An epistemic operator for description logics

F.M. Donini a,*, M. Lenzerini a,*, D. Nardi a,*, W. Nutt b,2, A. Schaerf a,1

a Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Roma, Italy
b German Research Center for Artificial Intelligence GmbH (DFKI), Saarbrücken, Germany

Received 15 November 1996

Abstract

Description logics (also called terminological logics, or concept languages) are fragments of first-order logic that provide a formal account of the basic features of frame-based systems. However, there are aspects of frame-based systems—such as nonmonotonic reasoning and procedural rules—that cannot be characterized in a standard first-order framework. Such features are needed for real applications, and a clear understanding of the logic underlying them is necessary for principled implementations.

We show how description logics enriched with an epistemic operator can formalize such aspects. The logic obtained is a fragment of a first-order nonmonotonic modal logic. We show that the epistemic operator formalizes procedural rules, as provided in many knowledge representation systems, and enables sophisticated query formulation, including various forms of closed-world reasoning. We provide an effective procedure for answering epistemic queries posed to a knowledge base expressed in a description logic and extend this procedure in order to deal with rules. We also address the computational complexity of reasoning with the epistemic operator, identifying cases in which an appropriate use of the epistemic operator can help in decreasing the complexity of reasoning. © 1998 Elsevier Science B.V.

Keywords: Knowledge representation; Description logics; Epistemic operators; Nonmonotonic reasoning; Closed-world reasoning; Query languages

1. Introduction

Frame-based systems are among the most widely used tools for the construction of Artificial Intelligence systems. They are based on the idea that knowledge can be repre-
presented by defining structural descriptions, called “frames”, which are arranged hierarchically, typically based on a generalization/specialization relation. The wide acceptance of frames as representation framework has a twofold justification. Clearly, frames allow the user to represent the taxonomies that arise in the construction of knowledge-based applications. Moreover, the hierarchical organization of frames allows for several interesting forms of reasoning, such as inheritance and classification, which are required for problem solving.

A substantial amount of research has been carried out in the last decade with the aim of providing a logical reconstruction of frame-based knowledge representation systems. Although a first-order semantics of frames was well known (see [33]), much of this work has taken place in the context of description logics (also called terminological logics, or concept languages). Description logics have been developed after the work of Brachman and others [6,8,12], with the aim of providing a tight formal setting for describing properties of both the representation language and the associated reasoning procedures of frame systems. Description logics are fragments of first-order predicate calculus that suffice to capture the logical content of frames and are limited enough to allow for effective procedures that perform the reasoning tasks of interest (e.g., subsumption).

However, the first-order semantics leave out several features that are typically provided in frame-based systems. The need for such features has often been discussed in the literature (see, for example, [29,59]). They can be classified as follows:

- **query features**, such as those typical of database systems;
- **nonmonotonic features**, which allow one to make assumptions based on incomplete knowledge;
- **procedural features**, which allow one to express knowledge in terms of procedural rules, attachments, and methods.

In this paper we present an extension of description logics with an epistemic operator that is interpreted in terms of knowledge in the style of Lifschitz and Reiter [40,51] and show that in the resulting language we can effectively address all the above three aspects, thus providing a formal basis for the behavior of implemented systems. The correspondence between theory and practice is made concrete by referring to some of the most recent frame systems based on description logics such as CLASSIC [10], BACK [49], LOOM [42] and CLASP [61]. The main contribution of the paper is therefore a new common framework for a formal characterization of several aspects of frame systems that are still lacking a clear semantic interpretation and techniques for the associated reasoning tasks.

### 1.1. Representing knowledge with description logics

In description logics, concepts are used to represent classes as sets of individuals, and roles are used to specify properties or attributes as binary relations. Typically, concepts are placed into hierarchies determined by the properties associated with them. More specific concepts inherit the properties of more general ones through the hierarchical structure. Concepts are often described through diagrams (e.g., see [12]), but
description logics provide formal languages, called concept languages, for describing the structure of concepts.

For example, the concept FatherOfSons (see Fig. 1) can be modeled through the concept expression $\text{Parent} \sqcap \text{Male} \sqcap \forall \text{CHILD.Male}$, which denotes the class of fathers (male parents) all of whose children are male. The symbol "$\sqcap$" denotes concept conjunction and is interpreted as set intersection. Similarly one can use disjunction "$\sqcup$" and negation "$\neg$", interpreted as set union and complement. The expression $\forall \text{CHILD.Male}$ denotes the set of individuals all of whose children are male, thus specifying a property which relates, through the role CHILD, individuals in the described class to other individuals. Expressions of the form $\forall R.C$ are called universal role quantifications. Similarly, $\exists \text{CHILD.Male}$ is an example of an existential role quantification, denoting the set of individuals with at least one male child. The basic language that we consider (called $\mathcal{ALC}$, see [56]) includes concept negation, conjunction, disjunction, universal role quantification and existential role quantification.

One of the goals motivating the study of description logics is the design of efficient methods for the classification of concepts [60] according to the subsumption relation. Essentially, subsumption of concepts is defined as logical implication. Early algorithms published in the literature and most of the procedures implemented in systems are based on the idea of comparing the syntactic structure of the expressions denoting concepts. Such a check is complete (with respect to first-order semantics) only for languages with limited expressivity [8] and in general more powerful methods are required to fully capture the logic of subsumption [56]. The subsumption problem has been studied for a wide range of concept languages [19, 20, 45, 54, 56] and the relationship between the expressive power of languages and the computational complexity of reasoning about concepts has been fully characterized (see [25]).

Knowledge bases combine intensional and extensional knowledge. The typical way (first proposed in the system KRYPTON [7, 11]) to realize this distinction is to divide the knowledge base into two components, called "TBox" ("T" for terminology) and "ABox" ("A" for assertions).

More specifically, the TBox contains concept definitions, which can be organized in a taxonomy according to the subsumption relation. Typically, such definitions take
the form $A \equiv C$, where $A$ is the concept name being defined and $C$ is a concept expression. As shown in [47], concept definitions are problematic from the point of view of reasoning. However, usually they are required to be acyclic, so that one can substitute the defined concepts with the corresponding definitions and perform the actual reasoning on concept expressions. When the hierarchy is not deep, this way of treating definitions is feasible.

The ABox contains knowledge about individuals specified as a set of assertions of the forms $C(a)$ or $R(a, b)$, where $C$ is a concept expression, $R$ is a role, and $a$ denotes an individual. For example, $\text{Male}(\text{andrea})$ asserts that Andrea is male. The system should then provide methods by which one can query the knowledge base for the individuals which are instances of a specified concept. Reasoning taking into account both the ABox and TBox is generally more difficult than checking subsumption with respect to the TBox [24, 53].

1.2. Non-standard representational features

The setting just outlined does not address a number of aspects of knowledge representation that are needed in practice and are often provided in an ad hoc way. We have already mentioned at least three of them, namely query facilities, nonmonotonic reasoning and procedural features, which we will address here.

Regarding query facilities, since concept expressions describe sets of individuals in a knowledge base, it is natural to use concepts as queries. The result of such a query comprises the set of individuals described by the corresponding expression. The use of concept languages as query languages has been investigated in [4, 13, 15, 36].

It has been argued that queries should be able to refer to aspects of the external world, as represented by the knowledge base, as well as to aspects of what the knowledge base knows about the external world (see [37, 40, 51]). The need for such a distinction is evident when a knowledge base contains incomplete information about individuals. For example, if we assert $\exists \text{FRIEND}. \text{Male}(\text{susan})$, the knowledge base cannot tell who is the male friend of Susan although it can tell that there is one. The query language should therefore allow one to express distinctly the query asking whether Susan has a male friend and the query asking whether in the knowledge base there is a known individual who is a friend of Susan. It is worth noticing that, for efficiency reasons, implemented systems sometimes restrict the reasoning to the known individuals. However, these systems do not provide the user with the ability to specify the distinction in the query language.

Many Artificial Intelligence applications require the representation of incomplete knowledge about a state of affairs. There are basically two ways through which incompleteness can be expressed in description logics: existential quantification and disjunction. For example, we have already seen an assertion stating that Susan has a male friend without specifying who Susan's friend is. As an example of a disjunction, a knowledge base may know that Andrea is a person and that every person is either male or female, without knowing which one is Andrea's sex. Note that such a disjunction can be a piece of knowledge and not just an integrity constraint.
From the very beginning, frame-based systems performed nonmonotonic inferences. Several extensions of basic description logics have been proposed that capture aspects of nonmonotonic reasoning. For example [2,50,57] discuss the introduction of defaults into the language. However, none of the existing proposals accounts for the forms of closed-world reasoning that one finds in implemented systems [10].

There are several kinds of procedural features that are often combined with frame-based knowledge representation languages. They range from procedural attachments or daemons that allow one to trigger procedures for specific computations, to so-called procedural rules that provide the ability to trigger forward reasoning on the knowledge base, to the integration of description logics with Datalog rules (see [26,39]). Both procedural attachments and procedural rules can be found in the system KEE [30]. Here we focus mainly on procedural rules, since we find them in systems based on description logics such as CLASSIC [5,10] and LOOM [42,43]. Procedural rules take the form \( C \Rightarrow D \), where \( C \) and \( D \) are concepts. The meaning of a rule is "if an individual is proved to be an instance of \( C \), then derive that it is also an instance of \( D \)." Indeed, in some systems (see, e.g., [42]) concept definitions are interpreted as rules of the above kind, and are treated by forward reasoning procedures that, according to the definitions, add assertions about the individuals in the knowledge base.

1.3. Approach, results and organization of the paper

In the paper we present an epistemic description logic which allows us to give a principled formalization of the non-standard features discussed above. Recent work on data and knowledge bases exploits the use of epistemic operators for improving both the expressiveness of knowledge representation languages and their associated querying facilities. The idea of using an epistemic query language was first proposed by Levesque [37]. Later, his framework was developed by Reiter [51], who investigates the use of the epistemic language to specify integrity constraints and proposes a method for query answering that is applicable to a class of logic programs. Even though the use of the \( O \) (only knowing) operator around the knowledge base makes Levesque's logic monotonic, both systems behave nonmonotonically, in the sense that the method adopted may turn the answer to a query from "yes" into a "no" after adding information to the knowledge base. This is because an implicit closure assumption is made on the knowledge base, when answering queries. This aspect is further investigated in [38] and later in [40], where the closure assumption is made explicit and related to the idea of maximal ignorance, which in turn is analogous to that of minimal knowledge [32,41].

The keystone of our proposal is a logic that is obtained by extending the description logic \( \text{ALC} \) with an epistemic operator both on roles and on concept expressions and by interpreting it in terms of minimal knowledge. The resulting epistemic description logic is called \( \text{ALCK} \), which was presented in [21] and further discussed in [22]. Initially, we use the epistemic language as a query language and assume that the knowledge base is non-modal. Thus, our setting is similar to that of [51]. We also treat integrity constraints similarly and our knowledge bases display an analogous nonmonotonic behavior. Subsequently, we admit epistemic sentences in the knowledge base in a very
restricted form that is sufficient to model procedural rules and weak forms of concept definitions.

As a result, our epistemic extension of description logics captures in a unified framework many non-first-order features that are commonly available in frame-based knowledge representation systems. This extension is both theoretically well-founded on the work on epistemic logics and strictly related to some of the state-of-the-art knowledge representation systems based on description logics.

In addition, we identify situations where nonmonotonic epistemic reasoning can effectively be used. In fact, we provide algorithms for answering epistemic queries in different settings of practical relevance, corresponding to description logics of different expressive power. Moreover, we provide a method for knowledge bases to reason with a class of epistemic sentences corresponding to procedural rules and weak forms of concept definitions.

The foundation of our proposal is the modal description logic \(\text{ALCK}\) (Section 2). We develop a technique (Section 3) for answering epistemic queries expressed in \(\text{ALCK}\), which is an extension of the tableaux-based method, which has already proved useful for solving reasoning and complexity problems in description logics. We then present an extensive example (Section 4) showing that epistemic operators can be useful for the design of more powerful knowledge representation systems based on description logics. In addition, we show that the epistemic operator enhances the expressive power of query languages without increasing the computational complexity of query answering (Section 5). We have also found interesting cases where the use of epistemic operators allows one to express queries (not expressible in first-order logic) that both have natural interpretations and are strictly less costly than their first-order counterparts. We finally show (Section 6) how \(\text{ALCK}\) can be used to provide a semantic characterization of a representation mechanism present in a number of frame-based systems, namely, procedural rules. Moreover, we show that epistemic sentences provide an account for weak forms of concept definitions similar to those found in other implemented systems. This formalization makes it clear that weak definitions provide a form of incomplete reasoning that is both computationally advantageous, and semantically well-founded. We conclude the paper (Section 7) by discussing the main outcomes and the further possible development of the proposed approach.

2. The formalism

In this section we introduce the concept language \(\text{ALC}\) and its epistemic extension \(\text{ALCK}\). Although we restrict our attention to \(\text{ALC}\), the epistemic extension can be applied to other languages as well.

2.1. The concept language \(\text{ALC}\)

The concept language \(\text{ALC}\) (see [19, 56]) allows one to express the knowledge about the classes of interest in a particular application through the notions of concept and role.
Intuitively, concepts represent the classes of objects in the domain to be modeled, while roles represent relationships between objects. Starting with concept names and role names, one can construct complex expressions by means of various concept-forming operators.

The syntax and semantics of $\mathcal{ALC}$ are as follows. We assume that two alphabets of symbols are given, one for atomic concepts, and one for atomic roles. The letter $A$ always denotes a concept name, and the letter $P$ denotes a role, which in $\mathcal{ALC}$ is always a name. The concepts (denoted by the letters $C$ and $D$) of the language $\mathcal{ALC}$ are built up according to the syntax rule:

$$ C, D \rightarrow A \mid \top \mid \bot \mid C \cap D \mid C \cup D \mid \neg C \mid \forall P.C \mid \exists P.C $$

(atomic concept) 
(top) 
(bottom) 
(conjunction) 
(disjunction) 
(negation) 
(universal quantification) 
(existential quantification).

We use parentheses whenever we have to disambiguate concept expressions. For example, we write $(\exists P.D) \cap E$ to indicate that the concept $E$ is not in the scope of $\exists P$.

A first-order 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:

$$ \top^\mathcal{I} = \Delta^\mathcal{I}, $$
$$ \bot^\mathcal{I} = \emptyset, $$
$$ (C \cap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}, $$
$$ (C \cup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}, $$
$$ (\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}, $$
$$ (\forall P.C)^\mathcal{I} = \{d_1 \in \Delta^\mathcal{I} \mid \forall d_2: (d_1, d_2) \in P^\mathcal{I} \rightarrow d_2 \in C^\mathcal{I}\}, $$
$$ (\exists P.C)^\mathcal{I} = \{d_1 \in \Delta^\mathcal{I} \mid \exists d_2: (d_1, d_2) \in P^\mathcal{I} \land d_2 \in C^\mathcal{I}\}. $$

A concept is satisfiable if there exists an interpretation $\mathcal{I}$ such that $C^\mathcal{I}$ is non-empty and unsatisfiable otherwise. We say that $C$ is subsumed by $D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$ for every interpretation $\mathcal{I}$.

In knowledge representation systems based on description logics, the knowledge base includes both an intensional part, called the "terminology" or simply the $TBox$, and an
extensional part, called the assertional box or simply the ABox. The TBox is constituted by a set of inclusion statements of the form

\[ C \subseteq D, \]

where \( C \), \( D \) are concepts. Inclusion statements are interpreted as set inclusions: an interpretation \( \mathcal{I} \) satisfies \( C \subseteq D \) if \( C^\mathcal{I} \subseteq D^\mathcal{I} \). An interpretation \( \mathcal{I} \) is a model for a TBox \( \mathcal{T} \) if it satisfies every inclusion in \( \mathcal{T} \). As pointed out in [14], inclusions are more general than definitions, since definitions like \( A = C \) can be expressed as \( A \subseteq C \) and \( C \subseteq A \). In particular, TBoxes consisting of arbitrary inclusion statements may contain cyclic definitions, which are interpreted under so-called descriptive semantics [45].

The ABox is constituted by a set of assertions that specify either that an individual is an instance of a concept or that a pair of individuals is an instance of a role. Let \( \mathcal{O} \) be an alphabet of symbols, called individuals. Syntactically, assertions are expressed in terms of membership statements of the form

\[ C(a), \quad P(a, b), \]

where \( a \) and \( b \) are individuals, \( C \) is a concept, and \( P \) is a role. The assertion \( C(a) \) means that \( a \) is an instance of \( C \), while \( P(a, b) \) means that \( a \) is related to \( b \) by means of \( P \). In order to give a formal semantics to assertions, we extend the interpretation to the elements of \( \mathcal{O} \). In particular, we interpret each individual by a unique domain element: if \( a \neq b \) then \( a \) and \( b \) are given different interpretations (Unique Name Assumption). An assertion \( C(a) \) is satisfied by \( \mathcal{I} \) if \( a^\mathcal{I} \in C^\mathcal{I} \). Similarly, an assertion \( P(a, b) \) is satisfied by \( \mathcal{I} \) if \( (a^\mathcal{I}, b^\mathcal{I}) \in P^\mathcal{I} \). A first-order interpretation \( \mathcal{I} \) is a first-order model for an ABox \( \mathcal{A} \) if it satisfies every assertion in \( \mathcal{A} \).

An \( \mathcal{ALC}\)-knowledge base is a pair \( \Sigma = (\mathcal{T}, \mathcal{A}) \), where \( \mathcal{T} \) is a set of inclusion statements, and \( \mathcal{A} \) is a set of membership assertions whose concepts and roles belong to the language \( \mathcal{ALC} \). A first-order interpretation \( \mathcal{I} \) is a first-order model for \( \Sigma = (\mathcal{T}, \mathcal{A}) \) if it is a model for both \( \mathcal{T} \) and \( \mathcal{A} \). We say that \( \Sigma \) is satisfiable if it has a first-order model. The set of models of \( \Sigma \) is denoted as \( \mathcal{M}(\Sigma) \). The knowledge base \( \Sigma \) entails \( \sigma \) (written \( \Sigma \models \sigma \)), where \( \sigma \) is either an inclusion statement or a membership assertion, if every model in \( \mathcal{M}(\Sigma) \) satisfies \( \sigma \).

The most common kind of query to a knowledge base \( \Sigma \) is asking whether \( C(a) \) (or \( P(a, b) \)) is entailed by \( \Sigma \). Notice that the semantics associated with concept languages is an open-world semantics, that is, no world closure is assumed. Consequently, the answer to a query \( Q \) will be YES if \( Q \) is true in every model for \( \Sigma \), NO if \( Q \) is false in every model, and UNKNOWN otherwise. Query answering over \( \mathcal{ALC}\)-knowledge bases is easily reducible to satisfiability (see, for example, [14]). A calculus for knowledge base satisfiability in \( \mathcal{ALC} \) was first presented in [23] and shown to be complete and terminating in nondeterministic exponential time [14].

