Description Logic. Seminar presentation.

Mikolaj Kalinowski

January 16, 2014

1 Introduction to Description Logic

Description Logics were developed to facilitate building high-level representations of the world, with the languages used to query it. Terminological knowledge is used to build knowledge databases and algorithms use that data to infer implicit consequences not directly stated in the system.

1.1 History of Research

Some early predecessors of Description Logics (DLs) were Frames and Knowledge Networks. Those systems provided a method of representing human knowledge in a more friendly way than first-order logic. Previous knowledge formalisms often aimed for rejection of logical formalisms, but shortly after bestowing DL with formal semantics, it turned out it is actually a curious subset of FOL [Hayes, 1979].

The development of Description Logic continued in the 1980s and first DL-based systems started to show up in 1985. Those systems provided a universal interface for knowledge specified with the new, functional approach and deductive algorithms performed simple reasoning on this knowledge representation. While expressiveness was limited at that time, an important point of focus was on avoiding techniques of FOL theorem provers and providing polynomial complexities.

During 1990s a next generation of reasoning algorithms has emerged, viz the tableau method. Increased expressiveness came with the price of non-polynomial worst-case complexity.

2000s turned out to be pretty big for Description Languages, when World Wide Web Consortium (W3C) founded an Ontology Working Group with the goal of designing a family of DLs for representation of ontology and Semantic Web.

1.2 Relation to other logics

Description Logics can be translated to a subset of first-order logic with formulas restricted syntactically to contain at most 2 variables. This subset, called also L^2, is decidable [Mortimer, 1975].

There is also a relationship between Description Logics and multi-modal **K**. In fact, concepts in Description Logics are immediately translated to and from **K**-formulas in multi-modal **K**. This correspondence let mathematicians unify a lot of research, including complexity results and reasoning techniques.

1.3 Basic Concepts

One represents the knowledge in DLs using concepts denoting sets of individuals and roles describing relations between those. Complex concepts are described with terms, that are build with concept constructors. To describe a man that is married to a doctor and has only kids that are doctors themselves, one would write:

$$\text{Man}(x) \wedge \exists y(\text{married}(x, y) \wedge \text{Doctor}(y)) \wedge \forall y(\text{child}(x, y) \rightarrow \text{Doctor}(y))$$

Systems implementing DLs are sometimes called Tell and Ask. The "Tell" part involves describing the knowledge base using the mentioned concept terms and you can "Ask" the system to perform the inferences for you. Two most basic inferences are subsumption and satisfiability inference. Determining subsumption, usually written as $C \sqsubset D$, comprises checking if the concept denoted by D is considered more general than the one denoted by C. Satisfiability means checking whether the concept term denotes the empty concept.

1.4 Knowledge Representation

The nature of the knowledge bases is twofold: intentional knowledge, inherent to the problem domain and extensional knowledge specific to a particular problem.

Intentional knowledge in Description Logics is defined in a TBox (T stands for terminology or taxonomy) built with concept declarations, i.e. concept terms and subjunction statements. (note the lattice-like structure).

Extensional knowledge is what comprises an ABox (Assertional). It contains role assignments of the individuals in the particular domain. While intentional knowledge is deemed immutable, knowledge in ABox really describes a particular situation and is variable.

Common inference tasks include validating if as set of assertions in an ABox is consistent (if there is a model).

Let us consider a description logic containing medical knowledge database. In this case a TBox will describe a whole body of Medicine and ABox will be a single Patient data. This system can now aid doctors, by answering questions in the form:

$$patient1 \sqsubseteq \forall hasCancer.Patients$$

2 Formalism

2.1 Description Languages

we willThere is a whole family of Description Logic languages and in the following, we will look at ALCN: It is an Attributed Language extended with Negation, Number Restriction.

In the abstract notation we will use letters A and B for atomic concepts, letter R for atomic roles and letters C and D for content descriptions.

Content description in ALCN is formed as follows:

C, D $\rightarrow$A	(atomic concept)
$\mid\top$	(universal concept)
$\mid\bot$	(bottom concept)
$\mid\neg$C	(atomic negation extended to concepts)
$\mid$C $\sqcap$ D	(conjunction)
$\mid$C $\sqcup$ D	(disjunction)
$\mid\forall$R.C	(value restriction)
$\mid\exists$R.C	(existential restriction)
$\mid \leq nR$	(number restriction)
$\mid \leq nR$C	(qualifying number restriction)

Suppose that Person and Female are atomic concepts and that hasChild is an atomic role.

