$\mathcal{ALC}^u_{\mathbb{P}}$: An Integration of Description Logic and General Rules

Jing Mei[1], Zuoquan Lin[1], Harold Boley[2]

[1] Department of Information Science
Peking University, Beijing 100871, China
{mayyam, lz} AT is.pku.edu.cn
[2] Institute for Information Technology - e-Business
National Research Council of Canada
Fredericton, NB, E3B 9W4, Canada
{Harold.Boley} AT nrc.gc.ca

Abstract. A unifying logic is built on top of ontologies and rules for the revised Semantic Web Architecture. This paper proposes $\mathcal{ALC}^u_{\mathbb{P}}$, which integrates a description logic (DL) that makes a *u*nique names assumption with general rules that have the form of Datalog $\mathbb{P}$rograms permitting default negation in the body. An $\mathcal{ALC}^u_{\mathbb{P}}$ knowledge base (KB) consists of a TBox $\mathcal{T}$ of subsumptions, an ABox $\mathcal{A}$ of assertions, and a novel PBox $\mathbb{P}$ of general rules that share predicates with DL concepts and DL roles. To model open answer set semantics, extended Herbrand structures are used for interpreting DL concepts and DL roles, while open answer sets hold for general rules. To retain decidability, a well-known weak safeness condition is employed. We develop DL tableaux-based algorithms for decision procedures of the KB satisfiability and the query entailment problems.

1 Introduction

Based on input from the Semantic Web Rules community, the Semantic Web Architecture has been recently reconsidered by Tim Berners-Lee [3]: ontologies and rules are now sitting side by side between RDF(S) and a unifying logic layer. The Web Ontology Language (OWL), whose formalization relies directly on Description Logic (DL) [2], dates back to a W3C Recommendation released on 10 February 2004 [23]. Subsequently, W3C announced the formation of the Rule Interchange Format (RIF [24]) Working Group on 7 November 2005, aiming to specify a format for rules in the Semantic Web chartered to allow "knowledge expressed in OWL and in rules to be easily used together". Not surprisingly, how to best combine OWL/DL and rules has become a topic of heated discussions in the Semantic Web community.

At the top-level, those integration approaches are either homogeneous or hybrid [1]. Early work in the *hybrid* direction comprises AL-log [7] and CARIN [16], both of which extend Datalog rules with DL constraints. Recent *homogeneous* work prefers a more general and tight integration, such as $\mathcal{DL}+log$ [22]

(originating from r-hybrid KBs [21]) and HEX-programs [8] (originating from dl-programs [9]). Their generality appeals to negation and disjunction in rules, namely Datalog$^{\neg,\vee}$, while their tightness calls for certain *safeness* conditions that limit the interaction between the DL component and rules.

In hybrid approaches, ontology and rule predicates are always kept distinct. Homogeneous frameworks instead permit predicate sharing in a syntactically and semantically coherent manner. DLP (Description Logic Programs [11]) has been proposed as the intersection of DL and Datalog rules, while SWRL (Semantic Web Rule Language [25]) is their union. Unfortunately, DLP seems too restrictive, and SWRL seems too expressive (hence is undecidable).

Reduction is another way to build a homogeneous platform. The paper [15] works on reducing DL KBs to disjunctive Datalog programs, getting ready for an extension with DL-safe rules [19] and even MKNF rules [18]. There, DL-safeness is imposed as the condition for grounding rule variables with named individuals.

Summarizing, from a practical perspective, hybrid approaches are component-based, using plug-ins of both DL reasoners and rule engines, in addition to well-defined interfaces. Most homogeneous approaches make use of translators from the DL component into rules (even first-order formulae), followed by running rule engines (even first-order provers) with support for those reduced languages. We totally agree that reusing existing reasoning tools (e.g., DL reasoners and rule engines) facilitates various applications on the Semantic Web. But, towards a unifying logic on top of ontologies and rules, as envisioned by [3], it makes sense to develop a novel algorithm specifically for the homogeneous integration of DL and general rules.

This paper extends a DL KB – consisting of a TBox $\mathcal{T}$ of subsumptions and an ABox $\mathcal{A}$ of assertions – with a PBox $\mathbb{P}$ of general rules, i.e., Datalog$^{\neg}$ rules permitting default negation for atoms in the body, in a homogeneous manner. Particularly, we show the following characteristics:
Sharing predicates Rule predicates are exactly DL concepts and DL roles, taking advantage of the expressivity and reasoning power of both DL and rules.
Negative atoms The default negation "`not`" is allowed to prefix atoms in the body, making non-monotonicity applicable. Note that the classical negation "$\neg$" is still preserved for DL's negative constructor.
Open Answer Set Semantics Extended Herbrand structures are used for interpreting DL concepts and DL roles, while open answer sets hold for general rules, making a unique names assumption. Unnamed individuals, e.g. as introduced by DL existential restrictions, also occur in the open domain. To retain decidability, a well-known weak safeness condition [22] is employed, grounding variables in the rule head with named individuals.
Tableau-based algorithms Decision procedures for the knowledge base (KB) satisfiability problem and the query entailment problem are designed on completion graphs, getting rules incorporated into classical DL tableaux algorithms (which originally work on completion forests or trees).

The remainder of this paper starts with preliminaries for integrating DL and general rules in our homogeneous language $\mathcal{ALC}^{u}_{\mathbb{P}}$. The syntax and semantics

are defined in Section 3. Section 4 elaborates on algorithms, giving decision procedures for the KB satisfiability problem and the query entailment problem. Finally, Section 5 is our conclusion. Because of paper space limitations, detailed proofs are available in an Online Appendix[1].

2 Preliminaries

Description Logic, as discussed in this paper, is a fragment of classical first-order logic (FOL). Therefore, the semantics for DL is based on first-order interpretations, of which the domain is arbitrary. However, a fixed domain, viz. the Herbrand universe, in which rule variables are instantiated, is the key to the semantics of logic programming (LP). When combining DL and rules, we first of all should figure out the domain in common. A good candidate is the so-called *open domain*, i.e., an arbitrary non-empty countable superset of the Herbrand universe, as proposed for open answer set programming to solve the lack of modularity in closed world answer set programming [12].

