Andrea Schaerf

Reasoning with Individuals in Concept Languages

Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza",
via Salaria 113, I-00198 Roma, Italy.

Reasoning with Individuals in Concept Languages

Sommario

Una delle principali caratteristiche dei sistemi per la rappresentazione della conoscenza basati sulla descrizione dei concetti è la netta distinzione tra conoscenza terminologica e conoscenza asserzionale. Sebbene questa caratteristica sia causa notevoli vantaggi sia computazionali che di rappresentazione, essa generalmente limita il potere espressivo del sistema. Per questa ragione sono stati fatti alcuni tentativi nella direzione di permettere qualche forma di fusione tra le due componenti ed una interazione più complessa tra esse. In particolare, uno di tali tentativi è basato sul permettere che gli individui vengano referenziati nelle espressioni dei concetti. Cio' e' generalmente fatto ammettendo un costruttore che formi un concetto da un insieme di individui enumerati.

In questo lavoro si indagano le conseguenze di introdurre costruttori di questo tipo nel linguaggio per la descrizione dei concetti. Viene anche fornita una procedura di ragionamento completa che permetta di gestire questi costruttori e vengono ottenuti alcuni risultati di complessità sul ragionamento con tali costrutti.

Abstract

One of the main characteristics of knowledge representation systems based on the description of concepts is the clear distinction between terminological and assertional knowledge. Although this characteristic leads to several computational and representational advantages, it usually limits the expressive power of the system. For this reason, some attempts have been done, allowing for a limited form of amalgamation between the two components and a more complex interaction between them. In particular, one of these attempts is based on letting the individuals to be referenced in the concept expressions. This is generally performed by admitting a constructor for building a concept from a set of enumerated individuals.

In this paper we investigate on the consequences of introducing constructors of this type in the concept description language. We also provide a complete reasoning procedure to deal with these constructors and we obtain some complexity results on it.

1 Introduction

The idea of developing knowledge representation systems based on a structured representation of knowledge was first pursued with semantic networks and frames. Later, concept description logics (also called terminological languages or *concept languages*) have been introduced with the aim of providing a simple and well-established first order semantics to capture the meaning of the most popular features of the structured representations of knowledge (see for example [LB87, Neb90a]).

In concept languages, concepts are used to represent classes as sets of individuals, and roles are binary relations used to specify their properties or attributes. Typically concepts are structured into hierarchies determined by the properties associated with them. The hierarchical structure is defined in such a way that more specific concepts inherit the properties of the more general ones.

One of the main characteristics of concept-description-based knowledge bases is the clear distinction between terminological and assertional knowledge (see [BPGL85, Neb90a, Mac91, NvL88]). The former deals with concepts and roles and their relationship, the latter with individuals and their membership to concepts and roles. The two kinds of knowledge are stored in two different components of the knowledge base and each component has its specialized reasoner. Moreover, the inferences of each component can be combined in order to obtain more complex inferences, called hybrid inferences. The advantage of this architecture is that the specialized reasoners are usually able to process their specific knowledge more efficiently than a general purpose reasoner.

On the other hand, the strict separation of the two components limits the expressive power of the overall system. In order to recover some of the expressive power, some attempts have been done, which allow for a limited mixing of the two components and/or a more complex interaction between them.

One of these attempts is to admit the presence of the individuals, which are generally only present in the assertional component, also in the terminological one. This is usually done by introducing new constructors in the languages for defining the concepts. The reason why this fact results in a mixing of the two components will clarified in the sequel.

In particular, one of these constructors is obtained by building a concept from a set of enumerated individuals. This constructor, called `ONE-OF` in [BBMAR89] and simply $\mathcal{O}$ in this paper, allows one to express many natural concepts. For example, the concept `Permanent_onu_member` can be defined as $\{$`china, france, russia, uk, usa`$\}$, where `china,..., usa` are individuals.

Another constructor of the same kind, called `FILLS` in [BBMAR89] and $\mathcal{B}$ here, is the one for denoting the set of objects having a particular individual as a filler of a specified role. For example the concept representing the set of USA citizens can be expressed as `CITIZENSHIP:usa`, where `CITIZENSHIP` is a role and `usa` is an individual.

The demand for the constructors $\mathcal{O}$ and $\mathcal{B}$ in concept-based systems is due to the significant increase of the expressiveness of the language they provide, as shown in Section 3. It is also confirmed by the fact that they are both included in the recent proposal for a standard concept-based system in [PS93] ($\mathcal{O}$ was also included in the previous proposal [BBH+91]).

Moreover, in Section 3, we show that the use of $\mathcal{O}$ is also related to the introduction of an epistemic operator **K** in the concept-based system, as proposed in [DLN+92]. The epistemic operator **K** turned out to be very useful both for providing a highly expressive query language and for a formal characterization of some procedural mechanisms usually considered in concept-based systems. However a complete understanding of the possible uses of the **K** operator is still missing and the analysis of $\mathcal{O}$ can be helpful for this purpose.

In this paper we investigate on the consequences of introducing $\mathcal{O}$ and $\mathcal{B}$ in the concept language, and in general of admitting the individuals in it. In particular in the following sections, we introduce concept languages and their reasoning services, together with their syntax and their semantics (Section 2). We give a survey of the various issues associated with the use of $\mathcal{O}$ and $\mathcal{B}$ (Section 3). We briefly describe some of the strategies chosen by the implementors of the actual systems in order to deal with $\mathcal{O}$ and $\mathcal{B}$ (Section 4). We extend the reasoning procedure proposed in [SSS91, DLNN91, BH91, DLNS92] in order to develop a complete technique for reasoning with $\mathcal{O}$ and $\mathcal{B}$ (Section 5). We present several complexity results (Section 6). We propose a limited use of $\mathcal{O}$ and $\mathcal{B}$ (Section 7) and, finally, in Section 8 we draw some conclusions.

2 Preliminaries

In this section we present the basic notions regarding concept languages, knowledge bases built up using concept languages, and reasoning services that must be provided for inferring information from such knowledge bases.

2.1 Concept Languages

We consider a family of concept languages, called $\mathcal{AL}$-languages, which includes most of the concept languages considered in the literature. The simplest language of this family, called $\mathcal{AL}$, is an extension of the basic language $\mathcal{FL}^-$ introduced in [BL84] including a constructor for denoting the complement of primitive concepts and the two special concepts $\top$ and $\bot$. Given an alphabet of primitive concept symbols $\mathcal{C}$ and an alphabet of role symbols $\mathcal{R}$, $\mathcal{AL}$-*concepts* (denoted by the letters C and D) are built by means of the following syntax rule

$$
\begin{array}{rcll}
C, D & \longrightarrow & A \mid & \text{(primitive concept)} \\
 & & \top \mid & \text{(top)} \\
 & & \bot \mid & \text{(bottom)} \\
 & & \neg A \mid & \text{(primitive complement)} \\
 & & C \sqcap D \mid & \text{(intersection)} \\
 & & \forall R.C \mid & \text{(universal quantification)} \\
 & & \exists R & \text{(unqualified existential quantification)}
\end{array}
$$

where R denotes a *role*, that in $\mathcal{AL}$ is always primitive (more general languages provide constructors for roles).

In the following, we use parentheses whenever we need to disambiguate concept expressions. For example, we shall write $(\forall R.D) \sqcap E$ to mean that the concept E is not in the scope of $\exists R$.

Both $\mathcal{FL}^-$ and $\mathcal{AL}$ provide a restricted form of existential quantification, called *unqualified*: the construct $\exists R$ denotes the set of objects d_1 such that there exists an object d_2 related to d_1 by means of the role R. The existential quantification is unqualified in the sense that no condition is stated to d_2 other than its existence.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a nonempty set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) that maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that the following equations are satisfied:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{d_1 \in \Delta^{\mathcal{I}} \mid \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \to \ d_2 \in C^{\mathcal{I}}\} \\
(\exists R)^{\mathcal{I}} &= \{d_1 \in \Delta^{\mathcal{I}} \mid \exists d_2 : (d_1, d_2) \in R^{\mathcal{I}}\}
\end{aligned}$$

An interpretation $\mathcal{I}$ is a *model* for a concept C if $C^{\mathcal{I}}$ is nonempty. A concept is *satisfiable* if it has a model and *unsatisfiable* otherwise. We say that C is *subsumed* by D if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$, and C is *equivalent* to D, written $C \equiv D$, if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$.

Example 2.1 Consider the following two $\mathcal{AL}$-concepts

$$\mathtt{Person}\sqcap\exists\mathtt{CHILD}, \quad \mathtt{Person}\sqcap\exists\mathtt{CHILD}\sqcap(\forall\mathtt{CHILD.Graduate})$$

The first one denotes the individuals having at least one child. The second one denotes the individuals having at least one child and having only graduate children. It is easy to see that they are both satisfiable and that the first one subsumes the second.

On the contrary, the following concept is not satisfiable and it is therefore trivially subsumed by both the others.

$$(\exists\mathtt{CHILD})\sqcap(\forall\mathtt{CHILD.Female})\sqcap(\forall\mathtt{CHILD.}\neg\mathtt{Female})$$

□

More general languages are obtained by adding to $\mathcal{AL}$ the following constructors:

- *qualified existential quantification* (indicated by the letter $\mathcal{E}$), written as $\exists R.C$, and defined by $(\exists R.C)^{\mathcal{I}} = \{d_1 \in \Delta^{\mathcal{I}} \mid \exists d_2 : (d_1, d_2) \in R^{\mathcal{I}} \wedge d_2 \in C^{\mathcal{I}}\}$;

- *union of concepts* (indicated by the letter $\mathcal{U}$), written as $C \sqcup D$, and defined by $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$;
- *complement of general concepts* (indicated by the letter $\mathcal{C}$), written as $\neg C$, and defined by $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$;
- *conjunction of roles* (indicated by the letter $\mathcal{R}$), written as $R \sqcap Q$, and defined by $(Q \sqcap R)^{\mathcal{I}} = Q^{\mathcal{I}} \cap R^{\mathcal{I}}$;
- *number restrictions* (indicated by the letter $\mathcal{N}$), written as $(\geq n\, R)$ and $(\leq n\, R)$, where n range over the nonnegative integers, and defined by

$$(\geq n\, R)^{\mathcal{I}} = \{d_1 \in \Delta^{\mathcal{I}} \mid \ |\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\}| \ \geq n\},$$
$$(\leq n\, R)^{\mathcal{I}} = \{d_1 \in \Delta^{\mathcal{I}} \mid \ |\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\}| \ \leq n\},$$

Using these constructors, alone or in combination, it is possible to construct more expressive $\mathcal{AL}$-languages. Unfortunately, besides of the gained expressive power, such constructors usually increase the complexity of reasoning in concept languages. An extensive study of the complexity of computing subsumption in these languages is performed in [DLNN91].

Apart from the above ones, two further particular constructors have been considered in concept languages. These constructors have the peculiarity to involve the elements of a new alphabet $\mathcal{A}$, called *individuals*:

- *collection of individuals* (indicated by the letter $\mathcal{O}$), written as $\{a_1, \ldots, a_n\}$, where each a_i belongs to $\mathcal{A}$.
- *role filler* (indicated by the letter $\mathcal{B}$), written as $R : a$, where R is a role and a belongs to $\mathcal{A}$

In order to assign a meaning to such constructors, the interpretation function $\cdot^{\mathcal{I}}$ is extended to individuals in such a way that $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ for each individual $a \in \mathcal{A}$ and $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$ (Unique Name Assumption). The semantics of $\{a_1, \ldots, a_n\}$ is then defined by

$$\{a_1, \ldots, a_n\}^{\mathcal{I}} = \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\},$$

and the semantics of $R : a$ is defined by

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

It is important to observe that the above constructor are not all independent of each other. In particular, the combination of union and qualified existential quantification gives the possibility to express complements of concepts, and conversely, union and qualified existential quantification can be expressed using complements. Moreover due to the following equivalence

$$R : a \equiv \exists R.\{a\}$$

$\mathcal{B}$ can be expressed in terms of $\mathcal{O}$ and $\mathcal{E}$. Hence, without loss of generality we will assume that $\mathcal{U}$ and $\mathcal{E}$ are available in languages that contain C, and vice versa; and that $\mathcal{B}$ is available in the languages including $\mathcal{O}$ and $\mathcal{E}$ (or $\mathcal{O}$ and $\mathcal{C}$).