Person $\sqcap$ Female denotes the concept: person that is female

Person $\sqcap \neg$Female denotes the concept: person that is not female

Person $\sqcap \exists$hasChild.$\top$ denotes the concept all people who have children

Person $\sqcap \forall$hasChild.Female denotes the concept all people who have only female children

Person $\sqcap \forall$hasChild.$\bot$ denotes the concept: all people without children

Person $\sqcap$ ($\leq$ 1hasChild $\sqcup$ ($\geq$ 3hasChild $\sqcap \exists$hasChild.Female)) describes those people who have either not more than one child or at least three children, one of whom are female

2.1.1 Semantics

The semantics of ALCN is defined with interpretations $I = (\Delta^I, \cdot^I)$.

Δ^I is a non-empty set representing the domain of interpretation. $\cdot^I$ is an interpretation function assigning to every atomic concept A a set $A^I \subseteq \Delta^I$ and to each role a binary relation $R^I \subseteq \Delta^I \times \Delta^I$. An interpretation I is a model of a concept description C if $C^I \neq \emptyset$. The interpretation function is then inductively extended to content descriptions as follows:

$$\top^I = \Delta^I$$
$$\bot^I = \emptyset$$
$$(\neg\text{C})^I = \Delta^I \backslash C^I$$
$$(\text{C} \sqcap \text{D})^I = C^I \cap D^I$$
$$(\text{C} \sqcup \text{D})^I = C^I \cup D^I$$
$$(\forall\text{R.C})^I = \{a \in \Delta^I | \forall b.(a,b) \in R^I \rightarrow b \in C^I\}$$
$$(\exists\text{R.C})^I = \{a \in \Delta^I | \exists b.(a,b) \in R^I \wedge b \in C^I\}$$
$$(\leq nR)^I = \{a \in \Delta^I || \{(a,b) \in R^I\}| \leq n\}$$
$$(\leq nR\text{C})^I = \{a \in \Delta^I || \{(a,b) \in R^I | b \in C^I\}| \leq n\}$$

With this definition we can say that two concepts are equivalent if they their models are equal for all interpretations I. For example we can easily verify that interpretations of

$$\forall \text{hasChild.Female} \sqcup \forall \text{hasChild.Female}$$

and

$$\forall \text{hasChild.(Female} \sqcup \text{Student)}$$

denote the same sets for every interpretation of atomic concepts and roles, so those compound concepts are equivalent.

2.1.2 Translation to FOL

Looking at semantics defined in section 2.1.1, we can come up with a way to translate any concept to a first-order predicate formula. Atomic concepts and roles are respectively unary and binary relations over Δ^I, a concept description is then a predicate logic formula $\phi_C(x)$ such that for every interpretation I, the set of elements satisfying it is equal to C^I. The atomic concept A is translated into the formula $A(a)$. Concepts with intersection, sum and negation are translated respectively to logical conjunction, alternative or negation. Assuming that b is a new variable, the rest of the constructors are translated to formulae:

$$\phi_{\exists \text{R.C}}(a) = \exists b.R(a,b) \wedge \phi_C(a)$$

$$\phi_{\forall \text{R.C}}(a) = \forall b.R(a,b) \rightarrow \phi_C(b)$$

$$\phi_{\leq nRC}(a) = \forall b_1, ...b_{n+1}.R(a,b_1) \wedge ... \wedge R(a,b_{n+1}) \wedge \bigwedge \phi_C(b_i) \rightarrow \bigvee_{i<j} b_i = b_j$$

All translations except for the number restriction can be expressed in formulas using only 2 variables, which place them in a decidable subset of FOL. Number restriction however can be translated to C2 - another subset of FOL with counting quantifiers. C2 has been shown to be decidable [Graedel et al., 1997b; Pacholski et al., 1997] in NExpTime-complete.

It should be noted that formulas translated in such a ways are much longer and decision procedures for them have higher complexity than necessary. In comparison, satisfiability of ALCN formulas is PSpace-complete.