Moreover, in the context of LP, rule variables are ultimately substituted by constants, receiving a grounded version of rules. Within a combined signature of DL and rules [5], constants are referred to as *named individuals*, which are asserted explicitly into the corresponding KB, and the Herbrand universe exactly consists of those constants. Nevertheless, DL existential restrictions would introduce *unnamed individuals*, and unnamed individuals also act as constants but unfortunately are beyond the Herbrand universe. Again, the open domain appears to be the right place for capturing unnamed individuals. Referring to [6], if there are no unnamed individuals in the domain, we say the *parameter names assumption (PNA)* applies. Not surprisingly, we prefer not to adopt PNA.

Open answer set semantics adheres to a *unique names assumption (UNA)*, which is not the case for DL. However, if desired, the UNA can be made explicit in DL by adding an assertion $a \neq b$ for each pair of differently named individuals to the KB. In this respect, we also adopt UNA. Since, the combination of PNA and UNA is called the *standard names assumption (SNA)*, our proposal is under UNA but neither PNA nor SNA, while $\mathcal{DL} + log$ [22] etc. adopted SNA.

Next, we should point out the (un)decidability issue. Actually, a decade ago, the undecidability of an unrestricted combination of DL and rules was proved with CARIN [16]. For reobtaining decidability, safeness conditions were proposed, e.g., role-safeness in [16], DL-safeness in [19] and weak safeness in [22], each of which ensures a certain separation between DL and rule predicates. Differing from those predicate-separated systems, this paper is based on rule predicates shared with DL concepts and DL roles. Thus, we merely require the most general Datalog safeness in the *syntax*, while we adopt a *semantic* weak safeness condition that relies on grounding variables in the rule head with named individuals to avoid undecidability.

On the other hand, DL is a family with many layers [2][13]. Bottom up, $\mathcal{ALC}$ is a basic and simple language, permitting concept descriptions via $C_1 \sqcap$

[1] http://www.is.pku.edu.cn/~mayyam/proof.pdf

$C_2, C_1 \sqcup C_2, \neg C, \forall R.C$, and $\exists R.C$, where C, C_1, C_2 are concepts and R is a role. Augmented by transitive roles, $\mathcal{ALC}$ becomes $\mathcal{ALC}_{\mathcal{R}^+}$, denoted by $\mathcal{S}$ in the following. $\mathcal{SI}$ is an extension to $\mathcal{S}$ with inverse roles, followed by $\mathcal{SHI}$ with role hierarchies. It is called $\mathcal{SHIF}$ if extending by functional restrictions, $\mathcal{SHIN}$ if by cardinality restrictions, and $\mathcal{SHIQ}$ if by qualified number restrictions. Support for datatype predicates (e.g., string, integer) brings up the concrete domain of $\mathbf{D}$, and using nominals $\mathcal{O}$ helps construct concepts with singleton sets. With the expected pervasive use of OWL, $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ are paid much attention: OWL Lite is a syntax variant of one, and OWL DL, of the other.

As the DL foundation, $\mathcal{ALC}$ is regarded as the right level for our homogeneous DL-rule integration, observing that the expressivity of extending $\mathcal{ALC}$ with Datalog rules covers that of $\mathcal{SHI}$, i.e., $\mathcal{ALC}_{\mathcal{R}^+}\mathcal{HI}$, except for the semantic weak safeness condition for rules. Specifically, the role hierarchy $\mathcal{H}$ of $R_1 \sqsubseteq R_2$ is captured by $R_2(x, y) \leftarrow R_1(x, y)$, and inverse roles $\mathcal{I}$ and transitive roles $\mathcal{R}^+$ are characterized, respectively, by $\texttt{Inv}(R)(y, x) \leftarrow R(x, y)$ and $S(x, z) \leftarrow S(x, y), S(y, z)$, where $R_1, R_2, R, \texttt{Inv}(R)$ and S are roles, in addition to $\texttt{Trans}(S)$ being `true`.

At this point, we are ready for a proposal of $\mathcal{ALC}^u_{\mathbb{P}}$, where the superscript "u" denotes the adoption of UNA, and the subscript "$\mathbb{P}$" means a program of *general* (*normal* [17]) rules. Even if the "`not`" operator of general rules in $\mathbb{P}$ is removed from $\mathcal{ALC}^u_{\mathbb{P}}$, our proposal develops $\mathcal{SHI}$ into a language with "more Datalog and less DL". The "more Datalog" characteristic is evident, since 'end-user' rules are integrated right into the foundation of the DL machinery. The "less DL" characteristic results from the '$\mathcal{SHI}$-desugaring' rules, translating the role hierarchy $\mathcal{H}$, inverse roles $\mathcal{I}$, and transitive roles $\mathcal{R}^+$, but because of their semantic weak safeness condition not providing full support for $\mathcal{SHI}$ on top of $\mathcal{ALC}$.

This finally brings us to the design of algorithms. DL tableaux algorithms yield completion trees (resp. forests) for checking DL concept satisfiability [13] (resp. ABox consistency [14]). Those completion trees or forests are finite, and represent a set of possibly infinite models. For getting general rules incorporated homogeneously into this setting, we will use a parameter l and develop l-completion graphs. Thus, our algorithm is still tableau-based, and we conclude that: (1) An $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K}$ is satisfiable iff the algorithm yields a complete and clash-free $l_{\mathcal{K}}$-completion graph; and (2) a query Q is entailed by an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K}$ iff Q is mappable to all complete and clash-free $l_{\mathcal{K},Q}$-completion graphs that the algorithm yields. More technical details are discussed below.

3 The $\mathcal{ALC}^u_{\mathbb{P}}$ Language

Syntactically, $\mathcal{ALC}^u_{\mathbb{P}}$, built on $\mathcal{ALC}$-concepts and $\mathcal{ALC}$-roles, has three parts: a TBox $\mathcal{T}$ of subsumptions, an ABox $\mathcal{A}$ of assertions, and additionally, a PBox $\mathbb{P}$ of general rules. Semantically, open answer sets are associated with extended Herbrand structures, treating weak safeness as a semantic condition for rules.

