Knowledge-Based Systems 24 (2011) 595–608

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Combining description logics and Horn rules with uncertainty in ARTIGENCE

Xiaoqing Zheng *

School of Computer Science, Fudan University, Shanghai 201203, China

ARTICLE INFO

Article history:
Received 18 March 2010
Received in revised form 18 January 2011
Accepted 18 January 2011
Available online 26 January 2011

Keywords:
Description logics
Logic programming
Uncertainty
Semantic Web
Horn rules

ABSTRACT

We present ARTIGENCE, a representation language that combines description logics and Horn rules with uncertainty. ARTIGENCE capabilities go beyond the similar hybrid systems presently available, and it contains three components: a highly expressive description logic $\mathcal{ACLNR}$, a set of probabilistic Horn rules and a set of ground facts. The new features described, often required in realistic application domains, can be summarized in three main points. First, we obtained a sound, complete and decidable algorithm for reasoning in ARTIGENCE knowledge base, with decidability being an important indicator that the computational complexity of the language might be essential issue for practical applications. Second, ARTIGENCE was designed not only to combine the expressive power of Horn rules and description logics, but also for its ability to deal with uncertainty. Third, we consider $\mathcal{ACLNR}$ as a description logic component of ARTIGENCE, which is one of the most expressive description logic with decidable inference procedures so far. We also show that the specific description logic $\mathcal{ACLNR}$ used in our proposed framework is not mandatory, and other decidable description logics, even their probabilistic versions can be accommodated to our framework.

1. Introduction

Description logics (DL) are the descendants of so-called "structured inheritance networks", which were introduced to overcome the ambiguities of early semantic networks and frames. In DL, concepts are defined by the sets of objects as unary predicates, and roles are defined by the relationships between objects as binary predicates. Horn rules are a natural representation language in many Artificial Intelligence applications for which several efficient inference engines have been developed. Description logics and Horn rules are strict and tractable subsets of first order logic. There is expressive overlap of DL with Horn rules, but neither Horn rules nor DL is fully included by the other.

Horn rules and DL are not reducible to each other although there is the expressive intersection between those two formalisms. Horn rules require that all variables are universally quantified at the outer level of the rules. As a result of the restriction on quantifiers, Horn rules lack the possibility to express the existence of individuals whose identities might not be expressed explicitly. For example, it is impossible to state that every person has a mother (known or unknown), which is easy with a DL axiom, $Person \sqsubseteq \exists\, Mother$. A Horn clause is said to be definite when only one of its literals is positive. Negation is not allowed within the body or head of a definite Horn rule. Thus, it is impossible to represent that all persons are either male or female (but not both). This would also be easily expressed by two DL axioms, $Person \sqsubseteq Male \sqcup Female$; $Male \sqsubseteq \neg\, Female$. On the other hand, DL with restricted quantification requires that the quantified variables must occur in a binary predicate along with the free variable [1]. The consequence of this restriction is that it is impossible to describe concepts whose instances are related to another anonymous individual via different role paths. For example, it is impossible to describe a concept, "home workers", whose individuals live and work at the same location. This can be easily represented by a Horn rule.

One of the significant limitations of Horn rules is that they are not expressive enough to represent domain knowledge with rich hierarchical structures. In contrast, description logic is a family of logical systems that has been developed especially to model rich hierarchies of concepts [2]. At the same time, DL systems also need rules to express dynamic knowledge and support complex applications. Description logics have become the cornerstone of the Semantic Web for its use in the design of ontologies. The OWL Full, DL, and Lite sub-languages of Web Ontology Language (OWL) are based on description logics [3]. The Semantic Web consists of several hierarchical layers, where the ontology layer in form of the OWL, is currently the highest layer of sufficient maturity. On top of the ontology layer, the rules, logic, and proof layers of the Semantic Web will be designed next. The OWL rule language is the first proposal for extending OWL by Horn rules [4]. DL and Horn rules are two orthogonal and complementary subsets of first order logic. Several applications, such as combing information from multiple heterogeneous sources, especially for Semantic Web applications, can significantly benefit from combing the expressive power of

* Tel.: +86 21 5135 5386.
E-mail address: zhengxq@fudan.edu.cn

0950-7051/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2011.01.006

both formalisms. The motivation for building this hybrid system is to improve on two basic features of knowledge representation formalisms, namely representational adequacy and deductive power.

The key problem in developing such hybrid systems is designing a sound, complete, and decidable inference procedure for answering queries in the systems. Query answering in the hybrid systems constituted by DL and Horn rules is more complex, and it requires nontrivial reasoning. As pointed out in [2], combining standard Horn rule inference procedures with intermediate DL reasoning steps does not result in a complete inference procedure. The reason for the incompleteness is that Horn rule reasoning procedures apply each rule in isolation, and they try to instantiate the antecedent of a rule in order to derive its consequence. Since an ABox of DL represents possibly infinitely many interpretations, namely its models (open-world semantics), a knowledge base of the hybrid system may entail the disjunction of the antecedents of several Horn rules without entailing any of them. Moreover, it cannot be done by deriving just one empty clause for the component of Horn rules in the hybrid systems, as in the classical resolution calculus or its generalizations [5]. The reason is that such an empty clause might still have some constraints which are only satisfied by some of the DL models, but not by all of them, which is also the consequence of the open-world assumption of DL's ABox.

This paper describes ARTIGENCE that combines the expressive power of description logics and Horn rules with uncertainty. A constrained logic scheme [5] has been introduced with a resolution principle for Horn clauses whose variables are constrained by the model of description logic. Constraints can be seen as quantifier restrictions filtering out the values that any interpretation of the underlying description logic can assign to the variables of the Horn clause with such restricted quantifiers. It shows that this constrained resolution is sound, complete and decidable in which a set of recursive function-free, and constrained Horn clauses is unsatisfiable over a certain description logic if and only if for each canonical interpretation of the description logic we can deduce a constrained empty clause whose constraint is satisfiable in that interpretation. We also show that the reasoning under uncertainty in ARTIGENCE is the instance of a certain type of linear programming model, typically with exponentially many variables. Query answering in ARTIGENCE can be reduced to the following procedures: for every model of DL knowledge base, check if there is at least one constrained SLD-refutation whose constraints are satisfied by that model; then for every SLD-refutation under each model of DL knowledge base, solve two linear programming problems to obtain an interval of confidence degree of a query, and take the intersection of all such intervals as the final confidence degree of the query.

The paper is organized as follows. In Section 2, we describe the three components of ARTIGENCE with defining the semantics of its knowledge bases, and point out the reason why traditional inference mechanisms for Horn rules are inadequate for ARTIGENCE knowledge bases. Section 3 presents a procedure for reasoning in ARTIGENCE based on the resolution principle for constrained logics without considering uncertainty, and proves its soundness, completeness and decidability. Section 4 discusses the decidable reasoning for description logic $\mathcal{ALCNR}$. In Section 5, we provide the globe algorithm for uncertainty reasoning in ARTIGENCE, and show how some description logics, even their probabilistic extensions, fit into our proposed framework. Section 6 presents a brief overview of related work. Finally, the conclusions and future work are summarized in Section 7.

2. Language ARTIGENCE

The ARTIGENCE knowledge base contains three components. The first is a description logic knowledge base, the second is a set of probabilistic Horn rules, and the third is a set of probabilistic ground facts. ARTIGENCE allows the concepts and roles defined in the DL component to appear as predicates in the antecedents of Horn rules. Predicates that do not appear in the DL are called ordinary predicates. Each component of ARTIGENCE is described below.

2.1. Description logic component

In description logics, concepts represent the classes of objects in the domain of interest, while roles represent binary relations between objects. Complex concepts and roles can be defined by means of suitable constructors applied to concepts and roles. Decidability and complexity of the inference problems depends on the expressive power of the DLs, and the expressive power is restricted in a set of allowed constructors for building concepts and roles. Description logics and their properties vary depending on the set of allowed constructors and the kinds of statements allowed in the knowledge base. On the one hand, very expressive DLs are likely to have inference problems of high complexity, or they may even be undecidable. On the other hand, very weak DLs may not be sufficiently expressive to represent the important concepts of a given application. Here we consider $\mathcal{ALCNR}$ as the description logic component of ARTIGENCE. $\mathcal{ALCNR}$ proposed in [6] is one of the most expressive description logics with decidable inference procedures so far.

Concepts and roles in $\mathcal{ALCNR}$ can be established by using the following syntax (where P_i denotes a primitive role name, C and D denote arbitrary concepts, and R an arbitrary role):

$C, D \rightarrow A\|$	(concept name)
$\top\|$	(top concept)
$\bot\|$	(bottom concept)
$C \sqcap D\|$	(conjunction)
$C \sqcup D\|$	(disjunction)
$\neg C\|$	(complement)
$\forall R.C\|$	(universal quantification)
$\exists R.C\|$	(existential quantification)
$(\geqslant n\ R)\|(\leqslant n\ R)$	(number restrictions)
$R \rightarrow P_1 \sqcap \ldots \sqcap P_k$	(role conjunction)

Description logics have a model-theoretic semantics, based on the notion of interpretation. A *canonical interpretation* $\mathcal{I}$ is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a nonempty set $\Delta^{\mathcal{I}}$ (called the *domain*) and an *interpretation function* $\cdot^{\mathcal{I}}$ mapping different individuals into different elements of $\Delta^{\mathcal{I}}$, primitive concepts into subsets of $\Delta^{\mathcal{I}}$, and primitive roles into subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation of complex concepts and roles must satisfy the following equations (#{} denotes the cardinality of a set):

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$$

$$\bot^{\mathcal{I}} = \varnothing$$

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$$

$$(\forall R.C)^{\mathcal{I}} = \{d_1 \in \Delta^{\mathcal{I}} | \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow d_2 \in C^{\mathcal{I}}\}$$

$$(\exists R.C)^{\mathcal{I}} = \{d_1 \in \Delta\mathcal{I} | \exists d_2 : (d_1, d_2) \in R^{\mathcal{I}} \wedge d_2 \in C^{\mathcal{I}}\}$$

$$(\geqslant n\ R)^{\mathcal{I}} = \{d_1 \in \Delta^{\mathcal{I}} | \#\{d_2 : |(d_1, d_2) \in R^{\mathcal{I}}\} \geqslant n\}$$

$$(\leqslant n\ R)^{\mathcal{I}} = \{d_1 \in \Delta^{\mathcal{I}} | \#\{d_2 : |(d_1, d_2) \in R^{\mathcal{I}}\} \leqslant n\}$$

$$(P_1 \sqcap \ldots \sqcap P_k)^{\mathcal{I}} = P_1^{\mathcal{I}} \cap \ldots \cap P_k^{\mathcal{I}}$$

A knowledge base of description logic is typically comprised of two components, a "TBox" and an "ABox". The TBox contains *intensional* knowledge in the form of a terminology and is built through declarations that describe general properties of concepts. The ABox contains *extensional* knowledge (also called assertional knowledge) that is specific to the individuals of the domain of discourse.

The *terminological axioms* in $\mathcal{ALCNR}$ are either *concept inclusions* or *concept definitions*. A concept inclusion has the form $C \sqsubseteq D$, where C and D are two arbitrary $\mathcal{ALCNR}$-concepts. The inclusion statement specifies that every instance of C is also an instance of D. A concept definition is a statement of the form $A \equiv D$, where A is a concept and D is a concept description. An interpretation $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies a definition $A \equiv D$ if $A^{\mathcal{I}} = D^{\mathcal{I}}$. A set of definitions is said to be *cyclic* if a concept name may occur directly or indirectly within its own definition. Terminological cycles are allowed in $\mathcal{ALCNR}$ statements, while role descriptions are limited to conjunctions of primitive roles.

A TBox, denoted by $\mathcal{T}$, is considered to be a set of concept inclusions and concept definitions. An interpretation $\mathcal{I}$ is a model of $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every concept inclusion $C \sqsubseteq D$ in the TBox, and $A^{\mathcal{I}} = D^{\mathcal{I}}$ for every concept definition $A \equiv D$.

In the ABox, one can describe instance-of relations between individuals and concepts, and between pairs of individuals and roles. Instance-of relations are expressed in terms of *membership assertions* that have the forms: $C(a)$, and $R(a, b)$, where a and b are individuals, C is an $\mathcal{ALCNR}$ concept, and R is an $\mathcal{ALCNR}$ role. Intuitively, the first form (called *concept assertions*) specifies that a is an instance of C, whereas the second form (called *role assertions*) specifies that a is related to b by means of the role R, or b is a filler of the role R for a. An ABox, denoted by $\mathcal{A}$, is a finite set of such assertions.

In order to give semantics to membership assertions, the function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ is extended to individuals. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ not only maps primitive concepts and roles to sets and relations, but in addition maps each individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ in such a way that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$. This property is called unique name assumption which ensures that different individuals are interpreted as different objects.

The interpretation $\mathcal{I}$ satisfies the concept assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and it satisfies the role assertion $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model for an ABox $\mathcal{A}$ if $\mathcal{T}$ satisfies every assertion in $\mathcal{A}$.