2.2 Terminologies

The concept language allows us to form complex descriptions of concepts to describe classes of objects. To represent relations between concepts and roles in a structured way, we will represent them using terminologies - sets of definitions by which we can introduce atomic concepts as abbreviations or names for complex concepts.

2.2.1 TBox

As mentioned before, intentional knowledge is represented by TBoxes which contain a finite set of concept definitions and equalities: $C \equiv D (R \equiv S)$

where C is a concept name and D is a concept description, such that no concept name appears on the left-hand side of two different concept definitions in a TBox T.

A concept is called a defined concept if it appears on the left-hand side of a concept definition and a primitive (base) concept otherwise.

We say that a concept name A *directly uses* a concept name B if there is a concept definition $A \equiv C \in TboxT$ and B occurs in C. Let *uses* mean the transitive closure of *directly uses*. Then a TBox T contains a terminological cycle if there is a concept name that *uses* itself. For example here is a TBox without terminological cycle; we will call it acyclic:

$$\begin{aligned}
\text{Woman} &\equiv \text{Person} \sqcap \text{Female} \\
\text{Man} &\equiv \text{Person} \sqcap \neg\text{Woman} \\
\text{Mother} &\equiv \text{Woman} \sqcap \exists\text{hasChild.Person} \\
\text{Father} &\equiv \text{Man} \sqcap \exists\text{child.Person} \\
\text{Parent} &\equiv \text{Mother} \sqcup \text{Father} \\
\text{Grandmother} &\equiv \text{Mother} \sqcap \exists\text{hasChild.Parent}
\end{aligned}$$

And here is an example of a TBox with a terminological cycle; we will call it cyclic:

$$\text{Human} \equiv \text{Adam} \sqcup \text{Eve} \sqcup \exists\text{parent.Human}$$

For acyclic TBoxes, the natural semantics is descriptive semantics: an interpretation I satisfies a concept definition $A \equiv C$ if $A^I = C^I$, and I is a model of the TBox T if it satisfies all concept definitions in T. Intuitively, acyclic TBoxes merely state that defined concepts are abbreviations for certain compound concept descriptions. These compound concepts can be made explicit by expanding the acyclic TBox T : exhaustively replace all concept names A on the left-hand side of concept definitions A ≡ C by their defining concept descriptions C. After this expansion, the compound concept abbreviated by a defined concept can simply be read off from the corresponding concept definition. For example, the defined concept Father abbreviates to the compound concept:

$$Person \sqcap \neg(Person \sqcap Female) \sqcap \exists child.Person$$

To define semantics of cyclic boxes we need more terminology. A base interpretation for T is an interpretation that interprets only the base symbols. Let J be such a base interpretation. An interpretation I that interprets also the name symbols is an extension of J if it has the same domain as J, i.e., $\Delta^I = \Delta^J$, and if it agrees with J for the base symbols. We say that T is definitorial if every base interpretation has exactly one extension that is a model of T . In other words, if we know what the base symbols stand for, and T is definitorial, then the meaning of the name symbols is completely determined. Obviously, if a terminology is definitorial, then every equivalent terminology is also definitorial. The question whether a terminology is definitorial or not is related to the question whether or not its definitions are cyclic.

It is easy to see that if terminology T is acyclic, then it is definitorial. If T' is expanded terminology of acyclic terminology T, then both terminologies have the same base and name symbols. At each step of the iterative expansion, we are replacing symbols with equivalent definition, so models of T and T' are equal.

Suppose that J is an interpretation of base symbols. We extend it to an interpretation I that covers also the name symbols by setting $A^I = C^J$ if $A \equiv C$ is the definition of A in T'.

Clearly I is a model of T' and it is the only extension of J that is a model of T'. This shows that T' is definitorial. Moreover T is definitorial aswell since it is equivalent to T'.