3.1 Syntax

Let $\mathbf{I}$ be a finite set of named individuals, and $\mathcal{V} = \{x, y, z, x_1, \cdots\}$ a countable set of variables. A (Datalog) term is a named individual or a variable.

Let N_C be a set of concept names and N_R a set of role names. The set $\mathbf{R}$ of $\mathcal{ALC}$-roles is N_R. The set $\mathbf{C}$ of $\mathcal{ALC}$-concepts is the smallest set such that (1) The top concept $\top$ and the bottom concept $\bot$ are $\mathcal{ALC}$-concepts; (2) Every concept name in N_C is an $\mathcal{ALC}$-concept; (3) If C, C_1, C_2 are $\mathcal{ALC}$-concepts and R is an $\mathcal{ALC}$-role, then $\neg C, C_1 \sqcap C_2, C_1 \sqcup C_2, \exists R.C, \forall R.C$ are also $\mathcal{ALC}$-concepts.

A concept is said to be in *negation normal form* (NNF) if negation occurs only in front of concept names. By pushing negations inwards using a combination of DeMorgan's laws, any concept can be translated to NNF in linear time, and we will assume that all concepts are in NNF in this paper. Given a concept C, $\texttt{clos}(C)$ is the smallest set that contains C and is closed under subconcepts and negation (in NNF). For a set of concepts M, $\texttt{clos}(M) = \bigcup_{C \in M} \texttt{clos}(C)$. The size of $\texttt{clos}(C)$ is linear in the length of C, and the size of $\texttt{clos}(M)$ is polynomial in the size of M.

Definition 1. *An $\mathcal{ALC}^u_{\mathbb{P}}$ KB has the form $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathbb{P})$, where*
TBox $\mathcal{T}$: Subsumptions are $C_1 \sqsubseteq C_2$ with $C_1, C_2 \in \mathbf{C}$
ABox $\mathcal{A}$: Assertions are $C(a)$ or $R(a, b)$ with $C \in \mathbf{C}$, $R \in \mathbf{R}$, and $a, b \in \mathbf{I}$
PBox $\mathbb{P}$: Rules are $r : p(\boldsymbol{u}) \leftarrow q_1(\boldsymbol{v_1}), \cdots, q_m(\boldsymbol{v_m}), \texttt{not}\ q_{m+1}(\boldsymbol{v_{m+1}}), \cdots \texttt{not}\ q_n(\boldsymbol{v_n})$ with $p, q_i \in \mathbf{C} \cup \mathbf{R}$, and $\boldsymbol{u}, \boldsymbol{v_i}$ are vectors of terms in $\mathbf{I} \cup \mathcal{V}$, for each $1 \leqslant i \leqslant m \leqslant n$ (each vector has length 1 or 2 since concepts from $\mathbf{C}$ become unary predicates and roles from $\mathbf{R}$ become binary predicates)

We remark that weak safeness [22] for general rules originally assumes a lexical separation of DL and rule predicates, while $\mathcal{ALC}^u_{\mathbb{P}}$ permits rule predicates shared with DL concepts and DL roles. Later, we will show how weak safeness is moved into the semantics. Syntactically, rule variables in the PBox $\mathbb{P}$ are merely required to satisfy the most general Datalog safeness condition. That is, every variable in a rule r must appear among at least one of the $\boldsymbol{v_1}, \cdots, \boldsymbol{v_m}$.

3.2 Semantics

Referring to [6], we introduce first-order and (extended) Herbrand structures.

Given a function-free first-order language $\mathcal{L}$, an $\mathcal{L}$-structure is a pair $\mathcal{I} = \langle U, I \rangle$, where the universe $U = (\mathcal{D}, \sigma)$ consists of a non-empty domain $\mathcal{D}$ and a function $\sigma : \mathbf{I} \cup \mathcal{D}' \to \mathcal{D}$ which assigns a domain value to each individual, and $\sigma(d) = d$ for all $d \in \mathcal{D}'$, given $\mathbf{I} \cap \mathcal{D}' = \emptyset$. Elements of $\mathcal{D}'$ are called *unnamed* individuals. We remark that the corresponding definitions in [6] are less clear, where $\sigma : \mathbf{I} \cup \mathcal{D} \to \mathcal{D}$ and any $d \in \mathcal{D}$ is defined as an *unnamed* individual if there is no $i \in \mathbf{I}$ such that $\sigma(i) = d$.

Using σ, we formalize UNA, PNA, and SNA as follows: in case σ is injective, the UNA applies; in case $\mathcal{D}'$ is empty, the PNA applies; the SNA is exactly the combination of UNA and PNA.

We call I an $\mathcal{L}$-interpretation over $\mathcal{D}$, which assigns a relation $p^I \subseteq \mathcal{D}^n$ to each n-ary predicate symbol p (here $n \geq 1$). Being a fragment of function-free first-order logic, DL can also rely on structures for interpreting concepts (here $n = 1$) and roles (here $n = 2$).

Answer set semantics is usually defined in terms of a *Herbrand structure* that has a fixed universe, namely *Herbrand universe* $H = (\mathbf{I}, id)$, where $id : \mathbf{I} \to \mathbf{I}$ is the identity function. Obviously, by $\mathbf{I}$ and id, the SNA applies here.

Relaxing the PNA, open answer set semantics considers an *extended Herbrand structure* based on an *extended Herbrand universe* $eH = (\mathcal{D}, id)$, where $id : \mathbf{I} \cup \mathcal{D}' \to \mathcal{D}$ is still an identity function and $id(d) = d$ for all $d \in \mathbf{I} \cup \mathcal{D}'$, given $\mathbf{I} \cap \mathcal{D}' = \emptyset$. Thus, by id, the UNA applies, but unnamed individuals reside in $\mathcal{D}'$.