2.2. The epistemic concept language \( \mathcal{ALCK} \)

The epistemic concept language \( \mathcal{ALCK} \), proposed for the first time in [21], is an extension of \( \mathcal{ALC} \) with an epistemic operator \( K \). Following [51], we use \( KC \).
to denote the set of individuals known to be instances of the concept $C$ in every model for the knowledge base. The syntax of $\mathcal{ALCK}$ is the following (where $C, D$ denote concepts, $R$ denotes a role, $A$ denotes an atomic concept and $P$ an atomic role):

$$
C, D \rightarrow A \mid \text{(atomic concept)}
$$

$$
\top \mid \text{(top)}
$$

$$
\bot \mid \text{(bottom)}
$$

$$
C \sqcap D \mid \text{(conjunction)}
$$

$$
C \sqcup D \mid \text{(disjunction)}
$$

$$
\neg C \mid \text{(negation)}
$$

$$
\forall R.C \mid \text{(universal quantification)}
$$

$$
\exists R.C \mid \text{(existential quantification)}
$$

$$
KC \mid \text{(epistemic concept)}
$$

$$
R \rightarrow P \mid \text{(atomic role)}
$$

$$
KP \mid \text{(epistemic role)}
$$

The semantics of $\mathcal{ALCK}$ relies on a Kripke style possible-world semantics, as proposed in [37,40,51]. Following an idea that can be traced back to Hintikka, we want to interpret $K$ as an epistemic operator. However, since we allow the epistemic operator $K$ to appear inside and outside of quantified concepts $\exists R.C$, $\forall R.C$, some issues typical of first-order modal systems arise. As noted by Fitting [31, p. 420], there is no single correct semantics, and the choice of a first-order semantics must depend on the application.

In [37,40,51], all variables range over a fixed domain of “parameters”, interpreted in the same way in every world. That is, there is an infinite set of Rigid Designators, and this is the domain for every world. Note that Rigid Designators enforce the Unique Name Assumption we made in the non-modal setting of the previous section. Moreover, having a constant domain allows us to interpret the concept $KC$ as the same set in every world, extending to the modal setting the idea of interpreting concepts as sets. In summary, the choices of a Constant Domain and of Rigid Designators are the most suitable for our modal first-order logic. This just amounts to say that the set of possible individuals is fixed and known in advance, an assumption which is perfectly reasonable in knowledge bases.

Of course, the same choices would not be appropriate if $\mathcal{ALCK}$ were used in an application where individuals can be created and destroyed, or the referent of a name can change depending on the world. However, up to now we do not envisage the use of $\mathcal{ALCK}$ in such settings.

Following our choices, the domain of each interpretation is the set of all individuals $O$. Therefore, from now on $O = \Delta^T$ and, consequently, we denote the interpretation of $a$ simply as $a$ itself.
An epistemic interpretation is a pair \((\mathcal{I}, \mathcal{W})\) where \(\mathcal{I}\) is a first-order interpretation and \(\mathcal{W}\) is a set of first-order interpretations. Every epistemic interpretation gives rise to a unique mapping \(\mathcal{I}, \mathcal{W}\) associating concepts and roles with subsets of \(\mathcal{O}\) and \(\mathcal{O} \times \mathcal{O}\), respectively, such that the following equations are satisfied:

\[
\begin{align*}
\top_{\mathcal{I}, \mathcal{W}} &= \mathcal{O}, \\
\bot_{\mathcal{I}, \mathcal{W}} &= \emptyset, \\
A_{\mathcal{I}, \mathcal{W}} &= A^{\mathcal{I}}, \\
p_{\mathcal{I}, \mathcal{W}} &= p^{\mathcal{I}}, \\
(C \cap D)_{\mathcal{I}, \mathcal{W}} &= C_{\mathcal{I}, \mathcal{W}} \cap D_{\mathcal{I}, \mathcal{W}}, \\
(C \cup D)_{\mathcal{I}, \mathcal{W}} &= C_{\mathcal{I}, \mathcal{W}} \cup D_{\mathcal{I}, \mathcal{W}}, \\
(\neg C)_{\mathcal{I}, \mathcal{W}} &= \mathcal{O} \setminus C_{\mathcal{I}, \mathcal{W}}, \\
(\forall R.C)_{\mathcal{I}, \mathcal{W}} &= \{a \in \mathcal{O} \mid \forall b. (a, b) \in R_{\mathcal{I}, \mathcal{W}} \rightarrow b \in C_{\mathcal{I}, \mathcal{W}}\}, \\
(\exists R.C)_{\mathcal{I}, \mathcal{W}} &= \{a \in \mathcal{O} \mid \exists b. (a, b) \in R_{\mathcal{I}, \mathcal{W}} \land b \in C_{\mathcal{I}, \mathcal{W}}\}, \\
(KC)_{\mathcal{I}, \mathcal{W}} &= \bigcap_{\mathcal{J} \in \mathcal{W}} (C_{\mathcal{J}, \mathcal{W}}), \quad (1) \\
(KP)_{\mathcal{I}, \mathcal{W}} &= \bigcap_{\mathcal{J} \in \mathcal{W}} (P_{\mathcal{J}, \mathcal{W}}). \quad (2)
\end{align*}
\]

Notice that, since the domain is the same in all first-order interpretations belonging to \(\mathcal{W}\), it is meaningful to refer in (1) and (2) to the intersection of the extensions of a concept in different first-order interpretations. It follows that \(KC\) is interpreted as the set of objects that are instances of \(C\) in every first-order interpretation belonging to \(\mathcal{W}\). In this sense, \(KC\) represents those individuals known to be instances of \(C\) in \(\mathcal{W}\). Observe also that if one discards \(K\) and \(\mathcal{W}\) in the equations, one obtains the standard semantics of \(\mathcal{ALC}\).

An \(\mathcal{ALCK}\)-knowledge base \(\Psi\) is a pair \(\langle \mathcal{T}, \mathcal{A} \rangle\), where \(\mathcal{T}\) is a set of inclusion statements, and \(\mathcal{A}\) is a set of membership assertions whose concepts and roles belong to the language \(\mathcal{ALCK}\).

Inclusion statements are interpreted in terms of set inclusion: an epistemic interpretation \((\mathcal{I}, \mathcal{W})\) satisfies \(C \sqsubseteq D\) if \(C_{\mathcal{I}, \mathcal{W}} \subseteq D_{\mathcal{I}, \mathcal{W}}\). An epistemic interpretation \((\mathcal{I}, \mathcal{W})\) satisfies a TBox \(\mathcal{T}\) if it satisfies every inclusion in \(\mathcal{T}\).

An assertion \(C(a)\) is satisfied by \((\mathcal{I}, \mathcal{W})\) if \(a \in C_{\mathcal{I}, \mathcal{W}}\). Similarly, an assertion \(P(a, b)\) is satisfied by \((\mathcal{I}, \mathcal{W})\) if \((a, b) \in P_{\mathcal{I}, \mathcal{W}}\). An epistemic interpretation \((\mathcal{I}, \mathcal{W})\) satisfies an ABox \(\mathcal{A}\) if it satisfies every assertion in \(\mathcal{A}\).

An epistemic model for an \(\mathcal{ALCK}\)-knowledge base \(\Psi = \langle \mathcal{T}, \mathcal{A} \rangle\) is a maximal non-empty set \(\mathcal{W}\) of first-order interpretations such that for each \(\mathcal{I} \in \mathcal{W}\), the epistemic interpretation \((\mathcal{I}, \mathcal{W})\) satisfies both \(\mathcal{T}\) and \(\mathcal{A}\). An \(\mathcal{ALCK}\)-knowledge base \(\Psi\) is said to be satisfiable if there exists an epistemic model for \(\Psi\), unsatisfiable otherwise. The
knowledge base $\Psi$ logically implies an assertion $\sigma$, written $\Psi \models \sigma$, if for every epistemic model $W$ of $\Psi$, we have that for every $I \in W$, the epistemic interpretation $(I, W)$ satisfies $\sigma$.

Note that the maximality of $W$ rules out proper subsets of $W$ as epistemic models, even if for each $I \in W$, the epistemic interpretation $(I, W)$ satisfies both $T$ and $A$. This maximality condition is intended to capture the idea of minimizing knowledge. In fact, by adding an interpretation to a set $W$ one can falsify any sentence that is satisfied in $W$, thus reducing the set of known facts. Notice that the semantics of an $\mathcal{ALCK}$-knowledge base can be rephrased in terms of an accessibility relation on a set of possible worlds, each of which is a first-order interpretation. More specifically, each epistemic model can be viewed as a possible-world structure in which each world is connected with all the others. Therefore, the accessibility relation would be an equivalence relation, as in the modal system S5. Based on this property, the epistemic models of a knowledge base correspond to those S5-models with a maximal set of worlds (i.e., such that no world can be added without compromising the property of being a model). In particular, in [27,28,44] it is shown that the semantics of $\mathcal{ALCK}$ corresponds to that of the ground nonmonotonic version of the modal logic S5 (see [35]).

Next we introduce the notion of answer to a query. Given an $\mathcal{ALCK}$-knowledge base $\Psi$, an $\mathcal{ALCK}$-concept $C$, and an individual $a$, the answer to the query $C(a)$ posed to $\Psi$ is

- **YES**, if $\Psi \models C(a)$,
- **NO**, if $\Psi \models \neg C(a)$,
- **UNKNOWN** otherwise.

Moreover, if we denote as $\mathcal{O}_\Psi$ the set of individuals appearing in $\Psi$, then the answer set of $C$ with respect to $\Psi$ is the set of individuals $\{a \in \mathcal{O}_\Psi \mid \Psi \models C(a)\}$. Notice that, in the answer set, we consider only individuals appearing in the knowledge base, as customary in query answering systems.

Following [37,51] we initially (Sections 3-5) do not admit the epistemic operator in the knowledge base, and consider the problem of answering epistemic queries to a non-modal knowledge base, focusing on knowledge bases without TBox. We use the symbol $\Sigma$ to denote such special knowledge bases. Therefore, from this point on, we assume a knowledge base $\Sigma$ to be just a set of membership assertions (i.e., an ABox) in $\mathcal{ALC}$.

We then consider TBox statements in Sections 6.2 and 6.3, where we introduce the notion of *rule*, which is captured by a particular class of epistemic sentences in the TBox.

Observe that, if $\Sigma$ is an $\mathcal{ALC}$-knowledge base, i.e., it does not contain epistemic operators, then its unique epistemic model is $M(\Sigma)$. In the following section we develop a calculus for answering epistemic queries (i.e., queries of the form $C(a)$, where $C$ is an $\mathcal{ALCK}$-concept) to an $\mathcal{ALC}$-knowledge base. To this end we exploit the following property: for any $\mathcal{ALCK}$-concept $C$, individual $a$, and $\mathcal{ALC}$-knowledge base $\Sigma$ it holds that $\Sigma \models C(a)$ if and only if there is no $I \in M(\Sigma)$ such that the epistemic interpretation $(I, M(\Sigma))$ satisfies $\Sigma \cup \{\neg C(a)\}$ (see also Proposition 3.1).
3. The calculus for answering queries

Methods for answering epistemic queries were designed in [3,37,51]. In [51] a procedure is presented that is sound and complete if the query satisfies some syntactic constraints. However, not all epistemic concepts belonging to \( \text{ALCK} \) satisfy those constraints; for example, the formula corresponding to \( \exists P.\neg KC(a) \) is not admissible in [51]. The method proposed in [37] has been conceived within the more general framework of first-order predicate calculus augmented with the epistemic operator, and its specialization to the case of description logics does not yield an effective procedure. The approach is further developed in [3], where an epistemic concept language based on the language of \text{CLASSIC} is studied. The method proposed for query answering is based on a translation of the epistemic query into an equivalent first-order query. However, the concept language is much less expressive than \( \text{ALC} \). Therefore, none of the previous approaches can be directly applied to our setting.

In this section we present a general method for answering epistemic queries to an \( \text{ALC} \)-knowledge base. The method computes with so-called constraint systems, which are closely related to tableaux branches in tableaux-based calculi. We introduce constraint systems and study their properties in Section 3.1. Constraint systems are manipulated by completion rules, which are introduced and discussed in Section 3.2.

3.1. Constraint systems

We recall that \( O \) is the alphabet of individuals. Generic elements of \( O \) are denoted as \( a, b, c, d, e \). We also introduce \( V \), a set of variables, denoted by \( x, y \). The elements of \( O \cup V \) (called objects) will be denoted by \( w, z \). A constraint is a syntactic structure of one of the forms

\[
 w: C, \quad wRz,
\]

where \( C \) is an \( \text{ALCK} \)-concept and \( R \) is an \( \text{ALCK} \)-role. A constraint system is a finite set of constraints of the above forms. Observe the strict analogy between constraints and membership statements, and between constraint systems and ABoxes.

We denote by \( O_S \) the set of individuals appearing in a constraint system \( S \). In order to assign a meaning to constraints, we need the following definitions. An assignment \( \alpha(\cdot) \) is a function from \( V \cup O \) to \( O \) such that for each \( d \in O \) we have \( \alpha(d) = d \). Let \((I,W)\) be an epistemic interpretation, and let \( \alpha \) be an assignment. The triple \((I,W,\alpha)\) is said to satisfy the constraint \( w: C \) if \( \alpha(w) \in C^{I,W} \). Similarly, \((I,W,\alpha)\) satisfies the constraint \( wRz \) if \( (\alpha(w),\alpha(z)) \in R^{I,W} \). Let \( S \) be a constraint system. The triple \((I,W,\alpha)\) is a solution of \( S \) if \((I,W,\alpha)\) satisfies all of its constraints. If \( \Sigma \) is an \( \text{ALC} \)-knowledge base, then \( S \) is said to be \( \Sigma \)-solvable if there is a triple \((I,M(\Sigma),\alpha)\) that is a solution of \( S \). If there is no such solution, then \( S \) is said to be \( \Sigma \)-unsolvable.

Given an \( \text{ALC} \)-knowledge base \( \Sigma \), we define \( S_\Sigma \) to be the constraint system that includes one constraint \( a: C \) for each assertion \( C(a) \) of \( \Sigma \), and one constraint \( aPb \) for each assertion \( P(a,b) \) of \( \Sigma \) (see [34]). The next proposition shows that answering an epistemic query posed to an \( \text{ALC} \)-knowledge base \( \Sigma \) can be reduced to checking a particular constraint system for \( \Sigma \)-unsolvability.
Proposition 3.1. Let $\Sigma$ be an $\mathcal{ALC}$-knowledge base, $C$ an $\mathcal{ALCK}$-concept, and $a$ an individual. Then, $\Sigma \models C(a)$ if and only if $S_\Sigma \cup \{a: -C\}$ is $\Sigma$-unsolvable.

Proof. Since $\Sigma$ contains no epistemic operator it has just one epistemic model, namely $\mathcal{M}(\Sigma)$, the set of all its first-order models.

$(\Rightarrow)$ Suppose that the constraint system $S_\Sigma \cup \{a: -C\}$ is $\Sigma$-solvable. Then there is a triple $(I, \mathcal{M}(\Sigma), \alpha)$ that satisfies all constraints in $S_\Sigma \cup \{a: -C\}$. Since the triple satisfies $S_\Sigma$, we have that $I \in \mathcal{M}(\Sigma)$. This implies that $(I, \mathcal{M}(\Sigma))$ is an epistemic interpretation that does not satisfy $C(a)$. Hence $\Sigma \not\models C(a)$.

$(\Leftarrow)$ Assume that $\Sigma \not\models C(a)$. This means that there is an epistemic interpretation $(I, \mathcal{M}(\Sigma))$ with $I \in \mathcal{M}(\Sigma)$ that does not satisfy $C(a)$. Hence, $(I, \mathcal{M}(\Sigma))$ satisfies $\neg C(a)$. Observe that $S_\Sigma \cup \{a: -C\}$ contains no variables. Thus, for any assignment $\alpha$, the triple $(I, \mathcal{M}(\Sigma), \alpha)$ is a solution of $S_\Sigma \cup \{a: -C\}$, i.e., $S_\Sigma \cup \{a: -C\}$ is $\Sigma$-solvable.

An $\mathcal{ALCK}$-concept is said to be in negation normal form if every negation appearing in it is either of the form $\neg A$ or of the form $\neg \mathcal{K}C$. It is easy to see that every $\mathcal{ALCK}$-concept can be rewritten in linear time into an equivalent concept in negation normal form (see [56]), which we call the negation normal form of $C$. In the rest of the paper we assume that all concepts are in negation normal form unless stated otherwise. In particular, we assume that concepts in constraint systems are in negation normal form.

In the following we prove a number of properties of constraint systems that have to do with the role played by the individuals. We start by considering constraints on roles and show that there is a direct correspondence between constraints on roles and their interpretations in the epistemic models of the knowledge base.

Lemma 3.2. Let $\Sigma$ be a satisfiable $\mathcal{ALC}$-knowledge base, $a, b$ two individuals in $\mathcal{O}$, and $P$ an atomic role in $\Sigma$. Then $a \mathrel{P} b \in S_\Sigma$ if and only if $(a, b) \in P^I,\mathcal{M}(\Sigma)$ for all first-order models $I \in \mathcal{M}(\Sigma)$.

Proof. $(\Rightarrow)$ If $a \mathrel{P} b \in S_\Sigma$, then $P(a, b)$ is in $\Sigma$, and $\Sigma \models P(a, b)$. Hence, $(a, b) \in P^I,\mathcal{M}(\Sigma)$ for every first-order model $I \in \mathcal{M}(\Sigma)$.

$(\Leftarrow)$ We show that if $a \mathrel{P} b \notin S_\Sigma$ then there is some $I \in \mathcal{M}(\Sigma)$ such that $(a, b) \notin P^I,\mathcal{M}(\Sigma)$.

Since $\Sigma$ is satisfiable, it follows from the results in [1] that there exists a first-order model $J$ of $\Sigma$ such that the extension of every atomic concept $A$ and every atomic role $Q$ is finite. If $(a, b) \notin J,\mathcal{M}(\Sigma)$ then the claim follows. Otherwise let $d \in \mathcal{O}$ be an individual not appearing in the extension of any $A$ or $Q$ in $\Sigma$. We construct the first-order interpretation $I$ in such a way that the only difference with $J$ is that $d$ is added to the extensions of concepts and roles in $I$ so that $d$ behaves exactly as $b$ in $J$, except for the role $\mathrel{P}$, where $b$ is replaced by $d$:

- for every atomic concept $A$, let
  $$A^I = \begin{cases} A^J & \text{if } b \notin A^J, \\ A^J \cup \{d\} & \text{if } b \in A^J. \end{cases}$$
for every role \( Q \neq P \), let

\[
Q^I = \begin{cases}
Q^J \cup \{(c, d) \mid (c, b) \in Q^J\} \\
\quad \cup \{(d, c) \mid (b, c) \in Q^J\} & \text{if } (b, b) \notin Q^I,
\end{cases}
\]

\[
Q^I = \begin{cases}
Q^J \cup \{(c, d) \mid (c, b) \in Q^J, \, c \neq b\} \\
\quad \cup \{(d, c) \mid (b, c) \in Q^J, \, c \neq b\} \\
\quad \cup \{(d, d)\} & \text{if } (b, b) \in Q^I,
\end{cases}
\]

\[\text{let } P^I = (P^J \setminus \{(a, b)\}) \cup \{(a, d)\}.
\]

Now, one can verify the following two claims for every concept \( C \) by a simultaneous induction on the structure of concepts:

(i) \( c \in C' \) if and only if \( c \in C'' \), for every individual \( c \) with \( c \neq d \),

(ii) \( b \in C' \) if and only if \( d \in C'' \).