Cyclic terminologies can be definitorial. Consider for instance the one consisting of the axiom

$$A \equiv \forall R.B \sqcup \exists R.(A \sqcap \neg A)$$

Since $\exists R.(A \sqcap \neg A)$ is equivalent to the bottom concept, that axiom is equivalent to the acyclic axiom

$$A \equiv \forall R.B$$

Let us consider this cyclic terminology that is not definitorial:

$$Human' \equiv Animal \sqcap \forall hasParent.Human'$$

Here Human' is a name symbol and Animal and hasParent are base symbols. For an interpretation where hasParent relates every animal to its progenitors, many extensions are possible to interpret Human' in a such a way that the axiom is satisfied: Human' can, among others, be interpreted as the set of all animals, as some species, or any other set of animals with the property that for each animal it contains also its progenitors. That means that unique extension does not exist and this terminology is not definitorial.

Example Suppose that we want to specify the concept of a "man who has only male offspring," for short Momo. In particular, such a man is a Mos, i.e. a "man who has only sons." A Mos can be defined without cycles as

$$Mos \equiv Man \sqcap \forall hasChild.Man.$$

For a Momo, however, we want to make a statement about the fillers of the transitive closure of the role hasChild. Here a recursive definition of Momo seems to be natural. A man having only male offspring is himself a man, and all his children are men having only male offspring:

$$Momo \equiv Man \sqcap \forall hasChild.Momo.$$

In order to achieve the desired meaning, we have to interpret this definition under an appropriate fixpoint semantics. We shall show below that greatest fixpoint semantics captures our intuition here.

Fixpoint semantics Because in a terminology T, every name symbol A occurs exactly once as a definition of a concept name in the form of $A \equiv C$, we can view T as a mapping that associates the concept description to a name symbol: $T(A) = C$. With this notation, an interpretation I is a model of T iff $A^I = (T(A))^I$.

This equation reminds us of a fixpoint equation, and we will use this similarity to define a family of mappings, such that an an interpretation is a model of T iff it is a fixpoint of such a mapping.

Let T be a terminology and let J be a fixed base interpretation of T. By Ext_J we denote the set of all extensions of J. Let $T_J : Ext_J \to Ext_J$ be the mapping that maps the extension I to the extension $T_J(I)$ defined by $A^{T_J(I)} = (T(A))^I$ for each name symbol A.

Now, I is a fixpoint of T_J iff $I = T_J(I)$, i.e., iff $A^I = A^{T_J(I)}$ for all name symbols. This means that for every definition $A \equiv C$ in T , we have $A^I = A^{T_J(I)} = (T(A))^I = C^I$, which means that I is a model of T . This proves the following result.

Proposition Let T be a terminology, I be an interpretation, and J be the restriction of I to the base symbols of T. Then I is a model of T iff I is a fixpoint of T_J .

According to the preceding proposition, a terminology T is definitorial iff every base interpretation J has a unique extension that is a fixpoint of T_J .

Example To see why cyclic terminologies are not definitorial, we will discuss as an example the terminology T^{Momo} that consists only of Axiom Momo. Consider the base interpretation J defined by

$$\Delta^J = \{Charles_1, Charles_2, ...\} \cup \{James_1, ..., James_{Last}\}$$

$$Man^J = \Delta^J$$

$$hasChild^J = \{(Charles_i, Charles_{(i+1)}) | i \geq 1\} \cup \{(James_i, James_{(i+1)}) | 1 \leq i \leq Last\}$$

This means that the Charles dynasty does not die out, whereas there is a last member of the James dynasty.

We want to identify the fixpoints of T_J^{Momo}. Note that an individual without children, i.e., without fillers of hasChild, is always in the interpretation of $\forall hasChild.Momo$, no matter how Momo is interpreted.

Therefore, if I is a fixpoint extension of J , then $James_{Last}$ is in $(\forall hasChild.Momo)^I$, and thus in $Momo^I$. We can conclude that every James is a Momo. Let I_1 be the extension of J such that $Momo^{I1}$ comprises exactly of the James dynasty. Then it is easy to check that I_1 is a fixpoint. If also some Charles is a Momo, then all the members of the Charles dynasty before and after him must belong to the concept Momo. We can easily check that the extension I_2 that interprets Momo as the entire domain is also a fixpoint. There are no other fixpoints.