Definition 2. *An extended Herbrand structure $\mathcal{I} = \langle (\mathcal{D}, id), I \rangle$ is defined for a set of named individuals $\mathbf{I}$, a set of concepts $\mathbf{C}$ and a set of roles $\mathbf{R}$, where*

$id : \mathbf{I} \cup \mathcal{D}' \to \mathcal{D}$ and $id(d) = d$ for all $d \in \mathbf{I} \cup \mathcal{D}'$, given $\mathbf{I} \cap \mathcal{D}' = \emptyset$

$I : \mathbf{C} \to 2^{\mathcal{D}}$ for concepts and $I : \mathbf{R} \to 2^{\mathcal{D} \times \mathcal{D}}$ for roles

such that for concepts $C, C_1, C_2 \in \mathbf{C}$ and roles $R \in \mathbf{R}$, the following are satisfied:
$\top^I = \mathcal{D} \quad \bot^I = \emptyset \quad (\neg C)^I = \mathcal{D} \backslash C^I \quad (C_1 \sqcap C_2)^I = C_1^I \cap C_2^I \quad (C_1 \sqcup C_2)^I = C_1^I \cup C_2^I$
$(\exists R.C)^I = \{e_1 \in \mathcal{D} | \exists e_2.(e_1, e_2) \in R^I \text{ and } e_2 \in C^I\}$
$(\forall R.C)^I = \{e_1 \in \mathcal{D} | \forall e_2.(e_1, e_2) \in R^I \text{ implies } e_2 \in C^I\}$

An associated valuation v_I of an interpretation I over $\mathcal{D}$ is a mapping s.t.
$v_I(C(d)) = true$, *if* $d \in C^I$, *where* $C \in \mathbf{C}$ *and* $d \in \mathcal{D}$
$v_I(R(d_1, d_2)) = true$, *if* $(d_1, d_2) \in R^I$, *where* $R \in \mathbf{R}$ *and* $d_1, d_2 \in \mathcal{D}$

An extended Herbrand structure $\mathcal{I}$ satisfies a TBox $\mathcal{T}$ if, $C_1^I \subseteq C_2^I$ for all $C_1 \sqsubseteq C_2$ in $\mathcal{T}$, where $C_1, C_2 \in \mathbf{C}$. Such a structure $\mathcal{I}$ is called a model of $\mathcal{T}$, written as $\mathcal{I} \models \mathcal{T}$. An extended Herbrand structure $\mathcal{I}$ satisfies an ABox $\mathcal{A}$ if, $id(a) = a \in C^I$ and $(id(a_1), id(a_2)) = (a_1, a_2) \in R^I$ for all $C(a)$ and $R(a_1, a_2)$ in $\mathcal{A}$, where $C \in \mathbf{C}$, $R \in \mathbf{R}$ and $a, a_1, a_2 \in \mathbf{I}$. Such a structure $\mathcal{I}$ is called a model of $\mathcal{A}$, written as $\mathcal{I} \models \mathcal{A}$.

To define a model of a PBox $\mathbb{P}$, we start by grounding $\mathbb{P}$. The grounding $\mathbb{P}_g$ of $\mathbb{P}$ w.r.t. an extended Herbrand universe $eH = (\mathcal{D}, id)$ is the set of all rules obtained as follows. For every rule r in $\mathbb{P}$,
(1) keep each named individual $a \in \mathbf{I}$ appearing in r unchanged as $id(a) = a \in \mathcal{D}$,
(2) replace each variable $v \in \mathcal{V}$ appearing in r with a certain $d \in \mathcal{D}$,
(3) replace each variable $v \in \mathcal{V}$ appearing in the head of r with a certain $d \in \mathbf{I}$.

In order to guarantee decidability, the semantic condition of (3) is proposed. While $\mathcal{DL} + log$ [22] has defined such a (syntactical) weak safeness condition for hybrid rules, we rephrase it semantically for homogeneous rules here. That is, only named individuals are legal for grounding the head of rules, so unnamed individuals cannot be propagated by rules.

Below, given an extended Herbrand structure $\mathcal{I} = \langle (\mathcal{D}, id), I \rangle$, we will first consider the grounded PBox $\mathbb{P}_g$ of rules without `not`, i.e., $m = n$ for all rules r in $\mathbb{P}_g$. Put differently, $\mathbb{P}_g$ corresponds to the positive (function-free Horn) case of traditional logic programming. The *extended Herbrand model* of $\mathbb{P}_g$ is a set S such that, for any rule $r : p(\boldsymbol{u}) \leftarrow q_1(\boldsymbol{v_1}), \cdots, q_m(\boldsymbol{v_m})$ in $\mathbb{P}_g$, if $q_i(\boldsymbol{v_i}) \in S$ for

all $1 \leqslant i \leqslant m$, then $p(\boldsymbol{u}) \in S$. By $\lambda(\mathbb{P}_g)$, we denote the least extended Herbrand model of $\mathbb{P}_g$ in which none of the rules contains `not`.

Next, suppose $\mathbb{P}_g$ is a grounded PBox of general rules and S a set. Similarly as in the Gelfond-Lifschitz transformation [10], we denote with $\Gamma(\mathbb{P}_g, S)$ the set of rules obtained from $\mathbb{P}_g$ by deleting
1. each rule $r \in \mathbb{P}_g$ that has a "`not` $q(\boldsymbol{v})$" in the rule body with $q(\boldsymbol{v}) \in S$;
2. all "`not` $q(\boldsymbol{v})$" occurrences in the bodies of the remaining rules.

Clearly, $\Gamma(\mathbb{P}_g, S)$ does not contain "`not`" any more, and its extended Herbrand model is already defined above. If the least extended Herbrand model of $\Gamma(\mathbb{P}_g, S)$ coincides with S, then we say that S is an *open answer set* of $\mathbb{P}_g$. In other words, open answer sets of $\mathbb{P}_g$ are characterized by the equation $S = \lambda(\Gamma(\mathbb{P}_g, S))$.

An extended Herbrand structure $\mathcal{I}$ satisfies a PBox $\mathbb{P}$ if the set $S = \{C(d) | v_I(C(d)) = true\} \cup \{R(d_1, d_2) | v_I(R(d_1, d_2)) = true\}$ is an open answer set of $\mathbb{P}_g$. Such a structure $\mathcal{I}$ is called a model of $\mathbb{P}$, written as $\mathcal{I} \models \mathbb{P}$.