Since \( J \) is a model of \( \Sigma \), this implies that \( I \), too, is a model: In \( \Sigma \), the individual \( d \) does not appear in any constraint. Hence, \( I \) satisfies every constraint of the form \( c: C \) in \( \Sigma \). By construction, \( I \) also satisfies every constraint of the form \( c Q c' \) and \( c P c' \).

We now show that the interpretation of the individuals which do not occur in a constraint system \( S \) is immaterial, that is, given a pair of such individuals, by exchanging them in a solution of \( S \) one obtains another solution of \( S \). We first prove a preliminary result concerning the exchange of individuals in the models of a first-order knowledge base. To this end we need the following definitions. For every pair \( d, e \) of elements of \( \mathcal{O} \), we define a function \( \rho_{d,e}: \mathcal{O} \rightarrow \mathcal{O} \) by:

- \( \rho_{d,e}(d) := e \), \( \rho_{d,e}(e) := d \),
- \( \rho_{d,e}(a) := a \) for any other \( a \in \mathcal{O} \).

Obviously, \( \rho_{d,e} \) is bijective, and \( \rho_{d,e} \circ \rho_{d,e} \) is the identity on \( \mathcal{O} \).

Let \( \Sigma \) be an \( \mathcal{ALC} \)-knowledge base. For any first-order interpretation \( I \) of \( \Sigma \), we define \( I_{d,e} \) as follows:

- \( A^{I_{d,e}} = \{\rho_{d,e}(a) \mid a \in A^I\} \) for every \( A \),
- \( P^{I_{d,e}} = \{(\rho_{d,e}(a), \rho_{d,e}(b)) \mid (a, b) \in P^I\} \) for every \( P \).

According to this definition, \( I_{d,e} \) is the first-order interpretation obtained from \( I \) by swapping \( d \) and \( e \) in the extension of each concept and role. The next lemma proves that, if \( d \) and \( e \) do not appear in \( \Sigma \), then the property of being a first-order model for \( \Sigma \) is preserved by the swapping.

**Lemma 3.3.** Let \( \Sigma \) be an \( \mathcal{ALC} \)-knowledge base, and let \( d, e \in \mathcal{O} \setminus \mathcal{O}_\Sigma \). If \( I \in \mathcal{M}(\Sigma) \), then \( I_{d,e} \in \mathcal{M}(\Sigma) \).

**Proof.** By definition of \( I_{d,e} \), the first-order interpretations \( I \) and \( I_{d,e} \) are isomorphic and \( \rho_{d,e} \) is an isomorphism from \( I \) to \( I_{d,e} \) (see [58, Definition 3.3.1] for the definition of isomorphisms between interpretations). The Isomorphism Lemma of predicate logic says that isomorphic interpretations satisfy the same sentences [58, Lemma 3.3.3]. Since \( \Sigma \) can be expressed as a set of first-order sentences, this yields the claim. \( \square \)
From the above lemma, we can easily prove that for any ALC-knowledge base $\Sigma$, and for any pair $d, e \notin \mathcal{O}_\Sigma$, the operation $\cdot_{d,e}$ of exchanging $d$ and $e$ in a first-order interpretation is a bijection on $\mathcal{M}(\Sigma)$. Formally, this is expressed by the following lemma.

**Lemma 3.4.** Let $\Sigma$ be an ALC-knowledge base, and $d, e \in \mathcal{O} \setminus \mathcal{O}_\Sigma$.

(i) For each $\mathcal{I} \in \mathcal{M}(\Sigma)$, there exists a $\mathcal{J} \in \mathcal{M}(\Sigma)$ such that $\mathcal{J}_{d,e} = \mathcal{I}$.

(ii) If $\mathcal{J}_{d,e} = \mathcal{J}'_{d,e}$, then $\mathcal{J} = \mathcal{J}'$, for any $\mathcal{J}, \mathcal{J}' \in \mathcal{M}(\Sigma)$.

**Proof.** To prove the first part, let $\mathcal{J} = \mathcal{I}_{d,e}$. Lemma 3.3 implies that $\mathcal{I}_{d,e} \in \mathcal{M}(\Sigma)$.

Moreover, $\mathcal{J}_{d,e} = (\mathcal{I}_{d,e})_{d,e} = \mathcal{I}$ by the definition of $\cdot_{d,e}$.

To prove the second part, suppose that $\mathcal{J}_{d,e} = \mathcal{J}'_{d,e}$. Then $\mathcal{J} = (\mathcal{J}_{d,e})_{d,e} = (\mathcal{J}'_{d,e})_{d,e} = \mathcal{J}'$. □

Our next goal is to prove that if a constraint system $S$ is satisfied by a triple $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$, then any other triple obtained by exchanging a pair of individuals appearing neither in $\Sigma$ nor in $S$ satisfies $S$ too. Observe that, differently from $\Sigma$, the constraint system $S$ may contain epistemic concepts and roles. Therefore, we next address the effects of applying $\cdot_{d,e}$ in the framework of epistemic interpretations. With abuse of notation we write $\cdot_{d,e}$ applied to sets of elements of $\mathcal{O}$ to denote the set resulting from the application of $\cdot_{d,e}$ to every element of the set. Similarly, when $\cdot_{d,e}$ is applied to the interpretation of a role, it denotes the set of pairs obtained by applying $\cdot_{d,e}$ to each element of every pair.

**Lemma 3.5.** Let $\Sigma$ be an ALC-knowledge base, and let $d, e \in \mathcal{O} \setminus \mathcal{O}_\Sigma$. Then for any ALCK-concept $C$ and any ALCK-role $R$ we have

(i) $C^{\mathcal{I}_{d,e},\mathcal{M}(\Sigma)} = \rho_{d,e}(C^{\mathcal{I},\mathcal{M}(\Sigma)}) = \{\rho_{d,e}(c) \mid c \in C^{\mathcal{I},\mathcal{M}(\Sigma)}\},$

(ii) $R^{\mathcal{I}_{d,e},\mathcal{M}(\Sigma)} = \rho_{d,e}(R^{\mathcal{I},\mathcal{M}(\Sigma)}) = \{(\rho_{d,e}(c), \rho_{d,e}(c')) \mid (c, c') \in R^{\mathcal{I},\mathcal{M}(\Sigma)}\}.$

**Proof.** Note that, if concepts and roles did not contain epistemic operators, the lemma would be a consequence of the Isomorphism Theorem for predicate logic (cf. proof of Lemma 3.3). For the sake of completeness, we provide a full proof of the lemma. The proof is by induction on the structure of concepts and roles.

For $\top$ and $\bot$ the lemma obviously holds. For atomic concepts and roles, the claim is an immediate consequence of the definition of $\mathcal{I}_{d,e}$. For concepts of the form $C_1 \sqcup C_2$, $C_1 \sqcap C_2$, and $\neg C$, we exploit the identities

$\rho_{d,e}(C_1^{\mathcal{I},\mathcal{M}(\Sigma)}) \cup \rho_{d,e}(C_2^{\mathcal{I},\mathcal{M}(\Sigma)}) = \rho_{d,e}(C_1^{\mathcal{I},\mathcal{M}(\Sigma)} \cup C_2^{\mathcal{I},\mathcal{M}(\Sigma)}),$

$\rho_{d,e}(C_1^{\mathcal{I},\mathcal{M}(\Sigma)}) \cap \rho_{d,e}(C_2^{\mathcal{I},\mathcal{M}(\Sigma)}) = \rho_{d,e}(C_1^{\mathcal{I},\mathcal{M}(\Sigma)} \cap C_2^{\mathcal{I},\mathcal{M}(\Sigma)}),$

$\mathcal{O} \setminus \rho_{d,e}(C^{\mathcal{I},\mathcal{M}(\Sigma)}) = \rho_{d,e}(\mathcal{O} \setminus C^{\mathcal{I},\mathcal{M}(\Sigma)}).$

All three identities follow from basic set theoretic results about set operations and mappings. Note that the first identity holds for arbitrary mappings while for the second and third we need that $\rho_{d,e}$ is a bijection on $\mathcal{O}$. 
As an example, we give a complete proof for concepts of the form $\neg C$:

\[-C_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)} = \mathcal{O} \setminus C_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)} = \mathcal{O} \setminus \rho_{d,e}(C_{\mathcal{J},\mathcal{M}(\Sigma)}) = \rho_{d,e}(\mathcal{O} \setminus C_{\mathcal{J},\mathcal{M}(\Sigma)}) = \rho_{d,e}(-C_{\mathcal{J},\mathcal{M}(\Sigma)}).
\]

Here, the first and the fourth identity follow from the definition of epistemic interpretations, the second one uses the induction hypothesis, and the third one has been explained above.

Now, consider concepts of the form $\exists R.C$. For an arbitrary $a \in \mathcal{O}$ we have $a \in (\exists R.C)_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)}$ if and only if there is an element $b$ such that $(a, b) \in R_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)}$ and $b \in C_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)}$. By the induction hypothesis, this holds if and only if there is a $b$ such that $(a, b) \in \rho_{d,e}(R_{\mathcal{J},\mathcal{M}(\Sigma)})$ and $b \in \rho_{d,e}(C_{\mathcal{J},\mathcal{M}(\Sigma)})$, which can be rewritten as $(\rho^{-1}_{d,e}(a), \rho^{-1}_{d,e}(b)) \in R_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)}$ and $\rho^{-1}_{d,e}(b) \in C_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)}$. The latter means that $\rho^{-1}_{d,e}(a) \in (\exists R.C)_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)}$. Given that $\rho_{d,e}$ is bijective, this is equivalent to the statement that $a \in \rho_{d,e}((\exists R.C)_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)})$. Since $a$ was chosen arbitrarily, this shows the claim. For concepts of the form $\forall R.C$ the proof is similar.

Finally, we consider epistemic concepts and roles. For epistemic concepts we can derive the following sequence of identities:

\[\begin{align*}
(KC)_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)} &= \bigcap_{\mathcal{J} \in \mathcal{M}(\Sigma)} C_{\mathcal{J},\mathcal{M}(\Sigma)} \\
 &= \bigcap_{\mathcal{J} \in \mathcal{M}(\Sigma)} C_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)} \\
 &= \bigcap_{\mathcal{J} \in \mathcal{M}(\Sigma)} \rho_{d,e}(C_{\mathcal{J},\mathcal{M}(\Sigma)}) \\
 &= \rho_{d,e}\left(\bigcap_{\mathcal{J} \in \mathcal{M}(\Sigma)} C_{\mathcal{J},\mathcal{M}(\Sigma)}\right) \\
 &= \rho_{d,e}((KC)_{d,e}^{\mathcal{J},\mathcal{M}(\Sigma)}).
\end{align*}\]

Here, the first and the fifth identity follow from the definition of epistemic interpretations (Eq. 1), the second relies on the fact that $\cdot_{d,e}$ is a bijection on $\mathcal{M}(\Sigma)$ (Lemma 3.4), the third one uses the induction hypothesis, and the fourth one follows from basic set theory, since $\rho_{d,e}$ is a bijection on $\mathcal{O}$. The case of epistemic roles is analogous. □

We are now ready to prove that by exchanging in a solution a pair of individuals not occurring in the constraint system, we obtain another solution of the constraint system. For any assignment $\alpha$, we define $\alpha_{d,e}$ as follows:

- $\alpha_{d,e}(y) := \rho_{d,e}(\alpha(y))$ for every variable $y$,
- $\alpha_{d,e}(a) := a$ for every individual $a$.

Note that $\alpha_{d,e}$ is still an assignment.
Lemma 3.6. Let $\Sigma$ be an $\mathcal{ALC}$-knowledge base, $S$ be a constraint system and $d, e$ be any pair of individuals in $O \setminus (O_S \cup O_\Sigma)$. If $(I, M(\Sigma), \alpha)$ is a solution of $S$, then $(I_{d,e}, M(\Sigma), \alpha_{d,e})$ is also a solution of $S$.

Proof. First, observe that for any object $w$ occurring in $S$, we have that $\alpha_{d,e}(w) = \rho_{d,e}(\alpha(w))$. Indeed, if $w$ is a variable, then the equality holds by definition; if $w$ is an individual, then $w \neq d$ and $w \neq e$, and therefore, $\alpha_{d,e}(w) = \alpha(w) = \rho_{d,e}(\alpha(w))$, by the definition of $\rho_{d,e}$. Based on this property, we prove that if a constraint of $S$ is satisfied by $(I, M(\Sigma), \alpha)$, then it is also satisfied by $(I_{d,e}, M(\Sigma), \alpha_{d,e})$.

Let $w: C$ be a concept constraint in $S$. If $(I, M(\Sigma), \alpha)$ satisfies $w: C$, then $\alpha(w) \in C_{I,M(\Sigma)}$. Hence, $\rho_{d,e}(\alpha(w)) \in \rho_{d,e}(C_{I,M(\Sigma)})$. Now, $\rho_{d,e}(\alpha(w)) = \alpha_{d,e}(w)$ as shown above, and $\rho_{d,e}(C_{I,M(\Sigma)}) = C_{I_{d,e},M(\Sigma)}$ by Lemma 3.5. Thus, $\alpha_{d,e}(w) \in C_{I_{d,e},M(\Sigma)}$, which implies that $(I_{d,e}, M(\Sigma), \alpha_{d,e})$ satisfies $w: C$.

The proof for role constraints $w_1 R w_2$ is similar.  

3.2. Completion rules

Our method to answer an epistemic query posed to an $\mathcal{ALC}$-knowledge base $\Sigma$ is based on checking the $\Sigma$-solvability of the constraint system associated with the query.

In order to check the $\Sigma$-solvability of a constraint system, we apply a set of so-called completion rules to it, and then verify whether the resulting system is free of obvious contradictions (called “clashes” and to be defined later on).

We say that $w R z \Sigma$-holds in a constraint system $S$ if either

(i) $R$ is $P$, and $w P z \in S$, or

(ii) $R$ is $KP$, $w, z \in O$, and $w P z \in S$.

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

The set of completion rules we use is the following ($S$ denotes a constraint system):

(i) $S \rightarrow_1 \{w: C_1, w: C_2\} \cup S$
   if $w: C_1 \cap C_2$ is in $S$, and $w: C_1$ and $w: C_2$ are not both in $S$,

(ii) $S \rightarrow_2 \{w: D\} \cup S$
   if $w: C_1 \cup C_2$ is in $S$, neither $w: C_1$ nor $w: C_2$ is in $S$, and $D = C_1$ or $D = C_2$,

(iii) $S \rightarrow_3 \{\exists R x, x: C\} \cup S$
   if $w: \exists R C$ is in $S$, there is no $z$ such that both $w R z$ and $z: C$ are in $S$, and $x$ is a new variable.

(iv) $S \rightarrow_4 \{z: C\} \cup S$
   if $w: \forall R C$ is in $S$, $w R z \Sigma$-holds in $S$, and $z: C$ is not in $S$,

(v) $S \rightarrow_5 S[x/a]$
   if $x: KC, x: KC, x: KP$ $w$, or $w KP x$ is in $S$, and $a \in O_S \cup O_\Sigma \cup \{\iota\}$, where $\iota$ is any of the individuals in $O \setminus (O_S \cup O_\Sigma)$.

Observe that the applicability condition of the last rule $\rightarrow_5$ not only refers to $S$ but also to $O$ and $\Sigma$. In other words, the rule is parametric with respect to $O$ and $\Sigma$. However, since they are both always fixed and clear from the context, in order to simplify our notation we omit these parameters from the specification of the calculus.
The next proposition states that the application of any completion rule preserves the $\Sigma$-solvability of a constraint system.

**Proposition 3.7.** Let $\Sigma$ be an ALC-knowledge base, and let $S, S'$ be two constraint systems. Then:

(i) If $S'$ is obtained from $S$ by the application of one of the rules $\to_\pi$, $\to_\exists$, $\to_\forall$, then $S$ is $\Sigma$-solvable if and only if $S'$ is $\Sigma$-solvable.

(ii) If the $\to_{\bot}$-rule can be applied to $S$, and $S'$ and $S''$ are the two constraint systems that are obtained from $S$ by choosing $D = C_1$ or $D = C_2$ respectively in the conditions of the rule, then $S$ is $\Sigma$-solvable if and only if either $S'$ or $S''$ is $\Sigma$-solvable.

(iii) If $S'$ is obtained from $S$ by the application of the $\to_K$-rule, then $S$ is $\Sigma$-solvable if $S'$ is $\Sigma$-solvable. Furthermore, if $S$ is $\Sigma$-solvable and the $\to_K$-rule applies to a constraint in $S$, then the rule can be applied to that constraint in a way that yields a $\Sigma$-solvable constraint system $S'$.

**Proof.** The proof of (i) and (ii) easily follows from the results in [14,56]. Let us focus on the proof of (iii).

$\Leftarrow$ Let $S' = S[x/a]$, and suppose that $S'$ is $\Sigma$-solvable. Let $(I, M(\Sigma), \alpha)$ be one of its solutions. Let $\alpha'$ be the assignment that coincides with $\alpha$ except that $\alpha(x) = a$. It is easy to see that $(I, M(\Sigma), \alpha')$ is a solution of $S$.

$\Rightarrow$ If $S$ is $\Sigma$-solvable, then there is a triple $(I, M(\Sigma), \alpha)$ that satisfies every constraint in $S$. We show that for some $a \in O_S \cup O_S \cup \{\iota\}$, where $\iota$ is any of the individuals in $O \setminus (O_S \cup O_S)$, the constraint system $S[x/a]$ is $\Sigma$-solvable. We distinguish between two cases.

In the first case, there is an $a \in O_S \cup O_S$ such that $a = \alpha(x)$. It is obvious that in this case $(I, M(\Sigma), \alpha)$ satisfies $S[x/a]$ too, i.e., $S[x/a]$ is $\Sigma$-solvable.

In the second case, $\alpha(x) = d$, and $d \notin (O_S \cup O_S)$. By Lemma 3.6 we have that $(I_{d,x}, M(\Sigma), \alpha_{d,x})$ is a solution of $S$. Since $\alpha_{d,x}(x) = \iota$, the constraint system $S[x/\iota]$ is $\Sigma$-solvable. $\square$

A constraint system is said to be complete if no rule is applicable 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 (the $\to_K$- and the $\to_\bot$-rules), more than one completion can be obtained starting from one constraint system. To check the solvability of the constraint system, we now introduce the notion of $\Sigma$-clash.