To make cyclic terminology T definitorial, it is necessary to single out a single fixpoint of the mapping T_J if there is more than one. For that we will define a partial ordering $\preceq$ on the extensions of J. We say that $I \preceq I'$ if $A^I \subseteq A^{I'}$ for every name symbol in T . In the above example, Momo is the only name symbol. Since $Momo^{I_1} \subseteq Momo^{I_2}$, we have $I_1 \preceq I_2$. In the Momo example, I_1 is the least and I_2 the greatest fixpoint of T_J.

Existence of fixpoint models Least and greatest fixpoint models need not exist for every terminology.

Example Consider the axiom

$$A \equiv \neg A.$$

If I is a model of this axiom, then $A_I = \Delta^I \backslash A^I$, which implies $\Delta_I = \emptyset$, an absurd.

A terminology containing this axiom thus does not have any models and therefore also no greatest (or least) fixed point models.

There are also cases where models (i.e. fixpoints) exist, but there is neither the least one nor the greatest one. As an example, consider the terminology T with the single axiom

$$A \equiv \forall R.\neg A$$

Let J be the base interpretation with $\Delta^J = \{a, b\}$ and $R^J = \{(a, b), (b, a)\}$. Then there are two fixpoint extensions I_1, I_2, defined by $A^{I_1} = \{a\}$ and $A^{I_2} = \{b\}$. They are not comparable.

We can show that $Ext_J, \preceq$ is a complete lattice and with T_J being monotone, by Knaster-Tarski theorem it has least and greatest fix point. There are a couple of ways to make a terminology monotone. The simplest one is to eliminate negation from the language. A less restrictive way is to only ensure that all cycles in the dependency graph have an even number of edges .

2.2.2 ABox

The second component of a knowledge base is the world description or ABox. In the ABox, one describes a specific state of affairs of an application domain in terms of concepts and roles. Some of the concept and role atoms in the ABox may be defined names of the TBox. In the ABox, one introduces individuals, by giving them names and one asserts properties about these individuals. We denote individual names as r, q, s. Using concepts C and roles R, one can make assertions of the following two kinds in an ABox:

$$C(r)$$

$$R(q, s)$$

.

By the first kind, called concept assertions, one states that r belongs to (the interpretation of) C, by the second kind, called role assertions, one states that s is a filler of the role R for q.

Example ABox:

$$MotherWithoutDaughter(MARY)$$

$$Father(PETER)$$

$$hasChild(MARY, PETER)$$

$$hasChild(PETER, HARRY)$$

$$hasChild(MARY, PAUL)$$

In a simplified view, an ABox can be seen as an instance of a relational database with only unary or binary relations. However, contrary to the "closed-world semantics" of classical databases, the semantics of ABoxes is an "open-world semantics," since usually knowledge representation systems used when knowledge in the KB is not complete.

While a database instance represents exactly one interpretation, namely the one where classes and relations in the schema are interpreted by the objects and tuples in the instance, an ABox represents many different interpretations, namely all its models. As a consequence, absence of information in a database instance is interpreted as negative information, while absence of information in an ABox only indicates lack of knowledge.

Semantics for ABoxes are done by extending interpretations to individual names. With that extension, interpretation $I = (\Delta^I, \cdot^I)$ not only maps atomic concepts and roles to sets and relations, but in addition maps each individual name r to an element $r^I \in \Delta^I$. We assume that distinct individual names denote distinct objects. Therefore, this mapping has to respect the unique name assumption (UNA), i.e. if r, q are distinct names, then $r^I \neq q^I$

. The interpretation I satisfies the concept assertion $C(r)$ if $r^I \in C^I$ and it satisfies the role assertion $R(r, q)$ if $(r^I, q^I) \in R^I$. An interpretation satisfies the ABox A if it satisfies each assertion in A. In this case we say that I is a model of the assertion or of the ABox. Finally, I satisfies an assertion α or an ABox A with respect to a TBox T if in addition to being a model of α or of A, it is a model of T. Thus, a model of A and T is an abstraction of a concrete world where the concepts are interpreted as subsets of the domain as required by the TBox and where the membership of the individuals to concepts and their relationships with one another in terms of roles respect the assertions in the ABox.