From this point on, we will identify a language with a string of the form

$$\mathcal{AL}[\mathcal{E}][\mathcal{U}][\mathcal{C}][\mathcal{R}][\mathcal{N}][\mathcal{O}][\mathcal{B}]$$

indicating which constructors are allowed in the language[1]. Due to the mentioned equivalences, different strings can identify the same language. For example, $\mathcal{ALEU}$ is the same language as $\mathcal{ALC}$ (and $\mathcal{ALEUC}$), $\mathcal{ALEO}$ is the same as $\mathcal{ALEOB}$ and so on.

We do not claim the list of the considered constructors to be exhaustive. The description of some other useful constructors can be found in [BBH+91] and in [PS93].

From this point on, we call the languages without $\mathcal{O}$ and $\mathcal{B}$ *pure languages* and those including at least one of them *mixed languages*. The reason for these names will be clearer in Section 3.

2.2 Knowledge Bases

The construction of knowledge bases using concept languages is realized by permitting concept and role expressions to be used in assertions on individuals. Given a concept language $\mathcal{L}$, an $\mathcal{L}$-assertion is a statement of one of the forms:

$$C(a), \quad R(a, b)$$

where C is a concept of $\mathcal{L}$, R is a role of $\mathcal{L}$, and a, b are individuals in $\mathcal{A}$. The semantics of the above assertions is straightforward: if $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ is

[1] We will also speak about languages without primitive complements. Such languages will be identified with a string of the form $\mathcal{FL}[\mathcal{E}][\mathcal{U}][\mathcal{R}][\mathcal{N}][\mathcal{O}][\mathcal{B}]^-$.

an interpretation, $C(a)$ is satisfied by $\mathcal{I}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and $R(a,b)$ is satisfied by $\mathcal{I}$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

A set Σ of $\mathcal{L}$-assertions is called an *$\mathcal{L}$-knowledge base*. An interpretation $\mathcal{I}$ is said to be a *model* of Σ if every assertion of Σ is satisfied by $\mathcal{I}$. Σ is said to be *satisfiable* if it admits a model. We say that Σ *logically implies* α, where α is either an assertion or a subsumption relation, if α is satisfied by every model of Σ (written $\Sigma \models \alpha$).

In the so-called terminological systems, the knowledge base also includes an intensional part, called *terminology*, expressed in terms of concept definitions. However, almost all implemented systems assume that such definitions are acyclic, i.e. in the definition of concept C no reference, direct or indirect, to C itself may occur (see [Neb91] for a discussion on terminological cycles). It is well known that any reasoning process over knowledge bases comprising an acyclic terminology can be reduced to a reasoning process over a knowledge base with an empty terminology, in particular by substituting in the assertions every concept name with the corresponding definition (see [Neb90b] for a discussion of this technique and its computational properties). For the above reason, in our analysis we do not take into account terminologies and, therefore, we conceive a knowledge base as just a set of $\mathcal{L}$-assertions.

Example 2.2 Let Σ_1 be the following $\mathcal{ALEO}$-knowledge base:

$$\begin{aligned}\Sigma_1 = \{&\exists\texttt{FRIEND}.\{\texttt{susan},\texttt{peter}\}(\texttt{john}),\\ &\forall\texttt{FRIEND}.\texttt{Married}(\texttt{john}),\\ &\neg\texttt{Married}(\texttt{peter})\}\end{aligned}$$

It is easy to see that Σ_1 is satisfiable. Moreover, some non-trivial conclusion can be drawn from Σ_1. For example, we can prove that $\Sigma_1 \models$`Married(susan)` and $\Sigma_1 \models$`FRIEND(john,susan)`. In fact, due to the last two assertions, Peter cannot be a friend of John. Therefore, according to the first assertion, the friend of John must be Susan and, consequently, she must be married, i.e. both `FRIEND(john,susan)` and `Married(susan)` are logically implied by Σ_1. $\square$

2.3 Reasoning Services

There are several reasoning services to be provided by knowledge bases expressed by means of concept languages. Some of them are concerned with reasoning about concept expressions and they fall under the name of TBox-reasoning. Some others require to reason on a set of assertions, they are called ABox-reasoning. In this paper, we are mainly interested in the following basic reasoning tasks (see [BBH$^+$91] for a list of more complex reasoning services).

Definition 2.3 *Let $\mathcal{L}$ be any concept language. Then, given an $\mathcal{L}$-knowledge base Σ, two $\mathcal{L}$-concepts C and D, and an individual a, we call:*

- concept satisfiability*, written as $C \not\equiv \bot$, the problem of checking whether C is satisfiable;*
- terminological subsumption *(or simply* subsumption*) the problem of checking whether C is subsumed by D;*
- knowledge base satisfiability*, written $\Sigma \not\models$, the problem of checking whether Σ is satisfiable;*
- instance checking *the problem of checking whether $\Sigma \models C(a)$;*
- hybrid subsumption *is the problem of checking whether $\Sigma \models C \sqsubseteq D$;*

Concept satisfiability and terminological subsumption are TBox-reasoning problems, whilst all the others are ABox-reasoning problems. The importance of TBox-reasoning has been stressed by several authors (see for example [Neb90a]). Knowledge base satisfiability is used for verifying whether the information contained in a knowledge base is coherent. Hybrid subsumption is the problem of checking whether a subsumption relation holds with respect to the set of models of a knowledge base[2]. Finally, instance checking is used to check whether an individual is an instance of a concept; it can be considered the central reasoning task for retrieving information on individuals in the knowledge-base. In fact, instance checking is a basic tool for more complex reasoning problems. For example, the problem of retrieving all the individuals which are instances of a concept can be easily reduced to instance checking.

[2] In Section 3 we provide a discussion of the relation between the two types of subsumption. Notice that it is possible to consider the hybrid version of concept satisfiability too; i.e. the satisfiability w.r.t. the set of models of a knowledge base.

The five reasoning problems mentioned above are not independent of each other. In particular, consider a knowledge base Σ, two concepts C, D, and an individual a not appearing in Σ, C, D, the following relations hold:

$$
\begin{aligned}
C \not\equiv \bot &\iff C \not\sqsubseteq \bot && (1)\\
C \not\equiv \bot &\iff \{C(a)\} \not\models && (2)\\
C \sqsubseteq D &\iff \{C(a)\} \models D(a) && (3)\\
C \sqsubseteq D &\iff \emptyset \models C \sqsubseteq D && (4)\\
\Sigma \not\models &\iff \Sigma \not\models \top \sqsubseteq \bot && (5)\\
\Sigma \not\models &\iff \Sigma \not\models \bot(a) && (6)
\end{aligned}
$$

It follows that concept satisfiability can be reduced to both the complement of subsumption (1) and knowledge base satisfiability (2). Subsumption can be reduced to instance checking (3) and hybrid subsumption (4). Finally, knowledge base satisfiability can be reduced to the complement of hybrid subsumption (5) and to the complement of instance checking (6).

These relations are shown in Figure 1, where the simple arrows mean a reduction from a problem to the other and the marked arrows mean a reduction from a problem to the complement of the other problem (the meaning of the dashed arrows is explained below).

Relations (1-6) are stated for pure languages. In fact, they do not necessarily hold when $\mathcal{O}$ is used. For example, the following counterexample shows that relation (2) doesn't hold for mixed languages. Consider the concept $\{b, c, d\}$; it is trivially satisfiable. The knowledge base $\{\{b, c, d\}(a)\}$ instead is not satisfiable (due to the Unique Name Assumption) contradicting relation (2). In Section 6 we discuss the relationship between the reasoning tasks in mixed languages.

For languages with the constructor for expressing the complement of concepts the following other relations hold too:

$$
\begin{aligned}
C \sqsubseteq D &\iff (C \sqcap \neg D) \equiv \bot && (7)\\
\Sigma \models C(b) &\iff \Sigma \cup \{\neg C(b)\} \models && (8)\\
\Sigma \models C \sqsubseteq D &\iff \Sigma \cup \{C \sqcap \neg D(a)\} \models && (9)
\end{aligned}
$$

Therefore, subsumption can be reduced to the complement of concept satisfiability (7). Instance checking and hybrid subsumption can be reduced to

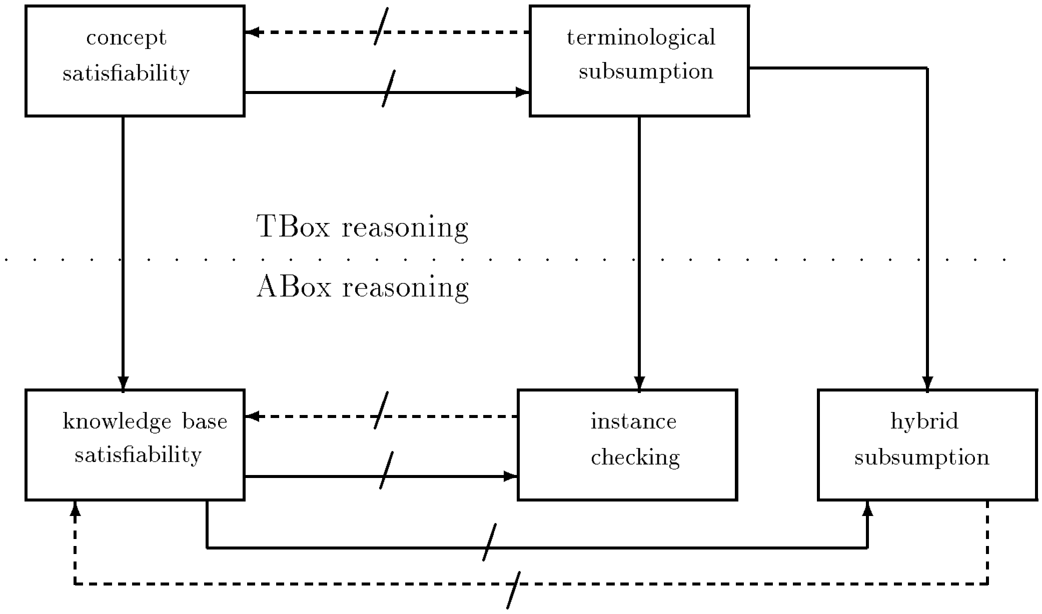

Figure 1: Reductions between reasoning tasks

the complement of knowledge base satisfiability (8, 9). These relations are shown by the dashed arrows in Figure 1

3 Reasoning with Mixed Languages

In this section we investigate the effects of the use of mixed languages in the reasoning process. In particular, in Sections 3.1–3.7 we give an overview of the most relevant issues related to $\mathcal{O}$ and we give an intuition of how these issues can make the reasoning process more complex than in the corresponding pure language (without $\mathcal{O}$), called the *underlying language*. In Section 3.8, we consider the issues related to $\mathcal{B}$. We refer to Section 5 for a general technique for reasoning with $\mathcal{O}$ and $\mathcal{B}$, and to Section 6 for an extensive study of its complexity in relation with the complexity of the underlying language.

3.1 Implicit Assertions

One of the characteristics of concept languages is the ability of describing incomplete knowledge. In particular, by means of existential quantification, it is possible to express information about objects that exist but whose identity is not known by the knowledge base. With regards to these unknown objects, it is also possible to state their membership to some concept. In particular, when $\mathcal{O}$ is used, it is possible to state the membership of an unknown object to a set of individuals. A consequence of this, is that the unknown object is bound to be one of the individuals of the set.

For example, consider the following $\mathcal{ALEO}$-assertion:

$$\exists R.(A \sqcap \{a\})(d).$$

It explicitly states the membership of d in $\exists R.(A \sqcap \{a\})$, but it also implicitly states that a must be in the extension of A. In fact, it says that there exists an object in $A \sqcap \{a\}$, therefore this object must be a and it must be in the extension of A, that is equivalent to stating the assertion $A(a)$.