Let $\Sigma$ be an ALC-knowledge base, and let $S$ be a constraint system. Then $S$ is said to contain a $\Sigma$-clash if at least one of the following conditions holds:

(i) $S$ contains a constraint of the form $w: \bot$;

(ii) $S$ contains two constraints of the form $w: A, w: \neg A$;

(iii) $S$ contains a constraint of the form $a: KC$, and there is at least one completion of $S_2 \cup \{a: C'\}$ without $\Sigma$-clashes, where $C'$ is the negation normal form of $\neg C$;

(iv) $S$ contains a constraint of the form $a: \neg KC$, and every completion of $S_2 \cup \{a: C'\}$ contains a $\Sigma$-clash, where $C'$ is the negation normal form of $\neg C$. 


(v) $S$ contains a constraint of the form $aKPb$, and $aPb \notin S_S$.

The completion calculus and the notion of clash are defined in such a way that constraint systems containing a clash are guaranteed to be unsatisfiable. The idea behind the $\neg K$-rule is that in principle, all the infinitely many objects of the domain would have to be substituted into a constraint with the $K$-operator when looking for an individual that satisfies the constraint. However, one can proceed in a much more clever way. It suffices to test those individuals that already have been mentioned and one which is a representative for those that have not. We illustrate the need to test $\iota$ in addition to the individuals present in a constraint system with the help of two examples.

**Example 3.8.** Suppose $\Sigma = \{\text{Student(}\text{susan})\}$ and consider the concept

$$C = \forall \text{FRIEND.} K\text{Student}$$

as a query on the individual susan. It is straightforward to see that

$$\Sigma \not\models \forall \text{FRIEND.} K\text{Student(}\text{susan}).$$

The constraint system $S = S_S \cup \{\text{susan: } C'\}$, where $C'$ is the negation normal form of $\neg C$, is the following one:

$$S = \{\text{susan: Student, susan: } \exists \text{FRIEND.} \neg K\text{Student}\}.$$

It is satisfied by any triple $(I, M(\Sigma), \alpha)$ with an interpretation $I$ where Susan has a friend other than herself.

Applying the completion rules to $S$ we obtain the constraint system

$$S_1 = S \cup \{\text{susanFRIENDx, x: } K\text{Student}\}.$$  

Because of the constraint $x: \neg K\text{Student}$, we must find a substitution for the variable $x$. The only individual in $O_S \cup O_S$ is susan. Substituting susan for $x$ yields the constraint system

$$S_2 = S \cup \{\text{susanFRIEND susan, susan: } \neg K\text{Student}\}.$$  

The system $S_2$ is clash free if and only if some completion of

$$S'_2 = S_2 \cup \{\text{susan: } \neg \text{Student}\} = \{\text{susan: Student, susan: } \neg \text{Student}\}$$

is clash free. However, $S'_2$ contains a clash.

Substituting $\iota$ for $x$ yields the constraint system

$$S_3 = S \cup \{\text{susanFRIEND } \iota, \iota: \neg K\text{Student}\}.$$  

The system $S_3$ is clash free if and only if some completion of

$$S'_3 = S_2 \cup \{\iota: \neg \text{Student}\} = \{\text{susan: Student, } \iota: \neg \text{Student}\}$$

is clash free. Obviously, $S'_3$ is clash free and complete.

The next example shows that it is necessary to use different $\iota$s, when there is more than one variable to be substituted. In other words, it is necessary to take into account
the individuals previously introduced in the constraint system where we make the substitution. It follows that we must pick the individual $\epsilon$ in $O \setminus (O_2 \cup O_3)$ and not simply in $O \setminus O_2$.

**Example 3.9.** Suppose $\Sigma = \{\text{Student}(\text{susan})\}$ and consider the concept

$$C = \forall \text{FRIEND}.(\text{KStudent} \cup \neg \text{Male}) \cup \forall \text{FRIEND}.(\text{KStudent} \cup \text{Male}).$$

We want to check whether $\Sigma \models C(\text{susan})$. Let $C'$ be the negation normal form of $\neg C$. The constraint system $S = S_2 \cup \{\text{susan}: C'\}$ is

$$S = \{\text{susan: Student, susan: FRIEND}(\neg \text{KStudent} \cap \text{Male}) \cap \exists \text{FRIEND}.(\neg \text{KStudent} \cap \neg \text{Male})\}.$$ 

Applying the completion rules to $S$ we obtain the constraint system

$$S_1 = S \cup \{\text{susanFRIEND}_x, x: \neg \text{KStudent}, x: \text{Male}, \text{susanFRIEND}_y, y: \neg \text{KStudent}, y: \neg \text{Male}\}.$$ 

The constraints on each of the variables $x, y$ force them to be different from $\text{susan}$, since, according to $\Sigma$, $\text{susan}$ is known to be a student. Hence, the only way to obtain a clash free completion is to substitute $\epsilon$ for $x$ and $y$. Since no individual can be male and not male at the same time, we have to substitute two distinct objects $\epsilon_1, \epsilon_2$ with $\epsilon_1 \neq \epsilon_2$: the constraint system $S_1[x/\epsilon_1][y/\epsilon_2]$ is clash free. This reflects the fact that $\Sigma \not\models C(\text{susan})$.

### 3.3. Decidability

We conclude this section by showing that $\Sigma$-solvability of $\text{ALCK}$-constraint systems is decidable. The next theorem enables us to check whether a complete constraint system is $\Sigma$-solvable, by looking for $\Sigma$-clashes.

**Theorem 3.10.** Let $\Sigma$ be an $\text{ALCK}$-knowledge base, and let $S$ be a constraint system. Then $S$ is $\Sigma$-solvable if and only if there exists at least one completion of $S$ that contains no $\Sigma$-clash.

**Proof.** The proof is by induction on the number $k$ of occurrences of the epistemic operator in the constraint system. If $k = 0$, then the theorem follows from the results in [11]. For $k > 0$, the induction hypothesis tells us that any constraint system $S'$ with $h$ occurrences (where $h < k$) of the epistemic operator is $\Sigma$-solvable if and only if there is a completion of $S'$ that contains no $\Sigma$-clash. Let $S$ be a constraint system with $k$ occurrences of the epistemic operator. We prove two claims.

**Claim 1.** If there exists at least one completion of $S$ that contains no $\Sigma$-clash, then $S$ is $\Sigma$-solvable.

**Claim 2.** If every completion of $S$ contains a $\Sigma$-clash, then $S$ is $\Sigma$-unsolvable.
Proof of Claim 1. Suppose that there exists a completion $S'$ of $S$ that contains no $\Sigma$-clash. We use $S'$ to define a triple $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$. First, let $\alpha$ map injectively each variable $x$ to a distinct element of $\mathcal{O} \setminus \mathcal{O}_S$. Second, define $\mathcal{I}$ as follows:

- for every atomic concept $A$ and every $d \in \mathcal{O}$, let $d \in A^\mathcal{I}$ if and only if there is a $w$ such that $\alpha(w) = d$ and $w: A$ is in $S'$;
- for every atomic role $P$ and every $d, e \in \mathcal{O}$, let $(d, e) \in P^\mathcal{I}$ if and only if there are $w, z$ such that $\alpha(w) = d$, $\alpha(z) = e$, and $w P z$ is in $S'$;
- for every complex concept $C$ and role $R$, the interpretations $C^\mathcal{I}$ and $R^\mathcal{I}$ are directly derived from the semantic equations given in Section 2.

We show that $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies every constraint in $S'$, and therefore $S'$ is $\Sigma$-solvable.

Consider any constraint of the form $w P z$. By the construction of $\alpha$ and $\mathcal{I}$, we have $(\alpha(w), \alpha(z)) \in P^{\mathcal{I}, \mathcal{M}(\Sigma)}$. Consider any constraint of the form $w K P z$. Since $S'$ is a completion, due to the $\rightarrow K$-rule, it follows that $w$ and $z$ are individuals. Moreover, since $S'$ has no $\Sigma$-clash, the constraint $w P z$ is in $S_\Sigma$, and therefore, by Lemma 3.2, $(w, z) \in (K P)^{\mathcal{I}, \mathcal{M}(\Sigma)}$. Therefore, $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies $w K P z$.

With regard to the constraints of the form $w: C$, we proceed by a secondary induction on the structure of $C$.

With regard to the cases where $C$ is either of the form $A$ or of the form $\neg A$, it follows that $\alpha(w) \in C^{\mathcal{I}, \mathcal{M}(\Sigma)}$ by construction of $\alpha$ and $\mathcal{I}$.

Now consider any constraint of the form $w: C \cap D$. Since $S'$ is complete, both $w: C$ and $w: D$ are in $S'$. By the secondary induction hypothesis on the structure of concepts, $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies both constraints, and therefore, $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies $w: C \cap D$ too.

The other forms of constraints, namely $w: T$, $w: \bot$, $w: C \cup D$, $w: \exists R.C$, and $w: \forall R.C$, can be treated analogously.

Regarding concepts of the form $\neg KC$ or $KC$, since $S$ is a completion, there cannot be constraints of the form $x: \neg KC$ or $x: KC$. Therefore, we consider only constraints of the form $a: \neg KC$ and $a: KC$.

Consider any constraint of the form $a: \neg KC$. Let $C'$ be the negation normal form of $\neg C$. Since $S'$ does not contain any $\Sigma$-clash, there is at least one completion of $S_\Sigma \cup \{a: C'\}$ that does not contain any $\Sigma$-clash. Since the number of occurrences of the epistemic operator in $S_\Sigma \cup \{a: C'\}$ is less than $k$, by the induction hypothesis, $S_\Sigma \cup \{a: C'\}$ is $\Sigma$-solvable, which means that there is a model $\mathcal{J}$ of $\Sigma$, such that $a \in (C')^{\mathcal{J}, \mathcal{M}(\Sigma)}$, and hence $a \notin C^{\mathcal{J}, \mathcal{M}(\Sigma)}$. Since $\mathcal{J} \in \mathcal{M}(\Sigma)$, it follows that $a \notin \bigcap_{\mathcal{J} \in \mathcal{M}(\Sigma)} C^{\mathcal{J}, \mathcal{M}(\Sigma)}$, hence—by definition of $(KC)^{\mathcal{I}, \mathcal{M}(\Sigma)}$—we have that $a \notin (KC)^{\mathcal{I}, \mathcal{M}(\Sigma)}$. Therefore, $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies $a: \neg KC$.

Consider any constraint of the form $a: KC$. Since $S'$ does not contain any $\Sigma$-clash, it follows that every completion of $S_\Sigma \cup \{a: C'\}$ contains a $\Sigma$-clash, where $C'$ is the negation normal form of $\neg C$.

Since the number of occurrences of the epistemic operator in $S_\Sigma \cup \{a: C'\}$ is less than $k$, by the induction hypothesis, $S_\Sigma \cup \{a: C'\}$ is $\Sigma$-unsolvable, which means that for every model $\mathcal{J}$ of $\Sigma$, we have $a \notin C^{\mathcal{J}, \mathcal{M}(\Sigma)}$, i.e., $a \notin \bigcap_{\mathcal{J} \in \mathcal{M}(\Sigma)} C^{\mathcal{J}, \mathcal{M}(\Sigma)}$, and hence $a \notin (KC)^{\mathcal{I}, \mathcal{M}(\Sigma)}$. Therefore, $(\mathcal{I}, \mathcal{M}(\Sigma), \alpha)$ satisfies $a: KC$.
In conclusion, we have shown that the triple \((I, \mathcal{M}(\Sigma), \alpha)\) is a solution of \(S'\), and therefore \(S'\) is \(\Sigma\)-solvable. Now, Proposition 3.7 implies that \(S\) is \(\Sigma\)-solvable too.

**Proof of Claim 2.** Proposition 3.7 tells us that if every completion of \(S\) is \(\Sigma\)-unsolvable, then \(S\) is \(\Sigma\)-unsolvable. Therefore, it suffices to show that any completion \(S'\) of \(S\) that contains a \(\Sigma\)-clash is \(\Sigma\)-unsolvable. We now consider each type of \(\Sigma\)-clash in turn, and show that if \(S'\) contains a \(\Sigma\)-clash of that type, then it is \(\Sigma\)-unsolvable.

If \(S'\) contains a \(\Sigma\)-clash of type (i) or (ii), then it contains a constraint of the form \(a: KC\), and there is at least one completion of \(S_2 \cup \{a: C'\}\) with no \(\Sigma\)-clash, where \(C'\) is the negation normal form of \(\neg C\). By the induction hypothesis, \(S_2 \cup \{a: C'\}\) is \(\Sigma\)-solvable, i.e., there is a triple \((I, \mathcal{M}(\Sigma), \alpha)\) that satisfies all constraints of \(S_2 \cup \{a: C'\}\), and in particular \(a: C'\). Therefore, \(a \notin C^{\mathcal{I},\mathcal{M}(\Sigma)}\), which implies that \(a \notin \bigcap_{\mathcal{J} \in \mathcal{M}(\Sigma)} C^{\mathcal{J},\mathcal{M}(\Sigma)}\).

It follows that the constraint \(a: KC\) cannot be satisfied by any triple \((I, \mathcal{M}(\Sigma), \alpha)\), and therefore \(S'\) is \(\Sigma\)-unsolvable.

If \(S'\) contains a \(\Sigma\)-clash of type (iv), then it contains a constraint of the form \(a: \neg KC\), and every completion of \(S_2 \cup \{a: C'\}\) contains a \(\Sigma\)-clash, where \(C'\) is the negation normal form of \(\neg C\). By the induction hypothesis, \(S_2 \cup \{a: C'\}\) is \(\Sigma\)-unsolvable. This means that for every \(I \in \mathcal{M}(\Sigma)\), since \((I, \mathcal{M}(\Sigma), \alpha)\) satisfies \(S_2\), the triple \((I, \mathcal{M}(\Sigma), \alpha)\) does not satisfy \(a: C'\), that is, \(a(a) \in C^{\mathcal{I},\mathcal{M}(\Sigma)}\). This implies that \(a \in \bigcap_{\mathcal{J} \in \mathcal{M}(\Sigma)} C^{\mathcal{J},\mathcal{M}(\Sigma)}\).

It follows that the constraint \(a: \neg KC\) cannot be satisfied by any triple \((I, \mathcal{M}(\Sigma), \alpha)\), and therefore \(S'\) is \(\Sigma\)-unsolvable.

If \(S'\) contains a \(\Sigma\)-clash of type (v), then it contains a constraint of the form \(a KP b\), and \(a \not\in S_2\). By Lemma 3.2 there is a model \(I \in \mathcal{M}(\Sigma)\) such that \((a, b) \notin P^{\mathcal{I},\mathcal{M}(\Sigma)}\). Hence \((a, b) \notin \bigcap_{\mathcal{J} \in \mathcal{M}(\Sigma)} P^{\mathcal{J},\mathcal{M}(\Sigma)}\), and therefore, the constraint \(a KP b\) cannot be satisfied by any triple \((I, \mathcal{M}(\Sigma), \alpha)\), which implies that \(S'\) is \(\Sigma\)-unsolvable.

The results reported in [1,24] show that one can effectively decide whether a constraint system that does not include any occurrence of the epistemic operator is \(\Sigma\)-solvable. With the same arguments as in the proofs in [1,24] one can easily show that the number of completions of an \(ALCK\)-constraint system is finite. Observe that, in order to decide whether a complete constraint system \(S\) has a \(\Sigma\)-clash or not, a finite number of \(\Sigma\)-solvability checks suffices, each one involving a constraint system whose number of epistemic constraints is less than in \(S\). By induction one can show that the completion rules described in this section provide us with an algorithm for checking an \(ALCK\)-constraint system for \(\Sigma\)-solvability.

**Theorem 3.11.** It is decidable whether for an \(ALCK\)-constraint system \(S\) and an \(ALC\)-knowledge base \(\Sigma\) the constraint system \(S\) is \(\Sigma\)-solvable.

Note that the decidability of \(\Sigma\)-solvability implies that we have an effective method both for checking whether \(\Sigma \models C(a)\), and for computing the answer set of \(C\) with respect to \(\Sigma\). For the study of the computational complexity of the problem of answering epistemic queries we refer to Section 5.
Professor □ Grad(john), Professor(bob),
Course(cs221), Course(cs324), IntermediateCourse(ee282),
∃ENROLLED.Grad(ee282),
Grad(mary), Student(susan), ~Grad(peter),
TEACHES(john, cs221), TEACHES(john, cs324), TEACHES(bob, ee282),
ENROLLED(cs221, mary), ENROLLED(cs221, susan), ENROLLED(ee282, peter)
ENROLLED(cs324, susan), ENROLLED(cs324, peter)

Fig. 2. The knowledge base Σ1.

Fig. 3. A pictorial representation of the knowledge base Σ1.

4. ALCCK as a query language

The goal of this section is to show that the use of epistemic operators in queries allows for a sophisticated interaction with the knowledge representation system. For this purpose we consider the knowledge base Σ1 of Fig. 2. The same knowledge base is also shown in graphical form in Fig. 3, where the nodes of the graph represent individuals, arcs denote assertions on roles, and concept expressions are drawn close to the individuals that are their instances. It can easily be verified that Σ1 is satisfiable and that it has indeed several different first-order models. In fact, it does not have complete knowledge about the represented world. For example, Σ1 does not know whether Susan
is a graduate or not. That is, there are first-order models of $\Sigma_1$ in which the individual Susan is in the extension of Grad as well as models in which it is in the extension of $\neg$Grad.

We provide various kinds of queries that can be posed to $\Sigma_1$ using the language $\mathcal{ALCK}$. In particular, in order to understand the role of the epistemic operator $K$, we consider both $\mathcal{ALC}$ queries and their modified versions containing $K$. The comparison between the corresponding meanings highlights the role of $K$ in the query language.

4.1. Incomplete information

We now show how the epistemic operator copes with incomplete information. We start with a pair of queries involving existential quantifiers:

**Query 1.** $\Sigma_1 \models \exists$Enrolled.Grad(\text{ee282})  
**Answer:** YES.

**Query 2.** $\Sigma_1 \models \exists K$Enrolled.KGrad(\text{ee282})  
**Answer:** NO.

Query 1 asks whether there is a graduate student enrolled in EE282. The answer is YES because it has been explicitly asserted in $\Sigma_1$. However, the enrolled student is unknown. It might either be one of the individuals named in $\Sigma_1$ or a different one about whom no information is given. Moreover, it is not even ensured that it is the same one in all models.

Conversely, Query 2 asks whether there exists an individual who is known both to be enrolled in EE282 and to be a graduate student. In other words, it asks for an individual, say Fred, such that both the assertions Enrolled(\text{ee282, fred}) and Grad(fred) hold in every first-order model for $\Sigma_1$. Such an individual does not exist, thus the answer to the query is NO.

The next pair of queries shows how the epistemic operator interacts with disjunction:

**Query 3.** $\Sigma_1 \models \text{Professor} \sqcup \text{Grad}(\text{john})$  
**Answer:** YES.

**Query 4.** $\Sigma_1 \models K\text{Grad} \sqcup K\text{Professor}(\text{john})$  
**Answer:** NO.

Query 3 asks whether John is either a graduate student or a professor. The answer is YES because this fact is explicitly stated in $\Sigma_1$. Query 4, instead, asks whether John is either known to be a graduate student or known to be a professor. It is easy to verify that none of the two cases holds and therefore the answer to this query is NO.