3 Reasoning

3.1 Typical Inferences

We have successfully defined a way to represent the knowledge in a DL system. But the purposes of knowledge based systems go beyond only storing concept definitions and assertions and it is possible to perform specific kinds of reasoning on this data. Because a knowledge base comprising TBox and ABox has semantics that makes it equivalent to a set of axioms in first-order predicate logic, like any other set of axioms, it contains implicit knowledge that can be made explicit through inferences. For example, from the previous TBox and Abox one can conclude that Mary is a grandmother, although this knowledge is not explicitly stated as an assertion.

When modeling a domain, we start by constructing a terminology T, by defining new concepts, possibly in terms of others that have been defined before. During this process, it is important to find out whether a newly defined concept makes sense or whether it is contradictory. From a logical point of view, a concept makes sense for us if there is some interpretation that satisfies the axioms of T (that is, a model of T) such that the concept denotes a nonempty set in that interpretation. A concept with this property is said to be satisfiable with respect to T and unsatisfiable otherwise.

Checking satisfiability of concepts is a key inference. As we shall see, a number of other important inferences for concepts can be reduced to the (un)satisfiability. For instance, in order to check whether a domain model is correct, or to optimize queries that are formulated as concepts, we may want to know whether some concept is more general than another one: this is the subsumption problem. A concept C is subsumed by a concept D if in every model of T the set denoted by C is a subset of the set denoted by D. Algorithms that check subsumption are also employed to organize the concepts of a TBox in a taxonomy according to their generality.

Satisfiability : A concept C is satisfiable with respect to T if there exists a model I of T such that C^I is nonempty. In this case we say also that I is a model of C.

Subsumption : A concept C is subsumed by a concept D with respect to T if $C^I \subseteq D^I$ for every model I of T. In this case we write $C \sqsubseteq_T D$ or $T \models C \subseteq D$.

Equivalence : Two concepts C and D are equivalent with respect to T if $C^I = D^I$ for every model I of T. In this case we write $C \equiv_T T$ or $T \models C \equiv D$.

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

For example Person subsumes Woman, both woman and Parent subsume Mother and Mother subsumes Grandmother. Moreover Woman and Man, and Father and Mother are Disjoint.

Reduction to Unsatisfiability For concepts C, D we have

(i) C is subsumed by D , $C \sqcup \neg D$ is unsatisfiable;

(ii) C and D are equivalent , both $(C \sqcup \neg D)$ and $(\neg C \sqcup D)$ are unsatisfiable;

(iii) C and D are disjoint , $C \sqcap D$ is unsatisfiable. The statements also hold with respect to a TBox.

The reduction of subsumption can easily be understood if we recall that for sets M, N, we have $M \subseteq N$ iff $M \backslash N = \emptyset$. The reduction of equivalence is correct because C and D are equivalent if, and only if, C is subsumed by D and D is subsumed by C. Finally, the reduction of disjointness is just a rephrasing of the definition.

Because of the above proposition, in order to obtain decision procedures for any of the four inferences we have discussed, it is sufficient to develop algorithms that decide the satisfiability of concepts, provided the language for which we can decide satisfiability supports conjunction as well as negation of arbitrary concepts.

3.2 Tableau method

Early day DL systems did not allow for negation and for such DLs, subsumption of concepts can usually be computed by so-called structural subsumption algorithms, i.e., algorithms that compare the syntactic structure of (possibly normalized) concept descriptions.

Instead of directly testing subsumption of concept descriptions, tableau algorithms use negation to reduce subsumption to (un)satisfiability of concept descriptions.

Before describing a tableau-based satisfiability algorithm for ALCN in more detail, we illustrate the underlying ideas by two simple examples. Let A, B be concept names, and let R be a role name.

As a first example, assume that we want to know whether $(\exists R.A) \sqcap (\exists R.B)$ is subsumed by $\exists R.(A \sqcap B)$. This means that we must check whether the concept description

$$C = (\exists R.A) \sqcap (\exists R.B) \sqcap \neg(\exists R.(A \sqcap B))$$

is unsatisfiable.

First, we push all negation signs as far as possible into the description, using de Morgan's rules and the usual rules for quantifiers. As a result, we obtain the description

$$C_0 = (\exists R.A) \sqcap (\exists R.B) \sqcap (\forall R.(\neg A \sqcup \neg B))$$

which is in negation normal form, i.e. negation occurs only in front of concept names.