An $\mathcal{ALCNR}$-knowledge base Σ is a pair $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a TBox and A is an ABox. An interpretation $\mathcal{I}$ is a model for Σ if it is both a model for $\mathcal{T}$ and $\mathcal{A}$.

Example 2.1. Consider the following description logic knowledge base $\Sigma_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$:

$$\begin{aligned}\mathcal{T}_1 = \{&\mathit{ForeignCompany} \sqcap \mathit{DomesticCompany} \sqsubseteq \bot\\ &\mathit{CompeteWithForeign} \equiv \exists\, \mathit{SameIndustry}.\mathit{ForeignCompany}\\ &\mathit{CompeteWithDomestic} \equiv \exists\, \mathit{SameIndustry}.\mathit{DomesticCompany}\\ &\mathit{ProtectedCompany} \equiv \forall\, \mathit{SameIndustry}.\neg \mathit{ForeignCompany}\\ &\mathit{ForeignCompany} \sqcap \mathit{ProtectedCompany} \sqsubseteq \bot\\ &\mathit{NonMonopoly} \equiv \mathit{CompeteWithForeign} \sqcup \mathit{CompeteWithDomestic}\}\\ \mathcal{A}_1 = \{&\mathit{NonMonopoly}(b)\}\end{aligned}$$

The concepts *ForeignCompany* and *DomesticCompany* are primitive concepts, and the first inclusion states that they are disjoint. The concept *CompeteWithForeign* (respectively, *CompeteWithDomestic*) is defined to be set of companies that compete with foreign companies (respectively, domestic companies) in the same industry, i.e., the set of individuals that at least one filler of the role *SameIndustry* is a member of *ForeignCompany* (respectively, *DomesticCompany*). The concept *ProtectedCompany* represents the set of companies that have no foreign competitors in their industry. The fifth inclusion states that concepts *ForeignCompany* and *ProtectedCompany* are also disjoint. The concept *NonMonopoly* is defined to be set of companies that belong either to *CompeteWithForeign* or to *CompeteWithDomestic*. The ABox contains only one assertion that states that b is an instance of concept *NonMonopoly*.

2.2. Horn rule component

A clause in first order logics has the form, $L_1 \vee \ldots \vee L_k$, where each L_i is a literal. A literal L has either the form A or $\neg A$, where A is atom. The literal is said to be *positive* in case A, or to be *negative* in case $\neg A$. It is said to be a Horn rule when at most one of its literals is positive. The Horn rule component $\mathcal{R}$ of ARTIGENCE knowledge base contains a set of Horn rules each having the form:

$$q(\overline{Y}) \vee \neg p_1(\overline{X}_1) \vee \ldots \vee \neg p_n(\overline{X}_n)$$

where $\overline{X}_1, \ldots, \overline{X}_n$, and $\overline{Y}$ are tuples of variables or constants. A Horn rule is said to be *safe* when a variable that appears in $\overline{Y}$ must also appear in $\overline{X}_1 \cup \ldots \cup \overline{X}_n$. The safety condition guarantees that the set of all ground facts derivable from a set of Horn rules is finite. The predicates $p_1, \ldots, p_n$ may be either concept or role names defined in $\mathcal{T}$, or *ordinary* predicates that do not appear in $\mathcal{T}$. The predicate q must be an ordinary predicate. Note that no restriction is placed on the arity of the ordinary predicates. ARTIGENCE does not allow concept and roles atoms to appear in the consequents of the Horn rules. The underlying assumption behind this restriction is that the description logic component completely describes the hierarchical structures in the domain. This design decision follows the idea that rules may not be used to derive the hierarchical structure knowledge. Since $\neg p \vee q$ and $p \rightarrow q$ are logically equivalent, a Horn rule can be written equivalently as:

$$p_1(\overline{X}_1) \wedge \ldots \wedge p_n(\overline{X}_n) \rightarrow q(\overline{Y})$$

where $q(\overline{Y})$ is called the *head* or *consequent* of the rule, and $p_1(\overline{X}_1) \wedge \ldots \wedge p_n(\overline{X}_n)$ is called the *body* or *antecedent* of the rule.

A predicate q is said to depend on a predicate p if p appears in the antecedent of the Horn rule whose consequent is q. A set of rules is said to be *recursive* if there is a cycle in the dependency relation among predicates. Recursive rules are allowed in the ARTIGENCE knowledge base.

Example 2.2. Consider the following rules $\mathcal{R}_1$, where *serviceBy*, *highQuality*, and *price* are ordinary predicates, as well as the description logic knowledge base Σ_1 from Example 2.1. We use the notation that ordinary predicate symbols are strings beginning with a lower case letter, whereas concept or role names are strings beginning with an upper case letter.

$$\begin{aligned}\mathcal{R}_1 = \{&r_1 : \mathit{serviceBy}(x,y) \wedge \mathit{ProtectedCompany}(y) \rightarrow \mathit{price}(x, \mathit{high})\\ &r_2 : \mathit{serviceBy}(x,y) \wedge \mathit{SameIndustry}(y,z) \wedge\ \mathit{ForeignCompany}(z)\\ &\wedge\ \mathit{highQuality}(y,x) \rightarrow \mathit{price}(x, \mathit{high})\}\end{aligned}$$

However, not all knowledge in the world is deterministic; uncertainty often plays an important role, and weights are required. In ARTIGENCE, a set of Horn rules can be given with some information about the level of confidence we may have in them. The confidence level for a rule is indicated by its probability mass, which is a number in the interval [0, 1]. Since the precise probability mass of a rule may be unknown, we will suppose that an interval $[l, u] \in [0, 1]$ is given, with which the mass lies. If nothing is know about the mass, we set $l = 0$, and $u = 1$.

2.3. Ground fact component

The ground fact component is a set of ground atomic facts each having the form $p(\bar{a})$, where $\bar{a}$ is a tuple of constants and p is a ordinary predicate.

Example 2.3. As an example of an ARTIGENCE knowledge base, we consider $\mathcal{K}_1 = \langle \Sigma_1, \mathcal{R}_1, \mathcal{F}_1 \rangle$, where $\Sigma_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ is the description logic component from Example 2.1, $\mathcal{R}_1$ is the set of Horn rules from Example 2.2, and $\mathcal{F}_1$ is the following set of ground facts:

$$\mathcal{F}_1 = \{f_1 : serviceBy(a, b),\ f_2 : highQuality(b, a)\}$$

Just as the rules in ARTIGENCE, every ground fact can be given with real numbers l, $u \in [0, 1]$ that indicate the degree of confidence we have in it.

2.4. Semantics of ARTIGENCE knowledge base

In ARTIGENCE, a knowledge base $\mathcal{K}$ is defined as a triple, $\mathcal{K} = \langle \Sigma, \mathcal{R}, \mathcal{F} \rangle$, where

- $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ is an $\mathcal{ALCNR}$ knowledge base;
- $\mathcal{R}$ is a set of function-free Horn rules;
- $\mathcal{F}$ is a set of ground atomic facts.

The semantics of ARTIGENCE is naturally derived from the semantics of its components. An interpretation $\mathcal{I}^*$ is a pair $\mathcal{I}^* = (\Delta^{\mathcal{I}^*}, \cdot^{\mathcal{I}^*})$ consisting of a nonempty domain $\Delta^{\mathcal{I}^*}$ and an interpretation function $\cdot^{\mathcal{I}^*}$ mapping every individuals a into an object $a^{\mathcal{I}^*} \in \Delta^{\mathcal{I}^*}$, and every predicate of arity n into a relation of arity n over the domain $\Delta^{\mathcal{I}^*}$. An interpretation $\mathcal{I}^*$ is a model of a knowledge base $\mathcal{K}$ if it is a model of each of its components. The model of the description logic component was defined in Section 2.1. An interpretation $\mathcal{I}^*$ is a model of a rule r if, whenever α is a mapping from the variables appearing in r to the domain $\Delta^{\mathcal{I}^*}$, such that $\alpha(\overline{X}_i) \in p_i^{\mathcal{I}^*}$ for every atom of the antecedent of r, then $\alpha(\overline{Y}) \in q^{\mathcal{I}^*}$, where $q(\overline{Y})$ is the consequent of r. $\mathcal{I}^*$ is a model of a ground fact $p(\bar{a})$ if $\bar{a}^{\mathcal{I}^*} \in p^{\mathcal{I}^*}$. Note that every constant occurring in $\mathcal{F}$ and $\mathcal{R}$ appears also in Σ. This is not a real limitation, because for each individual a not appearing in Σ we can add to $\mathcal{A}$ the assertion $\top(a)$. We also make the unique name assumption that if a and b are constants in $\Delta^{\mathcal{I}^*}$, then $a^{\mathcal{I}^*} \neq b^{\mathcal{I}^*}$.

2.5. Reasoning in ARTIGENCE

Given a ARTIGENCE knowledge base $\mathcal{K}$, a query Q is a conjunction of the form $p(\overline{X})$, where p can be any predicate, and $\overline{X}$ is a tuple of variables or constants.

An answer to a query Q is a substitution θ for the variables in Q. The answer θ is correct with respect to the knowledge base $\mathcal{K}$ if $\mathcal{K} \models Q\theta$, where $Q\theta$ is a conjunction of grounds. In other words, the answer to a query Q is the ground instance of Q which is logical consequence of $\mathcal{K}$. In particular, a query for $\mathcal{K}$ is written as the form $Q = p(\bar{a})$, where $\bar{a}$ is a tuple of constants.

The following example shows some of additional inferences can be drawn from ARTIGENCE but not from either of its components alone, which has also been observed by Levy and Rousset in [2].

Example 2.4. Given the ARTIGENCE knowledge base $\mathcal{K}_1 = \langle \Sigma_1, \mathcal{R}_1, \mathcal{F}_1 \rangle$, does $\mathcal{K}_1$ *price*(*a*, *high*)?

For this example, $\Sigma_1 \cup \mathcal{F}_1$ does not entail the antecedent of any single rule in $\mathcal{R}_1$. However, we can make the inference by the following cases: (i) if company b has at least one foreign competitor in the same industry, then the antecedent of rule r_2 will be entailed, and the consequent *price*(*a*, *high*) will follow; (ii) if b has no foreign competitors, then *ProtectedCompany*(b) will be entailed, and *price*(*a*, *high*) will follow by rule r_1.

The above example illustrated that an ARTIGENCE knowledge base may entail the *disjunction* of the antecedents of two rules without entailing either of them, and the reasoning may require making case analyses. Therefore, traditional Horn rule inference mechanisms that consider each rule in isolation are inadequate for ARTIGENCE knowledge base. In addition, it is possible for DL to express the existence of individuals whose identities might not be expressed explicitly. For the case (i), we suppose that company b has at least one foreign competitor (known or unknown), then the antecedent of rule r_2 is entailed. This case showed that an ARTIGENCE knowledge base may entail the antecedent of a rule without the antecedent being instantiated. In contrast, standard Horn rule inference procedures try to instantiate the antecedent in order to derive the consequent for each rule. These problems are caused by the "open-world semantics" of ABox that may represent many interpretations or models.

Since an ABox of DL represents possibly many interpretations, a query Q to an ARTIGENCE knowledge base is valid if and only if in any interpretation or model of the DL component, there exists at least one refutation for the component of Horn rules. It cannot be done by deriving just one empty clause for the component of Horn rules in ARTIGENCE, as in the classical resolution calculus or its generalizations. The reason is that such an empty clause might only be entailed by some of the DL models, but not by all of them. For Example 2.4, if we delete rule r_1 from the knowledge base $\mathcal{K}_1$, the query *price*(*a*, *high*) will be incorrect with respect to $\mathcal{K}_1$. This is because that all the models of Σ_1 can be divided into two classes: the first one in which company b has at least one foreign competitor; and another one in which b has no foreign competitors (see Example 3.2 for detail). Without rule r_1, the query *price*(*a*, *high*) will not follow in the models of the latter class.

In order to design the inference procedure for answering queries in ARTIGENCE, a constrained logic scheme has been introduced with a resolution principle for the Horn rules whose variables are constrained by $\mathcal{ALCNR}$ description logic. Our algorithm is mainly based on the technique of a resolution principle for constrained logics [5]. To summarize, the algorithm has three steps:

- **Step 1.** Build all the canonical interpretations for a DL knowledge base and represent them by Herbrand structures.
- **Step 2.** Collect all the SLD-derivations ending with the empty clauses for a set of function-free recursive Horn rules.
- **Step 3.** For every canonical interpretation of DL knowledge base, check if there is a constrained SLD-refutation.

Although the standard resolution of Horn clauses is semi-decidable, it is decidable for the above step 2. Note that Herbrand universe is finite when Horn clauses are function-free, even with recursive rules. Therefore, we can compute the refutation trees up to a finite depth depending on given a set of Horn clauses. In the next section we introduce the inference procedure based upon the resolution principle for constrained logics, and prove its soundness, completeness and decidability (step 2 and 3). In Section 4 we discuss the decidable reasoning in $\mathcal{ALCNR}$ description logic (step 1).