4.2. Closed-world reasoning

We now show that the use of the epistemic operator allows the user to express a form of closed-world reasoning. To this aim, we consider two queries that involve universal quantifiers:

**Query 5.** $\Sigma_1 \models \forall$TEACHES.IntermediateCourse(hob)  
**Answer:** UNKNOWN.
Query 6. $\Sigma_1 \models \forall \text{KTEACHES.KIntermediateCourse(bob)}$

*Answer*: YES.

Query 5 asks whether every course taught by Bob is an intermediate one. The answer is *UNKNOWN* because there are first-order models for $\Sigma_1$ in which Bob teaches only intermediate courses as well as models in which he teaches also courses that are not intermediate.

Query 6, instead, asks whether everything that is known to be taught by Bob is also known to be an intermediate course. Since the only course known to be taught by Bob is EE282, and it is indeed an intermediate course, the answer to Query 6 is YES.

The above queries show that the use of $K$ allows one to pose queries to a knowledge base $\Sigma$ under the assumption that $\Sigma$ has complete knowledge about a certain individual $a$ and a certain role $P$ (bob and TEACHES in the example), i.e., under the assumption that for every pair $(a, b)$ such that $\Sigma \not\models P(a, b)$, the assertion $P(a, b)$ is false in $\Sigma$.

Notice that this is not the same as assuming that knowledge about every role is complete, like for example can be done in CLASSIC [10] by means of the CLOSE operator. In fact, in our case the closure is applied only in computing the answer to the query, whereas in cited approaches the whole knowledge base has a closed-world semantics.

4.3. Case analysis and not knowing

We now consider a more complex query in which other forms of reasoning are involved. Let us consider the following three queries involving nested quantifiers:

Query 7. $\Sigma_1 \models \exists \text{TEACHES.} (\exists \text{ENROLLED.Grad} \land \exists \text{ENROLLED.} \neg \text{Grad})(\text{john})$

*Answer*: YES.

Query 8. $\Sigma_1 \models \exists \text{KTEACHES.K(ENROLLED.Grad} \land \exists \text{ENROLLED.} \neg \text{Grad})(\text{john})$

*Answer*: NO.

Query 9. $\Sigma_1 \models \exists \text{KTEACHES.K(ENROLLED.Grad} \land \exists \text{ENROLLED.} \neg \text{KGrad})(\text{john})$

*Answer*: YES.

Query 7 asks whether John teaches a course in which both a graduate and an undergraduate are enrolled. At a superficial reading of the query, it might seem that the answer should be NO. The intuitive answer NO is supported by the fact that none of the courses taught by John is known to meet the requested conditions, i.e., $\Sigma_1$ entails neither

$$\exists \text{ENROLLED.Grad} \land \exists \text{ENROLLED.} \neg \text{Grad}(cs221)$$

nor

$$\exists \text{ENROLLED.Grad} \land \exists \text{ENROLLED.} \neg \text{Grad}(cs324).$$

Nevertheless, the correct answer is YES, and in order to obtain it, one must reason by *case analysis*: As we have already remarked, the knowledge base does not provide the information as to whether Susan is a graduate or an undergraduate; however, in every first-order model she must be either one or the other. This fact ensures that in every
first-order model for \( \Sigma_1 \) either \( \text{Grad}(\text{susan}) \) or \( \neg\text{Grad}(\text{susan}) \) holds. Consider now the set of first-order models for \( \Sigma_1 \) in which \( \text{Grad}(\text{susan}) \) holds. In each of these models, the course CS324 is taken by both a graduate (Susan) and an undergraduate (Peter). Similarly, consider the set of the remaining first-order models for \( \Sigma_1 \), i.e., the ones in which \( \neg\text{Grad}(\text{susan}) \) holds. It is easy to see that in every model for this set the course CS221, in this case, is taken by both a graduate (Mary) and an undergraduate (Susan). In conclusion, in every first-order model for \( \Sigma_1 \), either CS324 or CS221 is in the extension of \( \exists\text{ENROLLED}.\text{Grad} \cap \exists\text{ENROLLED}.\neg\text{Grad} \). It follows that in every first-order model for \( \Sigma_1 \), the above assertion is true proving that the correct answer is YES.

On the other hand, Query 8 asks whether John is known to teach a course that is known to be in the extension of \( \exists\text{ENROLLED}.\text{Grad} \cap \exists\text{ENROLLED}.\neg\text{Grad} \). The courses known to be taught by John are CS221 and CS324, and therefore none of them falls within the conditions required by the query.

Query 9 is like Query 8, except that the concept \( \neg\text{Grad} \) is replaced with the concept \( \neg\text{TKGrad} \). In this case, since \( \neg\text{TKGrad}(\text{susan}) \) holds in \( \Sigma_1 \), we have that CS221 is in the extension of \( (\exists\text{ENROLLED}.\text{Grad} \cap \exists\text{ENROLLED}.\neg\text{TKGrad} ) \), and therefore the answer is YES. Notice how the reasoning required to answer Query 9 follows the idea of minimizing knowledge: \( \neg\text{TKGrad}(\text{susan}) \) holds because Susan is not known to be a graduate.

Query 7 shows how, in some cases, the first-order semantics of a query might not agree with its intuitive reading. In fact, most people tend to read Query 7 as requiring the reasoning pattern that is actually associated with the semantics of Query 8. In other words, they tend to rule out the case analysis from the computation. Some others may read it as Query 9. For this reason, in our opinion, it is important to have the operator \( K \), which gives us the possibility to distinguish and express in one framework the three alternative readings of the query.

5. Complexity of answering epistemic queries

In this section we investigate the complexity of answering epistemic queries. Specifically, in Section 5.1 we study the complexity of the general problem of answering \( \mathcal{ALCK} \)-queries posed to an \( \mathcal{ALC} \)-knowledge base. In the subsequent two sections, we focus on two cases of special interest: in Section 5.2 we show that a careful use of the \( K \)-operator in the queries decreases the complexity of reasoning, whereas in Section 5.3, we show a case in which the introduction of the \( K \)-operator substantially increases the complexity of reasoning.

The complexity of a problem is usually measured as a function of the size of the problem instances. Therefore, the complexity of checking whether \( \Sigma \models D(a) \) is a function of the sum of the size of \( \Sigma \) and \( D \) (the size of \( a \) is constant and can be neglected).

In Section 5.2 however, we consider a different complexity measure, namely the complexity with respect to the knowledge base \( \Sigma \) alone, as already proposed in [24,53]. We call this complexity measure knowledge base complexity, whereas the one taking into consideration both \( \Sigma \) and \( D \) is called combined complexity.
It is obvious that knowledge base complexity is meaningful in those cases where the size of the query can be neglected with respect to the size of the knowledge base. This is the case, for example, when the knowledge base contains many facts about individuals, like in database applications.

5.1. ACCK-queries

The calculus we presented in Section 3 can be turned into an effective procedure for answering ACCK-queries posed to an ALC-knowledge base $\Sigma$. The simplest way to derive such a procedure is to compute all the completions of the initial constraint system, and then check whether they are $\Sigma$-clash-free. Computing one completion simply means storing the initial constraint system in suitable data structures, and then adding new constraints by applying the completion rules. Unfortunately, completions might have exponential size with respect to the size of the initial constraint system, and therefore the above method requires exponential space in the worst case.

In this section we devise a new method for answering ACCK-queries posed to an ALC-knowledge base. The method works in polynomial space with respect to the size of the query and the knowledge base. Since answering ACCK-queries posed to an ALC-knowledge base is already a PSPACE-complete problem (see below), answering ACCK-queries is PSPACE-hard. Hence, the proposed procedure for answering ACCK-queries proves that the problem is PSPACE-complete.

In [56], both concept satisfiability and subsumption of ALC-concepts are proved to be PSPACE-complete. The upper bound is proved by exhibiting a linear-space algorithm whose main idea is as follows: Although the whole constraint system involved in the computation may have exponential size, one needs only to keep track of a polynomial part of it at a time. These parts, called traces, are mutually independent, and can be checked separately for a clash.

A trace is a set of constraints one obtains when applying exhaustively the completion rules in such a way that for each object $w$ in the constraint system, the $\rightarrow_{3}$-rule is only applied to one constraint of the form $w : \exists P.D$.

This means that, in computing a trace, we are using a variant of the $\rightarrow_{3}$-rule, called $\rightarrow_{3}$-rule (defined below). Intuitively, when the $\rightarrow_{3}$-rule is used instead of the $\rightarrow_{3}$-rule, among the set of possible constraints of the form $w : \exists P.D$ involving $w$, exactly one of them is nondeterministically chosen as the one to which the $\rightarrow_{3}$-rule is applied.

The above technique is extended to reason about ALC-knowledge bases in [1], where a PSPACE algorithm for both knowledge base satisfiability and instance checking is presented. Essentially, the algorithm relies on the same idea, although an ordering is imposed on the application of the rules to improve efficiency. In particular, the application of the $\rightarrow_{3}$-rule is postponed with respect to all other rules.

When the query is an ACCK-concept, the above method is no longer applicable. In particular, because of the presence of the $K$-operator, the $\Sigma$-solvability of one trace cannot be checked independently of the other traces. Indeed, a variable in a trace might be substituted with the same individual as a variable in a different trace, thus making the traces mutually dependent. The following example clarifies the point.
Example 5.1. Given the $\mathcal{ALC}$-knowledge base

$$\Sigma = \{\text{Student(susan)}, \forall \text{FRIEND.\text{Male}(peter)}, \forall \text{CHILD.\neg \text{Male}(peter)}\},$$

consider the query

$$D = \forall \text{CHILD.}(\neg K\text{Student}) \sqcup \forall \text{FRIEND.}(\neg K\text{Student})$$

for the individual peter.

The constraint system $S = S_\Sigma \cup \{peter:D'\}$, where $D'$ is the negation normal form of $\neg D$, is:

$$S = \{\text{susan: Student, peter: \forall \text{FRIEND.\text{Male}}, peter: \forall \text{CHILD.\neg \text{Male}}, peter: \exists \text{FRIEND.}K\text{Student} \sqcap \exists \text{CHILD.}K\text{Student}\}.$$

Applying the completion rules to $S$ we obtain the following constraint system:

$$S_0 = S \cup \{peter \text{FRIEND } x, x: K\text{Student}, x: \text{Male}, peter \text{CHILD } y, y: K\text{Student}, y: \neg \text{Male}\}.$$

Note that $S_0$ is $\Sigma$-unsolvable because of the two constraints $x: K\text{Student}$ and $y: K\text{Student}$. In fact, the substitution $[x/\text{susan}, y/\text{susan}]$ leads to a $\Sigma$-clash of type (ii) because of the constraints $x: \text{Male}$ and $y: \neg \text{Male}$ in $S$, whereas each one of the substitutions $[x/peter], [x/i], [y/peter], [y/i]$ yields directly a $\Sigma$-clash of type (iii).

Conversely, both traces derivable from $S$, i.e., $S_1 = S \cup \{peter \text{FRIEND } x, x: \text{Male}, x: K\text{Student}\}$ and $S_2 = S \cup \{peter \text{CHILD } y, y: K\text{Student}\}$, are $\Sigma$-solvable.

The example shows that a mere application of the notion of trace is not sufficient for capturing all the inferences needed to answer epistemic queries. Nevertheless, we show in the following that answering $\mathcal{ALC\forall}$-queries can be done in polynomial space. The method is based on the following idea: We still proceed by computing traces, but in order to prevent incompatible substitutions from being applied in different traces, we allow only substitutions that agree with a set of choices for the individuals that we make a priori. The key point is that this set of choices can be represented in polynomial space.

A subconcept of a concept $C$ is a substring of $C$ that is a concept. A subconcept of a constraint system $S$ is a subconcept of some concept appearing in a constraint in $S$. We denote by $\text{Subc}(S)$ the set of all the subconcepts of $S$.

Intuitively, the PSPACE algorithm for checking the $\Sigma$-solvability of a constraint system $S$ can be defined as follows:

- For each individual in $S$ and $\Sigma$, guess the subset of $\text{Subc}(S \cup S_\Sigma)$ consisting of the concepts whose extension contains the individual; polynomial space is sufficient for storing such a guess, because both the size of $\text{Subc}(S \cup S_\Sigma)$ and the number of individuals are linear with respect to $|S| + |\Sigma|$.
Explore all traces, but only one trace at a time. Compute each trace starting from the constraint system obtained from $S$ by adding a suitable set of constraints on individuals representing the guess.

Traces and guesses are formalized by two new completion rules, respectively:

(iii) $S \rightarrow_{r3} \{ w \ R y, y : C \} \cup S$

if $w : \exists R.C$ is in $S$, there are no $z, R'$ such that $z$ is an $R'$-successor of $w$ in $S$, and $y$ is a new variable,

(vi) $S \rightarrow_{cut} \{ a : D \} \cup S$

if $a \in O_S \cup O_S$, $C \in \text{Subc}(S \cup S_S)$, $D = C$ or $D = \neg C$, and neither $a : C$ nor $a : \neg C$ is in $S$.

The name of rule (vi) is justified by the fact that the rule is (a nondeterministic version of) the analytic cut rule in tableaux-based calculi [17].

We distinguish between the calculus constituted by rules (i)-(v) presented in Section 3, and the modified calculus, constituted by rules (i), (ii), (iii)', (iv), (v), (vi), i.e., the $\rightarrow_{r3}$, $\rightarrow_{l3}$, $\rightarrow_{r3}$, $\rightarrow_{k}$, $\rightarrow_{r3}$ and $\rightarrow_{cut}$-rules. Also, we call trace any constraint system to which we cannot apply any rule of the modified calculus. Soundness and completeness of the modified calculus are stated in the following two lemmas. The first one simply states that the $\rightarrow_{cut}$-rule preserves the solvability of the constraint system.

**Lemma 5.2.** Let $S$ be a constraint system, $b$ an individual in $O_S \cup O_S$, and $C$ a concept in $\text{Subc}(S \cup S_S)$, such that neither $b : C$ nor $b : \neg C$ is in $S$. Then $S$ is $\Sigma$-solvable if and only if there exists an $S' = \{ b : D \} \cup S$, where $D = C$ or $D = \neg C$, that is obtained from $S$ by the application of the $\rightarrow_{cut}$-rule, and is $\Sigma$-solvable.

**Proof.** ($\Rightarrow$) Suppose $S$ is $\Sigma$-solvable. Let $(T, W, \alpha)$, where $W = M(\Sigma)$ be a solution of $S$. Let $S'$ be obtained as follows: If $b \in C^{T,W}$ then $b : C \in S'$, otherwise $b : \neg C \in S'$. Obviously, $(T, W, \alpha)$ satisfies $S'$.

($\Leftarrow$) Since $S \subseteq S'$, if $S'$ is $\Sigma$-solvable, then so is $S$. \(\square\)

Given a constraint system $S$ and a knowledge base $\Sigma$, we call subconcept saturation of $S$ with respect to $\Sigma$ any constraint system (nondeterministically) obtained from $S$ by the exhaustive application of the $\rightarrow_{cut}$-rule only. One can think of a subconcept saturation also as a binary relation over $(O_S \cup O_S) \times \text{Subc}(S \cup S_S)$. When $D = C$ is chosen in the $\rightarrow_{cut}$-rule, the pair $(a, C)$ is in such a relation; when $D = \neg C$ is chosen, the pair $(a, C)$ is not in the relation.

We now want to group together all the traces that are part of one completion obtainable with the rules (i)-(v). Such groups can be incrementally built using the algorithm shown in Fig. 4.

We call a complete set of traces any set of traces that can be obtained from $S$ and $\Sigma$ as a result of the algorithm of Fig. 4.

Completions are in a one-to-one relation with complete sets of traces, as shown in the following lemma.

**Lemma 5.3.** Let $S$ be a constraint system, and $\Sigma$ an ALC-knowledge base. There is a completion of $S$ (obtained by applying rules (i)-(v) presented in Section 3)
Algorithm \textit{CompleteSetOfTraces}($S, \Sigma$);
\textbf{Input} constraint system $S$, ALC-knowledge base $\Sigma$;
\textbf{Output} a set of traces $\tau = \{T_1, \ldots, T_n\}$;
\begin{algorithmic}
  \State let $\tau$ be the singleton set composed by a subconcept saturation of $S$ w.r.t. $\Sigma$ in
  \State \textbf{while} (there is a trace in $\tau$ to which a trace rule is applicable)
    \If{the $\rightarrow_{cut}$-rule is applicable to a trace $T_i \in \tau$}
      \State apply $\rightarrow_{cut}$ to $T_i$
    \ElsIf{the $\rightarrow_{*}$-rule, $* \in \{\cap, \cup, \forall, \exists\}$, is applicable to a trace $T_i \in \tau$}
      \State apply $\rightarrow_{*}$ to $T_i$
    \ElsIf{the $\rightarrow_{\exists}$-rule is applicable to a trace $T_i \in \tau$}
      \State let $T_i^1, \ldots, T_i^k$ be all the different traces
        that can be obtained by applying $\rightarrow_{\exists}$ to $T_i$
      \State $\tau := (\tau \setminus \{T_i\}) \cup \{T_i^1, \ldots, T_i^k\}$
    \EndIf
  \EndWhile
  \State \textbf{return} $\tau$
\end{algorithmic}

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{algorithm.png}
\caption{The nondeterministic algorithm computing a complete set of traces.}
\end{figure}

without $\Sigma$-clashes if and only if there is a complete set of traces $\{T_1, \ldots, T_n\}$ that can be obtained from $S$ and $\Sigma$ such that, for each $1 \leq i \leq n$, $T_i$ does not contain any $\Sigma$-clash.

\textbf{Proof.} ($\Rightarrow$) Suppose there is a complete set of traces $\{T_1, \ldots, T_n\}$, computed from $S$ and $\Sigma$ by means of the algorithm in Fig. 4, and such that no $T_i$ contains a $\Sigma$-clash. Assume also that the variables generated in different $T_i$s are different. We show that there is a completion $S_2$ of $S$ such that $S_1 \subseteq S_1 = T_1 \cup \cdots \cup T_n$, and $S_2$ does not contain any $\Sigma$-clash. We first show that $S_1$ does not contain any $\Sigma$-clash. Indeed, $S_1$ cannot contain any $\Sigma$-clash of types (i), (iii), (iv), or (v), because, otherwise, such a $\Sigma$-clash would be present in some $T_i$. Assume now that $S_1$ contains a $\Sigma$-clash of type (ii), i.e., $S_1$ contains two constraints $w: A$ and $w: \neg A$. There are two cases:

\textit{Case 1:} the object $w$ is a variable. Due to the structure of the traces, two constraints involving the same variable are necessarily in the same trace. It follows that there exists a $T_i$ with a $\Sigma$-clash.