An extended Herbrand structure $\mathcal{I}$ satisfies an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathbb{P})$ if $\mathcal{I}$ is a model of $\mathcal{T}, \mathcal{A}$ and $\mathbb{P}$. Such a structure $\mathcal{I}$ is called a model of $\mathcal{K}$, written as $\mathcal{I} \models \mathcal{K}$. A KB $\mathcal{K}$ is *satisfiable* if there is a model of $\mathcal{K}$. Two KBs $\mathcal{K}_1$ and $\mathcal{K}_2$ are *equivalent* if the models of $\mathcal{K}_1$ are also the models of $\mathcal{K}_2$, and vice versa.

3.3 An Example

We demonstrate an example in Table 1, about the policy of "one family, one child" for the current generation in China.

Disjunction is exemplified by (1), a cyclic TBox by (2) and existential restriction by (3), all of which state the properties of a person. Among persons, the current generation is described in (4) having OnlyChild as descendants.

Recursive rules are used for the "descend" relationship by (5) and (6), of which (5) is the base and (6) is the propagation. Rule (7) describes a symmetric relationship of "hasSpouse". Rules (8) and (9) reflect the disjunction of male and female, according to (1), while role subsumptions of having parents are shown in (10). Excluding unmarried parents, we suppose, in (11), someone having a child has a spouse. As for (12) and (13), the OnlyChild has no sibling, while someone whose parents both have no siblings has no cousin.

Default negation appears in (14) and (15), also DL existential restriction participates in the rule head. The current generation, in general, has siblings (resp. cousins), but not in the case when there is an explicit statement of having no sibling (resp. having no cousin).

Classical negation, serving for DL negative concepts, appears in the head of rules (16), (17) and (18). It seems redundant to state both (18) and (13), whose heads are complements of each other. We remark that "hasSibling", in the rule body of (18), is possibly derived from (14), while (13) helps to override the default in (14). However, (14) merely concerns the current generation, and (13) is for all.

In (19), classical negation gets along with default negation, stating that OnlyChild generally has another OnlyChild as his/her spouse.

Table 1. An example of an $\mathcal{ALC}^u_{\mathbb{P}}$ KB

(1) Person $\sqsubseteq$ Male $\sqcup$ Female	Person is male or female.
(2) $\sqsubseteq \forall$ hasChild.Person	Any child of a person is a person.
(3) $\sqsubseteq \exists$ hasFather.Male $\sqcap \exists$ hasMother.Female	Any person has a father and a mother.
(4) CCG $\sqsubseteq$ Person $\sqcap \forall$ descend.OnlyChild	CCG: The current generation.
(5) $\text{descend}(x, y) \leftarrow \text{hasChild}(x, y)$.	The relationship of descend is the
(6) $\text{descend}(x, z) \leftarrow \text{descend}(x, y), \text{hasChild}(y, z)$.	transitive closure of having children.
(7) $\text{hasSpouse}(x, y) \leftarrow \text{hasSpouse}(y, x)$.	Having spouse is symmetric.
(8) $\text{hasFather}(x, y) \leftarrow \text{hasChild}(y, x), \text{Male}(y)$.	A male having a child is the father.
(9) $\text{hasMother}(x, y) \leftarrow \text{hasChild}(y, x), \text{Female}(y)$.	A female having a child is the mother.
(10) hasFather $\sqsubseteq$ hasParent hasMother $\sqsubseteq$ hasParent	Parents consist of father and mother.
(11) $\exists$ hasChild.Person $\sqsubseteq \exists$ hasSpouse.Person	One having a child has a spouse.
(12) $\text{NoSibling}(x) \leftarrow \text{OnlyChild}(x)$.	One in OnlyChild belongs to NoSibling.
(13) $\text{NoCousin}(x) \leftarrow \text{hasFather}(x, y), \text{NoSibling}(y)$.	One whose father and mother are both
$\text{hasMother}(x, z), \text{NoSibling}(z)$.	in NoSibling belongs to NoCousin.
(14) $\exists$ hasSibling.Person$(x) \leftarrow \text{CCG}(x)$,	One CCG not being known as NoSibling
$\texttt{not}\ \text{NoSibling}(x)$.	has some sibling.
(15) $\exists$ hasCousin.Person$(x) \leftarrow \text{CCG}(x)$,	One CCG not being known as NoCousin
$\texttt{not}\ \text{NoCousin}(x)$.	has some cousin.
(16) $\neg\ \text{NoSibling}(x) \leftarrow \text{hasSibling}(x, y)$.	One with sibling is outside of NoSibling.
(17) $\neg\ \text{NoCousin}(x) \leftarrow \text{hasCousin}(x, y)$.	One with cousin is outside of NoCousin.
(18) $\neg\ \text{NoCousin}(x) \leftarrow \text{hasParent}(x, y), \text{hasSibling}(y, z)$.	One special is outside of NoCousin.
(19) $\text{OnlyChild}(y) \leftarrow \text{hasSpouse}(x, y), \text{OnlyChild}(x)$,	The OnlyChild generally has
$\texttt{not}\ \neg\ \text{NoSibling}(y)$.	another OnlyChild as his/her spouse.

	Therefore	
[1]	CCG $\sqsubseteq \forall$descend.NoSibling	
[2]	CCG $\sqsubseteq \neg$NoSibling $\sqcap\neg$NoCousin $\sqcap\forall$hasChild.$\forall$descend.NoCousin	
[3]	Antecedent(*)	CCG(a). CCG(b). OnlyChild(a). $\forall$hasParent.OnlyChild(b).
	Consequence(*)	NoSibling(a). $\neg$NoCousin(a). NoCousin(b). $\neg$NoSibling(b).
	Suppose	AmusingFamily $\leftarrow$ amusedBy(x, y).
		$\text{amusedBy}(x_2, y_2) \leftarrow \text{hasSpouse}(x_1, y_1), \text{NoSibling}(x_1), \text{NoCousin}(y_1),$
		$\text{hasCousin}(x_1, x_2), \text{hasSibling}(y_1, y_2), \texttt{not}\ \text{amusedBy}(y_2, x_2)$.
	Conclude	AmusingFamily when Antecedent(*) holds in addition to hasSpouse(a, b).