3. Constrained resolution in ARTIGENCE

The language ARTIGENCE combines the expressive power of Horn rules and description logics by allowing the usage of concepts and roles from the description logic component as constraints on variables that appear in Horn rules. Constraints can be seen as quantifier restrictions filtering out the values that any interpretation of the underlying DL can assign to the variables of a Horn clause with such restricted quantifiers. It shows that this constrained resolution is sound, complete and decidable in which a set of constrained Horn clauses is unsatisfiable over a certain DL if and only if for each canonical interpretation of the DL we can deduce a constrained empty clause whose constraint is satisfiable in that interpretation.

3.1. Resolution principle for constrained logics

In order to link description logics with Horn rules, description logics will come in via the constraints that play the role of quantifier restrictions. The constraint theory that is a set of all Herbrand structures is a tractable case [5]. Therefore, all the canonical interpretations of the DL, as constraint theory, are required to be represented by Herbrand structure, which can be done trivially.

The Horn rules defined in Section 2.2 are called *constrained clauses*, denoted by *R*-clauses. The DL concepts and roles that appear in an *R*-clause are called the *constraints* or *restrictions* of the *R*-clause, denoted by *R*. *P* denotes the remainder of the *R*-clause after its constraints have been taken away and we call *P* the *kernel* of the *R*-clause. An *R*-clause is logically equivalent to a formula $\forall_{\overline{X}}(R \to P)$, where $\overline{X}$ is a finite set of variables. We usually denote constrained clauses by the form $P\|R$ instead of $\forall_{\overline{X}}(R \to P)$.

Example 3.1. The rule r_1 from Example 2.2 can be rewritten into
$\forall_y(ProtectedCompany(y)) \to (\neg serviceBy(x,y) \vee price(x,high))$
or $(\neg serviceBy(x,y) \vee price(x,high))\|(ProtectedCompany(y))$
where $R : ProtectedCompany(y)$
and $P : \neg serviceBy(x,y) \vee price(x,high)$

Definition 3.1. Let L_1 and L_2 be any pair of *R*-clauses, such that L_1 contains at least one positive literal $p(x_1, \ldots, x_n)$ and L_2 contains at least one negative literal $\neg p(y_1, \ldots, y_n)$ each with the same predicate p, where x_i, y_i $(i = 1,\ldots,n, n \geqslant 1)$ are variables or constants. Let θ be the *most general unifier* such that $x_i\theta = y_i\theta$, $i \in n$. The *constrained resolution rule* for L_1 and L_2 with substitution θ:

$$L_1 : p(x_1,\ldots,x_n) \cup P_1\|R_1$$
$$L_2 : \neg p(y_1,\ldots,y_n) \cup P_2\|R_2$$
$$L_3 : P_1\theta \cup P_2\theta \| R_1\theta \wedge R_2\theta$$

where P_1 and P_2 are the remaining parts of the two *R*-clauses. The derived clause L_3 is called a *constrained resolvent* of the two parent *R*-clauses.

A *constrained resolution step* $\mathbb{R} \to \mathbb{R}'$ transforms an *R*-clause set $\mathbb{R}$ into the next set $\mathbb{R}'$ by adding a derived *R*-clause that is a resolvent of two parent *R*-clauses in the previous set $\mathbb{R}$. A *constrained derivation* is a sequence of constrained resolution steps starting with an initial set $\mathbb{R}_0$ of *R*-clauses. A *constrained refutation* of an *R*-clause set $\mathbb{R}_0$ is a constrained derivation starting with the set $\mathbb{R}_0$, such that for each model of the constraint theory there is a set $\mathbb{R}_n$ of *R*-clauses in the derivation containing an empty *R*-clause $\square\|R$ whose restriction is satisfied by that model.

3.2. Proof of soundness, completeness and decidability

Let us first prove whether the constrained resolution is sound and complete method for checking if a query *Q* is the logical consequence of a knowledge base $\mathcal{K}$. We reformulate some results from [5] here, and show that query answering in ARTIGENCE is decidable.

Lemma 3.1. *An R-clause produced by the constrained resolution rule from two parent R-clauses also logically follows from them.*

Proof. Taking the signature from Definition 3.1, we first consider the kernel of *R*-clauses L_1 and L_2. Let θ be the most general unifier for L_1 and L_2, and through application of *universal instantiation*, we have

$$p(x_1,\ldots,x_n) \vee P_1 \vdash p(x_1,\ldots,x_n)\theta \vee P_1\theta$$
$$\neg p(y_1,\ldots,y_n) \vee P_2 \vdash \neg p(y_1,\ldots,y_n)\theta \vee P_2\theta$$

then $(p(x_1,\ldots,x_n) \vee P_1) \wedge (\neg p(y_1,\ldots,y_n) \vee P_2) \vdash (p(x_1,\ldots,x_n)\theta \vee P_1\theta) \wedge (\neg p(y_1,\ldots,y_n)\theta \vee P_2\theta)$
and $(A \vee B) \wedge (\neg A \vee C) \to (B \vee C)$is tautology,
then $(p(x_1,\ldots,x_n)\theta \vee P_1\theta) \wedge (\neg p(y_1,\ldots,y_n)\theta \vee P_2\theta) \vdash P_1\theta \vee P_2\theta$
hence $(p(x_1,\ldots,x_n) \vee P_1) \wedge (\neg p(y_1,\ldots,y_n) \vee P_2) \vdash P_1\theta \vee P_2\theta$.

Note that all variables in R_1 and R_2 be universally quantified. This allows full freedom these variables to be replaced by terms from the domain, and the constraints preserves validity. □

Definition 3.2. Let a Herbrand structure $\mathcal{M} = (H, I)$ be a model for a DL knowledge base Σ, where the domain of the structure is a Herbrand universe, denoted by H, and each Herbrand structure can be identified with its Herbrand interpretation, denoted by I. A constraint theory $\Re$ is the set of all such $\mathcal{M}$ models. A structure $\mathcal{M}^* = (H^*, I^*)$ expands the model $\mathcal{M}$ to the ordinary predicates in Horn rules and the ground fact component. The relationship of $\mathcal{M}^*$ to $\mathcal{M}$ is the same as $\mathcal{I}^*$ to $\mathcal{I}$ (see Section 2.4). Let α be an assignment that maps each variable into an element of the domain H^*.

- Let *R* be the constraint or restriction of an *R*-clause. If there is a model $\mathcal{M}$ and an assignment α that satisfy *R*, we write $(\mathcal{M}, \alpha) \models R$.
- Let *P* be the kernel of an *R*-clause. We call the triple $(\mathcal{M}, \alpha, P)$ a $\mathcal{M}$*-instance* of the *R*-clause iff $(\mathcal{M}, \alpha) \models R$.
- Let $\mathbb{R}$ be a set of *R*-clauses. Then a set $\{(\mathcal{M}, \alpha_i, P_i) : i \in n\}$ of $\mathcal{M}$-instances of *R*-clauses $P_i\|R_i$ $(i \in n)$ of $\mathbb{R}$ is called a $\mathcal{M}$*-instantiation* of $\mathbb{R}$ iff $(\mathcal{M}, \alpha_i) \models R_i$ for each $i \in n$.
- The set $\{(\mathcal{M}, \alpha, P) : P\|R$ in $\mathbb{R}$ and $(\mathcal{M}, \alpha) \models R\}$ of $\mathcal{M}$-instances of all *R*-clauses in $\mathbb{R}$ is called the $\mathcal{M}$*-base* of $\mathbb{R}$.
- We call a set $\mathbb{R}$ of *R*-clauses is satisfiable iff there is an expansion $\mathcal{M}^*$ of $\mathcal{M}$ satisfies the $\mathcal{M}$-base of $\mathbb{R}$; otherwise $\mathbb{R}$ is called unsatisfiable.

  The above definition is the generalization of classical notion of ground instances of clauses, which is considered as a triple consisting of a Herbrand universe, a ground assignment, and the clause to be instantiated.

Theorem 3.2 (*Herbrand Theorem*). *A set $\mathbb{R}$ of R-clauses is unsatisfiable iff for each model $\mathcal{M}$ in the constraint theory $\Re$ there is a finite $\mathcal{M}$-instantiation of $\mathbb{R}$ that is unsatisfiable (reformulated from [5]).*

Proof

($\Rightarrow$) Let $\mathbb{R}$ be unsatisfiable. Suppose there is a $\mathcal{M} \in \Re$ such that each finite $\mathcal{M}$-instantiation of $\mathbb{R}$ is satisfiable. By the Compactness Theorem for first order logics (note that description logics and Horn rules are strict subsets of first order logic) we have that the $\mathcal{M}$-base for $\mathbb{R}$ is satisfiable. Hence there is an expansion $\mathcal{M}^*$, such that for each $P\|R$ in $\mathbb{R}$ and each assignment α with $(\mathcal{M}, \alpha) \models R$ we have $(\mathcal{M}^*, \alpha) \models P$, which is a contradiction to the unsatisfiability of $\mathbb{R}$.
($\Leftarrow$) This direction is obvious. If for each model $\mathcal{M} \in \Re$ there is no $\mathcal{M}$-base of $\mathbb{R}$ that is satisfiable. Hence it is impossible to construct an expansion $\mathcal{M}^*$ of $\mathcal{M}$ which satisfies $\mathbb{R}$. □

Definition 3.3. In order to prove the unsatisfiability of $\mathbb{R}$ by using the unsatisfiability of a $\mathcal{M}$-instantiation we introduce the following $\mathcal{M}$*-resolution rule* for $\mathcal{M}$-instances which generalizes the constrained resolution rule (see Definition 3.1).

$$L_1' : \quad (\mathcal{M}, \alpha, p(x_1,\ldots,x_n) \cup P_1)$$
$$L_2' : \quad (\mathcal{M}, \alpha, \neg p(y_1,\ldots,y_n) \cup P_2)$$
$$L_3' : \quad (\mathcal{M}, \alpha, P_1\theta \cup P_2\theta) \text{ and } R_3 = R_1\theta \wedge R_2\theta$$

Where the constraint of R-clause L'_3 is $R_1\theta \wedge R_2\theta$ and $(\mathcal{M}, \alpha) \models R_1\theta \wedge R_2\theta$.

The following Lifting Lemma shows that a $\mathcal{M}$-resolvent of $\mathcal{M}$-instances of two R-clauses is also a $\mathcal{M}$-instance of the constraint resolvent of the two R-clauses.

Lifting Lemma 3.3. *Let $P_1\|R_1$ and $P_2\|R_2$ be two R-clauses and let $(\mathcal{M}, \alpha, P_1)$ and $(\mathcal{M}, \alpha, P_2)$ be the $\mathcal{M}$-instances of the two R-clauses. Then a $\mathcal{M}$-resolution step on the two $\mathcal{M}$-instances can be lifted to a constrained resolution step on the two R-clauses, such that the $\mathcal{M}$-resolvent $(\mathcal{M}, \alpha, P)$ is a $\mathcal{M}$-instance of the constrained resolvent $P\|R$.*

Proof. It follows from the definition of $\mathcal{M}$-instance and $\mathcal{M}$-resolvent. □

The following proposition says that $\mathcal{M}$-resolution can be used to deduce the empty $\mathcal{M}$-instance from every unsatisfiable $\mathcal{M}$-instantiation.

Proposition 3.4. *If a finite $\mathcal{M}$-instantiation $\mathbb{D}$ of a set $\mathbb{R}$ of R-clauses is unsatisfiable, then there exists a finite derivation containing an empty $\mathcal{M}$-instance through $\mathcal{M}$-resolution (reformulated from [5]).*

Proof. Mathematical induction has been used to prove this proposition. We consider two cases.

Case 1. If the empty $\mathcal{M}$-instance is already in $\mathbb{D}$, we finished.

Case 2. If the empty $\mathcal{M}$-instance is not in $\mathbb{D}$, we proceed by induction on the number N of excess literals in $\mathbb{D}$ which is the number of literal occurrences in $\mathbb{D}$ minus the number of $\mathcal{M}$-instance in $\mathbb{D}$.

$N = 0$ (then no non-unit $\mathcal{M}$-instance is in $\mathbb{D}$. Here an unit is a $\mathcal{M}$-instance in which the kernel P has only one literal): Because $\mathbb{D}$ is unsatisfiable there must be two complementary units $(\mathcal{M}, \alpha, L'_1)$ and $(\mathcal{M}, \alpha, L'_2)$ in $\mathbb{D}$ (the corresponding variables of the two $\mathcal{M}$-instances are assigned to the same elements in $\mathcal{M}$ by α). A $\mathcal{M}$-resolution step on the two units will deduce an empty $\mathcal{M}$-instance. Otherwise let be any expansion $\mathcal{M}^*$ of $\mathcal{M}$, such that $p(\alpha(x_1), \ldots, \alpha(x_n))$ is true in $\mathcal{M}^*$ if the unit $(\mathcal{M}, \alpha, p(x_1, \ldots, x_n))$ is in $\mathbb{D}$, and $p(\alpha(x_1), \ldots, \alpha(x_n))$ is false in $\mathcal{M}^*$ otherwise. Then $\mathcal{M}^*$ satisfies $\mathbb{D}$ by definition. Therefore the empty $\mathcal{M}$-instance can be derived.

