, structural subsumption plays an important role for solving nonstandard inferences.

For expressive DLs (in particular, DLs allowing for disjunction and/or negation), for which the structural approach does not lead to complete subsumption algorithms, *tableau algorithms* have turned out to be useful: they are complete and often behave quite well in practice. The first such algorithm was proposed by Schmidt-Schauß and Smolka [89] for the DL \mathcal{ALC} .¹ It quickly turned out that this approach for deciding subsumption can be extended to various other DLs [59, 58, 13, 2, 55, 46, 11, 28, 65, 67, 29] and also to other inference problems such as the instance problem [56, 51, 57]. Early on, DL researchers started to

¹ Actually, at that time the authors were not aware of the close connection between their rule-based algorithm working on constraint systems and tableau procedures for modal and first-order predicate logics.

call the algorithms obtained this way “tableau-based algorithms” since they observed that the original algorithm by Schmidt-Schauß and Smolka for \mathcal{ALC} , as well as subsequent algorithms for more expressive DLs, could be seen as specializations of the tableau calculus for first-order predicate logic (the main problem to solve was to find a specialization that always terminates, and thus yields a decision procedure).

After Schild [88] showed that \mathcal{ALC} is a syntactic variant of multi-modal \mathbf{K} , it turned out that the algorithm by Schmidt-Schauß and Smolka was actually a re-invention of the tableau algorithm for \mathbf{K} known from modal logics [34].

The first DL systems employing tableau-based algorithms (KRIS [14] and CRACK [45]) demonstrated that (in spite of the high worst-case complexity of the underlying DL \mathcal{ALCN}) such algorithms can be implemented in a practical way. The complexity barrier has been pushed even further back by the seminal system FACT [61]. Although FACT employs the very expressive DL \mathcal{SHIQ} , which has an EXPTIME-complete subsumption problem, its highly optimized tableau-based subsumption algorithm outperforms the early systems based on structural subsumption algorithms and KRIS by several orders of magnitude [63]. The equally well-performing system RACER [54] also provides for a highly-optimized implementation of the ABox-consistency and instance test for an extension of \mathcal{SHIQ} .

3. Nonstandard Inferences—Motivation and Definitions

In this section, we will first motivate the nonstandard inferences considered in this paper within a uniform application scenario, in which these inferences are used to support the design of DL knowledge bases. Then, we give formal definitions of the relevant nonstandard inferences, and briefly sketch different techniques for solving them. Each nonstandard inference will be considered in

Mathematical Problems from Applied Logic I

Logics for the XXIst Century

Gabbay, D.; Goncharov, S.; Zakharyashev, M. (Eds.)

2006, XXVIII, 348 p. 50 illus., Hardcover

ISBN: 978-0-387-28688-4