In the above example we have considered a collection formed by a single individual. If we consider collection formed by more the one element then the resulting implicit assertion can be disjunctive. For example, if we state the existence of an object in the concept $A \sqcap \{a, b\}$, then the resulting implicit assertion is $A(a) \vee A(b)$.

The following example shows how the implicit assertions play a role in the semantics of a concept.

Example 3.1 Consider the following $\mathcal{ALEO}$-concept formed by a conjunction of three existential quantifications

$$\exists R.(A \sqcap \{a, b\}) \sqcap \exists R.(\neg A \sqcap \{a\}) \sqcap \exists R.(\neg A \sqcap \{b\}).$$

Suppose that we want to check its satisfiability. The standard approach (e.g. [SSS91, BH91]) to this problem is to separately check for the satisfiability of the three concepts involved in the existential quantifications, namely $A \sqcap \{a, b\}$, $\neg A \sqcap \{a\}$, and $\neg A \sqcap \{b\}$. It is easy to see that this technique fails to recognize that the whole concept is unsatisfiable. In fact, although each of the conjuncts is separately satisfiable, the conjunction of their implicit assertion (i.e. $A(a) \vee A(b)$, $\neg A(a)$, and $\neg A(b)$) is unsatisfiable.

3.2 Mixing terminological and assertional knowledge

Another important characteristic of concept languages is that the reasoning process in the terminological component is in general not influenced by the assertional knowledge. More precisely, the following theorem holds for a large class of languages (in [Neb90a], here simplified slightly from the original version):

Theorem 3.2 ([Neb90a]) *Given a* satisfiable *knowledge base* Σ *then for every pair of concepts* C, D*:*

$$\Sigma \models C \sqsubseteq D \quad \Longleftrightarrow \quad C \sqsubseteq D.$$

The above theorem states that hybrid subsumption can be trivially reduced to terminological subsumption (plus knowledge base satisfiability). In other words, it says that the knowledge base, if satisfiable, plays no role in the reasoning about concepts. The above property is crucial for the efficiency of reasoning in concept-description-based knowledge representation systems. In fact, it allows for the maintenance of a static hierarchy of concepts; which is not influenced by the evolution of the knowledge base.

Unfortunately, such nice property does not hold when the language includes $\mathcal{O}$, as shown in the following example.

Example 3.3 Let $\Sigma_2 = \{A(a), A(b)\}$. It is easy to see that

$$\forall R.\{a,b\} \not\sqsubseteq \forall R.A.$$

In fact, given an interpretation $\mathcal{I}$ such that $R^{\mathcal{I}} = \{(d, a^{\mathcal{I}})\}$ and $A^{\mathcal{I}} = \emptyset$, then $d \in (\forall R.\{a,b\})^{\mathcal{I}}$ and $d \notin (\forall R.A)^{\mathcal{I}}$. On the other hand

$$\Sigma_2 \models (\forall R.\{a,b\} \sqsubseteq \forall R.A).$$

That is because, in every model of Σ_2 all the objects related only with a and b by means of R are obviously related only to object in A

For the above reason, when $\mathcal{O}$ is used, it is necessary to make the distinction between the two notions of subsumption. This is also the reason why we call mixed the languages with $\mathcal{O}$ and pure the languages satisfying the property stated in Theorem 3.2.

3.3 Abstraction

Abstraction is a well known mechanism in reasoning about individuals in concept-based systems. It consists in retrieving all the assertions relevant to a given individual a and collecting them into a single concept. Such concept has the property of been the most specific concept expressible in the language such that the individual a is an instance of. For this reason it is generally indicated by $MSC(a)$.

Abstraction, together with subsumption, allows to perform instance checking. In fact, given the problem of checking whether $\Sigma \models C(a)$, with the abstraction process, we compute $MSC(a)$ and, after that, instance checking can be performed by checking whether C subsumes $MSC(a)$. This technique, called Abstraction/Subsumption, has been broadly exploited in actual systems (see [Kin90, QK90, Neb90b]).

However, the problem of exploiting this technique is that, in general, it is not possible to completely fit the information relevant to an individual into a single concept of the language. For example, given the following $\mathcal{ALE}$-knowledge base $\Sigma = \{R(a,a), B(a)\}$, the abstraction for a in $\mathcal{ALE}$ returns $MSC(a) = B \sqcap \exists R.B$.

In $MSC(a)$, the information that the individual related to a is exactly a itself is lost. In general, any time an individual is referred twice in the knowledge base, the connection between the two occurrences may be lost.

For this reason, the algorithms for instance checking based on abstraction are, in general, incomplete. For instance, in the above example, the Abstraction/Subsumption technique fails to draw the conclusion that $\Sigma \models \exists R.\exists R.B(a)$.

As pointed out in [MB92] there are even other drawbacks about using the Abstraction/Subsumption technique. However they are out of the scope of this paper.

Nevertheless, if the language includes $\mathcal{O}$ it is possible to make a *lostless* abstraction. In the previous example, if the language is $\mathcal{ALEO}$, the abstraction for a gives $MSC(a) = \{a\} \sqcap B \sqcap \exists R.\{a\}$, and it is easy to see that the inference $\Sigma \models \exists R.\exists R.B(a)$ is captured since $\{a\} \sqcap B \sqcap \exists R.\{a\} \sqsubseteq \exists R.\exists R.B$ holds.

It follows that the use of $\mathcal{O}$ gives the possibility to complete reasoning using the Abstraction/Subsumption technique (see [DE92] for a detailed discussion on this topic).

3.4 Epistemic Operator

In [DLN$^+$92], the addition of an epistemic operator $\mathbf{K}$ to concept languages is investigated. Among other things, it is considered the possibility of enhancing the language used to query a knowledge by means of that operator. In particular, the constructor $\mathbf{K}C$ is inserted in such query language, called $\mathcal{ALCK}$. Ruoghly speaking[3], the concept $\mathbf{K}C$ denotes the set of individuals such that the knowledge base *knows* that are in the extension of C.

It that paper it is argued that a concept of the form $\mathbf{K}C$ is equivalent to the concept $\{a_1, \ldots, a_n\}$, where $a_1, \ldots, a_n$ are exactly the individuals for which $\Sigma \models C(a_i)$ holds. For this reason, as shown in [DLN$^+$93], reasoning with $\mathcal{O}$ turned out to be a basic tool for reasoning with $\mathbf{K}$.

3.5 Number Restrictions and Complements

The ability offered by the constructor $\mathcal{O}$ to express concepts of a fixed extension gives also the possibility to express implicit number restrictions on the roles. If, for example, we assert the membership of an individual d to the concept $\forall R.\{a, b, c\}$ it implies that d is also in the extension of $(\leq 3\, R)$. For this reason, an inconsistency can be generated by the conjunction of two concept of the form $\forall R.\{a_1, \ldots, a_n\}$ and $(\geq m\, R)$, in the case $m > n$.

As pointed out in [BMPS$^+$91], using $\mathcal{O}$ it is possible to express the complement of a concept with respect to another concept. Let clarify this point by means of the following example (which is a slight modification of the example in [BMPS$^+$91, page 44]).

Example 3.4 Consider the following three concepts

$$
\begin{aligned}
C &= \forall R.\{a, b\} \sqcap (\geq 1\, R) \sqcap (\leq 1\, R), \\
D_1 &= \forall R.\{a\} \sqcap (\geq 1\, R), \\
D_2 &= \forall R.\{b\} \sqcap (\geq 1\, R).
\end{aligned}
$$

The concept C describes the individuals which have a single filler for the role R and such filler is a or b; D_1 and D_2 describe the individuals which have a single filler for the role R and such fillers are respectively a and b. Therefore, we have that $D_1 \sqcap D_2 = \bot$ and $D_1 \sqcup D_2 = C$, i.e. D_1 and D_2 are complementary with respect to C. □

[3]We refer to the cited paper for a precise definition of the semantics of $\mathbf{K}$.

3.6 Logical Connectives

Example 3.1 shows that, exploiting the implicit assertion, it is possible to express logical connectives between assertions. The explicit use of such connectives is usually not allowed in concept based systems. In fact assertions like $R(a,b) \vee C(b)$ are not allowed and only atomic assertions of the form $C(a)$ and $R(a,b)$ are generally considered[4].

On the other hand, if the concept language includes $\mathcal{O}$ and $\mathcal{C}$, all the connectives can be simulated by atomic assertions. In order to show this point, we consider the language $\mathcal{ALCO}$ and we call *complex* $\mathcal{ALCO}$-knowledge base a knowledge base obtained combining atomic assertions with the usual propositional connectives $\vee$, $\wedge$, and $\sim$. We assume that complex $\mathcal{ALCO}$-knowledge bases are provided with the standard semantics for connectives.

We now prove that every complex $\mathcal{ALCO}$-knowledge base Π can be transformed in a simple (i.e. as defined in Section 2) knowledge base $\Sigma = \Phi(\Pi)$ such that Π is satisfiable if and only if Σ is satisfiable.

Consider a generic assertion α in Π. For the sake of simplicity, we suppose that α is in conjunctive normal form (CNF); however it is possible to show that our results hold for general formulae as well. Therefore, α has the form $c_1 \wedge \cdots \wedge c_m$, where each c_i has the form $l_1 \vee \cdots \vee l_n$, and each l_j has the form p or $\sim p$ (where p has either the form $C(a)$ or $R(a,b)$).

As a notation, if α has the form $C(a)$, we call C_α the concept C involved in α and a_α the individual a. Furthermore, we assign to each clause in each assertion an individual i_k that does not appear in Π; where k is an integer that takes a different value for each clause. The transformation Φ is then defined by the following rules (where Q is a role not appearing in Π):

$$
\begin{aligned}
\Phi(R(a,b)) &= \exists R.\{b\}(a) && (10)\\
\Phi(C(a)) &= C(a) && (11)\\
\Phi(\sim p) &= \neg\Phi(p) && (12)\\
\Phi(l_1 \vee \cdots \vee l_n) &= \exists Q.(C_{\Phi(l_1)} \sqcap \{a_{\Phi(l_1)}\}) \sqcup \cdots \sqcup \\
&\quad\; \exists Q.(C_{\Phi(l_n)} \sqcap \{a_{\Phi(l_n)}\})(i_k) && (13)\\
\Phi(c_1 \wedge \cdots \wedge c_m) &= \{\Phi(c_i) \mid i = 1, \ldots, m\} && (14)\\
\Phi(\Pi) &= \bigcup_{\alpha \in \Pi} \Phi(\alpha) && (15)
\end{aligned}
$$

[4]except for KRIPTON [BPGL85], which allows all the propositional connectives.

Example 3.5 Consider the following complex $\mathcal{ALCO}$-knowledge base Π_1:

$$\Pi_1 = \{\exists R.D(a) \vee R(b,c),\ \sim R(a,b)\}$$

Applying the reduction Φ, we obtain:

$\Phi(\Pi_1) = \{\exists Q.(\exists R.D \sqcap \{a\}) \sqcup \exists Q.((\exists R.\{c\}) \sqcap \{b\})(i_1),\ \exists Q.((\neg \exists R.\{b\}) \sqcap \{a\})(i_2)\}$

Lemma 3.6 *A complex $\mathcal{ALCO}$-knowledge base Π is satisfiable iff the $\mathcal{ALCO}$-knowledge base $\Phi(\Pi)$ is satisfiable.*

Proof. We prove the claim by showing that all the rules 10-15 are satisfiability preserving. In particular, it is easy to see that for each rule, but rule (13), $\Phi(\Pi)$ is trivially equivalent to Π. Therefore in order to prove the claim it is sufficient to show that $\Pi \cup \{l_1 \vee \cdots \vee l_n\}$ is satisfiable iff $\Pi \cup \{\Phi(l_1 \vee \cdots \vee l_n)\}$ is satisfiable.