$N > 0$: let the Proposition be true for all $\mathbb{D}$ with fewer than N excess literals. Then there is at least one non-unit $\mathcal{M}$-instance, say $(\mathcal{M}, \alpha, L')$, that has at least one excess literal. Let $L'' := L' \backslash \{p\}$ for some literal p in L'. Then the set

$$\mathbb{D}' := (\mathbb{D} \setminus \{(\mathcal{M}, \alpha, L')\}) \cup \{(\mathcal{M}, \alpha, L'')\}$$

is also an unsatisfiable set. Since it contains one fewer excess literal, by the induction hypothesis there is a deduction of the empty $\mathcal{M}$-instance. Similarly, the set

$$\mathcal{D}'' := (\mathbb{D} \setminus \{(\mathcal{M}, \alpha, L')\}) \cup \{(\mathcal{M}, \alpha, \{p\})\}$$

is unsatisfiable and there is a deduction of the empty $\mathcal{M}$-instance by induction hypothesis. If p is not used in the deduction for $\mathbb{D}'$, then it works also for $\mathbb{D}$. Otherwise a deduction of the empty $\mathcal{M}$-instance for $\mathbb{D}$ can be constructed as follows. We add p back into L' and all its descendants in the deduction for $\mathbb{D}'$ in order to get a deduction for $\mathbb{D}$. Now, this modified deduction contains either the empty $\mathcal{M}$-instance or the unit $(\mathcal{M}, \alpha, \{p\})$ resulting from the empty $\mathcal{M}$-instance after adding p to it. In the latter case we get a deduction of the empty $\mathcal{M}$-instance for $\mathbb{D}$ by appending the deduction of the empty $\mathcal{M}$-instance from $\mathbb{D}''$ onto the end of this modified deduction. □

Theorem 3.5 (*Soundness and Completeness for the Constrained Resolution*). *A set $\mathbb{R}$ of R-clauses is unsatisfiable iff for each model $\mathcal{M} \in \Re$ there exists a constrained derivation from $\mathbb{R}$ containing an empty R-clause $\square\|R$, such that $\mathcal{M} \models R$ (reformulated from [5]).*

Proof.

Soundness ($\Leftarrow$): Assume $\mathbb{R}$ were satisfiable, then there exists a $\mathcal{M} \in \Re$ such that $\mathcal{M}^* \models \mathbb{R}$ for some expansion $\mathcal{M}^*$ of $\mathcal{M}$. By Lemma 3.1 $\mathcal{M}^* \models \forall\overline{X}(R \rightarrow P)$ for each R-clause $P\|R$ that can be derived by the constrained resolution from $\mathbb{R}$. By the precondition an empty R-clause $\square\|R$ with $\mathcal{M} \models R$ is derivable from $\mathbb{R}$. Therefore there is an assignment α with $(\mathcal{M}^*, \alpha) \models \square$. This is a contradiction.

Completeness ($\Rightarrow$): Let $\mathcal{M}$ be a model in $\Re$. If $\mathbb{R}$ is unsatisfiable, then for each $\mathcal{M} \in \Re$ there must be a finite $\mathcal{M}$-instantiation that is unsatisfiable (Herbrand Theorem 3.2), and we can deduce the empty $\mathcal{M}$-instance by $\mathcal{M}$-resolution (Proposition 3.4). By the Lifting Lemma 3.3 this refutation can be lifted and hence the collected restriction R of the empty $\mathcal{M}$-instance is satisfied by $\mathcal{M}$. □

Note that it is enough to consider for each $\mathcal{M} \in \Re$ only those R-clauses of $\mathbb{R}$ whose constraints are satisfiable by that $\mathcal{M}$. The same holds for the resolvents that are derived. The following theorem shows that our algorithm always terminates.

Theorem 3.6 (*Decidability*). *Query answering in ARTIGENCE is decidable.*

Proof. In order to answer a query Q to an ARTIGENCE knowledge base $\mathcal{K}$, our algorithm has three steps. First, we build all models for an $\mathcal{ALCNR}$ description logic knowledge base and represent them by Herbrand structures, which has proved to be a decidable problem (see Section 4.3). Second, we collect all the SLD-derivations ending with the empty clauses and the corresponding constraints for a set of constrained clauses. Note that Herbrand universe is finite when Horn clauses are function-free, even with recursive rules since the number of constants is finite. It is well-known that SLD-resolution for Datalog is complete if one computes the refutation trees up to a finite depth [7]. Let *nconst* denote the number of distinct constant symbols which occur in $\mathcal{R}$, let *npreds* denote the number of distinct predicate symbols occurring in $\mathcal{R}$ and let *maxargs* denote the maximum arity of all predicates in $\mathcal{R}$. The maximum depth of refutation trees is $nconst \cdot npreds^{maxargs}$. Since there is a straightforward one-to-one mapping between constrained SLD-derivations and the Datalog derivations, we can compute the refutation trees up to a finite depth depending on given a set of Horn clauses, and this step is also decidable. Finally, for every canonical interpretation of the DL knowledge base, check if there exists at least one SLD-refutation whose constraints are satisfied by that model. There are two kinds of constraints, $C(a)$ and $R(a,b)$, where a and b are individuals, C is an $\mathcal{ALCNR}$ concept, and R is an $\mathcal{ALCNR}$ role. Checking satisfaction of the constraints can be done by performing a lookup in a canonical interpretation I of the DL knowledge base. If we are performing a lookup for a fact of the form $C(a)$, we check whether $a^{\mathcal{I}} \in C^{\mathcal{I}}$. Lookups for role atoms can be done similarly. □

Example 3.2. Given the ARTIGENCE knowledge base $\mathcal{K}_1 = \langle \Sigma_1, \mathcal{R}_1, \mathcal{F}_1 \rangle$, does *price*(*a*, *high*) logically follow from $\mathcal{K}_1$?

Considering the description logic knowledge base $\Sigma_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ from Example 2.1, the following two canonical interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ satisfy all the inclusions in $\mathcal{T}_1$ and all the assertions in $\mathcal{A}_1$,

and therefore they are models for Σ_1. It can be proved that all other models of Σ_1 are equivalent to $\mathcal{I}_1$ or $\mathcal{I}_2$. For clarity, we have already omitted redundant instances that are irrelevant to the query.

$\mathcal{I}_1$:	$\mathcal{I}_2$:
$\Delta^{\mathcal{I}_1} = \{b, v_1\}$	$\Delta^{\mathcal{I}_2} = \{b, v_1\}$
$ForeignCompany^{\mathcal{I}_1} = \{v_1\}$	$ForeignCompany^{\mathcal{I}_2} = \{\varnothing\}$
$DomesticCompany^{\mathcal{I}_1} = \{\varnothing\}$	$DomesticCompany^{\mathcal{I}_2} = \{v_1\}$
$CompeteWithForeign^{\mathcal{I}_1} = \{b\}$	$CompeteWithForeign^{\mathcal{I}_2} = \{\varnothing\}$
$CompeteWithDomestic^{\mathcal{I}_1} = \{\varnothing\}$	$CompeteWithDomestic^{\mathcal{I}_2} = \{b\}$
$ProtectedCompany^{\mathcal{I}_1} = \{\varnothing\}$	$ProtectedCompany^{\mathcal{I}_2} = \{b, v_1\}$
$NonMonopoly^{\mathcal{I}_1} = \{b\}$	$NonMonopoly^{\mathcal{I}_2} = \{b\}$
$SameIndustry^{\mathcal{I}_1} = \{(b, v_1)\}$	$SameIndustry^{\mathcal{I}_2} = \{(b, v_1)\}$

where the instance v_1 is added during the procedure of building the canonical interpretation for $\mathcal{ALCNR}$ knowledge base Σ_1, and the name v_1 can be substituted by any other arbitrary symbol.

The above two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ of the $\mathcal{ALCNR}$ knowledge base Σ_1 can be represented by Herbrand structures $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively.

$\mathcal{M}_1 = (H_1, I_1)$	$\mathcal{M}_2 = (H_2, I_2)$
$H_1 = \{b, v_1\}$	$H_2 = \{b, v_1\}$
$I_1 = \{ForeignCompany(v_1),$	$I_2 = \{DomesticCompany(v_1),$
$CompeteWithForeign(b),$	$CompeteWithDomestic(b),$
$NonMonopoly(b),$	$NonMonopoly(b),$
$SameIndustry(b, v_1)\}$	$ProtectedCompany(b),$
	$ProtectedCompany(v_1),$
	$SameIndustry(b, v_1)\}$

The query $price(a, high)$ can be rewritten to the corresponding R-clause $\neg price(a, high)\|\square$, and all constrained clauses of this example are given below:

$$r_1: \quad \neg serviceBy(x,y) \vee price(x,high)\|ProtectedCompany(y) \tag{1}$$

$$r_2: \quad \neg serviceBy(x,y) \vee \neg highQuality(y,x) \vee price(x,high)\| \\ ForeignCompany(z) \wedge SameIndustry(y,z) \tag{2}$$

$$f_1: \quad serviceBy(a,b)\|\square \tag{3}$$

$$f_2: \quad highQuality(b,a)\|\square \tag{4}$$

$$Q: \quad \neg price(a,high)\|\square \tag{5}$$

For the model $\mathcal{M}_1$, we can deduce the following constrained refutation.

$$\neg serviceBy(a,y) \vee \neg highQuality(y,a)\|ForeignCompany(z) \\ \wedge SameIndustry(y,z) \tag{6}$$

by (5) and (2) with $\{a/x\}$

$$\neg highQuality(b,a)\|ForeignCompany(z) \wedge SameIndustry(b,z) \tag{7}$$

by (6) and (3) with $\{b/y\}$

$$\square\|ForeignCompany(z) \wedge SameIndustry(b,z) \tag{8}$$

by (7) and (4) with $\{\varnothing\}$, and by mapping variable z to v_1, $\mathcal{M}_1 \models (ForeignCompany(z) \wedge SameIndustry(b,z))$ holds.

For the model $\mathcal{M}_2$, we also have another constrained refutation.

$$\neg serviceBy(a,y)\|ProtectedCompany(y) \tag{9}$$

by (5) and (1) with $\{a/x\}$

$$\square\|ProtectedCompany(b) \tag{10}$$

by (9) and (3) with $\{b/y\}$, and $\mathcal{M}_2 \models ProtectedCompany(b)$ holds.

For every canonical interpretation of the $\mathcal{ALCNR}$ knowledge base Σ_1 we can deduce a constrained empty clause whose constraint is satisfiable in that interpretation. Hence $\mathcal{K}_1 \models price(a, high)$.

4. Decidable algorithm for $\mathcal{ALCNR}$

The fundamental deduction in the DL knowledge base Σ is checking whether Σ is satisfiable. If Σ is satisfiable we can build all models for the knowledge base Σ by an algorithm based on tableaux-like calculus. The algorithm makes use of the notion of *constraint system* [8,9], and begins with an initial constraint system translated from Σ. The initial constraint system represents the set of all models of Σ. Then several *propagation rules* are applied to generate a set of *completions*. A constraint system is *complete* if no propagation rule can apply to it. Each completion is an elaboration of the initial constraint system, in which every implicit constraint has been made explicit. Every completion may contain a *clash* or a clash-free tableau branch. For each clash-free completion that represents a model of Σ, it is always possible to build a canonical interpretation for Σ on the basis of the completion. We rephrase results with minor modification from [2,6].

4.1. Constraint system

An $\mathcal{ALCNR}$-knowledge base Σ is a pair $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is the *intensional* one, called TBox and $\mathcal{A}$ is the *extensional* one, called ABox. We denote the set of *individuals* that appear in Σ by $\mathcal{O}$, and introduce a new alphabet of variable symbols $\mathcal{V}$, with a well-founded total ordering $\prec$ on $\mathcal{V}$. The alphabet $\mathcal{V}$ is disjoint from $\mathcal{O}$. The elements of $\mathcal{V}$ are denoted by the letters u, v, w, x, y, z. The term *object* is an element of $\mathcal{O} \cup \mathcal{V}$. Objects are denoted by the letters s, t, and individuals are denoted by the letters a, b.

A constraint system is a finite nonempty set of constraints of the forms:

$$s : C, \quad s\ P\ t, \quad \forall x.x : C, \quad s \neq t$$

where C is a concept and P is a primitive role name. Given an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, an *$\mathcal{I}$-assignment* α is defined as a function that maps every variable of $\mathcal{V}$ to an element of $\Delta^{\mathcal{I}}$, and every individual a to $a^{\mathcal{I}}$.

A pair $(\mathcal{I}, \alpha)$ *satisfies* the constraint $s: C$ if $\alpha(s) \in C^{\mathcal{I}}$, the constraint $s\ P\ t$ if $(\alpha(s), \alpha(t)) \in P^{\mathcal{I}}$, the constraint $\forall x.\ x: C$ if $C^{\mathcal{I}} = \Delta^{\mathcal{I}}$, and the constraint $s \neq t$ if $\alpha(s) \neq \alpha(t)$. A constraint system $\mathcal{S}$ is *satisfiable* if there is a pair $(\mathcal{I}, \alpha)$ that satisfies every constraint in $\mathcal{S}$.