Then, we try to construct a finite interpretation I such that $C_0^I \neq \emptyset$. This means that there must exist an individual in Δ^I that is an element of C_0^I.

The algorithm just generates such an individual, say b, and imposes the constraint $b \in C_0^I$ on it. Since C_0 is the conjunction of three concept descriptions, this means that b must satisfy the following three constraints: $b \in (\exists R.A)^I$, $b \in (\exists R.B)^I$, and $b \in (\forall R.(\neg A \sqcup \neg B))^I$.

From $b \in (\exists R.A)^I$ we can deduce that there must exist an individual c such that $(b, c) \in R^I$ and $c \in A^I$. Analogously, $b \in (\exists R.B)^I$ implies the existence of an individual d with $(b, d) \in R^I$ and $d \in B^I$. In this situation, one should not assume that c = d since this would possibly impose too many constraints on the individuals newly introduced to satisfy the existential restrictions on b. Thus:

For any existential restriction the algorithm introduces a new individual as role filler, and this individual must satisfy the constraints expressed by the restriction. Since b must also satisfy the value restriction $(\forall R.(\neg A \sqcup \neg B))^I$, and c, d were introduced as R-fillers of b, we obtain the additional constraints $c \in (\neg A \sqcup \neg B)^I$ and $d \in (\neg A \sqcup \neg B)^I$. Thus:

The algorithm uses value restrictions in interaction with already defined role relationships to impose new constraints on individuals. Now $c \in (\neg A \sqcup \neg B)^I$ means that $c \in (\neg A)^I$ or $c \in (\neg B)^I$, and we must choose one of these possibilities. If we assume $c \in (\neg A)^I$, this clashes with the other constraint $c \in A^I$, which means that this search path leads to an obvious contradiction. Thus we must choose $c \in (\neg B)^I$. Analogously, we must choose $d \in (\neg A)^I$ in order to satisfy the constraint $d \in (\neg A \sqcup \neg B)^I$ without creating a contradiction to $d \in B^I$. Thus:

For disjunctive constraints, the algorithm tries both possibilities in successive attempts. It must backtrack if it reaches an obvious contradiction, i.e., if the same individual must satisfy constraints that are obviously conflicting. In the example, we have now satisfied all the constraints without encountering an obvious contradiction. This shows that C_0 is satisfiable, and thus $(\exists R.A) \sqcup (\exists R.B)$ is not subsumed by $\exists R.(A \sqcap B)$. The algorithm has generated an interpretation I as witness for this fact: $\Delta^I = \{b, c, d\}$; $R^I = \{(b, c), (b, d)\}$; $A^I = \{c\}$ and $B^I = \{d\}$.

For this interpretation, $b \in C_0^I$. This means that $b \in ((\exists R.A) \sqcap (\exists R.B))^I$, but $b \notin (\exists R.(A \sqcap B))^I$.

In our second example, we add a number restriction to the first concept of the above example, i.e., we now want to know whether $(\exists R.A) \sqcap (\exists R.B) \sqcap \leq 1R$ is subsumed by $\exists R.(A \sqcap B)$. Intuitively, the answer should now be "yes" since $\leq 1R$ in the first concept ensures that the R-filler in A coincides with the R-filler in B, and thus there is an R-filler in $A \sqcap B$. The tableau-based satisfiability algorithm first proceeds as above, with the only difference that there is the additional constraint $b \in (\leq 1R)^I$. In order to satisfy this constraint, the two R-fillers c, d of b must be identified with each other. Thus:

If an at-most number restriction is violated then the algorithm must identify different role fillers. In the example, the individual $c = d$ must belong to both A^I and B^I, which together with $c = d \in (\neg A \sqcup \neg B)^I$ always leads to a clash. Thus, the search for a counterexample to the subsumption relationship fails, and the algorithm concludes that

3.2.1 Complexity

This algorithm is PSpace-complete since it is a non-deterministic algorithm using only polynomial space, i.e., for every non-deterministic rule we may simply assume that the algorithm chooses the correct alternative.

References

[1] D. L. McGuiness, D. Nardi, P. F. Patel-Schneider : THE DESCRIPTION LOGIC HANDBOOK: Theory, implementation, and applications.

[2] Franz Baader and Carsten Lutz : Description Logic from Handbook of Modal Logic