"$\Rightarrow$" Suppose $\Pi \cup \{\Phi(l_1 \vee \cdots \vee l_n)\}$ satisfiable. Let $\mathcal{I}$ be a model of $\Pi \cup \{\Phi(l_1 \vee \cdots \vee l_n)\}$. Since $\mathcal{I}$ satisfies $\Phi(l_1 \vee \cdots \vee l_n)$ there must be $j \in \{1, \ldots, n\}$ such that $\mathcal{I} \models \exists Q.C_{\Phi(l_j)} \sqcap \{a_{\Phi(l_j)}\}(i_k)$. Hence, in $\Delta^{\mathcal{I}}$ there must exist an element d such that $(i_k^{\mathcal{I}}, d) \in Q^{\mathcal{I}}$ and $d \in (C_{\Phi(l_j)} \sqcap \{a_{\Phi(l_j)}\})^{\mathcal{I}}$ Therefore $d = a_{\Phi(l_j)}$ and consequently $a_{\Phi(l_j)} \in C_{\Phi(l_j)}$. It follows that $\mathcal{I}$ is a model of l_j, therefore it is a models of $l_1 \vee \cdots \vee l_n$ too.

"$\Leftarrow$" Suppose $\Pi \cup \{l_1 \vee \cdots \vee l_n\}$ satisfiable. Let $\mathcal{I}$ be a model of $\Pi \cup \{l_1 \vee \cdots \vee l_n\}$. There must exist one $j \in \{1, \ldots, n\}$ such that $\mathcal{I} \models l_j$. Let $\mathcal{I}'$ be the interpretation equal to $\mathcal{I}$ except that $({i_k}^{\mathcal{I}'}, {a_{\Phi(l_j)}}^{\mathcal{I}'}) \in Q^{\mathcal{I}'}$. Since i_k does not appear in $\Pi \cup \{l_1 \vee \cdots \vee l_n\}$, it follows that $\mathcal{I}'$ is a model of Π. Since $a^{\mathcal{I}}_{\Phi(l_j)} \in C^{\mathcal{I}}_{\Phi(l_j)}$, it follows that $a^{\mathcal{I}'}_{\Phi(l_j)} \in C^{\mathcal{I}'}_{\Phi(l_j)}$, and therefore $a^{\mathcal{I}'}_{\Phi(l_j)} \in (C_{\Phi(l_j)} \sqcap \{a^{\mathcal{I}'}_{\Phi(l_j)}\})$. Moreover $({i_k}^{\mathcal{I}'}, {a_{\Phi(l_j)}}^{\mathcal{I}'}) \in Q^{\mathcal{I}'}$ holds (by construction of $\mathcal{I}'$). In conclusion, $\mathcal{I}'$ is a model of $\Pi \cup \{\Phi(l_1 \vee \cdots \vee l_n)\}$ and therefore $\Pi \cup \{\Phi(l_1 \vee \cdots \vee l_n)\}$ is satisfiable.

$\square$

Theorem 3.7 *Knowledge base satisfiability in $\mathcal{ALCO}$ and knowledge base satisfiability of complex $\mathcal{ALCO}$-knowledge bases are problems polynomially reducible to each other.*

Proof. Knowledge base satisfiability is trivially reducible to complex knowledge base satisfiability, being a particular case of it. The other direction is proved by Lemma 3.6 and the observation that Φ is polynomial. $\square$

3.7 Discussion on Reasoning with $\mathcal{O}$

The above (not exhaustive) list of issues helps in understanding why reasoning with $\mathcal{O}$ is generally hard. This hardness has a twofold explanation: on one side, it is related to the implicit disjunction carried by the use of sets with more then one object. On the other side, it is due to the identification of unknown objects with individuals.

It is well known, that concept-based assertions can be translated into first-order formulae. The above explanation can be clarified looking at the translation in first-order formulae of assertions with $\mathcal{O}$. For example, an assertion of the form

$$\exists R.\{a_1, \ldots, a_n\}(b)$$

is translated into the formula

$$R(b, x) \wedge (x = a_1 \vee \cdots \vee x = a_n).$$

This formula explicitly contains both disjunction and equality; they can be easily recognized as the causes of the hardness of reasoning with $\mathcal{O}$.

It is important to note how, in some cases, the standard semantics gives results which are hard to be intuitively understood. In particular, the implicit assertion are difficult to be recognized and their role in the semantics can be mistaken or overlooked.

These two facts together, i.e. hardness and lack of intuition, explain why $\mathcal{O}$ is usually treated in a non-standard way in the actual systems, as shown in Section 4.

3.8 Reasoning with $\mathcal{B}$

As shown in Section 2, a concept of the form $R : a$ is equivalent to the concept $\exists R.\{a\}$. Hence $\mathcal{B}$ can be viewed as a limited form of the combination of $\mathcal{E}$ and $\mathcal{O}$.

Reasoning with $\mathcal{E}$ has been proved to be generally hard (see [DHL$^+$92, DLNS92]). However, the hardness of $\mathcal{E}$ is related to the possibility of nesting arbitrary numbers of existential quantification. Since $\mathcal{B}$ does not offer the possibility of nesting, the issues related to $\mathcal{E}$ do not regard $\mathcal{B}$.

Regarding the relation with $\mathcal{O}$, the main difference between $\mathcal{O}$ and $\mathcal{B}$ is that $\mathcal{B}$ involves always a single individual. Therefore, all the issues related to the use of sets with more then one individual are not concerned to $\mathcal{B}$.

The other difference between $\mathcal{O}$ and $\mathcal{B}$ is that the set of individuals involved in $\mathcal{B}$ is always in the scope of an existential quantification. Even though this is a obvious limitation of the use of single-element sets, we now show that the issues related to the use of single-element sets are pertinent to $\mathcal{B}$ too.

For instance, using $\mathcal{B}$, it is possible to express implicit assertions. As an example, it is easy to see that the assertion

$$(\forall R.C) \sqcap (R : b)(a)$$

carries the implicit assertion $C(b)$.

As said before, since $\mathcal{B}$ involves only single-element sets, disjunctive assertions cannot be expressed with $\mathcal{B}$. However, as shown in Section 3.6, disjunctive implicit assertions can be obtained anyway by using single-element sets together with $\mathcal{U}$. For example the following assertion

$$((\forall R.A) \sqcap (R : a)) \sqcup ((\forall R.B) \sqcap (R : b))(c)$$

carries the implicit assertion $A(a) \vee B(b)$.

Moreover, Theorem 3.2, that ensures the independence of reasoning about concepts from the knowledge base, does not hold also in presence of $\mathcal{B}$, as shown by the following counterexample.

Example 3.8 Let $\Sigma = \{A(a)\}$. Then

$$(R : a) \not\sqsubseteq \exists R.A.$$

Conversely, we have that

$$\Sigma \models ((R : a) \sqsubseteq \exists R.A).$$

□

In conclusion, reasoning with $\mathcal{B}$ has all the characteristic of reasoning with $\mathcal{O}$ related to the presence of individual. On the other hand, the issues related to the implicit disjunction of $\mathcal{O}$ are obviously not pertinent to $\mathcal{B}$. However, when the language is supported with explicit disjunction ($\mathcal{U}$), the use of $\mathcal{B}$ becomes complex like the use of $\mathcal{O}$.

4 How actual systems deal with mixed languages

In this section we briefly describe the methods chosen by the implementors of the actual systems for dealing with $\mathcal{O}$ and $\mathcal{B}$, and with individuals in the concept descriptions in general. For this purpose we have chosen to describe two systems, namely $CLASSIC$ and BACK. A more detailed description of them can be found in [BMPS+91, BPS92] and [QK90] respectively.

In $CLASSIC$, individuals are treated with a non-standard semantics. The reason why the $CLASSIC$ designers have left the standard semantics is mostly related to the drawbacks described in Section 3 (in particular in Sections 3.1 and 3.2), and to the computational intractability of subsumption (see Section 6), which, in their opinion (see [BPS92]), is not relegated only to few non-practical worst cases.

Roughly speaking, the individuals appearing in concept descriptions are interpreted as primitive disjoint concepts, i.e. as subsets of the domain, instead of as single elements of it. This semantics eliminates the effects of implicit assertions. In fact the existence of an object in the concept $C \sqcap \{a\}$ does not tell that $a^{\mathcal{I}}$ is in $C^{\mathcal{I}}$ but only that $a^{\mathcal{I}}$ and $C^{\mathcal{I}}$ intersect each other. The fact that $a^{\mathcal{I}}$ and $C^{\mathcal{I}}$ intersect each other does not guarantee that $\{a^{\mathcal{I}}\} \subseteq C^{\mathcal{I}}$ and does not exclude the possibility that even $a^{\mathcal{I}}$ and $\neg C^{\mathcal{I}}$ intersect each other.

Moreover, in $CLASSIC$, the assertions on the individuals are not taken into account while reasoning with concepts. In other words, even when a knowledge base is involved the type of subsumption considered is always the terminological one. This semantics is weaker than the standard one, in fact it fails to draw several conclusions that are entailed in the standard semantics.

The following example is taken from [BPS92, page 13]. The names are modified w.r.t. the original version.

Example 4.1 Let Σ_3 be the following $CLASSIC$-knowledge base:

$$\Sigma_3 = \{\forall\texttt{FRIEND}.\{\texttt{susan}\}(\texttt{john}),\ \texttt{Married}(\texttt{susan})\}$$

The proposed semantics fails to draw the correct conclusion that $\Sigma_3 \models \forall\texttt{FRIEND}.\texttt{Married}(\texttt{john})$.

In BACK, $\mathcal{O}$ and $\mathcal{B}$ are not allowed. However, in BACK it is possible to express collections of elements, but these elements, called *attributes*, belong to an alphabet disjoint from the alphabet of the individuals. Moreover, the domain of interpretation of the concepts is disjoint form the domain of interpretation of the attributes. A collection of attributes is not considered a concept and it is not allowed to be in conjunction with any concept, but it can appear only in the range of the quantification of a role.

This treatment avoids the reasoning complications of Section 3. It is simple and efficient; in fact reasoning with collections of attributes in BACK is just a matter of computing intersection, union, and difference between sets. The possible usefulness of sets of non-individual elements is argued also in [Bra92], where they are called *Host Individuals*. However, they miss the expressive power of the full use of $\mathcal{O}$.

5 A technique for complete reasoning

The technique we present here is a refinement of the tableaux calculus for first order logic [BM77], and is employed in [DHL+92, DLNN91, DLNS92, HNSS90, SSS91, Hol90] both for the design of algorithms for the various reasoning tasks, and for studying their computational properties. The calculus in this paper is a straightforward extension, to deal with $\mathcal{O}$ and $\mathcal{B}$, of the calculus in the cited papers.

For the sake of simplicity, we restrict our attention to the language $\mathcal{ALCO}$ ($\mathcal{ALCOB}$) and its sublanguages. The calculus can be easily extended to other languages (following the line of [DLNN91]).

The calculus operates on constraints consisting of individuals, variables, concepts and roles. Concepts are assumed to be simple, i.e. they contain complements only of one of the forms $\neg\{a_1, \ldots, a_n\}$ or $\neg A$, where A is a primitive concept. Arbitrary concepts can be rewritten into equivalent simple concepts in linear time[5].

[5] the concept $\neg(R : a)$, being equivalent to $\neg\exists R.\{a\}$, is rewritten as $\forall R.\neg\{a\}$.

Consider an alphabet of variable symbols $\mathcal{V}$. The elements of $\mathcal{V}$ are denoted by the lower case letters x, y, z. From this point on, we use the term *object* as an abstraction for individual and variable (i.e. an object is an element of $\mathcal{A} \cup \mathcal{V}$). Objects are denoted by the symbols s, t and, as in the previous sections, individuals are denoted by a, b.

A *constraint* is a syntactic entity of one of the forms

$$s\colon C, \quad sRt,$$

where C is a concept and R is a role. Given an interpretation $\mathcal{I}$, we define an $\mathcal{I}$*-assignment* α as a function that maps every variable in $\mathcal{V}$ to an element of $\Delta^{\mathcal{I}}$ (not necessarily injectively), and every individual to its interpretation (i.e. $\alpha(a) = a^{\mathcal{I}}$ for $a \in \mathcal{A}$).

A constraint of the form $s\colon C$ is satisfied by the pair $(\mathcal{I}, \alpha)$ if $\alpha(s) \in C^{\mathcal{I}}$. A constraint of the form sRt is satisfied by $(\mathcal{I}, \alpha)$ if $(\alpha(s), \alpha(t)) \in R^{\mathcal{I}}$.