\textit{Case 2:} the object $w$ is an individual. Suppose that $w: A$ appears in $T_j$ and $w: \neg A$ appears in $T_h$, with $j \neq h$. Let $S'$ be the subconcept saturation of $S$ with respect to $\Sigma$ chosen by the algorithm. Since $w \in O_S \cup O_S'$, due to the $\rightarrow_{cut}$-rule, either $w: A$ or $w: \neg A$ is in $S'$, hence it is in $S_1$ too. Suppose now, without loss of generality, that $w: A$ is in $S'$. Then $w: A$ is in every trace, and thus it is in $T_h$, too. Hence $T_h$ contains a $\Sigma$-clash.

Now, one can construct a completion $S_2$ of $S$ by repeatedly eliminating from $S_1$ unnecessary constraints. Since $S_2$ is a subset of $S_1$, it does not contain any $\Sigma$-clash.
Suppose that there is a completion \( S_1 \), obtained by applying the rules of the calculus presented in Section 3, without \( \Sigma \)-clashes. From Theorem 3.10, \( S_1 \) is \( \Sigma \)-solvable, i.e., there is a triple \((T, M(\Sigma), \alpha)\) that is a solution of \( S_1 \). Let \( S_2 \) be a constraint system obtained from \( S_1 \) by the exhaustive application of the \( \neg_{\text{cat}} \) rule only, using \((T, M(\Sigma), \alpha)\) to guide the applications of the rule: if \( a \in C^{\Sigma,M(\Sigma)} \), then choose \( D = C \), otherwise choose \( D = \neg C \). Obviously, \((T, M(\Sigma), \alpha)\) is a solution of \( S_2 \) too, hence \( S_2 \) cannot contain any \( \Sigma \)-clash.

Now, split \( S_2 \) into the set of its traces \( \{T_1, \ldots, T_n\} \). This set of traces can be derived from \( S \) and \( \Sigma \) by means of the algorithm of Fig. 4, choosing a subconcept saturation contained in \( S_2 \), and using \( S_2 \) to guide the application of nondeterministic rules: add the constraint already present in \( S_2 \) whenever a choice in the application of a rule must be made. Therefore, \( \{T_1, \ldots, T_n\} \) is a complete set of traces; moreover, for each \( 1 \leq i \leq n \), the trace \( T_i \) does not contain any \( \Sigma \)-clash.

Lemma 5.3 tells us that each trace can be checked for a \( \Sigma \)-clash independently. In fact, no \( \Sigma \)-clash can involve two variables belonging to different traces in a constraint system where the \( \neg_{\text{cat}} \) rule is not applicable. Based on this property, we now show that the method sketched at the beginning of this section leads to an algorithm that works with polynomial space.

Before presenting the detailed algorithm, we need one more definition. A constraint \( \sigma \) is closed in a constraint system \( S \) in the following three cases:

- \( \sigma \) is \( w : C \sqcap \neg D \), and both \( w : C \) and \( w : D \) are in \( S \);
- \( \sigma \) is \( w : C \sqcap \neg D \), and either \( w : C \) or \( w : D \) is in \( S \);
- \( \sigma \) is \( w : \exists R.C \), and there exists \( z \) such that \( w R z \) and \( z : C \) are in \( S \).

Intuitively, if a constraint is closed no rule applies to it. A constraint in \( S \) is open if it is not closed.

The algorithm for instance checking is shown in Fig. 5. It faithfully follows the modified calculus, except that traces are checked independently, and closed constraints are removed from the constraint system. The information regarding the choices for individuals (represented by the subconcept saturation \( S' \)) is present in each trace. Observe also that since in this algorithm \( S_2 \subseteq S \), also \( \mathcal{O}_2 \subseteq \mathcal{O}_S \). Hence, when introducing a \( \iota \) with the \( \neg_{\text{K}} \) rule, it is sufficient to check that \( \iota \notin \mathcal{O}_S \).

**Theorem 5.4.** The algorithm \( \text{Instance}_{\text{ALC}}(\Sigma, a, C) \) is correct and terminating.

**Proof.** Follows from Lemmata 5.2 and 5.3 and from the results in [1,56] about the independence of the traces in \( \text{ALC} \). 

We now turn our attention to the complexity of the algorithm. First of all, notice that the size of each subconcept saturation of \( S \) with respect to \( \Sigma \) is polynomially bounded by the size of \(|S| + |\Sigma|\), as proved in the following lemma.

**Lemma 5.5.** Let \( \Sigma \) be an \( \text{ALC} \)-knowledge base, \( S \) a constraint system, and \( S' \) be a subconcept saturation of \( S \) with respect to \( \Sigma \). Then

\[ |S'| = O((|S| + |\Sigma|)^3). \]
Algorithm $\text{Instance}_{\text{ALC/ACCK}}(\Sigma, a, C)$;
Input $\text{ALC}$-knowledge base $\Sigma$, individual $a$, $\text{ACCK}$-concept $C$;
Output true if $\Sigma \models C(a)$, false otherwise;
begin
$S := S_2 \cup \{a: C'\}$, where $C'$ is the negation normal form of $\neg C$;
while (not all possible subconcept saturations of $S$ w.r.t. $\Sigma$ have been considered)
let $S'$ be a new subconcept saturation of $S$ w.r.t. $\Sigma$
in if $\text{Solvable}(S', \Sigma)$
then return false
endwhile;
return true
end.

Algorithm $\text{Solvable}(S, \Sigma)$;
Input constraint system $S$, $\text{ALC}$-knowledge base $\Sigma$;
Output true if $S$ is $\Sigma$-solvable, false otherwise;
begin
if $(w: A, w: \bot \in S)$ or $(aKPb \in S \text{ and } aPb \notin S_2)$
then return false (* $\Sigma$-clash of type (i), (ii), or (v) *)
elseif $a: KC \in S$
then return $\text{Instance}_{\text{ALC/ACCK}}(\Sigma, a, C)$ and
\hspace{1cm} $\text{Solvable}(S \setminus \{a: KC\}, \Sigma)$ (* $\Sigma$-clash of type (iii) *)
elseif $a: \neg KC \in S$
then return (not $\text{Instance}_{\text{ALC/ACCK}}(\Sigma, a, C)$) and
\hspace{1cm} $\text{Solvable}(S \setminus \{a: \neg KC\}, \Sigma)$ (* $\Sigma$-clash of type (iv) *)
elseif $(x: KC \in S \text{ or } x: \neg KC \in S \text{ or } x KP y \in S \text{ or } y KP x \in S)$
then return (there exists $b \in O_S \cup \{t\}$ with $t \notin O_S$: \hspace{1cm} $\text{Solvable}(S[x/b], \Sigma)$)
elseif $w: C_1 \sqcap C_2 \in S$
then return $\text{Solvable}( (S \setminus \{w: C_1 \sqcap C_2\}) \cup \{w: C_1, w: C_2\}, \Sigma)$
elseif $w: C_1 \cup C_2 \in S$
then return ( $\text{Solvable}( (S \setminus \{w: C_1 \cup C_2\}) \cup \{w: C_1\}, \Sigma)$ or \hspace{1cm} $\text{Solvable}( (S \setminus \{w: C_1 \cup C_2\}) \cup \{w: C_2\}, \Sigma)$)
elseif $(w: \forall R.C \in S)$ and $(wRz \text{ holds in } S)$ and $(z: C \notin S)$
then return $\text{Solvable}(S \cup \{z: C\})$
elseif $w: \exists KP.C \in S$
then return $\text{Solvable}( (S \setminus \{w: \exists KP.C\}) \cup \{wKPx, x: C\}, \Sigma)$
elseif $w: \exists P.C \in S$
then return $\text{Solvable}(S \cup \{wPx, x: C\} \cup \bigcup_{Q,C} \{z: \exists Q.D\}, \Sigma)$ and \hspace{1cm} $\text{Solvable}(S \setminus \{w: \exists P.C\}, \Sigma)$
else return true
end.

Fig. 5. The algorithm for instance checking in $\text{ALC/ACCK}$. 
Proof. Let $n$ be the cardinality of the set $\text{Subc}(S \cup S')$. Each subconcept saturation $S'$ contains all constraints in $S$, plus $n \cdot |O_3 \cup O_2|$ constraints of the form $a : D$. The size of each constraint in $S'$ is obviously bounded by $|S| + |\Sigma|$. Therefore, $|S'| \leq |S| + (|S| + |\Sigma|) \cdot n \cdot |O_3 \cup O_2|$. Since $n$ and $|O_3 \cup O_2|$ are bounded by $|S| + |\Sigma|$, the claim follows. □

**Theorem 5.6.** Instance$_{ACC/ALC\bar{K}}(\Sigma, a, C)$ works with polynomial space with respect to $|\Sigma| + |C|$.

**Proof.** The proof is by induction on the number $k$ of occurrences of the $K$-operator in the concept $C$.

*Base case: $k = 0*. We first prove that Solvable($S'$, $\Sigma$) runs in polynomial space with respect to the size of the subconcept saturation $S'$ of $S$ with respect to $\Sigma$. Note that Solvable($S'$, $\Sigma$) actually computes, one by one, every trace $T_i$ in a complete set of traces of $S$ and $\Sigma$. The number of variables involved in each trace $T_i$ is bounded by the maximal nesting depth of existential quantifiers in $S$, which is linear in $|S|$. Also, the number of individuals in $T_i$ is linear in $|S|$. In addition, the number of constraints is polynomially bounded by the number of objects (which is polynomial in $|S'|$). It follows that the size of the trace involved in any recursive call of the algorithm Solvable is polynomial with respect to the size of the initial subconcept saturation $S'$ of $S$ with respect to $\Sigma$, where $S = S_2 \cup \{a : \neg C\}$. Since Lemma 5.5 tells us that the size of each subconcept saturation of $S$ with respect to $\Sigma$ is polynomial with respect to $|S| + |\Sigma|$, it follows that Instance$_{ACC/ALC\bar{K}}(\Sigma, a, C)$ runs in polynomial space with respect to $|S| + |\Sigma|$, and, since $|S| = |\Sigma| + |C|$, we can conclude that Instance$_{ACC/ALC\bar{K}}(\Sigma, a, C)$ works with polynomial space with respect to $|\Sigma| + |C|$. 

*Induction step: $k \geq 1*. The cost of the algorithm in the case $k \geq 1$ is the same of the case $k = 0$ plus the cost of the recursive calls to Instance$_{ACC/ALC\bar{K}}$ issued during the execution of Solvable. The number of calls of Instance$_{ACC/ALC\bar{K}}$ is globally limited to $k$ times the number of individuals in the constraint system, and therefore is polynomial in the size of the initial constraint system. Since for each call of Instance$_{ACC/ALC\bar{K}}$, at least one occurrence of the $K$-operator is eliminated, by the inductive hypothesis each one requires polynomial space. It follows that the whole algorithm works with polynomial space. □

Notice that the algorithm Instance$_{ACC/ALC\bar{K}}$ is meant only for the purpose of stating the complexity upper bound. In order to obtain a more efficient algorithm, several optimizations are possible. However, the analysis of such optimizations is outside the scope of this paper.

### 5.2. Queries with restricted existential quantification

The examples of Section 4 show that existential quantification allows one to express queries that require reasoning by case analysis. In [24,53], it is shown that this kind of reasoning makes deductions about concepts computationally hard. In the examples given in Section 4 we showed that the use of $K$ may allow us to express queries ruling out case analysis. In particular, this is done by replacing the concepts of the form $\exists P \cdot D$
with concepts of the form \( \exists K P. K D \). Those examples suggest that a decrease of the complexity of reasoning is possible by the use of \( K \). In this section, we obtain a general result about this possibility, by analyzing the complexity of query answering when two sublanguages of \( \mathcal{ALC} \), namely \( \mathcal{ALE} \) and \( \mathcal{AL} \), are used.

The language \( \mathcal{ALE} \) consists of all concepts in negation normal form which do not contain the union constructor, whereas \( \mathcal{AL} \) consists of all the \( \mathcal{ALE} \)-concepts whose existential quantifications are of the form \( \exists P. T \). In [53], it has been proved that the problem of checking whether \( \Sigma \models N C (a) \), where \( \Sigma \) is an \( \mathcal{AL} \)-knowledge base and \( C \) is an \( \mathcal{ALE} \)-concept, is \( \text{coNP} \)-hard with respect to the size of \( \Sigma \) (knowledge base complexity). In [24], the same problem for the case of \( \mathcal{ALE} \)-knowledge bases is proved to be \( \text{PSPACE} \)-complete with respect to the size of \( \Sigma \) and \( C \) (combined complexity).

We call \( \mathcal{ALEK} \) the language obtained by adding the \( K \)-operator to \( \mathcal{ALE} \), and we call \( \mathcal{ALEK}^- \) the sublanguage of \( \mathcal{ALEK} \) consisting of the concepts where the existential quantifications are only of the form \( \exists K P. K D \).

We prove that the answer to a query over an \( \mathcal{AL} \)-knowledge base can be computed in polynomial time with respect to the size of the knowledge base (knowledge base complexity) provided the query is in \( \mathcal{ALEK}^- \). This result, compared with the \( \text{coNP} \)-hardness result in [53], confirms the fact that there are cases where we can decrease the complexity of reasoning with a careful use of the \( K \)-operator.

Specifically, we have developed a polynomial-time algorithm (shown in Fig. 6) that checks whether \( \Sigma \models C (a) \), where \( \Sigma \) is an \( \mathcal{AL} \)-knowledge base and \( C \) is an \( \mathcal{ALEK}^- \)-concept. The algorithm is an implementation of the calculus of Section 3, specialized to deal with an \( \mathcal{AL} \)-knowledge base and an \( \mathcal{ALEK}^- \)-query. The specialization amounts to disallowing certain rule applications of the general calculus that cannot take place in our case.

First, since \( \mathcal{ALEK} \) does not have disjunction, no conjunction occurs in the negation normal form of any negated \( \mathcal{ALEK}^- \)-concept. Let \( C \) be an \( \mathcal{ALEK}^- \)-concept and \( C' \) be the negation normal form of \( -C \). Since the \( \rightarrow_{\forall \neg} \)-rule is the only one that can generate two open constraints on the same variable, it follows that each object can be in at most one open constraint involving a subconcept of \( C' \). This open constraint is represented by the second and the third parameter of the algorithm \( \text{ClashFree} \), and is kept separate from the constraint system \( \Sigma \).

Note that the \( \rightarrow_{\forall} \), \( \rightarrow_{\exists} \), and \( \rightarrow_{\forall \neg} \)-rule are applied implicitly in the algorithm, when computing the completion of \( \Sigma \). In addition, the constraints of the form \( w; KC \) do not occur because only negated \( \mathcal{ALEK}^- \)-concepts must be considered, and \( \mathcal{ALEK}^- \) does not allow for general negation.

Notice that, since the whole constraint system has polynomial size with respect to the knowledge base (see below), there is no need to use the modified calculus developed in Section 5.1. The algorithm, called \( \text{Instance}_{\mathcal{AL}/\mathcal{ALEK}^-} \), is shown in Fig. 6.

The following lemma states the correctness of the algorithm, and shows that its time complexity is polynomial with respect to the size of \( \Sigma \).

**Lemma 5.7.** Let \( \Sigma \) be an \( \mathcal{AL} \)-knowledge base, \( a \) an individual, and \( C \) be an \( \mathcal{ALEK}^- \)-concept. Then \( \text{Instance}_{\mathcal{AL}/\mathcal{ALEK}^-} (\Sigma, a, C) \) terminates, returning true if \( \Sigma \models C (a) \), and false otherwise. Moreover, it runs in polynomial time with respect to \( |\Sigma| \).
Algorithm Instance_{\mathcal{AL}/\mathcal{ALEK}^\neg}(\Sigma, a, C)
Input \mathcal{AL}-knowledge base \Sigma, individual \textit{a}, \mathcal{ALEK}^\neg-concept \textit{C};
Output true if \Sigma \models C(a); false otherwise
begin
  \textit{S} := completion of \textit{S}_\Sigma;
  if \textit{S} contains a \Sigma-clash
    then return true
    else return not ClashFree(\textit{S}, \Sigma, \textit{a}, \neg C)
end.

Algorithm ClashFree(\textit{S}, \Sigma, \textit{w}, \textit{E})
Input constraint system \textit{S}, \mathcal{AL}-knowledge base \Sigma, object \textit{w}, negated \mathcal{ALEK}^\neg-concept \textit{E};
Output true if \textit{S} \cup \{\textit{w} : \textit{E}\} is \Sigma-solvable, false otherwise
begin
  \textbf{case} \textit{E} of
  \textbf{\textit{\neg \top}}: \textbf{return} false;
  \textbf{\textit{\neg \bot}}: \textbf{return} true;
  \textbf{\textit{\neg A}}: \textbf{return} (\textit{w} : \textit{A} \notin \textit{S});
  \textbf{\textit{\neg \neg A}}: \textbf{return} (\textit{w} : \neg \textit{A} \notin \textit{S});
  \textbf{\textit{\neg (C_1 \cap C_2)}}: \textbf{return} ClashFree(\textit{S}, \Sigma, \textit{w}, \neg C_1) \textbf{or} ClashFree(\textit{S}, \Sigma, \textit{w}, \neg C_2);
  \textbf{\textit{\forall P.C}}: \textbf{return} ClashFree(completion of \textit{S} \cup \{\textit{w} P \textit{x}\}, \Sigma, \textit{x}, \neg \textit{C}),
  \text{where} \textit{x} \text{is a new variable};
  \textbf{\textit{\neg \neg KC}}: if \textit{w} is an individual
    then \textbf{return} ClashFree(\textit{S}, \Sigma, \textit{w}, \neg C)
    \textbf{else return there exists} \textit{a} \textbf{such that} \textit{a} \in O_\Sigma \cup \{e\}
    \textbf{and} ClashFree(\textit{S}, \Sigma, \textit{a}, \neg C)
  \textbf{\textit{\neg \exists KP.C}}: if \textit{w} is an individual
    then \textbf{return forall} \textit{b} \textbf{such that} (\textit{w} P \textit{b} \in \textit{S}_\Sigma)
    ClashFree(\textit{S}, \Sigma, \textit{b}, \neg \textit{C})
    \textbf{else return there exists} \textit{a} \textbf{such that} \textit{a} \in O_\Sigma
    \textbf{and forall} \textit{b} \textbf{such that} (\textit{a} P \textit{b} \in \textit{S}_\Sigma)
    ClashFree(S[w/a], \Sigma, \textit{b}, \neg \textit{C})
  \textbf{\textit{\neg \forall KP.C}}: if \textit{w} is an individual
    then \textbf{return there exists} \textit{b} \textbf{such that} (\textit{w} P \textit{b} \in \textit{S}_\Sigma)
    \textbf{and} ClashFree(\textit{S}, \Sigma, \textit{b}, \neg \textit{C})
    \textbf{else return there exist} \textit{a}, \textit{b} \textbf{such that} (\textit{a} P \textit{b} \in \textit{S}_\Sigma)
    \textbf{and} ClashFree(S[w/a], \Sigma, \textit{b}, \neg \textit{C})
endcase
end.