Let $\mathcal{S}$ be a constraint system and R be a role defined by the description $R = P_1 \sqcap \ldots \sqcap P_k$, $(k \geqslant 1)$. We say that an object t is an *R-successor* of an object s in $\mathcal{S}$ if $s\ P_1 t, \ldots, s\ P_k t$ are in $\mathcal{S}$. We say that t is a *direct successor* of s in $\mathcal{S}$ if t is an R-successor of s for some role R. The *direct predecessor* is the inverse of the direct successor. The *successor* denotes the transitive closure of the direct successor relation, and the *predecessor* denotes its inverse.

We say that s and t are *separated* in $\mathcal{S}$ if $s \neq t$ is in $\mathcal{S}$. We denote by $\mathcal{S}[u/t]$ the constraint system obtained from $\mathcal{S}$ by replacing each occurrence of the variable u by the object t. We assume that variables are introduced in a constraint system according to the ordering $\prec$. If v is introduced to a constraint system $\mathcal{S}$, then $u \prec v$ for all variables u that are already in $\mathcal{S}$. Given a constraint system $\mathcal{S}$ and an object u, we define the function $\sigma(\mathcal{S}, u) := \{C | u : C \in \mathcal{S}\}$. We say that two variables u and v are *concept–equivalent* if $\sigma(\mathcal{S}, u) = \sigma(\mathcal{S}, v)$. Intuitively, two concept–equivalent variables have the same properties, and they may represent the same element in the domain, unless they are separated in $\mathcal{S}$. A constraint system contains a *clash*, if it has one of the following forms:

- $\{s\colon \bot\}$ or
- $\{s\colon C, s\colon \neg C\}$, where C is a concept name, or
- $\{s\colon (\leqslant n\ R)\} \cup \{s\ P_1 t_i, \ldots, s\ P_k\ t_i | i \in 1, \ldots, n+1\} \cup \{t_i \neq t_j | i, j \in 1, \ldots, n+1,\ i \neq j\}$ where $R = P_1 \sqcap \ldots \sqcap P_k$.

4.2. Algorithm description

Given an ARTIGENCE knowledge base $\mathcal{K} = \langle \Sigma, \mathcal{R}, \mathcal{F} \rangle$, its description logic component, the $\mathcal{ALCNR}$ knowledge base $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$, can be translated into the constraint system $\mathcal{S}_\Sigma$ as follows.

(S1) For every concept assertion $C(a)$, put the constraint $a\colon C$ in $\mathcal{S}_\Sigma$.
(S2) For every inclusion $C \sqsubseteq D \in \mathcal{T}$, put the constraint $\forall\ x.\ x\colon \neg C \sqcup D$ in $\mathcal{S}_\Sigma$.
(S3) For every $R(a, b) \in \mathcal{T}$, put the constraints $a\ P_1\ b, \ldots, a\ P_k\ b$ if $R = P_1 \sqcap \ldots \sqcap P_k$, $(k \geqslant 1)$ in $\mathcal{S}_\Sigma$.
(S4) For every pair (a, b) of individuals appearing in $\mathcal{A} \cup \mathcal{F}$, put the constraint $a \neq b$ in $\mathcal{S}_\Sigma$.
(S5) For every concept C that appears in $\mathcal{R}$, put the constraint $\forall x.\ x\colon C \sqcup \neg C$ in $\mathcal{S}_\Sigma$.

The last set of constraints added in [2] is necessary, which forces an object s to belong either to a concept C or its negation for every object s and concept C appearing in $\mathcal{R}$ in every completion that the algorithm generates. It is obvious that Σ is satisfiable iff $\mathcal{S}_\Sigma$ is satisfiable. We assume that all the concepts in a constraint system are *simple*, i.e., the only complements they contain are of the form $\neg C$, where C is a primary concept name. Every $\mathcal{ALCNR}$ concept can be rewritten into equivalent simple concept in linear time [8].

In order to check a knowledge base $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ for satisfiability, the algorithm starts with $\mathcal{S}_\Sigma$, and adds constraints to $\mathcal{S}_\Sigma$ until either a clash is generated or an interpretation satisfying $\mathcal{S}_\Sigma$ can be obtained from the resulting system. Constraints are added on the basis of a suitable set of *propagation rules*. The seven propagation rules are [2]:

(R1) $\mathcal{S} \rightarrow_\sqcap$ $\{s : C_1, s :\ C_2\} \cup \mathcal{S}$
if (1) $s\colon C_1 \sqcap C_2$ is in $\mathcal{S}$,
(2) $s\colon C_1$ and $s\colon C_2$ are not both in $\mathcal{S}$.

(R2) $\mathcal{S} \rightarrow_\sqcup$ $\{s : D\} \cup \mathcal{S}$
if (1) $s\colon C_1 \sqcup C_2$ is in $\mathcal{S}$,
(2) neither $s\colon C_1$ nor $s\colon C_2$ are in $\mathcal{S}$,
(3) $D = C_1$ or $D = C_2$.

(R3) $\mathcal{S} \rightarrow_\forall$ $\{t : C\} \cup \mathcal{S}$
if (1) $s\colon \forall R.C$ is in $\mathcal{S}$,
(2) t is an R-successor of s,
(3) $t\colon C$ is not in $\mathcal{S}$.

(R4) $\mathcal{S} \rightarrow_\exists$ $\{sP_1y, \ldots, sP_ky, y : C\} \cup \mathcal{S}$
if (1) $s\colon \exists R.C$ is in $\mathcal{S}$,
(2) $R = P_1\ \ldots\ P_k$,
(3) y is a new variable,
(4) there is no t such that t is an R-successor of s in $\mathcal{S}$ and $t\colon C$ is in $\mathcal{S}$,
(5) if s is not blocked.

(R5) $\mathcal{S} \rightarrow_\geqslant$ $\{sP_1y_i, \ldots, sP_ky_i | i \in 1, \ldots, n\} \cup \{y_i \neq y_j | i, j \in 1, \ldots, n, i \neq j\} \cup \mathcal{S}$
if (1) $s\colon (\geqslant n\ R)$ is in $\mathcal{S}$,
(2) $R = P_1 \sqcap \ldots \sqcap P_k$,
(3) $y_1, \ldots, y_n$ are new variables,
(4) there do not exist n pairwise separated R-successors of s in $\mathcal{S}$,
(5) if s is not blocked.

(R6) $\mathcal{S} \rightarrow_\leqslant$ $\mathcal{S}[y/t]$
if (1) $s\colon (\leqslant n\ R)$ is in $\mathcal{S}$,
(2) s has more than n R-successors in $\mathcal{S}$,
(3) y, t are two R-successors of s which are not separated.

(R7) $\mathcal{S} \rightarrow_{\forall x}$ $\{s : C\} \cup \mathcal{S}$
if (1) $\forall x.\ x\colon C$ is in $\mathcal{S}$,
(2) s appears in $\mathcal{S}$,
(3) $s\colon C$ is not in $\mathcal{S}$.

We call the rules R2 and R6 *nondeterministic rules*, because they can be applied in at least two different ways to the same constraint system. All the others are said to be *deterministic rules*. The rules R4 and R5 are called *generating rules*, since they add new variables to the constraint system. All the others are called *nongenerating rules*.

A naïve application of the propagation rules may cause infinite chains of application of generating rules and may not terminate. Therefore, the generating rules can be applied only on variables that are not *blocked*, which is related to the goal of keeping the constraint system finite.

If we only consider the variables in a constraint system, it forms a forest of trees whose nodes are the variables and there is an arc from u to v if v is a direct successor of u. Examining the propagation rules reveals that the generating rules introduce new nodes in the forest, and the rule R6 unifies two successors of the same node. All the other rules do not change the forest. The *depth* of a variable in a constraint system is defined to be its depth in the tree to which it belongs. With this structure, the notion of *n-tree equivalence* among variables in a constraint system can be defined as follows.

Definition 4.1. The *n-tree* of a variable v in a constraint system Σ is the tree that consists of the variable v and its successors, whose distance from v is at most n arcs of direct successors. The set of variables in the n-tree of v are denoted by $V_n(v)$.

Two variables $v, u \in \Sigma$ are said to be *n-tree equivalent* if there is an isomorphism $\Psi\colon V_n(v) \rightarrow V_n(u)$ such that

- $\Psi\ (v) = u$,
- for every $s, t \in V_n(v), s\ P\ t \in \mathcal{S}$ iff $\Psi(s)\ P\ \Psi(t) \in \mathcal{S}$, and
- for every $s \in V_n(v), \sigma(\mathcal{S}, \Psi(s)) = \sigma(\mathcal{S}, s)$.

If there is two n-tree equivalent variables, u and v, such that $u \prec v$ then we say that u is a witness of v. The leaves of the n-tree whose root is v will be blocked. We denote by D_R the maximum number of roles with the same name in all empty R-clause of SLD-refutations, which is designed especially to check satisfiability of the restrictions of the empty R-clauses. Given the definition of n-tree equivalent, we can define the notion of a *witness* of a variable.

Definition 4.2. A variable u is a *witness* of a variable v if

- u is D_R-tree equivalent to v,
- v is not in the D_R-tree of u, and
- there is no other variable w, such that $w \prec u$, and w satisfies the first two conditions.

A variable u is said to be *blocked* if u is a leaf of a D_R-tree whose root is v, and v has a witness. Since more than one rule could be applicable to a constraint system $\mathcal{S}$, the propagation rules should be always applied according to the following strategy.

- Apply a rule to a variable only if no rule is applicable to individual.

- Apply a rule to a variable v only if no rule is applicable to a variable u such that $u \prec v$.
- Apply a generating rule only if no nongenerating rule can be applied.

It should be noted that once a generating rule has been applied to a variable v in a constraint system $\mathcal{S}$ according to the strategy above, $\sigma(\cdot, v)$ is stable, i.e., $\sigma(\mathcal{S}', v) = \sigma(\mathcal{S}, v)$, where $\mathcal{S}'$ is any constraint system resulting from applying propagation rules to $\mathcal{S}$. This stability has been proved by Lemma 3.2 in [6]. A variable can be blocked only after an application of a generating rule, and every variable is blocked by a single witness. A constraint system is said to be *complete* when no propagation rule applies to it. A complete system derived from a constraint system $\mathcal{S}$ is called a *completion* of $\mathcal{S}$. Any constraint system containing a clash is evidently unsatisfiable.

Given a clash-free completion $\mathcal{S}$, we define its *canonical interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and the *canonical $\mathcal{I}$-assignment* $\alpha^{\mathcal{I}}$ as follows:

(1) $\Delta^{\mathcal{I}} := \{s | s \text{ is an object in } \mathcal{S}\}$
(2) $\alpha^{\mathcal{I}}(s) := s$.
(3) For a primitive concept $D, s \in D^{\mathcal{I}}$ iff s: D is in $\mathcal{S}$.
(4) $(s,t) \in R^{\mathcal{I}}$ iff
  (a) $s\ R\ t \in \mathcal{S}$, or
  (b) s is blocked, s is a leaf of the D_R-tree whose root is v, w is the witness of v, Ψ is an isomorphism between the D_R-trees rooted with v and w, and $\Psi(s)\ R\ t \in \mathcal{S}$.

If $\mathcal{S}$ is a completion of $\mathcal{S}_\Sigma$ and $\mathcal{S}$ contains no clash, it is always possible to build a model for Σ on the basis of $\mathcal{S}$. The following Theorem 4.1 is proved by Theorem 3.6 in [6].

Theorem 4.1 (*Correctness*). *Let $\mathcal{S}$ be a clash-free complete constraint system, and let $\mathcal{I}$ be its canonical interpretation. Then, $\mathcal{S}$ is satisfiable and $\mathcal{I}$ is a model of $\mathcal{S}$.*

Proof. It follows from the Theorem 3.6 in [6] and the Lemma 3.2 in [2]. □

Example 4.1. Consider the $\mathcal{ALCNR}$ knowledge base $\Sigma_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ from Example 2.1.

The corresponding constraint system $\mathcal{S}_\Sigma$ is:

$\mathcal{S}_\Sigma = \{b : \exists SameIndustry.ForeignCompany \sqcup \exists SameIndustry.DomesticCompany$
$\forall x.x : ForeignCompany \sqcup \neg ForeignCompany$
$\forall x.x : \neg ForeignCompany \sqcup \neg DomesticCompany$
$\forall x.x : \exists SameIndustry.ForeignCompany \sqcup \forall SameIndustry.\neg ForeignCompany$
$\forall x.x : \exists SameIndustry.ForeignCompany \sqcup \neg ForeignCompany$
$a \neq b\}$

Here, we consider 1-tree, namely, $D_R = 1$. An interpretation $\mathcal{I}$ can be constructed as follows which is $\mathcal{I}_1$ in Example 3.2, where instances v_1, v_2, v_3, v_4, are added during the procedure of building the canonical interpretation.