A *constraint system* is a finite nonempty set of constraints. A constraint system S is *satisfiable* if there is an interpretation $\mathcal{I}$ and an $\mathcal{I}$-assignment α such that $(\mathcal{I}, \alpha)$ satisfies every constraint in S.

A knowledge base Σ can be translated into a constraint system S_Σ by replacing every membership assertion $C(a)$ (resp. $R(a, b)$) with the constraint $a\colon C$ (resp. aRb).

It is easy to see that all the reasoning tasks considered in this paper can be reduced to the satisfiability of a constraint system. In fact, a concept C is satisfiable if and only if $\{x\colon C\}$ is satisfiable, C is subsumed by D if and only if $\{x\colon C \sqcap \neg D\}$ is satisfiable, and Σ is satisfiable if and only if S_Σ is satisfiable. Furthermore, $\Sigma \models C(a)$ if and only if the constraint system $S_\Sigma \cup \{a\colon \neg C\}$ is unsatisfiable and $\Sigma \models C \sqsubseteq D$ if and only if $S_\Sigma \cup \{x\colon C \sqcap \neg D\}$ is unsatisfiable.

In order to check a constraint system S for satisfiability, our technique adds constraints to S until either a contradiction is generated or an interpretation satisfying it can be obtained from the resulting system. Constraints are added on the basis of a suitable set of so-called *propagation rules*.

If a is an individual, then we denote by $S[x/a]$ the constraint system obtained from S by substituting every occurrence of the variable x with the individual a.

The *propagation rules* are:

1. $S \rightarrow_\sqcap \{s\colon C_1,\ s\colon C_2\} \cup S$

   if $s\colon C_1 \sqcap C_2$ is in S, and $s\colon C_1$ and $s\colon C_2$ are not both in S

2. $S \rightarrow_{\sqcup} \{s\colon D\} \cup S$

   if $s\colon C_1 \sqcup C_2$ is in S, neither $s\colon C_1$ nor $s\colon C_2$ is in S, and $D = C_1$ or $D = C_2$

3. $S \rightarrow_{\exists} \{sRx,\ x\colon C\} \cup S$

   if $s\colon \exists R.C$ is in S, there is no t such that both sRt and $t\colon C$ are in S and x is a new variable.

4. $S \rightarrow_{\forall} \{t\colon C\} \cup S$

   if $s\colon \forall R.C$ is in S, sRt is in S, and $t\colon C$ is not in S

5. $S \rightarrow_{[/]} S[x/a_i]$

   if $x\colon \{a_1, \ldots, a_n\}$ is in S and $i \in \{1, \ldots, n\}$

6. $S \rightarrow_{:} \{sRa\} \cup S$

   if $s\colon (R : a)$ is in S and sRa is not in S

We call the rules $\rightarrow_{\sqcup}$ and $\rightarrow_{[/]}$ *nondeterministic* rules, because they can be applied in more than one way. All the other rules are called *deterministic* rules. A constraint system is said to be *complete* if no propagation rule applies to it. Any complete constraint system obtained from a constraint system S by applying the above rules is called a *completion* of S. Notice that, due to the presence of the nondeterministic rules, more than one completion can be obtained starting from a constraint system.

The following theorem ensures the correctness of the rules.

Theorem 5.1 (Correctness) *Let S be a constraint system. Then:*

1. *If S' is obtained from S by the application of a deterministic rule, then S is satisfiable if and only if S' is satisfiable.*

2. *If S' is obtained from S by the application of the nondeterministic rule, then S is satisfiable if S' is satisfiable. Furthermore, if the nondeterministic rule applies to S, then it can be applied in a way that it yields a constraint system S' such that S' is satisfiable if S is satisfiable.*

Proof. The correctness of rules 1–4 is stated in [Hol90], the extension to rules 5 and 6 is straightforward. $\square$

A *clash* is a set of constraints of one of the following forms

1. $\{s\colon \bot\}$,
2. $\{s\colon A, s\colon \neg A\}$,
3. $\{a\colon \{a_1, \ldots, a_n\}\}$ with $a \neq a_i$ for all $i = 1, \ldots, n$,
4. $\{a\colon \neg\{a_1, \ldots, a_n\}\}$ with $a = a_i$ for some $i = 1, \ldots, n$.

The following theorem ensures the termination of the calculus. It is a straightforward extension of the corresponding theorem in [Hol90].

Theorem 5.2 *Let S be a constraint system. Then:*

1. *If S is complete then it is satisfiable if and only if it contains no clash.*
2. *S has a finite number of completions and every completion of S has finite size*

Due to the above results, the calculus can be turned in an correct and terminating procedure, thus providing an effective method for carrying out the various reasoning services.

6 Complexity of reasoning with mixed languages

We start this section analyzing the relationship between the complexity of the various reasoning tasks in mixed languages. Referring to Figure 1, we show that some other arcs can be drawn for such languages.

In the following sections, we investigate on the complexity of the above problems in the specific languages. To this aim, we consider various languages that do not use $\mathcal{O}$ and $\mathcal{B}$ and the corresponding languages obtained by adding them. In particular we focus on the pure languages $\mathcal{ALC}$, $\mathcal{ALE}$, and $\mathcal{AL}$, which are a good representative of the various degrees of expressiveness (and complexity), and we achieve some complexity results on the corresponding languages with $\mathcal{O}$ and $\mathcal{B}$. In those sections, we concentrate on the terminological subsumption problem. Results for the other reasoning tasks easily follow.

6.1 Relationship between Reasoning Tasks in Mixed Languages

Before starting, we need one round of definitions. Given a concept C, we call *subconcept* of C any substring of C (including C itself) that is a concept, according to the syntax rules. Notice that, if $|C|$ denotes the size of C, then the number of subconcepts of C is bounded by $|C|$. Moreover, we call $\mathcal{A}_C$ the set of individuals appearing in C.

In a similar way, we call subconcept of a knowledge base Σ any subconcept of some concept in Σ and we call $\mathcal{A}_\Sigma$ the set of all the individuals appearing in Σ (either within the concepts or as the individual the assertion states the membership of). The number of subconcepts of Σ is bounded by $|\Sigma|$, the size of Σ.

We first discuss the validity for mixed languages of relations (1-6) stated in Section 2. It is easy to see that relations (1,4-6) hold for mixed languages as well. Conversely, relations (2,3) do not hold as they are, as shown by the example at hte end of Section 2 for relation (2). The intuitive reason is that relations (2,3) involve assertions of the form $C(a)$, and, if C involves some individuals, there might be an interaction between C and a.

Nevertheless, a variant of relations (2,3) hold for mixed languages, as stated by the following lemma.

Lemma 6.1 *Let $\mathcal{L}$ be a mixed language, C and D two $\mathcal{L}$-concepts, and b an individual not appearing in C and D:*

$$C \not\equiv \bot \iff \exists a \in (\mathcal{A}_C \cup \{b\}) \mid \{C(a)\} \not\models \tag{16}$$

$$C \sqsubseteq D \iff \forall a \in (\mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}) \mid \{C(a)\} \models D(a) \tag{17}$$

Proof.

(16)

"$\Leftarrow$" Suppose C satisfiable and $\forall a \in (\mathcal{A}_C \cup \{b\}) \mid \{C(a)\}$ is unsatisfiable. Since C is satisfiable, there exists an interpretation $\mathcal{I}$ such that $d \in \Delta^\mathcal{I}$ and $d \in C^\mathcal{I}$. The element d can be either the interpretation of an individual in $\mathcal{A}_C \cup \{b\}$ or not. We show that in both cases we reach a contradiction.

- $\exists a \in \mathcal{A}_C : d = a^{\mathcal{I}}$. Contradicts the hypothesis that $\{C(a)\}$ is unsatisfiable.
- $\forall a \in \mathcal{A}_C : d \neq a^{\mathcal{I}}$. Let $\mathcal{I}'$ be the interpretation equal to $\mathcal{I}$ except that $b^{\mathcal{I}'} = d$. Since b does not appear in C, it follows that the interpretation of b as no influence on the interpretation of C, and therefore $b^{\mathcal{I}'} \in C^{\mathcal{I}'}$. This contradicts the hypothesis that $\{C(b)\}$ is unsatisfiable.

"$\Rightarrow$" Suppose $\exists a \in (\mathcal{A}_C \cup \{b\}) \mid \{C(a)\}$ is satisfiable. It easily follows that C is satisfiable.

(17)

"$\Leftarrow$" Assume $C \sqsubseteq D$ and $\exists a \in (\mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}) : \{C(a)\} \not\models D(a)$. From $\{C(a)\} \not\models D(a)$ it follows that there exists an interpretation $\mathcal{I}$ such that $a^{\mathcal{I}} \in C^{\mathcal{I}}$, $a^{\mathcal{I}} \notin D^{\mathcal{I}}$. This contradicts the hypothesis that $C \sqsubseteq D$.

"$\Rightarrow$" Assume $\forall a \in (\cup \mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}) : \{C(a)\} \models D(a)$ and $C \not\sqsubseteq D$. From $C \not\sqsubseteq D$, it follows that there exists an interpretation $\mathcal{I}$ and an element d of $\Delta^{\mathcal{I}}$ such that $d \in C^{\mathcal{I}}$, and $d \notin D^{\mathcal{I}}$. The element d can be either the interpretation of an individual in $\mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}$ or not. We show that in both cases we reach a contradiction.

- Suppose that $d = a^{\mathcal{I}}$ for some $a \in \mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}$; then $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $a^{\mathcal{I}} \notin D^{\mathcal{I}}$ contradicting the hypothesis that $\{C(a)\} \models D(a)$.
- Suppose that $d \neq a^{\mathcal{I}}$ for all $a \in \mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}$ then let $\mathcal{I}'$ be the interpretation equal to $\mathcal{I}$ except that $b^{\mathcal{I}'} = d$. Since b does not appear in C and D, it follows that $b^{\mathcal{I}'} \in C^{\mathcal{I}'}$, $b^{\mathcal{I}'} \notin D^{\mathcal{I}'}$. This contradicts the hypothesis that $\{C(b)\} \models D(b)$.

□

Next lemma shows a bidirectional reduction between hybrid subsumption and instance checking in any language including $\mathcal{O}$ and states its correctness.

Lemma 6.2 *Let $\mathcal{L}$ be a concept language including $\mathcal{O}$, Σ an $\mathcal{L}$-knowledge base, C, D two $\mathcal{L}$-concepts, a an individual, and b an individual not in $\mathcal{A}_\Sigma \cup$*

$\mathcal{A}_C \cup \mathcal{A}_D$, *then:*

$$\Sigma \models D(a) \iff \Sigma \models \{a\} \sqsubseteq D \tag{18}$$

$$\Sigma \models C \sqsubseteq D \iff \forall a \in (\mathcal{A}_\Sigma \cup \mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}) : \Sigma \cup \{C(a)\} \models D(a) \tag{19}$$

Proof.

(18) If Σ is unsatisfiable, then both $\Sigma \models D(a)$ and $\Sigma \models \{a\} \sqsubseteq D$ trivially hold. Therefore, we can suppose Σ satisfiable. If Σ is satisfiable then it has at least one model. Let $\mathcal{I}$ be a generic model, obviously $a^\mathcal{I} \in D^\mathcal{I}$ if and only if $\{a^\mathcal{I}\} \subseteq D^\mathcal{I}$. The claim follows.

(19)

"$\Leftarrow$" Assume $\Sigma \models C \sqsubseteq D$ and $\exists a \in (\mathcal{A}_\Sigma \cup \mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}) : \Sigma \cup \{C(a)\} \not\models D(a)$. From $\Sigma \cup \{C(a)\} \not\models D(a)$ it follows that there exists an interpretation $\mathcal{I}$ such that $\mathcal{I}$ is a model of Σ, and $a^\mathcal{I} \in C^\mathcal{I}$, $a^\mathcal{I} \notin D^\mathcal{I}$. This contradicts the hypothesis that $\Sigma \models C \sqsubseteq D$.