Fig. 6. The algorithm for Instance Checking in \mathcal{AL}/\mathcal{ALEK}^\neg.
Proof. The correctness of the algorithm follows from the soundness and completeness of the calculus and the above observations. For the termination, it is sufficient to observe that in any recursive call of the algorithm ClashFree the parameter corresponding to $E$ decreases in length.

With respect to the complexity, first notice that the completion of a constraint system in $\mathcal{ACL}$ has polynomial size [36]. Since all the other operations performed by each call of the procedure ClashFree are polynomial, it follows that each call of the procedure ClashFree runs in polynomial time. In addition, the number of calls is bounded by $|\Sigma|^{|C|}$. In fact, each call can fire a number of calls that is at most the cardinality of $C$, which is bounded by $|\Sigma|$. The depth of the tree of recursive calls is bounded by the number of nested constructors in $C$, and therefore is bounded by $|C|$. Since we are measuring the complexity with respect to the size of $\Sigma$ only, $|C|$ is a constant, and we can conclude that the whole algorithm works in polynomial time with respect to knowledge base complexity.

Theorem 5.8. Answering $\mathcal{ALEK}^-$-queries posed to an $\mathcal{ACL}$-knowledge base can be done in polynomial time with respect to knowledge base complexity.

Proof. Follows from Lemma 5.7. 

It is interesting to observe that the same result does not hold if we measure the complexity by taking into account the size of the query. In fact, the algorithm Instance$_{\mathcal{ACL}/\mathcal{ALEK}^-}$ runs in polynomial time with respect to the size of the knowledge base, but in (deterministic) exponential time with respect to the size of the query. Note, however, that looking at Instance$_{\mathcal{ACL}/\mathcal{ALEK}^-}$ as a nondeterministic algorithm, it works in (nondeterministic) polynomial time, and this allows us to deduce that the problem is in coNP. In the next section, we prove that the same problem is coNP-hard with respect to combined complexity, thus showing that answering $\mathcal{ALEK}^-$-queries posed to an $\mathcal{ACL}$-knowledge base is a coNP-complete problem (with respect to combined complexity).

5.3. Limits to tractability

In this section we prove that reasoning with the $K$-operator is coNP-hard with respect to combined complexity. We prove this result for a language of very low expressivity, namely $\mathcal{ACL}_0$, which is defined below. The result obviously extends to more expressive languages, such as $\mathcal{ACL}$ and $\mathcal{ALE}$.

The language $\mathcal{ACL}_0$ is obtained from $\mathcal{ACL}$ by eliminating the constructor for existential quantification, and the language $\mathcal{ACL}_0K$ is $\mathcal{ACL}_0$ plus the epistemic operator. We now show that answering $\mathcal{ACL}_0K^-$-queries posed to an $\mathcal{ACL}_0$-knowledge base is coNP-hard.

We prove the claim by a reduction from the complement of the problem Uniform-3SAT, which is known to be NP-complete. Uniform-3SAT consists of deciding the satisfiability of a set of propositional clauses, each one consisting of exactly three literals that are either all positive or all negative.
Let \( I = \{ \gamma_1, \ldots, \gamma_n \} \) be such a set of propositional clauses. Without loss of generality, we assume that the clauses \( \gamma_1, \ldots, \gamma_k \) are composed by positive literals and the clauses \( \gamma_{k+1}, \ldots, \gamma_n \) are composed by negative ones, with \( 1 \leq k \leq n \). We construct an \( \mathcal{AL}_0 \)-knowledge base \( \Sigma \) and an \( \mathcal{AL}_0 \mathcal{K} \)-query \( D(c_1) \) such that \( I \) is unsatisfiable if and only if \( \Sigma \models D(c_1) \).

Suppose the propositional symbols occurring in \( I \) are \( p_1, \ldots, p_m \). We consider each \( p_i \) as an individual, and we assume that there are individuals \( c_1, \ldots, c_{n+1} \) which are distinct from \( p_1, \ldots, p_m \). Moreover, we assume that \( A \) is an atomic concept, and \( P, Q_1, \ldots, Q_n \) are primitive roles.

Now, let \( \Sigma \) be the knowledge base containing the assertions

- \( \forall P. A(c_i) \) for \( i = 1, \ldots, k \);
- \( \forall P. \neg A(c_i) \) for \( i = k + 1, \ldots, n \);
- \( Q_i(q_i^1, c_{i+1}), Q_i(q_i^2, c_{i+1}), Q_i(q_i^3, c_{i+1}) \) for \( i = 1, \ldots, n \), where \( q_i^1, q_i^2, q_i^3 \) are the propositional symbols occurring in the clause \( \gamma_i \).

Let \( D \) be the concept

\[
\forall P. \forall KQ_1. \forall P. \forall KQ_2. \ldots. \forall P. \forall KQ_n. \bot,
\]

that is, \( D \) consists of a universally quantified chain of roles where the roles \( P \) and \( KQ_i \) alternate.

**Lemma 5.9.** Let \( I', \Sigma \) and \( D \) be defined as above. Then \( I' \) is unsatisfiable if and only if \( \Sigma \models D(c_1) \).

**Proof.** Recall that \( \Sigma \models D(c_1) \) holds if and only if the constraint system \( S = S_\Sigma \cup \{ c_1 : \neg D \} \) is \( \Sigma \)-unsolvable.

The constraint system \( S \), after rewriting \( \neg D \) into negation normal form, assumes the following form:

\[
S = \{ c_1 : \forall P. A, \ldots, c_k : \forall P. A, c_{k+1} : \forall P. \neg A, \ldots, c_n : \forall P. \neg A, \quad q_1^1 Q_1 c_2, q_1^2 Q_1 c_2, q_1^3 Q_1 c_2, \\
\cdots, \\
q_n^1 Q_n c_{n+1}, q_n^2 Q_n c_{n+1}, q_n^3 Q_n c_{n+1}, \\
c_1 : \exists P. \exists KQ_1. \exists P. \exists KQ_2. \ldots. \exists P. \exists KQ_n. \top \}.
\]

We will prove the lemma by showing that \( I' \) is satisfiable if and only if \( S \) is \( \Sigma \)-solvable, which by Theorem 3.10 is equivalent to the fact that \( S \) has a clash free completion.

Applying the completion rules to \( S \), the only constraint system obtainable (up to variable renaming) is:

\[
S_1 = S \cup \{ c_1 P x_1, x_1 KQ_1 y_2, y_2 P x_2, x_2 KQ_2 y_3, \ldots, \\
y_n P x_n, x_n KQ_n y_{n+1}, y_{n+1} : \top, x_1 : A \}.
\]

The \( \neg K \)-rule can be applied to \( S_1 \) substituting individuals for variables. It is easy to see that for the variables \( y_i \), where \( i = 2, \ldots, n + 1 \), the only substitution that does not
yield a \( \Sigma \)-clash of type (v) is \([y_i/c_i]\). In fact, the constraints in \( S_2 \) regarding the role \( Q_i \) are \( q_1^i Q_i c_{i+1}, q_2^i Q_i c_{i+1}, q_3^i Q_i c_{i+1} \), and all the three of them involve the individual \( c_{i+1} \) as second argument. Moreover, no other clash results from this substitution. Let

\[
S_2 = S_1 [y_2/c_2, \ldots, y_{n+1}/c_{n+1}].
\]

Because of this substitution, \( S_2 \) contains the constraints \( c_1 P x_1, \ldots, c_n P x_n \). In addition, \( S_2 \) contains the constraints \( c_i: \forall P.A \) for \( 1 \leq i \leq k \), and \( c_i: \forall P.\neg A \), for \( k + 1 \leq i \leq n \). Applying the \( \to \sigma \)-rule to these constraints results in the system

\[
S_3 = S \cup \{ c_1 P x_1, x_1 KQ_1 c_2, c_2 P x_2, x_2 KQ_2 c_3, \ldots, c_n P x_n, x_n KQ_n c_{n+1}
\]

\[
\neg x_i, x_1: A, \ldots, x_k: A, x_{k+1}: \neg A, \ldots, x_n: \neg A \}.
\]

Again, in \( S_3 \) the \( \to K \)-rule can be applied to the constraints \( x_i KQ_i c_{i+1} \). After replacing all variables \( x_i \) by individuals, the resulting system will be complete. However, it need not be clash free. Due to the relational constraints in \( \Sigma \) that involve \( Q_i \), there are three candidate substitutions for each \( x_i \), namely \([x_i/q_1^i], [x_i/q_2^i], [x_i/q_3^i]\]. Any other substitution yields immediately a \( \Sigma \)-clash of type (v).

Therefore, the proof of the lemma will be complete once we have verified the following fact:

**Claim.** The set of clauses \( \Gamma \) is satisfiable if and only if in each clause \( \gamma_i \) there is a propositional variable \( p_i \) such that \( S_3 [x_1/p_1, \ldots, x_n/p_n] \) is clash free.

\((\Leftarrow)\) Let \( \theta = [x_1/p_1, \ldots, x_n/p_n] \) be a substitution such that \( S_3 \theta \) is clash free. Note that each \( p_i \) is a propositional variable that occurs in the clause \( \gamma_i \). We define the propositional assignment \( \delta \) by

\[
\delta(p_i) := \begin{cases} 
\text{true} & \text{if } p_i: A \text{ is in } S_3 \theta, \\
\text{false} & \text{otherwise}, 
\end{cases}
\]

where \( l = 1, \ldots, m \).

Let \( 1 \leq i \leq k \). Since \( S_3 \theta \) contains the constraint \( p_i: A \), we have that \( \delta(p_i) = \text{true} \) so that \( \delta \) satisfies the clause \( \gamma_i \). Let \( k + 1 \leq i' \leq n \). Then \( S_3 \theta \) contains the constraint \( p_{i'}: \neg A \). Since \( S_3 \theta \) is clash free, there is no constraint \( p_{i'}: A \) in \( S_3 \theta \). Hence, \( \delta(p_{i'}) = \text{false} \) so that \( \delta \) satisfies the clause \( \gamma_{i'} \). Summarizing, we have shown that \( \delta \) satisfies each clause in \( \Gamma \), hence \( \Gamma \) is satisfiable.

\((\Rightarrow)\) Let \( \delta \) be a propositional assignment that satisfies each clause in \( \Gamma' \). We define a substitution \( \theta = [x_1/p_1, \ldots, x_n/p_n] \) such that \( S_3 \theta \) is clash free.

Let \( 1 \leq i \leq k \). Since \( \delta \) satisfies \( \gamma_i \), we conclude that for one of \( q_1^i, q_2^i, q_3^i \), call it \( p_i \), we have that \( \delta(p_i) = \text{true} \). We substitute \( p_i \) for \( x_i \). As argued above, this substitution does not give rise to a clash of type (v). Let \( k + 1 \leq i' \leq n \). Since \( \delta \) satisfies \( \gamma_{i'} \), we conclude that for one of \( q_1^i, q_2^i, q_3^i \), say \( p_{i'} \), we have that \( \delta(p_{i'}) = \text{false} \). We substitute \( p_{i'} \) for \( x_{i'} \). Again, this substitution does not give rise to a clash of type (v).

Define \( \theta := [x_1/p_1, \ldots, x_n/p_n] \). We argue that \( S_3 \theta \) is clash free. Clearly, there is no clash of type (v). Neither is there a clash of type (i), (ii), or (iv), since there is
no constraint in $S_{3}\theta$ of the form $w: \bot$, $w: KC$, or $w: \neg KC$, respectively. If $S_{3}\theta$ contains a constraint $w: A$, then $w$ is a propositional variable that has been substituted for one of $x_{1}, \ldots, x_{k}$. By definition of $\theta$, we have $\delta(w) = \text{true}$. Similarly, if $S_{3}\theta$ contains a constraint $w: \neg A$, then $w$ is a propositional variable and $\delta(w) = \text{false}$. Since $\delta$ does not assign two different truth values to one propositional variable, $S_{3}\theta$ contains no clash of type (ii).

Lemma 5.9 implies the following theorem.

**Theorem 5.10.** Answering $\mathcal{ALC}$-queries posed to an $\mathcal{ALC}$-knowledge base is coNP-hard with respect to combined complexity.

Observe that the theorem implies also that answering $\mathcal{ALC}K$-queries posed to an $\mathcal{ALC}$-knowledge base is coNP-hard with respect to combined complexity.

### 6. Rules and definitions

In the previous sections we considered knowledge bases constituted only by membership statements in $\mathcal{ALC}$—i.e., an empty TBox and an ABox without any occurrence of the epistemic operator. We now consider knowledge bases with a TBox containing epistemic sentences of a special kind, and an ABox without epistemic sentences as before. We show that this extension formalizes the use of procedural rules as provided in many implemented systems based on description logics [5,43], and allows for a weakening of definitions that is both semantically well-founded and computationally advantageous.

#### 6.1. Epistemic rules

In this subsection we discuss intuitions and formal properties of the class of epistemic sentences that we admit in the TBox. We consider knowledge bases of the form $(\mathcal{R}, \mathcal{A})$, where $\mathcal{A}$ is an $\mathcal{ALC}$-ABox and $\mathcal{R}$ is a TBox containing only of the form

$$KC \subseteq D,$$

where $C$ and $D$ are $\mathcal{ALC}$-concepts. We call these sentences epistemic rules, or simply rules. The concept $C$ is called the antecedent of the rule while concept $D$ is called the consequent. Rules can be instantiated with individuals. Given an individual $a$, we also call $C(a)$ an antecedent and $D(a)$ a consequent of the rule instance.

We remind the reader that an epistemic interpretation $(\mathcal{I}, \mathcal{W})$ satisfies the rule $KC \subseteq D$ if $(KC)^{2,\mathcal{W}} \subseteq D^{2,\mathcal{W}}$. An epistemic model of a knowledge base of the form $(\mathcal{R}, \mathcal{A})$ must satisfy all epistemic rules in $\mathcal{R}$ and all membership statements in $\mathcal{A}$. Therefore, for any epistemic model $\mathcal{W}$ of a knowledge base $(\mathcal{R}, \mathcal{A})$, we have that $\mathcal{W} \subseteq \mathcal{M}(\mathcal{A})$, where $\mathcal{M}(\mathcal{A})$ is the set of first-order models of $\mathcal{A}$. This means that the set of epistemic sentences $\mathcal{R}$ restricts the set of first-order models of $\mathcal{A}$ to a maximal subset that satisfies every rule in $\mathcal{R}$. Because of the form of epistemic rules there is a unique
epistemic model of the knowledge base $\Psi = \langle \mathcal{R}, \mathcal{A} \rangle$, as proved in the following proposition.

**Proposition 6.1.** Let $\Psi = \langle \mathcal{R}, \mathcal{A} \rangle$ be a knowledge base. If $\Psi$ is satisfiable then there is a unique epistemic model for $\Psi$.

**Proof.** By contradiction. Suppose that $\mathcal{W}$ and $\mathcal{W}'$ are two different epistemic models of $\Psi$. We prove that $\mathcal{W} \cup \mathcal{W}'$ is also an epistemic model of $\Psi$, contradicting the maximality condition for epistemic models for $\mathcal{W}$ and $\mathcal{W}'$.

Consider a generic interpretation $I \in \mathcal{W} \cup \mathcal{W}'$. Suppose that $I$ is in $\mathcal{W}$ (the other case $I \in \mathcal{W}'$ is symmetric). From (1), for every rule $KC \subseteq D$ in $\mathcal{R}$ we have that

$$\bigcap_{J \in \mathcal{W}} C^J \subseteq D^I.$$

Further, since

$$\bigcap_{J \in \mathcal{W} \cup \mathcal{W}'} C^J = \left( \bigcap_{J \in \mathcal{W}} C^J \right) \cap \left( \bigcap_{J \in \mathcal{W}'} C^J \right) \subseteq \left( \bigcap_{J \in \mathcal{W}} C^J \right),$$

it follows that

$$\bigcap_{J \in \mathcal{W} \cup \mathcal{W}'} C^J \subseteq D^I$$

proving that $(I, \mathcal{W} \cup \mathcal{W}')$ satisfies $KC \subseteq D$.

Since both $\mathcal{W}$ and $\mathcal{W}'$ are subsets of $\mathcal{M}(\mathcal{A})$, also $\mathcal{W} \cup \mathcal{W}'$ is a subset of $\mathcal{M}(\mathcal{A})$, hence $(I, \mathcal{W} \cup \mathcal{W}')$ satisfies every assertion of $\mathcal{A}$. It follows that $\mathcal{W} \cup \mathcal{W}'$ is an epistemic model of $\Psi$. \(\square\)

Observe that $KC$ is equivalent to $T$ if $C$ is equivalent to $T$. For such concepts, the epistemic rule $KC \subseteq D$ is equivalent to the inclusion statement $T \subseteq D$ without the epistemic operator in the antecedent. Dealing with implications of this kind requires some extra machinery and is computationally demanding (see, e.g., [14]). Therefore in this paper we restrict our attention to "genuine" epistemic rules, i.e., rules whose antecedent is not equivalent to $T$.

Let us show through an example the effects of epistemic rules in the knowledge base.

**Example 6.2** (see [5, p. 62]). Consider the knowledge base $\Psi_2 = \langle \mathcal{R}_2, \mathcal{A}_2 \rangle$, where

$$\mathcal{R}_2 = \{KC_{Student} \subseteq \forall V E A S . J u n k F o o d \},$$
$$\mathcal{A}_2 = \{Student(john)\}.$$

The epistemic rule in $\mathcal{R}_2$ states that "those that are known to be students eat only junk food". Therefore, if we know that john is a student we can conclude that he eats only junk food. In every first-order interpretation of the epistemic model $\mathcal{W}$ of $\Psi_2$, we have
that \texttt{Student(john)} is true, thus \texttt{john} is known to be a student. Therefore, in order to satisfy the rule \( K\texttt{Student} \subseteq \forall \texttt{EATS.JunkFood} \), every first-order interpretation in \( \mathcal{W} \) must satisfy \( \forall \texttt{EATS.JunkFood(john)} \). Thus, the semantics of the epistemic rule gives the desired conclusion that \texttt{john} eats only junk food.

One of the features of epistemic rules is that they represent a weak form of implication, since they rule out contrapositive reasoning. This feature, which is relevant for both uses of epistemic rules that are discussed below, is illustrated in the following example.

\textbf{Example 6.3.} Consider the knowledge base \( \Psi_3 = (\mathcal{R}_2, \mathcal{A}_3) \), where \( \mathcal{R}_2 \) is the same as in the previous example:

\[
\mathcal{R}_2 = \{ K\texttt{Student} \subseteq \forall \texttt{EATS.JunkFood} \},
\]

\[
\mathcal{A}_3 = \{ \neg \forall \texttt{EATS.JunkFood(john)} \}.
\]

In this case, \( \neg \forall \texttt{EATS.JunkFood(john)} \) is true in every first-order interpretation of the epistemic model \( \mathcal{W} \). However, in \( \mathcal{W} \) there is a first-order interpretation in which \( \texttt{Student(john)} \) is true and another one in which \( \neg \texttt{Student(john)} \) is true. Therefore, \texttt{john} is not known to be a student and the rule is satisfied because the antecedent is false.