So far, we conclude definitely that [1]: descendants of the current generation are those without any sibling, as derived from (4)(12). By default, we conclude that [2]: (14)(16) and (15)(17) respectively state the current generation has siblings and cousins, while (13) implies that children of the current generation will have descendants without any cousin. For [3], when the antecedent (*) arises, we have one person a and the other person b as the current generation. Being OnlyChild, a has no sibling – an exception to (14). An exception to (15) is b, whose parents are both OnlyChild, and b has no cousin.

Suppose an amusing family, in which one is amused by the other. For a couple, one has a cousin but no sibling, and the other has a sibling but no cousin. As to such a case, the cousin is amused by the sibling, or in the converse direction. Given the antecedent (*) plus hasSpouse(a, b), an amusing family does exist.

4 Algorithms

In the following, if not stated otherwise, for an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathbb{P})$, we denote that: (1) Σ_C is the closure of concepts occurring in $\mathcal{T}$, $\mathcal{A}$ and $\mathbb{P}$; (2) Σ_R is the set of roles occurring in $\mathcal{T}$, $\mathcal{A}$ and $\mathbb{P}$; (3) Σ_I is the set of named individuals occurring in $\mathcal{A}$ and $\mathbb{P}$.

The algorithm first rewrites TBox subsumptions with rules, and $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathbb{P})$ becomes $\mathcal{K}' = (\emptyset, \mathcal{A}, \mathbb{P}^{\mathcal{T}})$, followed by $\mathcal{K}'' = (\emptyset, \mathcal{A}, \mathbb{P}^{\mathcal{T},E})$ to compute extensions of complex $\mathcal{ALC}$-concepts. Upon that, we will establish completion graphs, towards the decision procedure for the KB satisfiability problem and the query entailment problem, respectively.

4.1 Preprocessing

Given an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathbb{P})$, concept subsumptions in the TBox $\mathcal{T}$ are rewritten such that $\mathbb{P}^{\mathcal{T}} = \mathbb{P} \cup \{C_2(x) \leftarrow C_1(x) | C_1 \sqsubseteq C_2 \in \mathcal{T}\}$, and the KB is updated to $\mathcal{K}' = (\emptyset, \mathcal{A}, \mathbb{P}^{\mathcal{T}})$.

Next, the computation, along with reasoning, is used to evaluate extensions of complex $\mathcal{ALC}$-concepts, and now the KB becomes $\mathcal{K}'' = (\emptyset, \mathcal{A}, \mathbb{P}^{\mathcal{T},E})$. Starting from $\mathbb{P}^{\mathcal{T}}$, we obtain $\mathbb{P}^{\mathcal{T},E}$ by appending rules for each computable $\mathcal{ALC}$-concept in Σ_C, that is $C_1 \sqcap C_2, C_1 \sqcup C_2$ and $\exists R.C$. We observe that classical negation appears in $\neg C$ and $\forall R.C$, where $\forall R.C$ concerns a case for the negation of R. Incomplete information possibly leads to neither positive nor negative atoms, which motivates us to introduce *universal* concepts of $C \sqcup \neg C$ and $\forall R.C \sqcup \exists R.\neg C$ for $\neg C$ and $\forall R.C$, respectively. So that, individuals in the top concept $\top$ are assigned into those universal concepts.

$C_1 \sqcap C_2(x) \leftarrow C_1(x), C_2(x).$ $\quad C_1 \sqcup C_2(x) \leftarrow C_1(x).$ $\quad C_1 \sqcup C_2(x) \leftarrow C_2(x).$

$\exists R.C(x) \leftarrow R(x,y), C(y).$ $\quad C \sqcup \neg C(x) \leftarrow \top(x).$ $\quad \forall R.C \sqcup \exists R.\neg C(x) \leftarrow \top(x).$

Above, computation rules are specified for every concept $\neg C, C_1 \sqcap C_2, C_1 \sqcup C_2, \exists R.C$ and $\forall R.C$ appearing in Σ_C. Since these 'system-level' rules are designed for DL concepts, named individuals and facts need not to stay here. We also realize that, in a similar but more elaborate manner, [12] simulates DLs via open answer set programming. Interestingly, support for DL reasoning totally replies on running simulation rules in [12], while we would develop DL tableaux-based algorithms getting the above computation rules involved.

In the Online Appendix[1], it shows that: The $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathbb{P})$ and its updated version $\mathcal{K}' = (\emptyset, \mathcal{A}, \mathbb{P}^{\mathcal{T}})$ as well as $\mathcal{K}'' = (\emptyset, \mathcal{A}, \mathbb{P}^{\mathcal{T},E})$ are equivalent.

4.2 Completion Graphs

Observing that role assertions are possibly refreshed by rules when such a role occurs in the rule head, completion graphs instead of completion forests [14] or completion trees [13] are studied in this paper.

A *completion graph* is a (directed) graph G where each node u is labeled with a set $\mathcal{L}(u) \subseteq \Sigma_C$ and each edge $\langle u, v \rangle$ is labeled with a set $\mathcal{L}(\langle u, v \rangle) \subseteq \Sigma_R$. If there is an edge $\langle u, v \rangle$ in G, then we say that v is the successor of u and u is the

predecessor of v. The transitive closure of predecessor (resp. successor) is called ancestor (resp. descendant). For a node u, $\mathcal{L}(u)$ is said to contain a *clash* if, for some concept C, $\{C, \neg C\} \subseteq \mathcal{L}(u)$.

Initially, we construct a graph $G_{\mathcal{A}}$ for an ABox $\mathcal{A}$ as follows.
– A node u_a is created, for each named individual $a \in \Sigma_I$.
– An edge $\langle u_a, u_b \rangle$ is created, if $R(a,b) \in \mathcal{A}$ for some role $R \in \Sigma_R$ and $a, b \in \Sigma_I$.
– The labels of these nodes and edges are initialized by
$\mathcal{L}(u_a) = \{C | C(a) \in \mathcal{A}\}$ and $\mathcal{L}(\langle u_a, u_b \rangle) = \{R | R(a,b) \in \mathcal{A}\}$.