"$\Rightarrow$" Assume $\forall a \in \mathcal{A}_\Sigma \cup \mathcal{A}_C \cup \mathcal{A}_D \cup \{b\} : \Sigma \cup \{C(a)\} \models D(a)$ and $\Sigma \not\models C \sqsubseteq D$. From $\Sigma \not\models C \sqsubseteq D$, it follows that there exists an interpretation $\mathcal{I}$ and an element d of $\Delta^\mathcal{I}$ such that $\mathcal{I}$ is a model of Σ, $d \in C^\mathcal{I}$, and $d \notin D^\mathcal{I}$. The element d can be either the interpretation of an individual in $\mathcal{A}_\Sigma \cup \mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}$ or not. We show that in both cases we reach a contradiction.

- Suppose that $d = a^\mathcal{I}$ for some $a \in \mathcal{A}_\Sigma \cup \mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}$; then $a^\mathcal{I} \in C^\mathcal{I}$ and $a^\mathcal{I} \notin D^\mathcal{I}$ contradicting the hypothesis that $\Sigma \cup \{C(a)\} \models D(a)$.
- Suppose that $d \neq a^\mathcal{I}$ for all $a \in \mathcal{A}_\Sigma \cup \mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}$ then let $\mathcal{I}'$ be the interpretation equal to $\mathcal{I}$ except that $b^{\mathcal{I}'} = d$. Since b does not appear in Σ, C, and D, it follows that $\mathcal{I}'$ is a model of Σ, and $b^{\mathcal{I}'} \in C^{\mathcal{I}'}$, $b^{\mathcal{I}'} \notin D^{\mathcal{I}'}$. This contradicts the hypothesis that $\Sigma \cup \{C(b)\} \models D(b)$.

□

Lemma 6.2 states that in order to solve hybrid subsumption, we make a linear number of instance checking tests, one for each individual appearing in the concepts and in the knowledge base plus one new. It is interesting to observe that, when $\mathcal{O}$ is not used, it is sufficient to make a single instance checking test, by considering only the new individual b. The following example shows that, on the contrary when $\mathcal{O}$ is used, this is not true. Let $\Sigma = \emptyset$, it is easy to see that $\Sigma \not\models (C \sqcup \{a\} \sqsubseteq C)$. However, due to the Unique Name Assumption, $\{C \sqcup \{a\}(b)\} \models C(b)$ holds. In fact, the latter relation holds for every individual except a, therefore it is necessary to include a in the set of individuals we consider.

Theorem 6.3 *Hybrid subsumption and instance checking are polynomially reducible to each other in any language including $\mathcal{O}$.*

Proof. Follows from Lemma 6.2 and from the fact that the size of $\mathcal{A}_\Sigma \cup \mathcal{A}_C \cup \mathcal{A}_D \cup \{b\}$ is linear with respect to the size of Σ. □

Next we show that in the languages with $\mathcal{O}$ and $\mathcal{B}$, knowledge base satisfiability can be reduced to concept satisfiability. In order to achieve this result, we present the transformation ϕ from a knowledge base to a concept defined as follows: Let $\mathcal{L}$ be a concept language including $\mathcal{O}$ and $\mathcal{B}$, Σ an $\mathcal{L}$-knowledge base, C, D two $\mathcal{L}$-concepts, and a, b two individuals, then:

$$\begin{aligned} \phi(C(a)) &= \exists Q_i \sqcap \forall Q_i.(\{a\} \sqcap C) \\ \phi(R(a,b)) &= \exists Q_i \sqcap \forall Q_i.(\{a\} \sqcap R : b) \\ \phi(\Sigma) &= \sqcap_{(\alpha \in \Sigma)} \phi(\alpha) \end{aligned}$$

where i has a different value for each assertion and Q_i does not appear in Σ for each i[6]. Intuitively, ϕ "encodes" the knowledge base Σ in the implicit assertions of the concept $\phi(\Sigma)$. Such encoding is done in a way that the only possible cause of unsatisfiability of $\phi(\Sigma)$ comes from the implicit assertions. It follows that $\phi(\Sigma)$ is satisfiable if and only if Σ is satisfiable, as stated by the following lemma.

Lemma 6.4 *Given a language $\mathcal{L}$ and an $\mathcal{L}$-knowledge base Σ, then Σ is satisfiable iff $\phi(\Sigma)$ is satisfiable*

[6]If $\mathcal{L}$ includes $\mathcal{E}$ then the concept $\exists Q_i \sqcap \forall Q_i.(\ldots)$ can be simplified in $\exists Q_i.(\ldots)$ and the condition on i can be dropped. In fact this condition is needed to avoid the interaction between the universal and existential quantifications coming up from different assertions.

Proof.

"$\Rightarrow$" Suppose $\phi(\Sigma)$ satisfiable and let $\mathcal{I}$ be one of its models. There exists an element d in $\Delta^{\mathcal{I}}$ such that $d \in \phi(\Sigma)^{\mathcal{I}}$. By definition of ϕ, for each assertion in Σ of the form $C(a)$ (resp. $R(a,b)$) we have that $(d, a^{\mathcal{I}}) \in Q_k^{\mathcal{I}}$, for some k, and $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$). Hence $\mathcal{I}$ is a model of Σ.

"$\Leftarrow$" Suppose Σ satisfiable and let $\mathcal{I}$ be one of its models. Since for each i, Q_i does not appear in Σ, we can assume, without lost of generality, that $Q_i^{\mathcal{I}} = \emptyset \times \emptyset$ for each i. Let $\mathcal{I}'$ be the interpretation such that $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}} \cup \{d\}$ and $\cdot^{\mathcal{I}'} = \cdot^{\mathcal{I}}$ except that for each conjunct of $\phi(\Sigma)$ we have $(d, a^{\mathcal{I}'}) \in Q_k^{\mathcal{I}'}$. It is easy to see that $d \in (\phi(\Sigma))^{\mathcal{I}'}$ and therefore $\phi(\Sigma)$ is satisfiable.

$\square$

Exploiting the same idea of Lemma 6.4, next lemma shows that in the languages with $\mathcal{O}$ and $\mathcal{B}$, instance checking can be reduced to terminological subsumption.

Lemma 6.5 *Give a language $\mathcal{L}$, an $\mathcal{L}$-knowledge base Σ, an individual a, and an $\mathcal{L}$-concept C*[7] *then $\Sigma \models C(a)$ iff $\phi(\Sigma) \sqcap \{a\} \sqsubseteq C$*

Proof.

"$\Rightarrow$" Assume $\Sigma \models C(a)$ and $\phi(\Sigma) \sqcap \{a\} \not\sqsubseteq C$. Since $\phi(\Sigma) \sqcap \{a\} \not\sqsubseteq C$, there exists an interpretation $\mathcal{I}$ and an element $d \in \Delta^{\mathcal{I}}$ such that $d \in (\phi(\Sigma) \sqcap \{a\})^{\mathcal{I}}$ and $d \notin C^{\mathcal{I}}$. From $d \in (\phi(\Sigma) \sqcap \{a\})^{\mathcal{I}}$, it follows that $a^{\mathcal{I}} = d$ and therefore $a^{\mathcal{I}} \in (\phi(\Sigma))^{\mathcal{I}}$ and $a^{\mathcal{I}} \notin C^{\mathcal{I}}$. Since $a^{\mathcal{I}} \in (\phi(\Sigma))^{\mathcal{I}}$ holds, $\mathcal{I}$ is a model of $\phi(\Sigma)$ and therefore (see proof of Lemma 6.4) it is a model of Σ too. Since $a^{\mathcal{I}} \notin C^{\mathcal{I}}$ and $\mathcal{I}$ is a model of Σ, the assumption $\Sigma \models C(a)$ is contradicted.

"$\Leftarrow$" Assume $\phi(\Sigma) \sqcap \{a\} \sqsubseteq C$ and $\Sigma \not\models C(a)$. From $\Sigma \not\models C(a)$ it follows that there exists an interpretation $\mathcal{I}$ such that $\mathcal{I}$ satisfies Σ and $a^{\mathcal{I}} \notin C^{\mathcal{I}}$. Since $\mathcal{I}$ satisfies Σ, there exists $\mathcal{I}'$ that satisfies $\phi(\Sigma)$ (see proof of

[7] We assume that Q_i (for each i) does not appear in C. Otherwise it is possible to modify Φ in a way such that this condition is fulfilled.

Lemma 6.4). Suppose $d \in \phi(\Sigma)^{\mathcal{I}'}$. Since for each i, Q_i does not appear in C and the only necessary property of d concern Q_i, it is possible to assume that $d \notin C^{\mathcal{I}'}$. Let $\mathcal{I}''$ be the interpretation obtained from $\mathcal{I}'$ assigning $a^{\mathcal{I}''} = d$. Then $a^{\mathcal{I}''} \notin C^{\mathcal{I}''}$ and $a^{\mathcal{I}''} \in \phi(\Sigma)^{\mathcal{I}''}$. Therefore $a^{\mathcal{I}''} \in (\phi(\Sigma) \sqcap \{a\})^{\mathcal{I}''}$ contradicting the assumption $\phi(\Sigma) \sqcap \{a\} \sqsubseteq C$.

□

Theorem 6.6 *In any language including $\mathcal{O}$ and $\mathcal{B}$, knowledge base satisfiability is polynomially reducible to concept satisfiability and instance checking is polynomially reducible to hybrid subsumption.*

Proof. Follows from Lemmata 6.4 and 6.5 and the observation that ϕ is polynomial. □

Summarizing the results obtained since now, for concept languages with $\mathcal{O}$ and $\mathcal{B}$ instance checking and hybrid subsumption are reducible to each other, knowledge base satisfiability is reducible to concept satisfiability and instance checking is reducible to subsumption[8].

The relations stated in this section are shown in Figure 2. Combining the result of Theorem 6.6 with the properties of the languages with $\mathcal{C}$ we obtain that for languages including $\mathcal{O}$ and $\mathcal{C}$ all the five problems are reducible to each other (or to their complement).

The results of Theorem 6.6 are important because they relate ABox problems with TBox ones. To this regard, in [DLNS92] it was already proved that for a large class of languages concept satisfiability and knowledge base satisfiability are in the same complexity class, and therefore the latter is reducible to the former. However, that result is achieved considering each language separately. The result obtained here is stronger, in the sense that it is proved independently of the single language (provided that $\mathcal{O}$ and $\mathcal{B}$ are included). This result is important, since it relates the complexity of an ABox-problem to the complexity of a TBox-problem. Such a relationship is crucial for the design of efficient reasoning algorithms and it is still not completely clear.

In [Sch93] and [DLNS92] it was also shown that there are languages such that instance checking and subsumption are in different complexity classes.

[8]Since no primitive complement are considered, the results are valid even for the less expressive language than the ones considered here (i.e. $\mathcal{FLOB}^-$).