$\Delta^{\mathcal{I}} = \{b, v_1, v_2, v_3, v_4\}$

$ForeignCompany^{\mathcal{I}} = \{b, v_1, v_2, v_3, v_4\}$
$DomesticCompany^{\mathcal{I}} = \{\varnothing\}$
$CompeteWithForeign^{\mathcal{I}} = \{b, v_1, v_2, v_3\}$
$CompeteWithDomestic^{\mathcal{I}} = \{\varnothing\}$
$ProtectedCompany^{\mathcal{I}} = \{\varnothing\}$
$NonMonopoly^{\mathcal{I}} = \{b\}$
$SameIndustry^{\mathcal{I}} = \{(b, v_1), (v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_3)\}$

4.3. Proof of termination and complexity

The algorithm begins with an initial constraint system translated from Σ. Then, the propagation rules are applied to generate a set of completions. Each completion is a refinement of the initial constraint system, in which every implicit constraint has been made explicit. Therefore, in order to prove that the algorithm terminates, it suffices to show that there is a bound on the number of the constraints in completions.

Theorem 4.2 (*Termination and Complexity*). *Let Σ be an $\mathcal{ALCNR}$ knowledge base. Every completion of $\mathcal{S}_\Sigma$ translated from Σ is finite.*

Proof. The number of times that apply the propagation rules with the given strategy to an object in a constraint system is bounded by the size of the TBox $\mathcal{T}$. The number of new variables added by each application of a propagation rule is also bounded by the largest number appearing in the number restrictions in $\mathcal{T}$, i.e., $(\geqslant n\ R)$ and $(\leqslant n\ R)$. We use the following notation:

- Let D_R be the maximum number of roles with the same name in all empty R-clauses.
- Let N be the maximum number appearing in the number restrictions in $\mathcal{T}$.
- Let K be the number of concepts appearing in $\mathcal{S}_\Sigma$.
- Let L be the number of individuals appearing in $\mathcal{A}$.

Let $\mathcal{S}'$ be any constraint system resulting from applying the propagation rules to $\mathcal{S}_\Sigma$. Each constraints $s : C \in \mathcal{S}'$ may contain only concepts of $\mathcal{S}$. Since there are K such concepts, the number of different sets of constraints s: C in $\mathcal{S}'$ is at most 2^K. An application of a propagation rule to a variable x adds no more than N new variables. Then, any D_R-tree rooted with x has at most N^{D_R} new variables. Observe that the number of variables that can become the roots of D_R-trees is bounded by 2^K. Therefore, there are at most $2^K \times N^{DR} + 2^K$ total variables in $\mathcal{S}'$. Since the number of individuals is L, the total number of objects in $\mathcal{S}'$ is at most $2^K \times N^{DR} + 2^K + L$. The number of different constraints of the forms s: C and $\forall x.\ x$: C in which each object s can be involved is also bounded by 2^K. Hence, the total size of these constraints is bounded by

$$2^K \times (2^K \times N^{D_R} + 2^K + L)$$

The number of constraints of the form $s\ P\ t$, $s \neq t$, is bounded by

$$(2^K \times (2^K \times N^{D_R} + 2^K + L))^2$$

To simplify notation, let $M = Max(N,K,L)$. We obtain

$$(2^M \times (2^M \times M^{D_R} + 2^M + M))^2$$

In conclusion, we have that the size of every completion of $\mathcal{S}_\Sigma$ is $O(M^{2D_R} \cdot 2^{4M})$. If $D_R = 0$, it would lead to a bound of $O(2^{4M})$. □

As pointed out in [10], a naïve approach to building a model for description logic knowledge base may expand indefinitely by applying a set of propagation rules to an initial constraint system, and will therefore not terminate. For decidable DLs such as $\mathcal{ALCNR}$ and $\mathcal{SHIQ}$, termination can be ensured without losing completeness because we can transform the infinite tree model of DL knowledge base into a finite structure by setting the appropriate termination conditions, i.e., by defining the blocked variable and the strategy that the generating rules can be applied only on variables that are not blocked. As for function-free Horn rules, reasoning can be done by grounding the rules, i.e., replacing the variables in the rules with the individuals from the knowledge base. Through

grounding, first-order reasoning becomes propositional. For a finite program, the number of grounds is also finite, and satisfiability of a set of Horn rules is decidable. If we want to extend a DL such as $\mathcal{ALCNR}$ with function-free Horn rules, we should ensure that the canonical interpretations constructed from the clash-free completions of DL knowledge base are *enough* for use in the following grounding of a set of Horn clauses. This is ensured by defining D_R to be the maximum number of roles with the same name in all empty R-clauses, which is also one of major differences between the approach in [2] and the approach proposed here.

Theorem 4.3 (*Decidability*). *Given an $\mathcal{ALCNR}$ knowledge base Σ, checking whether Σ is satisfiable and building all the canonical interpretations for Σ if it is satisfiable are decidable problems.*

Proof. This follows from Theorems 4.1 and 4.2. □

Notice that it is necessary to apply the propagation rules based on D_R-tree equivalence of variables, because terminological cycles are allowed in $\mathcal{ALCNR}$ statements. Since the value of D_R depends on the result of SLD-derivations ending with empty R-clauses, we may change the order of step 1 and step 2 of our algorithm without more complexity. Specifically, given an ARTIGENCE knowledge base $\mathcal{K}$, for the set of constrained clauses $\mathcal{R}$, all the SLD-derivations ending with the empty clauses are collected, and the corresponding constraints are obtained at first. Then, let D_R be the maximal number of roles with the same names in these constraints. Secondly, we build all the canonical interpretations for the DL knowledge base Σ by the tableaux-like calculus based on the technique of constraint system, and represent them by Herbrand structures. Finally for every canonical interpretation check if there is at least one SLD-refutation, such that the corresponding constraints are satisfied by that interpretation.

The proof of Theorem 4.2 shows that the number of constraints of each completion is at most doubly exponential in the size of $\mathcal{T}$. Consequently, the time complexity of checking whether the constraints of empty R-clauses are satisfied is also at most doubly exponential in the size of $\mathcal{T}$, which problem we consider is at least as hard as the KB-satisfiability problem studied in [6]. A low bound of the complexity of KB-satisfiability problem is obtained by examining previous results about the language $\mathcal{ALC}$. Since the language $\mathcal{ALC}$ does not allow number restrictions and role conjunction, it is considered to be a sublanguage of $\mathcal{ALCNR}$. We know from [6] that KB-satisfiability in $\mathcal{ACL}$ knowledge bases is EXPTIME-hard. Hence it is hard for $\mathcal{ALCNR}$ knowledge bases too. Notice that we have provided only a worst-case complexity analysis. Therefore, the above conclusion is a coarse upper bound for theoretical purposes. We expect that the actual size to be much smaller than that in practical cases, and the issue of optimization is outside the scope of this paper.

5. Uncertainty reasoning in ARTIGENCE

In Sections 3 and 4, our inference procedures followed the model of reasoning: from correct premises, sound inference rules produce new correct conclusion. However, there are many situations that will not fit this method, and sometimes we may draw useful conclusions from uncertain evidence by using unsound inference rules. Probabilistic analysis is appropriate when it is impossible to know and measure all causes and their interactions well enough to predict consequences. Towards reasoning techniques that allow for probabilistic uncertainty in the Horn rule and ground fact components of ARTIGENCE, we show that the uncertainty reasoning in ARTIGENCE is an instance of a certain type of linear programming model based on probabilistic logic. In uncertain situation, for each model of the DL it is not sufficient to check if there is just one constrained SLD-refutation whose constraints are satisfied by that model, and we should consider every constrained SLD-refutation, such that its constraints are satisfied by the model instead, because each of them may yield a different range of confidence levels for a query.

5.1. Probabilistic logic

Several logics for reasoning under uncertainty can be viewed as probability mass distribution problems, and the solution of this distribution problem obtains a range of confidence level for the conclusion. In probabilistic logic, probabilities are assigned to propositions to indicate levels of confidence. The goal is to calculate the degree of confidence one can have in a conclusion derived from these propositions. In other words, the inference problem is to determine how much confidence we can place in the conclusion whose probability mass is constrained by a set of interrelated propositions for which masses are given. It has been shown that inference in probabilistic logic can be formulated as a linear programming problem, typically with exponentially many variables [11].

In probabilistic logic, we are given a set of propositions, $F_1, \ldots, F_h$, and the degrees of confidence we have in them. The confidence degree for F_i ($i = 1,\ldots,h$) is indicated by its probabilistic mass, which is an interval $[l_i, u_i] \in [0,1]$, and $l_i \leqslant u_i$. Let Pr be a function that maps each *possible world* $w \in W$ into a real number in the interval [0, 1] as follows.

$$Pr: \ W \to [0,1], \ \text{and} \ \sum_{w \in W} Pr(w) = 1, \ 0 \leqslant Pr(w) \leqslant 1$$

Let S_i be the set of possible worlds that make proposition F_i true, and let $\mu(S_i)$ be the mass of S_i, which denotes the sum of probabilistic masses that the function Pr can assign to the possible worlds in which F_i is true. The sets $S_1,\ldots,S_h$ need not all be distinct.

The propositions, $F_1,\ldots,F_h$ contain a set of atomic propositions $x_1,\ldots,x_n$, and a possible world is an assignment $w{:}\{x_1,\ldots,x_n\} \to \{0,1\}^n$ of truth values to the atomic propositions. The proposition F_i is true in a possible world w when the assignment w makes it true, which is denoted by $w \models F_i$. Then, the mass of S_i under a function Pr is

$$\mu(S_i) = \sum_{w \in Si} Pr(w), S_i = \{w | w \models F_i\}$$

A function Pr is an *interpretation* of a set of probabilistic logic propositions F_i ($i = 1,\ldots,h$) if $\mu(S_i) \in [l_i,u_i]$ for every F_i. The fundamental problem is to determine how much confidence we can have in a new proposition F_q that is logical consequence of $F_1,\ldots,F_h$ whose probabilistic masses are given. Note that the probability of all possible worlds must sum to one, and we can place bounds on the mass of S_q by solving the following two optimization problems.

$$\begin{aligned}
&\text{minimize/maximize } \mu(S_q) \text{ subject to} \\
&\mu(S_i) \leqslant u_i \quad i = 1,\ldots,h \\
&\mu(S_i) \geqslant l_i \quad i = 1,\ldots,h \\
&\sum_{w \in W} Pr(w) = 1, \ 0 \leqslant Pr(w) \leqslant 1
\end{aligned}$$

5.2. Reasoning under uncertainty in ARTIGENCE

Probabilistic logic requires that the formulas must appear in the form of propositions, i.e., there is no variable in the formulas of probabilistic logic. As we all know that Horn clauses can be instantiated from the resolution refutation process, and the sequence of substitutions (unifications) used to make predicates equivalent gives us the value of variables in the clauses. Hence, retaining information on the unification substitutions made in the resolution refutation give information for the instantiation, which replaces

each clause by its instance. An instance of a clause can be regarded as a proposition. Note that an instantiation might contain more than one instance of the same clause. Now we are prepared to present the global algorithm for reasoning under uncertainty in ARTIGENCE, as shown in Algorithm 5.1.

Algorithm 5.1. Global algorithm for reasoning under uncertainty in ARTIGENCE

Input: An ARTIGENCE knowledge base $\mathcal{K} = \langle \Sigma, \mathcal{P}, \mathcal{F} \rangle$, where $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ is an $\mathcal{ALCNR}$ knowledge base, and Horn rules and ground facts are given with their confidence levels;
A query Q to the knowledge base $\mathcal{K}$.
Output: A ground instance of Q with confidence level.
Begin.
1. Collect all the SLD-derivations ending with the empty clauses and the corresponding constraints for a set of constrained clauses $\mathcal{R}$ and ground facts $\mathcal{F}$.
2. Build all the canonical interpretations for the description logic knowledge base Σ, and represent them by Herbrand structures.
3. **if** for each canonical interpretation of Σ there is at least one SLD-refutation whose constraints are satisfied by that interpretation. **then**
4. **for** each canonical interpretation of Σ **do**
5. **for** each SLD-refutation whose constraints are satisfied by the interpretation **do**
6. Construct and solve the corresponding maximization and minimization linear programming problems to obtain an interval $[l, u]$ for the query Q.
7. **end for**
8. **end for**
9. **if** $maximum(l) \leqslant minimum(u)$ **then**
10. **return** $[maximum(l), minimum(u)]$
11. **else** the query Q is not logical consequence of $\mathcal{K}$.
End.

For each interpretation of Σ, we need to build a linear programming model for every SLD-refutation whose constraints are satisfied by that interpretation, because each of them may have a different set of possible worlds and a different function Pr that assigns each possible world in the set with a probabilistic mass. In other words, for each SLD-refutation under a model of Σ, a query Q may be logically related to different sets of clauses for which masses are given. How much confidence we can have in the query Q is constrained by the fact that S_q intersects those sets of possible worlds that make the instances of clauses true and these clauses logically entail the query Q, where S_q is the set of possible worlds that make the query Q true.