Running $\mathsf{CompGraph}(G_{\mathcal{A}})$, the algorithm proceeds.

Procedure CompGraph
Input: A graph G_{in}
Output: A set of graphs G_{out}
begin $G_{out} := \emptyset$
  for each g in $\mathsf{ExpGraph}(G_{in})$ **do**
    for each g' in $\mathsf{SmsGraph}(g)$ **do**
      if g' is *complete* **then** $G_{out} := G_{out} \cup \{g'\}$
        else $G_{out} := G_{out} \cup \mathsf{CompGraph}(g')$
  return G_{out}
end

Expansion principles in Table 2 are ready for the procedure of ExpGraph. The application of $\sqcup$-principle is non-deterministic, branching graphs. The $\exists$-principle is a generating principle since, possibly, fresh new nodes are inserted into the graph. Besides, we call nodes having been located in $G_{\mathcal{A}}$ as a-nodes, each of which represents a named individual, and b-nodes for the others, each of which represents a unnamed individual (e.g., being introduced by the $\exists$-principle).

Table 2. Expansion Principles for ExpGraph

$\sqcap$:	if $C_1 \sqcap C_2 \in \mathcal{L}(u)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(u)$ then $\mathcal{L}(u) := \mathcal{L}(u) \cup \{C_1, C_2\}$
$\sqcup$:	if $C_1 \sqcup C_2 \in \mathcal{L}(u)$ and $\{C_1, C_2\} \cap \mathcal{L}(u) = \emptyset$ then $\mathcal{L}(u) := \mathcal{L}(u) \cup \{C_1\}$ or $\mathcal{L}(u) := \mathcal{L}(u) \cup \{C_2\}$
$\forall$:	if $\forall R.C \in \mathcal{L}(u)$ and $R \in \mathcal{L}(\langle u, v \rangle)$ but $C \notin \mathcal{L}(v)$ then $\mathcal{L}(v) := \mathcal{L}(v) \cup \{C\}$
$\exists$:	if $\exists R.C \in \mathcal{L}(u)$ where u is an a-node or a b-node not being l-blocked, there does not exist any node v such that $R \in \mathcal{L}(\langle u, v \rangle)$ and $C \in \mathcal{L}(v)$ then create a b-node v with $\mathcal{L}(v) := \{C\}$ and an edge $\langle u, v \rangle$ with $\mathcal{L}(\langle u, v \rangle) := \{R\}$

Referring to definitions of n-tree equivalence, n-witness and n-blocking in [16][20], as restated below, we present l-blocking for the $\exists$-principle in Table 2. The parameter of l will take the value of $l_{\mathcal{K}}$ for the KB satisfiability problem and of $l_{\mathcal{K},Q}$ for the query entailment problem, given that
$l_{\mathcal{K}}$: The maximal of l_r for all r in $\mathbb{P}^{\mathcal{T},E}$ where l_r is the number of variables in r
l_Q: The length of a query Q and Section 4.4 presents more details
$l_{\mathcal{K},Q}$: The maximal of $l_{\mathcal{K}}$ and l_Q

Definition 3. *The n-tree of a node u is the tree that includes the node u and its descendants, whose distance from u is at most n successor edges. We denote the set of nodes in the n-tree of u by $V_n(u)$.*

Two nodes u, v in a graph G are said to be n-tree equivalent if there is an isomorphism $\psi : V_n(v) \to V_n(u)$ s.t. (1) $\psi(v) = u$; (2) $\mathcal{L}(s) = \mathcal{L}(\psi(s))$, for every $s \in V_n(v)$; (3) $\mathcal{L}(\langle s, t\rangle) = \mathcal{L}(\langle \psi(s), \psi(t)\rangle)$, for every $s, t \in V_n(v)$.

A node u is an n-witness of a node v in a graph G if (1) u is an ancestor of v, (2) u is n-tree equivalent to v, and (3) v is not in the n-tree of u.

A node w is n-blocked in a graph G if (1) one of its ancestors is n-blocked, or (2) w is in an n-tree of which root has an n-witness. Suppose u be an n-witness of v and $\psi : V_n(v) \to V_n(u)$ the isomorphism. For any node w in the n-tree of v, w is n-blocked by $\psi(w)$.

A completion graph G is called *complete* when for some node u in G, $\mathcal{L}(u)$ contains a clash, or when none of the expansion principles is applicable. The output of ExpGraph consists of complete completion graphs, each of which again becomes the input of SmsGraph. Intuitively, the procedure of ExpGraph serves for DL constructors, while the procedure of SmsGraph for general rules.

With an input graph g to SmsGraph, a "bottom" set B_g and a "top" set T_g are built. The former collects those labels in g such that $B_g = \{C(u)|C \in \mathcal{L}(u)\} \cup \{R(u, v)|R \in \mathcal{L}(\langle u, v\rangle)\}$, and the latter concerns all possible constituents s.t. $T_g = \{C(u)|C \in \Sigma_C$ and u appears in $g\} \cup \{R(u, v)|R \in \Sigma_R$ and u, v appear in $g\}$. By the Gelfond-Lifschitz transformation [10], we denote a *stable set* S_g s.t.
1. $B_g \subseteq S_g \subseteq T_g$;
2. For a rule $r : p(\boldsymbol{u}) \leftarrow q_1(\boldsymbol{v_1}), \cdots, q_m(\boldsymbol{v_m}), \texttt{not}\ q_{m+1}(\boldsymbol{v_{m+1}}), \cdots \texttt{not}\ q_n(\boldsymbol{v_n})$ satisfying all $q_j(\sigma(\boldsymbol{v_j})) \notin S_g$ and $m + 1 \leqslant j \leqslant n$, in $\mathbb{P}^{\mathcal{T},E}$, where σ is a term assignment w.r.t. g and r, if $q_i(\sigma(\boldsymbol{v_i})) \in S_g$ for each $1 \leqslant i \leqslant m$ then $p(\sigma(\boldsymbol{u})) \in S_g$.