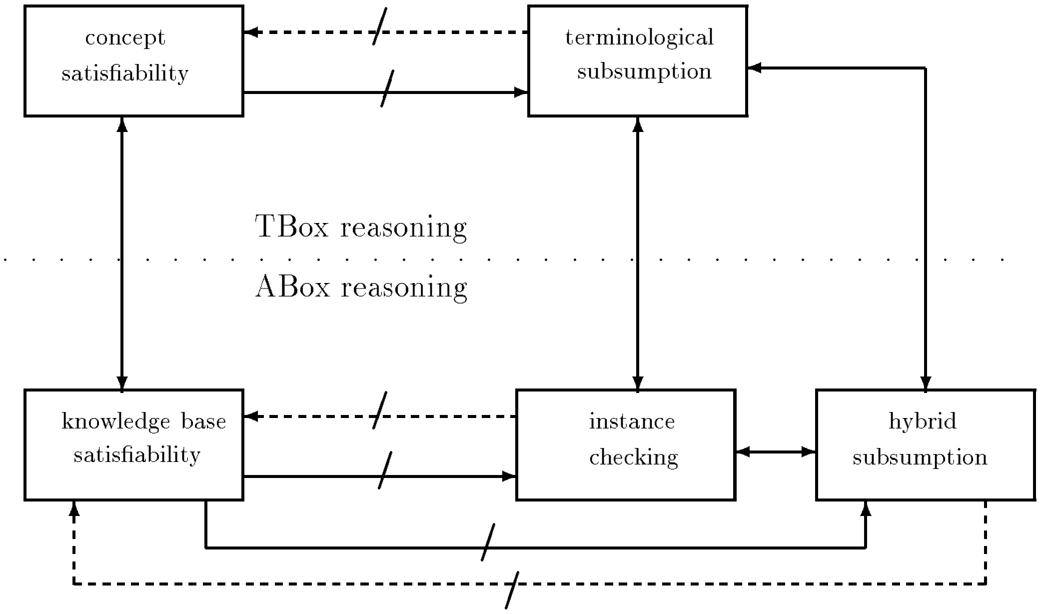

Figure 2: Reductions between reasoning tasks in languages with $\mathcal{O}$ and $\mathcal{B}$

In particular, in [DLNS92] it is shown that in $\mathcal{ALE}$ instance checking is PSPACE-complete while subsumption is NP-complete. Therefore (assuming NP$\neq$PSPACE), such result proves that, in $\mathcal{ALE}$, instance checking is strictly harder than subsumption. On the other hand, the result in Theorem 6.6 ensures that, as far as $\mathcal{O}$ and $\mathcal{B}$ are included in the language, no result of that kind are possible since the two problems are reducible to each other.

6.2 Reasoning in $\mathcal{ALCO}$

Using a technique based on constraint systems, in [SSS91] both concept satisfiability and subsumption in $\mathcal{ALC}$ are proved to be PSPACE-complete problems. That paper also presents a linear space algorithm for these problems. The basic idea of the algorithm is that, although the whole constraint system involved in the computation might have exponential size, it is possible to keep track of only a polynomial part of it at a time. These parts, called *traces*, are obtained considering, for each variable, one existential quantification at the time. Traces are mutually independent, and can be checked for a clash separately.

However, when the language is $\mathcal{ALCO}$, the above method is no longer valid. In fact, because of the presence of implicit assertions, the traces are not independent. On the contrary, the satisfiability of one trace can depend on the constraints in the other traces, as shown by Example 3.1. In the calculus shown in Section 5, this possibility is taken into account by the $\rightarrow_{[/]}$-rule which allows for the substitution of variable in different traces with the same individual.

Nevertheless reasoning in $\mathcal{ALCO}$ can be done in PSPACE. In fact, in [DLN+93], following the idea in [DLN+92], a polynomial space algorithm for instance checking $\Sigma \models D(a)$, where Σ is an $\mathcal{ALC}$-knowledge base and D is an $\mathcal{ALCK}$-concept. It is also shown that such problem is at least as hard as the same problem where C is an $\mathcal{ALCO}$-concept. The latter problem, in its turn, is at least as hard as checking the satisfiability of the $\mathcal{ALCO}$-concept $\neg D$. It follows that concept satisfiability in $\mathcal{ALCO}$, being at least as easy as a PSPACE problem, is in PSPACE too. In conclusion, we have the following theorem, whose proof derives from the results in [DLN+93] and the above observation.

Theorem 6.7 *Terminological subsumption in $\mathcal{ALCO}$ is a PSPACE-complete problem.*

The intuition of the fact that reasoning in $\mathcal{ALCO}$ is still in PSPACE is that, although the constraint system has in general exponential size, the informations regarding the individuals are polynomial. In fact, since the number of concepts to be considered is linear and the number of individuals is linear too, the membership relation of the individuals to the subconcepts can be stored in a table of polynomial size.

6.3 Reasoning in $\mathcal{ALEO}$

In [DLNS92] it is proven that instance checking in $\mathcal{ALE}$ is PSPACE-complete. It follows that instance checking in $\mathcal{ALEO}$ is PSPACE-hard, and therefore (Proposition 6.5) subsumption in $\mathcal{ALEO}$ is PSPACE-hard too. Since subsumption in $\mathcal{ALCO}$ is PSPACE-complete, it is in PSPACE for $\mathcal{ALEO}$ too (remind that $\mathcal{ALCO}$ is a superlanguage of $\mathcal{ALEO}$).

Theorem 6.8 *Terminological subsumption in $\mathcal{ALEO}$ is a PSPACE-complete problem.*

6.4 Reasoning in $\mathcal{ALO}$

In this section we state that subsumption in $\mathcal{ALO}$ is coNP-complete, besides the fact that in $\mathcal{AL}$ both subsumption and instance checking are in P, as proved in [DLNN91] and [LS91a] respectively. Since $\mathcal{ALO}$ does not have $\mathcal{E}$, $\mathcal{ALO}$ is not equivalent to $\mathcal{ALOB}$ and therefore (unlike previous sections) the results obtained for the language with $\mathcal{O}$, are not directly extended to the language with $\mathcal{O}$ and $\mathcal{B}$. Anyway, the results we obtain here for $\mathcal{ALO}$ are valid for $\mathcal{ALOB}$, as can be easily proved.

In order to prove the coNP-hardness of subsumption, we now prove the NP-hardness of concept satisfiability which obviously implies the coNP-hardness of subsumption (remind relation (1) in Section 2).

This proof is based on a reduction from SAT, the satisfiability problem for a propositional conjunctive normal form (CNF) formula, to the concept satisfiability problem in $\mathcal{ALO}$. This reduction has been sketched in [LS91a], here it is proposed in the full version and its correctness is proven.

We define a *pos-neg* CNF-formula, a CNF-formula Γ' such that every clause of Γ' is either positive (i.e. it is constituted by positive literals) or negative (i.e. it is constituted by negative literals).

Notice, first of all, that any CNF formula Γ can be transformed in polynomial time into a pos-neg CNF formula Γ' such that Γ is satisfiable if and only if Γ' is so. This is done by replacing every clause of Γ by two clauses, as follows:

$$p_1 \vee \ldots \vee p_n \vee \sim q_1 \vee \ldots \vee \sim q_m \quad \Rightarrow$$

$$(p_1 \vee \ldots \vee p_n \vee r), \quad (\sim q_1 \vee \ldots \vee \sim q_m \vee \sim r)$$

where r is a new propositional variable.

Let Ψ be the transformation from a pos-neg CNF formula $\Gamma = \alpha_1^+ \wedge \cdots \wedge \alpha_n^+ \wedge \alpha_1^- \wedge \cdots \wedge \alpha_m^-$ to the $\mathcal{ALO}$-concept $\Psi(\Gamma) = C_1^+ \sqcap \cdots \sqcap C_n^+ \sqcap C_1^- \sqcap \cdots \sqcap C_m^-$, specified by the following equations:

$$\begin{aligned} C_h^+ &= \exists R_h^+ \sqcap \forall R_h^+.(obj(\alpha_h^+) \sqcap A), \\ C_k^- &= \exists R_k^- \sqcap \forall R_k^-.(obj(\alpha_k^-) \sqcap \neg A) \end{aligned}$$

where α_i^+ (resp. α_i^-) denotes a positive (resp. negative) clause, A is a primitive concept, R_h^+ and R_k^- (for $h = 1, \ldots, n$ and $k = 1, \ldots, m$) are roles, and $obj(\alpha)$ denotes the concept $\{p_1, \ldots, p_k\}$, where $p_1, \ldots, p_k$ are all the propositional letters in the clause α. In other words, we associate with every propositional letter of Γ an individual with the same name, and with every clause α of Γ the collection of individuals $obj(\alpha)$.

For example, if $\Gamma = (p \vee q) \wedge (\sim p \vee \sim r)$, then the corresponding $\mathcal{ALO}$-concept is:

$$\Psi(\Gamma) = \exists R_1^+ \sqcap \forall R_1^+.(\{p, q\} \sqcap A) \sqcap \exists R_1^- \sqcap \forall R_1^-.(\{p, r\} \sqcap \neg A).$$

Lemma 6.9 *A pos-neg CNF formula Γ is satisfiable if and only if the corresponding $\mathcal{ALO}$-concept $\Psi(\Gamma)$ is satisfiable.*

Proof.

"$\Leftarrow$" Suppose $\Psi(\Gamma)$ satisfiable. Then there exists an interpretation $\mathcal{I}$ and an object d such that $d \in (\Psi(\Gamma))^{\mathcal{I}}$. Looking at the structure of $\Psi(\Gamma)$, one can verify that there are $n + m$ objects $d_1^+, \ldots, d_n^+, d_1^-, \ldots, d_m^-$ in $\Delta^{\mathcal{I}}$ such that

$$(d, d_1^+) \in (R_1^+)^{\mathcal{I}}, \ldots, (d, d_n^+) \in (R_n^+)^{\mathcal{I}},$$
$$(d, d_1^-) \in (R_1^-)^{\mathcal{I}}, \ldots, (d, d_m^-) \in (R_m^-)^{\mathcal{I}},$$

and for each $i \in \{1, \ldots, n\}$, $d_i^+ \in (obj(\alpha_i^+))^{\mathcal{I}}$ and $d_i^+ \in A^{\mathcal{I}}$, and for each $i \in \{1, \ldots, m\}$, $d_i^- \in (obj(\alpha_i^-))^{\mathcal{I}}$ and $d_i^- \in (\neg A)^{\mathcal{I}}$. Now construct a truth assignment J for Γ as follows: for every letter p, if $p^{\mathcal{I}} \in A^{\mathcal{I}}$, then $J(p) = true$, else $J(p) = false$. Due to the above properties, for every clause α_i^+ (resp. α_j^-), J assigns $true$ (resp. $false$) to at least one literal in α_i^+ (resp. α_j^-), and therefore J satisfies every clause of Γ.

"$\Rightarrow$" Suppose Γ satisfiable. There exists one truth assignment J that satisfies Γ. Let $\mathcal{I}$ be the interpretation such that:

- $A^{\mathcal{I}} = \{p \mid J(p) = true\}$
- for each $i \in \{1, \ldots, n\}$, $(R_i^+)^{\mathcal{I}} = \{(d, q^{\mathcal{I}})\}$, where q is a literal in c_i such that $J(q) = true$

- for each $i \in \{1, \dots, m\}$, $(R_i^-)^{\mathcal{I}} = \{(d, q^{\mathcal{I}})\}$, where $\sim q$ is a literal in c_i such that $J(q) = false$

It is easy to see that $\mathcal{I}$ is a model for $\Psi(\Gamma)$. □

Lemma 6.10 *Concept satisfiability in $\mathcal{ALO}$ is coNP-hard.*

Proof. Follows from Lemma 6.9 and the fact that the reduction Ψ is clearly polynomial with respect to the size of Γ.

We now show that satisfiability of an $\mathcal{ALO}$-concept C is in NP. This result is achieved showing that the algorithm obtained by the application of the propagation rule runs in nondeterministic polynomial time.

The first step is to show that, for any $\mathcal{ALO}$-concept C, each completion of the constraint system $\{x\colon C\}$ has polynomial size. In [SSS91] it is shown that for every $\mathcal{AL}$-concept C, the constraint system $\{x\colon C\}$ has the unique completion (up to variable renaming), which has linear size w.r.t. $|C|$. However, in $\mathcal{ALO}$, the size of the completion can be bigger, as shown by the following example:

$$C = \{a\} \sqcap \exists R \sqcap \forall R.\{a\} \sqcap \forall R.\forall R.\cdots\forall R.A$$

The only completion of the constraint system $\{x\colon C\}$ has quadratic size w.r.t. the length n of the chain of universal quantifications $\forall R.\forall R.\cdots\forall R.A$. In fact, such a completion contains the constraints $a\colon \forall R.\forall R.\cdots\forall R.A, \dots, a\colon \forall R.A$, $a\colon A$ for any size of the chain from 1 to N.

The example shows that the introduction of constraints on the individuals can increase the size of the single completion. Nevertheless, we show that its size is still polynomial. In particular, given the constraint system $\{x\colon C\}$, we associate with it a constraint system called S_C and we show the two following properties: (i) the completion of S_C is polynomial with respect to $|C|$ and (ii) the completion of S_C is bigger that the completion of the constraint system $\{x\colon C\}$.