Example 5.1. Consider the ARTIGENCE knowledge base $\mathcal{K}_1 = \langle \Sigma_1, \mathcal{R}_1, \mathcal{F}_1 \rangle$ from Example 3.2, and here $\mathcal{R}_1$ and $\mathcal{F}_1$ are given with confidence levels as follows.

$$\begin{aligned}\mathcal{R}_1 = \{r_1 :\ & serviceBy(x,y) \wedge ProtectedCompany(y) \rightarrow price(x, high)\\ r_2 :\ & serviceBy(x,y) \wedge SameIndustry(y,z) \wedge ForeignCompany(z)\\ & \wedge highQuality(y,x) \rightarrow price(x, high)\\ & Pr(r_1) \in [0.75, 0.90], Pr(r_2) \in [0.60, 0.80]\}\\ \mathcal{F}_1 = \{f_1 :\ & serviceBy(a,b), f_2 : highQuality(b,a)\\ & Pr(f_1) = 0.95, Pr(f_2) \in [0.70, 0.85]\}\end{aligned}$$

How much confidence we can have in the query *price*(*a*, *high*)?

Continue the Example 3.2 but in uncertain situation. For the model $\mathcal{M}_1 = (H_1, I_1)$, we can have the following propositions of probabilistic logic.

$$\begin{aligned}&Pr(serviceBy(a,b)) = 0.95\\ &Pr(highQuality(b,a)) \in [0.70, 0.85]\\ &Pr(serviceBy(a,b) \wedge highQuality(b,a) \rightarrow price(a, high)) \in [0.60, 0.80]\end{aligned}$$

Since every concept or role assertion in the model of Σ is considered to be definitely true, we can remove them from the propositions without affecting the result. Recall that as quantifier restrictions the descript logic component has already filtered out the values that its interpretation can assign to the variables of Horn clauses. We will discuss how some probabilistic versions of description logics can be accommodated to our framework later.

Let (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8) denote the probabilities of the eight possible worlds (*serviceBy*(*a*, *b*), *highQuality*(*b*, *a*), *price*(*a*, *high*)) as shown in Table 1. We can place bounds on the mass of the set of possible worlds that make the query *price*(*a*, *high*) true by solving the two optimization problems.

$$\text{minimize/maximize} \quad p_2 + p_4 + p_6 + p_8 \quad \text{subject to}$$

$$\begin{Bmatrix} 0&0&0&0&1&1&1&1\\ 0&0&1&1&0&0&1&1\\ 1&1&1&1&1&1&0&1\\ 0&0&1&1&0&0&1&1\\ 1&1&1&1&1&1&0&1\\ 1&1&1&1&1&1&1&1 \end{Bmatrix} \begin{Bmatrix} p_1\\ p_2\\ p_3\\ p_4\\ p_5\\ p_6\\ p_7\\ p_8 \end{Bmatrix} \begin{Bmatrix} =\\ \leqslant\\ \leqslant\\ \geqslant\\ \geqslant\\ =\end{Bmatrix} \begin{Bmatrix} 0.95\\ 0.85\\ 0.80\\ 0.70\\ 0.60\\ 1.00 \end{Bmatrix}$$

$$p_i \geqslant 0, \quad i = 1, \ldots, 8.$$

When finding the minimum value of the objective function, we have $\{p_1 = 0,\ p_2 = 0,\ p_3 = 0.05,\ p_4 = 0,\ p_5 = 0.3,\ p_6 = 0,\ p_7 = 0.4,\ p_8 = 0.25\}$, and $p_2 + p_4 + p_6 + p_8 = 0.25$. When we want to maximize the mass of the set of possible worlds in which the query *price*(*a*, *high*) is true, we obtain $\{p_1 = 0,\ p_2 = 0.0125,\ p_3 = 0,\ p_4 = 0.0375,\ p_5 = 0,\ p_6 = 0.2875,\ p_7 = 0.2,\ p_8 = 0.4625\}$, and $p_2 + p_4 + p_6 + p_8 = 0.8$. Hence, the confidence level we can have in the query *price*(*a*, *high*) under the model $\mathcal{M}_1$ is an interval [0.25, 0.8].

Similarly, for the model $\mathcal{M}_2 = (H_2, I_2)$ we have the following set of propositions.

$$\begin{aligned}&Pr\ (serviceBy(a,b)) = 0.95\\ &Pr\ (serviceBy(a,b) \rightarrow price(a, high)) \in [0.75, 0.90]\end{aligned}$$

Let (p_1, p_2, p_3, p_4) be the probabilities of the four possible worlds (*serviceBy*(*a*, *b*), *price*(*a*, *high*)) as shown in Table 2.

Now let us write the linear programming for the model $\mathcal{M}_2$.

Table 1
Truth table for the model $\mathcal{M}_1$ in Example 5.1.

Probability	*serviceBy*(*a*,*b*)	*highQuality*(*b*,*a*)	*price*(*a*,*high*)	r_2
p_1	0	0	0	1
p_2	0	0	1	1
p_3	0	1	0	1
p_4	0	1	1	1
p_5	1	0	0	1
p_6	1	0	1	1
p_7	1	1	0	0
p_8	1	1	1	1

Table 2
Truth table for the model $\mathcal{M}_2$ in Example 5.1.

Probability	*serviceBy*(*a*,*b*)	*price*(*a*,*high*)	r_1
p_1	0	0	1
p_2	0	1	1
p_3	1	0	0
p_4	1	1	1

$$\text{minimize/maximize } p_2 + p_4 \quad \text{subject to}$$
$$\begin{Bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{Bmatrix} \begin{Bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{Bmatrix} \begin{Bmatrix} = \\ \leqslant \\ \geqslant \\ = \end{Bmatrix} \begin{Bmatrix} 0.95 \\ 0.90 \\ 0.75 \\ 1.00 \end{Bmatrix}$$
$$p_i \geqslant 0, \quad i = 1, \ldots, 8.$$

When we want to minimize the objective function, we have $\{p_1 = 0.05,\ p_2 = 0,\ p_3 = 0.25,\ p_4 = 0.7\}$, and $p_2 + p_4 = 0.7$. Otherwise, we obtain $\{p_1 = 0,\ p_2 = 0.05,\ p_3 = 0.1,\ p_4 = 0.85\}$, and $p_2 + p_4 = 0.9$. Thus, the confidence level of the query *price*(*a*, *high*) under the model $\mathcal{M}_2$ is an interval [0.7,0.9].

We say $[l^*, u^*]$ is a confidence level of a query *Q*, if l^* is maximum value of the left bounds of all resulting intervals, u^* is minimum value of the right bounds of all resulting intervals to all interpretations *Pr* of $\mathcal{K}$, and l^* is less than or equal to u^*. In this example, $l^* = Max(0.25, 0.7) = 0.7$, $u^* = Min(0.8, 0.9) = 0.8$, and $0.7 \leqslant 0.8$. Therefore, the confidence level we can have in the query *price*(*a*, *high*) is [0.7,0.8], denoted by $\mathcal{K}_1 \models price(a, high)[0.7, 0.8]$.

Now, the procedure of our algorithm has been described in detail, we are prepared to prove the correctness, completeness and decidability of our global algorithm.

Proposition 5.1 (*Soundness and Completeness*). *Let Q be a query to an ARTIGENCE knowledge base* $\mathcal{K}$. *Then* $\mathcal{K} \models Q[l^*, u^*]$ *iff for every canonical interpretation of* Σ *these is at least one SLD-refutation whose constraints are satisfied by that interpretation; and for every SLD-refutation under each interpretation of* Σ *there exists an interpretation Pr, such that the confidence level of Q is an interval [l, u] under the Pr; and* l^* *(respectively,* u^**) is maximum (respectively, minimum) value of l (respectively, u) of all such intervals, and* $l^* \leqslant u^*$.

Proof. The claim follows from Theorem 3.5 without considering uncertainty. Here, we focus our attention on the part of uncertainty reasoning.

Soundness ($\Leftarrow$): By the precondition, any number that lies between l^* and u^* is also included in any range of the confidence levels we can have in the query *Q* to all interpretations *Pr* computed for every SLD-refutation under each interpretation of Σ. In other words, that we can trust in *Q* with the confidence level $[l^*, u^*]$ is supported by every interpretation *Pr* of $\mathcal{K}$.

Completeness ($\Rightarrow$): Suppose the probability mass of *Q* is included in an interval $[0, l^*)$. By the definition of l^*, l^* is maximum value of the left bounds of all interpretations *Pr* of $\mathcal{K}$. Then, there must be an interpretation *Pr*, such that the confidence level of *Q* under that interpretation does not intersect the interval $[0, l^*)$. This is a contradiction. We have the same conclusion for an interval $(u^*, 1]$. □

Theorem 5.2 (*Decidability*). *Query answering under uncertainty in ARTIGENCE is decidable.*

Proof. It follows from Theorems 3.6 and 4.3 and the fact that linear programming problems are solvable in exponential time. □

For some applications one may wish to express uncertainty in the component of description logic, although we do not pursue this possibility in detail here. But we point out that some probabilistic versions of description logics such as [12,13] can be easily accommodated to our framework without affecting the complexity of reasoning. For example, in [12] a concept assertion $C(a)$ of ABox is generalized by a probabilistic assertion $Pr(C(a)) \in [l,u]$ to express uncertain knowledge. In these cases, probabilistic concept and role assertions in the canonical interpretation can be imposed by adding the corresponding condition equations to linear programming models as ground facts with confidence levels, and other steps of our global algorithm remain unchanged.

In probabilistic logic, we assume that the confidence levels of formulas F_i are provided from a single *evidence source*. However, it is possible to allow for multiple evidence sources to supply probabilities (or estimates of these) to the formulas in question [11]. It is useful to extend the ordinary model of probabilistic logic, especially for inherent open and dynamic Web.

Suppose we have *k* evidence sources, denoted by E_j ($j = 1,\ldots,k$). If $k \geqslant 2$, we get conditional probabilities $Pr(F_i|E_j)$, $i = 1,\ldots,h$. The interpretation of these probabilities is: the probability of F_i given that evidence source *j* is reliable. For each of F_i, there are *k* probabilities, each of which is obtained from one of the *k* evidence sources. Let S_i be the set of possible worlds that make proposition F_i true, and R_j the set of possible worlds in which evidence source *j* is reliable. If evidence source *j* delivers information that the probability of F_i is in the interval $[l_i^j, u_i^j]$, this is equivalent to the set of condition equations:

$$l_i^j \leqslant Pr(F_i|E_j) \leqslant u_i^j, \quad i = 1,\ldots,h,\ j = 1,\ldots,k$$

The above condition equations can be rewritten to

$$0 \leqslant Pr(S_i \cap R_j) - l_i^j \times Pr(R_j), \quad i = 1,\ldots,h,\ j = 1,\ldots,k$$
$$Pr(S_i \cap R_j) - u_i^j \times Pr(R_j) \leqslant 0, \quad i = 1,\ldots,h,\ j = 1,\ldots,k$$

It is also possible to specify probability intervals $[l^j, u^j]$ for each evidence source. This gives rise to the set of condition equations:

$$l^j \leqslant Pr(E_j) \leqslant u^j, j = 1,\ldots,k$$

Then, the extended linear programming model is:

$$\text{minimize/maximize} \quad \mu(S_q) \quad \text{subject to}$$
$$0 \leqslant Pr(S_i \cap R_j) - l_i^j \times Pr(R_j), \quad i = 1,\ldots,h,\ j = 1,\ldots,k$$
$$Pr(S_i \cap R_j) - u_i^j \times Pr(R_j) \leqslant 0, \quad i = 1,\ldots,h,\ j = 1,\ldots,k$$
$$l^j \leqslant Pr(E_j) \leqslant u^j, \quad j = 1,\ldots,k$$
$$\sum_{w \in W} Pr(w) = 1, \quad 0 \leqslant Pr(w) \leqslant 1$$

6. Related to previous work

Related work on the combination of Horn rules and description logics with uncertainty can be divided into (a) hybrid systems using description logics as input to logic programs; (b) approaches reducing description logics reasoning to logic programming; (c) probabilistic extension of description logics. Below we review some representatives for these three types of related work.

Some hybrid systems using description logics as input to logic programs are works by [2,10,14,15]. Donini et al. present an integrated system, called $\mathcal{AL}$-log, based on description logic $\mathcal{ALC}$ and a set of constrained Datalog clauses, each of them is variants of Horn rules [14]. The interaction between the two components is realized by allowing the specification of constraints in Datalog clauses, where constraints are expressed using $\mathcal{ALC}$. Constraints on variables require them to range over the set of instances of a

specified concept, where constraints on individual objects require them to belong to a concept. $\mathcal{AL}$-log allows recursive clauses, but a weaker description logic $\mathcal{ALC}$, and only unary predicates from the description logic are allowed in the constrained Datalog clauses. We treat a more expressive logic and other decidable description logics can be easily accommodated to our framework.