We now introduce the \textit{extension} of an ABox \( \mathcal{A} \) with respect to a TBox \( \mathcal{R} \). We say that an epistemic rule \( KC \subseteq D \) is \textit{applicable} to an individual \( a \) in a knowledge base \( \Psi = (\mathcal{R}, \mathcal{A}) \), if its antecedent \( C(a) \) is true in \( \Psi \), i.e., \( \Psi \models C(a) \). The result of the application of the epistemic rule to the individual \( a \) is that the consequent \( D(a) \) of the rule instance is added to \( \mathcal{A} \) (obviously without changing the epistemic models of \( \Psi \)). The \textit{extension} of \( \mathcal{A} \) with respect to \( \mathcal{R} \) is the ABox \( \mathcal{A}_R \) that is obtained from \( \mathcal{A} \) as a result of the systematic application of the epistemic rules \( \mathcal{R} \) to all individuals in \( \mathcal{A} \). The \textit{first-order extension} of a knowledge base \( \Psi = (\mathcal{R}, \mathcal{A}) \) is the \( \mathcal{ALC} \)-knowledge base \( \Psi_R = (\emptyset, \mathcal{A}_R) \).

More precisely, \( \Psi_R \) is the knowledge base computed by the algorithm in Fig. 7. The idea behind the algorithm is simple: the application of every epistemic rule is attempted on every individual, until no rule is applicable. Each rule application adds the consequent of the rule to the final result and discards the pair rule-object that has fired the rule. As shown below, the result is unique and independent of the order of application of the rules.

\textbf{Example 6.4.} It is easy to verify that in Example 6.2 the rule is applicable and the first-order extension is

\[
(\Psi_2)_R = (\emptyset, \{ \texttt{Student(john)}, \forall \texttt{EATS.JunkFood(john)} \}).
\]

On the other hand, in Example 6.3 the rule is not applicable because the antecedent is false and the first-order extension is

\[
(\Psi_3)_R = (\emptyset, \{ \neg \forall \texttt{EATS.JunkFood(john)} \}).
\]
Algorithm ApplyRules(Ψ);
Input knowledge base Ψ = (R, A);
Output ALC-knowledge base Ψ_R = (∅, A_R)
begin
  K := R × O_A; (* x is the Cartesian product *)
  A' := A;
  loop
    finished := true;
    forall (KC ⊆ D, a) ∈ K do
      if A' ⊨ C(a) then begin
        A' := A' ∪ {D(a)};
        remove((KC ⊆ D, a), K);
        finished := false
      end;
  end for;
  if finished then return (∅, A')
  endloop

end.

Fig. 7. The algorithm for computing first-order extensions.

Notice that the number of assertions that can be added to A is at most equal to |R| · |O_A| (the number of epistemic rules times the number of individuals in the ABox), i.e., the number of possible rule applications. However, when an epistemic rule is applied, all rules that have not yet been applied must be reconsidered. Therefore the algorithm for computing the first-order extension requires a number of steps which is quadratic in |R| · |O_A|, the most expensive one being instance checking in ALC for which a calculus is given in [1,24].

First-order extensions represent the result of a forward reasoning process on a knowledge base and a set of epistemic rules. We now show that they correctly capture the semantics of knowledge bases with epistemic rules.

Proposition 6.5. Let Ψ = (R, A) be a knowledge base, W be its epistemic model, and Ψ_R = (∅, A_R) be its first-order extension. Then $M(A_R) = W$.

Proof. We first prove that $W \subseteq M(A_R)$. Let $A = A_0, A_1, \ldots, A_n = A_R$ be the sequence of ABoxes generated by the algorithm ApplyRules adding one assertion of the form $D(a)$ at a time. We have that $W \subseteq M(A_0)$; we now prove that if $W \subseteq M(A_i)$, then $W \subseteq M(A_{i+1})$, from which the claim follows by induction.

By contradiction: assume that $W \not\subseteq M(A_{i+1})$. Then there exists an interpretation $I$ such that $I \in W$, $I \in M(A_i)$, and $I \not\in M(A_{i+1})$.

Let $KC \subseteq D$ be the rule applied in $A_i$ and $D(a)$ be the assertion added to $A_i$ so as to obtain $A_{i+1}$, i.e., $\{D(a)\} = A_{i+1} \setminus A_i$. From $I \in M(A_i)$ and $I \not\in M(A_{i+1})$, it follows that $a \notin D^I$. 


Since $D(a)$ is in $A_{i+1}/A_i$ and $D(a)$ has been included to $A_{i+1}$ because of the application of the rule $KC \subseteq D$, we have that $A_i \models C(a)$.

By definition of $(\mathcal{I}, \mathcal{W})$ we have that $(\mathcal{I}, \mathcal{W})$ satisfies $KC \subseteq D$. It follows that $\bigcap_{J \in \mathcal{J}} C^J \subseteq D^J$. Now, since $a \notin D^J$, we have that $a \notin \bigcap_{J \in \mathcal{J}} C^J$. The two facts: $A_i \models C(a)$ and $a \notin \bigcap_{J \in \mathcal{J}} C^J$ contradict the inductive assumption that $\mathcal{W} \subseteq \mathcal{M}(A_i)$.

We now prove that $\mathcal{M}(A_R) \subseteq \mathcal{W}$. By contradiction: assume that $\mathcal{M}(A_R) \notin \mathcal{W}$; since we have already proved that $\mathcal{W} \subseteq \mathcal{M}(A_R)$, we have that $\mathcal{W} \neq \mathcal{M}(A_R)$.

Now, since $\mathcal{W}$ is the maximal set satisfying all rules in $\mathcal{R}$, it follows that $\mathcal{M}(A_R)$ cannot satisfy all rules, and therefore there exists a rule $KC \subseteq D$ and an interpretation $\mathcal{I} \in \mathcal{M}(A_R)$ such that $KC^{\mathcal{I}, \mathcal{M}(A_R)} \not\subseteq D^{\mathcal{I}, \mathcal{M}(A_R)}$. Let us consider the set $(KC)^{\mathcal{I}, \mathcal{M}(A_R)} \setminus D^{\mathcal{I}, \mathcal{M}(A_R)}$; in this set there is an individual $a$ such that $A_R \models C(a)$. However, we know from $a \notin D^{\mathcal{I}, \mathcal{M}(A_R)}$ that $D(a)$ is not in $A_R$. It follows that the rule $KC \subseteq D$ is applicable to $a$ in $A_R$, contradicting the assumption that $A_R$ is the first-order extension of $(\mathcal{R}, A)$. $\square$

The algorithm ApplyRules can therefore be effectively used in the computation of the first-order extension of a knowledge base.

6.2. Procedural rules

In this subsection we address a direct use of epistemic rules in the formalization of the so-called procedural rules (or trigger rules), that are commonly available in frame-based systems. In fact, following the idea of combining frames and rules [30], many knowledge representation systems based on description logics provide a mechanism for expressing knowledge, that we here refer to as procedural rules (see, for instance, CLASSIC [5] and LOOM [43]). Such rules take the form

$$C \Rightarrow D,$$

where $C, D$ are concepts. The intuitive meaning of a procedural rule is "if an individual is proved to be an instance of $C$, then derive that it is also an instance of $D$" and the behavior of procedural rules is usually described in terms of a forward reasoning process that adds to the knowledge base the assertion $D(a)$ whenever $C(a)$ is proved to hold. Consequently, a procedural rule cannot be interpreted in terms of logical implication since it does not support reasoning by contraposition—i.e., from $\neg D(a)$ infer $\neg C(a)$. Indeed, the main difference between procedural rules and implications is that the former are intended to provide a reasoning mechanism which applies them in one direction only, namely from the antecedent to the consequent.

The semantics of procedural rules in frame-based systems is often defined informally. Attempts to precisely capture the meaning of procedural rules are based either on viewing them as knowledge base updates (see for example the TELL operation of [37]), or on ad hoc semantics (see [55]). Procedural rules in frame-based systems can be nicely formalized as epistemic rules. In fact, the procedural rule $C \Rightarrow D$ can be represented by the epistemic rule $KC \subseteq D$. Procedural rules are therefore interpreted as implications, but the epistemic operator in the antecedent leads to a weaker form of inclusion, which rules out reasoning by contraposition. Epistemic rules correctly capture
this property, as shown in the previous section, and thus can be effectively used to give a formal account of procedural rules. In order to clarify the correspondence between procedural and epistemic rules we present an example where epistemic rules can be read as procedural rules.

**Example 6.6.** Consider the knowledge base $\mathcal{P}_4 = (\mathcal{R}_4, \mathcal{A}_4)$:

\[
\mathcal{R}_4 = \{ K\text{Grad} \subseteq \forall \text{TEACHES.BasicCourse}, \quad K\text{BasicCourse} \subseteq \forall \text{ENROLLED.\neg Grad} \},
\]

\[
\mathcal{A}_4 = \{ \text{Grad}(\text{bill}), \text{TEACHES}(\text{bill}, \text{cs}248), \text{ENROLLED}(\text{cs}248, \text{ann}) \}.
\]

Consequences of the procedural rules are directly expressed by the extension of $\mathcal{A}_4$ with respect to $\mathcal{R}_4$, which is the following:

\[
(\mathcal{A}_4)_{\mathcal{R}_4} = \{ \text{Grad}(\text{bill}), \text{TEACHES}(\text{bill}, \text{cs}248), \text{ENROLLED}(\text{cs}248, \text{ann}), \quad \forall \text{TEACHES.BasicCourse}(\text{bill}), \forall \text{ENROLLED.\neg Grad}(\text{cs}248) \}.
\]

Applying the rule $K\text{Grad} \subseteq \forall \text{TEACHES.BasicCourse}$ to the individual bill adds to the knowledge base the consequent $\forall \text{TEACHES.BasicCourse}(\text{bill})$, which implies $\text{BasicCourse}(\text{cs}248)$, thus firing the rule $K\text{BasicCourse} \subseteq \forall \text{ENROLLED.\neg Grad}$, which in turn adds $\forall \text{ENROLLED.\neg Grad}(\text{cs}248)$. Obviously, $(\mathcal{P}_4)_{\mathcal{R}_4} \models \neg \text{Grad(ann)}$.

One can verify that in every first-order interpretation $\mathcal{I}$ of the epistemic model $\mathcal{W}$ for $\mathcal{P}_4$, we have $\text{cs}248 \in \text{BasicCourse}^\mathcal{I,\mathcal{W}}$ and $\text{ann} \in \text{\neg Grad}^\mathcal{I,\mathcal{W}}$.

### 6.3. Weak inclusions

In this subsection we provide a weak form of concept definition by exploiting the use of epistemic rules in the knowledge base that is both semantically well-founded and strictly related to the actual behavior of implemented frame-based systems.

Recent studies on the formal properties of description logics [14, 16, 18, 46, 47] show that the treatment of inclusions in the TBox is one of the critical aspects of the realization of knowledge representation systems based on description logics. In fact, reasoning on knowledge bases with a TBox is coNP-hard even for TBoxes using very simple concept languages [46]. In the case of the language ACC, reasoning with an empty TBox is PSPACE-complete, while admitting inclusions in the knowledge base makes reasoning EXPTIME-hard [14].

We propose a suitable use of epistemic rules that leads to so-called *weak inclusions*. The idea is to substitute general inclusions with epistemic rules, thus losing some of the inferences that are sanctioned by the semantics of inclusions, but gaining in the efficiency of deduction. We believe that this weaker setting has an intuitive meaning based on the semantics of epistemic rules. Recall that a concept definition $A \equiv C$ can be seen as a shorthand for the two inclusions $A \subseteq C$ and $C \subseteq A$. One can verify that the treatment of definitions provided in, e.g., LOOM [42], which limits their use to known individuals and disregards many inferences based on the use of contrapositives, is similar to considering the two inclusions of a definition as epistemic rules.
Let $R$ be a set of epistemic rules, $T$ a set of inclusions between $\mathcal{ALC}$-concepts (no epistemic operator), and $A$ be an $\mathcal{ALC}$-ABox. The weakening of $\Psi = \langle R \cup T, A \rangle$ is the $\mathcal{ALCK}$-knowledge base
\[
\Psi^- = \langle R', A \rangle
\]
obtained by replacing every inclusion statement $C \subseteq D$ in $T$ by an epistemic statement $KC \subseteq D$ in $R$. More formally,
\[
R' = R \cup \{KC \subseteq D \mid (C \subseteq D) \in T\}.
\]
Intuitively, every inference we can make in $\Psi^-$ can be done in $\Psi$ as well, while the converse is not true. Hence, $\Psi^-$ can be regarded as a sound and incomplete approximation of $\Psi$.

Let us now consider the computational advantages of weakening inclusions in an $\mathcal{ALCK}$-knowledge base. In what follows, we assume that no rule in $R'$ has an antecedent which is equivalent to $T$.

Query answering over an $\mathcal{ALCK}$-knowledge base $\Psi = \langle R \cup T, A \rangle$ with $T \neq \emptyset$ is EXPTIME-hard [14]. Hence we do not expect to find any algorithm for answering queries in $\Psi$ working in polynomial space, unless EXPTIME = PSPACE. On the other hand, query answering in $\Psi^- = \langle R', A \rangle$ is the same as query answering in $(\Psi^-)_{R'}$, which is the first-order extension of $\Psi^-$. Observing that $(\Psi^-)_{R'}$ is a knowledge base $\langle \emptyset, A_{R'} \rangle$ constituted by an ABox only, we know from [24] that this problem can be solved in polynomial space. Since the size of $(\Psi^-)_{R'}$ is polynomially related to the size of $\Psi^-$, and therefore of $\Psi$ too, the above observation shows that weakening the inclusions of an $\mathcal{ALCK}$-knowledge base leads to an exponential decrease of the space required in the worst case for query answering.

We finally discuss an example of weakening, which illustrates the effects of the transformation on the conclusions that can be drawn from the knowledge base.

**Example 6.7.** Consider the knowledge base $\Psi_5 = \langle T_5, A_1 \rangle$, where $A_1$ is the same ABox of Fig. 2, and $T_5$ is shown in Fig. 8 (note that initially there are no rules, i.e., $R_5 = \emptyset$). Recall that all definitions of the form $C \cong D$ are a shorthand for $C \subseteq D$ and $D \subseteq C$.

The weakening $\Psi_5^-$ will be $\langle R_5', A_1 \rangle$, where $R_5'$ is shown in Fig. 9. Due to the weakening the answer to the queries posed to $\Psi_5$ and $\Psi_5^-$, is not always the same. For example,

**Query 10.** $\Psi_5 \models \exists T E A C H E S . I n t e r m e d i a t e C o u r s e ( j o h n)$  

**Answer:** YES.

**Query 11.** $\Psi_5^- \models \exists T E A C H E S . I n t e r m e d i a t e C o u r s e ( j o h n)$  

**Answer:** UNKNOWN.

In $\Psi_5$ the answer to the query is YES because of a case analysis on Susan—as in $\Sigma_1$ of Section 4. In fact, according to $T_5$, the two concepts $\text{Grad}$ and $\text{Undergrad}$ partition the concept $\text{Student}$. Being a student, Susan can be either a graduate or an undergraduate. In the first case, the course CS221 is an intermediate course, while in the second case CS324 is an intermediate course. Hence, in both cases John teaches an intermediate course.
IntermediateCourse =
Course ∩ ∃ENROLLED.Grad ∩ ∃ENROLLED.Undergrad,
Grad = Student ∩ ∃DEGREE.Bachelor,
Undergrad = Student ∩ ¬Grad

Fig. 8. The TBox $T_5$.

On the contrary, it is easy to see that this does not happen in $\Psi_5^-$, because in $\Psi_5^-$ the two concepts Grad and Undergrad do not partition the concept Student. Indeed, in $\Psi_5^-$ only those individuals known to be undergraduates are inferred to be students and non-graduates, and vice versa, only individuals known to be students and non-graduates are inferred to be undergraduates. Since Susan satisfies neither of the two conditions, we cannot infer anything about her. In fact, the epistemic model for $\Psi_5^-$ includes first-order interpretations where Susan is neither a graduate nor an undergraduate. Therefore, the answer to the query is UNKNOWN.

One can also verify that contrapositives are not applicable to $\Psi_5^-$. Compare the answer to $\neg\exists$DEGREE.Bachelor(peter) in the two knowledge bases:

Query 12. $\Psi_5 \models \neg\exists$DEGREE.Bachelor(peter)  
Answer: YES.

Query 13. $\Psi_5^- \models \neg\exists$DEGREE.Bachelor(peter)  
Answer: UNKNOWN.

In fact, in $\Psi_5$ Peter is known to be an undergraduate, hence a student who is a non-graduate. Since graduates are defined as students with a Bachelor's degree, we can infer that Peter has none by using the contrapositive of the inclusion

$(\text{Student} \cap \exists$DEGREE.Bachelor) $\subseteq$ Grad.
Instead, in $\Psi_5^-$ we only can infer that Peter is a student and a non-graduate. This does not activate the contrapositive of the epistemic rule

$$K(\text{Student} \sqcap \exists \text{DEGREE.Bachelor}) \sqsubseteq \text{Grad}.$$ 

7. Conclusions

We have presented a framework for adding epistemic operators to description logics. By enriching description logics with an epistemic operator, so as to distinguish what is known to the knowledge base as opposed to what is true in the external world, we have been able to formally characterize several aspects of frame-based systems.

We have considered the use of the epistemic operator in the query language, and shown that this richer query language may be used to reduce the complexity of reasoning. In addition, by virtue of the epistemic operator one can naturally specify forms of closed-world reasoning in the queries, without resorting to a closed-world semantics for the knowledge base. Finally, we have shown how to formalize mechanisms such as procedural rules.

We believe that one of the most important benefits of this work is that a single representation mechanism allows for the treatment of a large number of features which are necessary in real applications. This helps fill the gap between theoretical work on description logics and implemented frame-based systems. This is supported by the fact that the approach taken in this paper has been followed by various researchers (see, e.g., [3, 52]) and that the semantics for procedural rules we proposed has been adopted in the proposal for a standard concept-based system in [48].

At the same time, we believe that our investigation of the epistemic operator raises a number of interesting issues related to the use of concept languages in practical systems. First of all, the class of epistemic sentences proposed for formalizing rules and definitions might be extended so as to capture more aspects, while retaining the nice computational properties. Moreover, it is unknown whether epistemic sentences are powerful enough to express (some form of) default reasoning. Preliminary work in this direction is reported in [27].

Acknowledgements

We thank Riccardo Rosati for many fruitful discussions that contributed to the paper. This work has been partly funded by the Esprit Long Term Research Project No. 22469 “Foundations of Data Warehouse Quality (DWQ)”, and by ASI (Agenzia Spaziale Italiana), CNR (Consiglio Nazionale delle Ricerche) and MURST (Ministero dell’Università e della Ricerca Scientifica e Tecnologica).

References