Definition 6.11 *Given a constraint system $\{x\colon C\}$, we call S_C the constraint system obtained adding to $\{x\colon C\}$ all the constraints $a : E$, for every individual a in C and every subconcept E of C (including C itself). If C does not contain any individual then $S_C = \{x\colon C\}$.*

Since both the number of subconcept of C and the number of individual in C are linear w.r.t. $|C|$ it follows that S_C has a quadratic number of constraints w.r.t. $|C|$. Therefore the size of S_C is almost cubic w.r.t. $|C|$. Since S_C contains all the constraints of the form $a\colon E$, the application of the $\rightarrow_{[/]}$-rule to a constraint system S obtained from S_C do not add any new constraint to S. Therefore S_C has a single completion S'. Now, it is easy to see that each completion of $\{x\colon C\}$ is smaller than S'; in fact S_C contains at least the constraint $x\colon C$.

We now show that S' is polynomial w.r.t. S_C (and therefore w.r.t. C). To this aim, in [LS91a], it is shown that for every $\mathcal{AL}$-knowledge base Σ, the unique completion of the constraint system S_Σ, has polynomial size w.r.t. $|\Sigma|$. Since the $\rightarrow_{[/]}$-rule does not add any constraint to S_C, it follows that the completion of S_C is equal to the completion obtained for an $\mathcal{AL}$ constraint system. Therefore from the above results concerning $\mathcal{AL}$, it follows that the completion of S_C has polynomial size. Hence every completion of $\{x\colon C\}$ has polynomial size too.

The fact that each completion has polynomial size ensures that it can be computed in polynomial time. In fact, the application of each rule either increase the size of the constraint system or increase the number of constraints on the individuals (the $\rightarrow_{[/]}$-rule). Since the number of constraints on the individuals is obviously bound by the size of the constraint system it follows that only a polynomial number of applications of the rules can be done. Since each completion is obtained doing a polynomial guess in the application of the $\rightarrow_{[/]}$-rule, it follows that the whole algorithm works in nondeterministic polynomial time. Therefore we have proved the following lemma:

Lemma 6.12 *Concept satisfiability in $\mathcal{ALO}$ is an NP problem.*

Theorem 6.13 *Terminological subsumption in $\mathcal{ALO}$ is coNP-complete.*

Proof. The NP-hardness follows from Lemma 6.10. Regarding the upper bound, using the technique in [DLNN91, Sec. 5], Lemma 6.12 can be easily extended to state that subsumption in $\mathcal{ALO}$ is in NP too. □

In [LS91a], it is observed that it is not necessary to have a constructor for primitive negation for intractability, but it suffices to have the possibility to express disjointness between concepts. Therefore the intractability is directly extended to several other languages. For example, it applies to $CLASSIC$,

	without $\mathcal{O}$			with $\mathcal{O}$
	subsumption	instance checking		subsumption & inst. ch.
$\mathcal{AL}$	P [SSS91]	P [LS91a]	$\mathcal{ALO}$	coNP
$\mathcal{ALE}$	NP [DHL^{+}92]	PSPACE [DLNS92]	$\mathcal{ALEO}$	PSPACE
$\mathcal{ALC}$	PSPACE [SSS91]	PSPACE [BH91]	$\mathcal{ALCO}$	PSPACE

Table 1: Complexity of reasoning

which extend $\mathcal{FL}^-$ in several ways including $\mathcal{O}$ and $\mathcal{N}$ In fact using $\mathcal{N}$, it is possible to express the following two concepts which are obviously disjoint: $(\leq 2\, R)$ and $(\geq 3\, R)$.

On the other hand, if the underlying language does not include any form of disjointness, reasoning with $\mathcal{O}$ is in general polynomial. For example, in [FMV90], it is shown that in the language $\mathcal{OOL}$, which extends $\mathcal{FLO}^-$, subsumption can be checked in polynomial time.

The Table 1 summarize the results obtained in this section together with previous known results on the underlying languages. In the table, each entry means that the problem is complete for the given class, except for P that has the simple meaning that the problem is in the class P.

7 On the use of $\mathcal{O}$ in the query language

In the previous section we have shown that the use of $\mathcal{O}$ generally increases the complexity of reasoning. Opposite to this negative result, there exists one possibility of exploiting $\mathcal{O}$ in a useful way without increasing the complexity of reasoning: Admitting it only in the query language, i.e. allowing $\mathcal{O}$ in the expression of the query concept but not in the assertion of the knowledge base.

The usefulness of $\mathcal{O}$ in the query language is discussed in [LS91b]. In particular, using $\mathcal{O}$ it is possible to express various forms of selection that can be usually admitted in database query languages but are missing in standard concept languages. For example it is possible to ask for the books whose author is Newton and whose subject it mathematics:

$$\texttt{Book}\sqcap\exists\texttt{AUTHOR}.\{\texttt{newton}\}\sqcap\exists\texttt{SUBJECT}.\{\texttt{math.}\}$$

In [LS91b], it is shown that it is possible to query an $\mathcal{AL}$-knowledge base using $\mathcal{ALO}$ concepts in polynomial time, opposite to the fact that reasoning in $\mathcal{ALO}$ is in general coNP-complete. Our conjecture is that this result is quite general, in the sense that for many languages $\mathcal{L}$, it is possible to query an $\mathcal{L}$-knowledge base using concepts in $\mathcal{LO}$ with the same computing resources of reasoning in $\mathcal{L}$.

8 Discussion and Conclusions

We have shown an extended analysis of the various issues related to the use of concept constructors involving individuals. This analysis gives an insight of the problem of reasoning with individuals and allows to understand the intuitive aspects which makes reasoning difficult.

In addition, we have presented a complete procedure for reasoning with $\mathcal{O}$. This procedure is developed within the well established framework of constraint systems. The benefit of that is the possibility to extend this procedure to other languages.

Another result of the paper are a set of complexity results which formally confirm that reasoning with individuals is generally hard. In fact, in some languages, they increases the complexity of reasoning ($\mathcal{AL}$,$\mathcal{ALE}$). Whereas, in those cases in which reasoning is in the same complexity class as the underlying language ($\mathcal{ALC}$), the algorithms are generally more complex and less efficient (in term of both time and space) than in the underlying language (see [DLN$^+$93]).

We have also identified an intuitive explanation of this intractability: On one side, it is related to the implicit disjunction carried by the use of sets with more then one object. On the other side, it is due to the implicit equality associated with individuals in concept expressions.

In our opinion, the solutions proposed in actual systems to overcome the computational intractability are not completely satisfying. Therefore a deeper insight of the problem can also be useful for the development of better incomplete reasoners.

Acknowledgements

I am in debt to Francesco Donini, Daniele Nardi, Werner Nutt and, particularly, to Maurizio Lenzerini, for their support and collaboration. I would like to thank Enrico Franconi and Marco Schaerf for useful comments on earlier drafts of the paper. I also acknowledges Yoav Shoham for his hospitality at the Computer Science Department of Stanford University, where part of this research as been done.

This work has been supported by the ESPRIT Basic Research Action N.6810 (COMPULOG 2) and by the Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of the CNR (Italian Research Council).

References

[BBH+91] Franz Baader, Hans-Jürgen Bürkert, Jochen Heinson, Bernhard Hollunder, Jürgen Müller, Bernard Nebel, Werner Nutt, and Hans-Jürge Profitlich. Terminological knowledge representation: A proposal for a terminological logic. Technical Report TM-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz, Postfach 2080, D-6750 Kaiserslautern, Germany, 1991.

[BBMAR89] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *ACM SIGMOD*, 1989.

[BH91] Franz Baader and Bernhard Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of the Workshop on Processing Declarative Knowledge, PDK-91*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1991.

[BL84] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc.*

of the 4th Nat. Conf. on Artificial Intelligence AAAI-84, pages 34–37, 1984.

[BM77] John L. Bell and Moshe Machover. *A Course in Mathematical Logic*. North-Holland, 1977.

[BMPS+91] Ronald J. Brachman, Deborah L. McGuinness, Peter F. Patel-Schneider, Lori Alperin Resnick, and Alex Borgida. Living with CLASSIC: when and how to use a KL-ONE-like language. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, 1991.

[BPGL85] Ronald J. Brachman, Victoria Pigman Gilbert, and Hector J. Levesque. An essential hybrid reasoning system: Knowledge and symbol level accounts in KRYPTON. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, pages 532–539, Los Angeles, Cal., 1985.

[BPS92] Alexander Borgida and Peter F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. Submitted for publication, 1992.

[Bra92] Ronald J. Brachman. "reducing" CLASSIC to practise: Knowledge representation meets reality. In *Proc. of the 3nd Int. Conf. on Principles of Knowledge Representation and Reasoning KR-92*, pages 247–258, 1992.

[DE92] Francesco M. Donini and Angelo Era. Most specific concepts for knowledge bases with incomplete information. In Y. Yesha, editor, *Information and Knowledge Management, CIKM-92*, pages 545–551, 1992.

[DHL+92] Francesco M. Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, and Werner Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.

[DLN+92] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Andrea Schaerf. Adding epistemic operators to concept languages. In *Proc. of the 3nd Int. Conf. on*

Principles of Knowledge Representation and Reasoning KR-92, pages 342–353, 1992.

[DLN+93] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, and Andrea Schaerf. Adding epistemic operators to concept languages. Technical report, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1993. Forthcoming.

[DLNN91] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proc. of the 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning KR-91*, pages 151–162. Morgan Kaufmann, 1991.

[DLNS92] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. From subsumption to instance checking. Technical Report 15.92, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1992.

[FMV90] Anna Formica, Michele Missikoff, and S. Vazzana. An object-oriented data model for artificial intelligence applications. In J. W. Schmidt and A. A. Stogny, editors, *next generation information systems technology: first int. east/west database workshop*, number 504 in Lecture Notes in Computer Science, pages 26–41, Kiev, USSR, October 1990. Springer-Verlag.

[HNSS90] Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proc. of 9th the European Conf. on Artificial Intelligence ECAI-90*, London, 1990. Pitman.

[Hol90] Bernhard Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proc. of the German Workshop on Artificial Intelligence*. Springer-Verlag, 1990.

[Kin90] Carsten Kindermann. Class instances in a terminological framework–an experience report. In *Proc. of the German Workshop on Artificial Intelligence*, pages 48–57. Springer-Verlag, 1990.

[LB87] Hector J. Levesque and Ron J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[LS91a] Maurizio Lenzerini and Andrea Schaerf. Concept languages as query languages. In *Proc. of the 9th Nat. Conf. on Artificial Intelligence AAAI-91*, 1991.

[LS91b] Maurizio Lenzerini and Andrea Schaerf. Querying concept-based knowledge bases. In *Proc. of the Workshop on Processing Declarative Knowledge, PDK-91*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1991.

[Mac91] Robert MacGregor. The evolving technology of classification-based knowledge representation systems. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann, 1991.

[MB92] Robert MacGregor and David Brill. Recognition algorithms for the loom classifier. In *Proc. of the 10th Nat. Conf. on Artificial Intelligence AAAI-92*, pages 774–779, 1992.

[Neb90a] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*. Lecture Notes in Artificial Intelligence. Springer-Verlag, 1990.

[Neb90b] Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.

[Neb91] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, 1991.

[NvL88] Bernhard Nebel and Kai von Luck. Hybrid reasoning in BACK. In *Proc. of the 3rd Int. Symp. on Methodologies for Intelligent Systems ISMIS-88*, pages 260–269. North-Holland, 1988.

[PS93] P.F. Patel-Schneider and Bill Swartout. Working version (draft): Description logic specification from the krss effort. January 1993. Unpublished Manuscript.

[QK90] Joachim Quantz and Carsten Kindermann. Implementation of the BACK system version 4. Technical Report KIT-Report 78, FB Informatik, Technische Universität Berlin, Berlin, Germany, 1990.

[Sch93] Andrea Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. In *Proc. of the 7th Int. Symp. on Methodologies for Intelligent Systems ISMIS-93*, pages 508–517, 1993. An extended version will appear in *Journal of Intelligent Information Systems*.

[SSS91] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.