Levy and Rousset present CARIN that extends Horn rules with the expressive power of the description logic $\mathcal{ALCNR}$, which is most closely related to ARTIGENCE [2]. CARIN combines the two formalisms by allowing the concepts and roles, defined in the description logic, to appear as predicates in the antecedents of Horn rules. In contrast to $\mathcal{AL}$-log, CARIN does not require the safety condition that a variable appearing in a concept atom had to appear in an atom of ordinary predicate in the body of a rule. It has been shown that such hybrid system is not straightforward, and that some restrictions must be imposed in order to retain decidability of the reasoning services. We provide a proof of decidability and give a decision procedure combining tableau calculus with constrained resolution.

Motic et al. present an approach for extending OWL-DL with function-free Horn rules which yields a logic with decidable reasoning algorithms in [10]. However, the rules are required to be DL-safe: each variable in the rule is required to occur in a non-DL-atom in the rule body. As a consequence, rules apply only to individuals explicitly introduced in the ABox. In applications requiring intensional reasoning such as natural language processing, DL-safety is a severe restriction, as many conclusions drawn involve unnamed objects. The concept and role atoms must not appear in the consequents of the Horn rules in our approach, but the rules are not required to be DL-safe, and we can deal with uncertainty. Hence when compared to [10], our approach is slightly general in some, and slightly more general in other aspects.

Lukasiewicz describes probabilistic logic programs that combine with description logic under the answer set semantics and the well-founded semantics with Poole's independent choice logic [16]. It has been shown that the query processing in such probabilistic logic programs can be reduce to computing all answer sets of logic programs, solving linear optimization problems, and to computing the well-founded model of logic programs, respectively [15]. Lukasiewicz also allows concept and role constraints from description logic in the rules of logic programs, but a different kind of description logic $\mathcal{SHIF}(\mathbf{D})$. $\mathcal{SHIF}(\mathbf{D})$ is somewhat less expressive than indicated by its name since the use of roles in number restrictions is restricted: roles that have a transitive subrole must not occur in number restrictions.

Description logic programs language (DLP) proposed in [1] based on the expressive intersection of description logics with Horn rules as early representative of reducing description logic reasoning to logic programming. Actually, description logics and Horn rules are strict (decidable) subsets of first-order logic. Antoniou and Wagner present defeasible extension to the description logic programs by introducing a superiority relation among the rules [17]. Nottelmann and Fuhr propose pDAML + OIL that is a probabilistic generalization of the description logic programs by mapping a part of $\mathcal{SHOIQ}(\mathbf{D})$ onto probabilistic Horn logics [18]. The results are obviously decidable languages, but those are necessarily less expressive than either the description logics or Horn rules from which are formed. It seems that such languages are insufficient for the modeling of finer details of real applications.

The works in [12,13,19] are representatives of probabilistic extension of description logics. Jaeger provides a probabilistic extension of the description logic $\mathcal{ALC}$ by introducing statements about conditional probabilities between concepts and statements to express uncertain knowledge about a specific object [12]. The focus was on completing partial statistical information from a small set of probabilistic statements by cross-entropy minimization. Koller et al. present P-CLASSIC, a probabilistic version of the description logic CLASSIC, in order to express the degree of overlap between concepts, which is based on Bayesian networks [19]. The main reasoning problem is to determine the exact probabilities for conditionals between concept expressions. However, probabilistic knowledge about ground facts (i.e., Abox) is not allowed in the knowledge base of P-CLASSIC. Giugno and Lukasiewicz propose P-$\mathcal{SHOQ}(\mathbf{D})$, another probabilistic extension of description logic $\mathcal{SHOQ}(\mathbf{D})$ [13]. P-$\mathcal{SHOQ}(\mathbf{D})$ allows to express probabilistic knowledge about concepts and instances by utilizing the notion of probabilistic lexicographic entailment from probabilistic default reasoning [20,21]. A fuzzy extension of $\mathcal{ALC}$, combining Zadeh's fuzzy logic with a classical description logic, is described in [22], which is less closely related, as fuzzy uncertainty deals with vagueness, rather than likelihood. Pan et al. proposed f-SWRL, a fuzzy extension to SWRL (Semantic Web Rule Language) to include fuzzy assertions and rules in [23]. Lukasiewicz and Straccia give an overview of approach to managing uncertainty and vagueness in description logics [24].

Several other works also have discussed the integration of Horn rules and description logics. Their reasoning procedure was modified either by taking extra steps to consider the resolutions sanctioned by the description logic component or by modifying the unification substitutions underlying the reasoning engine. These approaches are either incomplete or some restriction must be imposed in order to retain decidability. ARTIGENCE differs from previous works in several ways: it has a sound, complete, and decidable inference procedure; it not only combines the expressive power of Horn rules and description logics, but also can deal with uncertainty; $\mathcal{ALCNR}$ is considered as the description logic component of ARTIGENCE, which is one of the most expressive description logics with decidable inference procedures. Other decidable description logics, even their probabilistic versions can be easily accommodated to our framework. Some applications of such hybrid systems can be found in [25,26].

7. Conclusions

We have described ARTIGENCE, a representation language that combines description logics and Horn rules with uncertainty. We show that query answering in ARTIGENCE can be reduced to for every model of DL knowledge base checking if there is at least one constrained SLD-refutation whose constraints are satisfiable in that model, and for every SLD-refutation under each model of DL knowledge base solving two linear programming problems to obtain an interval of confidence level of the query, and taking the intersection of all such intervals as the resulting confidence level we can have in the query. As a result, we obtained a sound, complete, and decidable algorithm for reasoning in ARTIGENCE knowledge base. We also discussed several possible extensions both to the underlying description logic and to the probabilistic Horn rule component.

It is worth noting that our framework allows a family of description logics not necessarily $\mathcal{ALCNR}$ but any other description logics, even their probabilistic versions if they have decidable algorithm to build all the canonical interpretations for their knowledge bases. The main goal of our work is to combine the expressive power of probabilistic Horn rules and description logics. In fact, there is still lack of some useful features, such as integration with relational model, and the ability to perform nonmonotonic reasoning in the component of description logic. Description logics naturally capture the way in which people encode their knowledge by defining basic concepts, their properties, and the relations between them. Unfortunately, it is severely limited in its ability to represent defeasible inference. Nonmonotonic systems are important in

practice since they can model phenomena like exceptions and priorities naturally in declarative way. So it makes sense to study how description logics are equipped with nonmonotonic reasoning to deal with inconsistencies, and how these nonmonotonic extensions of description logics fit into our framework.

ARTIGENCE can be used in at least two contexts. Description logics are orthogonal to Horn rules: none of proper subset of the other. Applications that utilize our framework can significantly benefit from combing the expressive power of both formalisms, and improve on deductive power and representational adequacy. More importantly, reasoning in ARTIGENCE is decidable as well as sound and complete, and it can deal with uncertainty, which enable us to obtain query answers that were not possible before this framework. Another possible use of ARTIGENCE for Semantic Web is mentioned in the introduction. Although OWL adds more vocabulary for describing properties and classes, and enhances considerable expressive power to the Semantic Web, it still has expressive limitations, particularly with respect to what can be said about properties. A feasible way to overcome some of expressive restrictions of OWL would be to extend it with Horn rules. ARTIGENCE is such representation language that can combines information from multiple evidence sources with its ability in uncertainty reasoning, which follows the philosophy of the Semantic Web that anybody can produce information or utilize anyone else's information on open environment full of dynamic and uncertainty.

Although the focus of this paper is on the question of decidability and soundness of the reasoning problem in ARTIGENCE, our work raises the critical performance issue of how to efficiently reason in the system. The inference procedure of ARTIGENCE requires that for every model of DL knowledge base, we need to deduce all (at least one) constrained empty clauses whose constraints are satisfiable in that model. One of the possible optimizations is to replace the traditional "blind" search in automated deduction by more directed search or still better by deterministic resolution under the guidance of the semantic information from the constraint model. A second direction is to reduce the size and number of the completions in order to obtain sufficient but least models for DL knowledge base. In our context, we attempt to stop applying the propagation rules to the tableau branch that is proved to be equivalent to a completion already created before, or unite two or more equivalent tableau branches into one for a specific query as early as possible. An implementation with practically efficient method for reasoning is planned, and we are currently looking into applying ARTIGENCE as a representational and reasoning framework for multi-agent applications and information integration.

Acknowledgements

I am grateful to Hans-Jürgen Bürckert for discussion on the decidability of constrained resolution. The work was supported by a grant from the National Natural Science Foundation of China (No. 60903078) and a grant from Fudan Research Fund for Young Scholars.

References

[1] B.N. Grosof, I. Horrocks, R. Volz, S. Decker, Description logic programs: combining logic programs with description logic, in: Proc. 13th Int. World Wide Web, 2003, pp. 48–57.
[2] A.Y. Levy, M.-C. Rousset, Combining Horn rules and description logics in CARIN, Artificial Intelligence 104 (1998) 165–209.
[3] I. Horrocks, P.F. Patel-Schneider, F.V. Harmelen, From $\mathcal{SHIQ}$ and RDF to OWL: the making of a Web ontology language, Journal of Web Semantics 1 (1) (2003) 7–26.
[4] I. Horrocks, P.F. Patel-Schneider, A proposal for an OWL rules language, in: Proc. 14th Int. World Wide Web, 2004, pp. 723–731.
[5] H-J. Bürckert, A resolution principle for constrained logics, Artificial Intelligence 66 (1994) 235–271.
[6] M. Buchheit, F.M. Donini, A. Schaerf, Decidable reasoning in terminological knowledge representation systems, Journal of Artificial Intelligence Research 1 (1993) 109–138.
[7] S. Ceri, G. Gottlob, L. Tanca, Logic Programming and Databases, Springer-Verlag, Berlin, 1990.
[8] F.M. Donini, M. Lenzerini, D. Nardi, W. Nutt, The complexity of concept languages, in: Proc. 2nd Int. Principles of Knowledge Representation and Reasoning, 1991, pp. 151–162.
[9] M. Schmidt-Schauβ, G. Smolka, Attributive concept descriptions with complements, Artificial Intelligence 48 (1) (1991) 1–26.
[10] B. Motik, U. Sattler, R. Studer, Query answering for OWL-DL with rules, in: Proc. 5th Int. International Semantic Web Conference, 2004, pp. 549–563.
[11] K.A. Andersen, J.N. Hooker, A linear programming framework for logics of uncertainty, Decision Support Systems 16 (1996) 39–53.
[12] M. Jaeger, Probabilistic reasoning in terminological logics, in: Proc. 4th Int. Principles of Knowledge Representation and Reasoning, 1994, pp. 305–316.
[13] R. Giugno, T. Lukasiewicz, P-$\mathcal{SHOQ}$(**D**): a probabilistic extension of $\mathcal{SHOQ}$(**D**) for probabilistic ontologies in the Semantic Web, in: Proc. 9th European Conference on Logics in Artificial Intelligence, 2002, pp. 86–97.
[14] F.M. Donini, M. Lenzerini, D. Nardi, A. Schaerf, $\mathcal{AL}$-log: integrating datalog and description logics, Journal of Intelligent Information Systems 10 (1998) 227–252.
[15] T. Lukasiewicz, Probabilistic description logic programs, International Journal of Approximate Reasoning 45 (2) (2006) 288–307.
[16] D. Poole, The independent choice logic for modeling multiple agents under uncertainty, Artificial Intelligence 94 (1–2) (1997) 7–56.
[17] G. Antoniou, G. Wagner, Rules and defeasible reasoning on the Semantic Web, in: Proc. 2nd Int. Rules and Rule Markup Languages for the Semantic Web, 2003.
[18] H. Nottelmann, N. Fuhr, pDAML+OIL: a probabilistic extension to DAML + OIL based on probabilistic Datalog, in: Proc. 10th Int. Information Processing and Management of Uncertainty in Knowledge-based Systems, 2004.
[19] D. Koller, A. Levy, A. Pfeffer, P-CLASSIC: a tractable probabilistic description logic, in: Proc. 4th National Conference on Artificial Intelligence, 1997, pp. 390–397.
[20] T. Lukasiewicz, Probabilistic logic programming under inheritance with overriding, in: Proc. 17th Uncertainty in Artificial Intelligence, 2001, pp. 329–336.
[21] T. Lukasiewicz, Probabilistic default reasoning with conditional constraints, Annals of Mathematics and Artificial Intelligence 34 (1–3) (2002) 35–88.
[22] U. Straccia, Reasoning within fuzzy description logics, Journal of Artificial Intelligence Research 14 (2001) 137–166.
[23] J.Z. Pan, G. Stoilos, G.B. Stamou, V. Tzouvaras, I. Horrocks, f-SWRL: a fuzzy extension of SWRL, Journal of Data Semantics 6 (2006) 28–46.
[24] T. Lukasiewicz, U. Straccia, Managing uncertainty and vagueness in description logics for the Semantic Web, Journal of Web Semantics 6 (4) (2008) 291–308.
[25] Q.L. Guo, M. Zhang, Question answering based on pervasive agent ontology and Semantic Web, Knowledge-Based Systems 22 (6) (2009) 443–448.
[26] P.F. Liu, B. Raahemi, M. Benyoucef, Knowledge sharing in dynamic virtual enterprises: a socio-technological perspective, Knowledge-Based Systems 24 (3) (2011) 427–443.