A *term assignment* σ w.r.t. g and r is a mapping which assigns
(1) a node in g to every variable in r,
(2) an a-node u_a in g to every named individual a in r,
(3) an a-node in g to every variable appearing in the head of r.
Naturally, the assignment of (3) concerns the *semantic* weak safeness.

After receiving all stable sets, we "repay" the input graph g with an output set of graphs, SmsGraph(g), each of which is constructed by a stable set S_g s.t.
– Nodes are created the same as g;
– An edge $\langle u, v\rangle$ is created, if $R(u, v) \in S_g$ for some R;
– Labels are $\mathcal{L}(u) = \{C|C(u) \in S_g\}$ and $\mathcal{L}(\langle u, v\rangle) = \{R|R(u, v) \in S_g\}$.

Since, the completion graph g' in SmsGraph(g) updates g (e.g., having new edges or more labels of nodes and edges), those expansion principles in Table 2 are possibly again applicable to g'. When g' happens not to being complete, a call to CompGraph(g') is stacked. If completion graphs, obtained from procedures of both ExpGraph and SmsGraph, are totally complete, the algorithm terminates. We remark l-blocking, which occurs in the $\exists$-principle at Table 2, is crucial to termination, and the following two subsections will elaborate on the parameter l: one takes $l_{\mathcal{K}}$ for the KB satisfiability problem, and the other is $l_{\mathcal{K},Q}$ for the query entailment problem.

4.3 The KB Satisfiability Problem

The above algorithm that yields $l_{\mathcal{K}}$-completion graphs is shown up as a decision procedure for the satisfiability problem w.r.t. an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathbb{P})$.

Recalling that weak safeness plays a role in our algorithm, the Online Appendix[1] declares the termination: For an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K}$, the algorithm terminates.

As to the soundness, which states that if the algorithm yields a complete and clash-free $l_{\mathcal{K}}$-completion graph then $\mathcal{K}$ is satisfiable, we need to introduce canonical structures for completion graphs (cf. [16] and [20]).

Definition 4. *Suppose G be an $l_{\mathcal{K}}$-completion graph generated by the algorithm for $\mathcal{K}$. A canonical structure $\mathcal{I}_G = \langle(\mathcal{D}_G, id), I_G\rangle$ for G is defined such that*
1. $\mathcal{D}_G := \{u \mid u$ is a node in $G\}$
2. For each named individual $a \in \Sigma_I$, $id(a) \in \mathcal{D}_G$ corresponds to its a-node u_a
3. For each concept $C \in \Sigma_C$, $C^{I_G} := \{u \mid C \in \mathcal{L}(u)\}$
4. For each role $R \in \Sigma_R$, $(u, v) \in R^{I_G}$ if and only if (1) $R \in \mathcal{L}(\langle u, v\rangle)$; or (2) $R \in \mathcal{L}(\langle\psi(u), v\rangle)$ where u is $l_{\mathcal{K}}$-blocked, t is the root of the $l_{\mathcal{K}}$-tree to which u belongs, s is the witness of t, $\psi : V_{l_{\mathcal{K}}}(t) \rightarrow V_{l_{\mathcal{K}}}(s)$ is an isomorphism between the $l_{\mathcal{K}}$-trees rooted with t and s.

Note that, for an $l_{\mathcal{K}}$-blocked node u, *explicit* edges, e.g., $\langle u, v\rangle$, are not available, but *implicit* edges, e.g., $\langle\psi(u), v\rangle$, rather contribute to interpreting roles.

Specifically, if the algorithm yields a complete and clash-free $l_{\mathcal{K}}$-completion graph G for an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K}$, a canonical structure $\mathcal{I}_G = \langle(\mathcal{D}_G, id), I_G\rangle$ for G is proved (in the Online Appendix[1]) as the model of $\mathcal{K}$, so that $\mathcal{K}$ is satisfiable.

Next is the completeness, which states that if $\mathcal{K}$ is satisfiable, then the algorithm yields a complete and clash-free $l_{\mathcal{K}}$-completion graph.

Since, $\mathcal{K}$ is satisfiable, by definitions, there is an extended Herbrand structure $\mathcal{I} = \langle(\mathcal{D}, id), I\rangle$ satisfying the ABox $\mathcal{A}$, the TBox $\mathcal{T}$ and the PBox $\mathbb{P}$. Referring to [14], we use $\mathcal{I}$ to trigger the application of the expansion principles such that they yield a complete and clash-free $l_{\mathcal{K}}$-completion graph. To this propose, a function π is defined, mapping nodes in a graph G to the domain $\mathcal{D}$, as follows. (1) For a named individual a, $\pi(u_a) := a$ where u_a is the corresponding a-node. (2) If $\pi(u) = s$ is already defined, and a successor v of u was generated for $\exists R.C \in \mathcal{L}(u)$, then $\pi(v) := t$ for some $t \in \mathcal{D}$ with $t \in C^I$ and $(s, t) \in R^I$.

For all nodes u, v in a completion graph G, we claim a condition that
$\mathcal{L}(u) \subseteq \{C|\pi(u) \in C^I\}$ and $\mathcal{L}(\langle u, v\rangle) \subseteq \{R|(\pi(u), \pi(v)) \in R^I\}$. (*)

As shown in the Online Appendix[1], the algorithm ends up with certain complete $l_{\mathcal{K}}$-completion graph, denoted by $G_{\mathcal{I}}$, satisfying the condition (*). Because $\mathcal{I} = \langle(\mathcal{D}, id), I\rangle$ is a model of $\mathcal{K}$, we have $C^I \cap (\neg C)^I = \emptyset$ for any concept $C \in \Sigma_C$, which implies $G_{\mathcal{I}}$ is clash-free. Otherwise, there exists a clash such that $\{D, \neg D\} \subseteq \mathcal{L}(u) \subseteq \{C|\pi(u) \in C^I\}$ in $G_{\mathcal{I}}$, for some concept D and some node u, making $\pi(u) \in D^I \cap (\neg D)^I$ conflict with the model $\mathcal{I}$. Summing up, $G_{\mathcal{I}}$ is the complete and clash-free $l_{\mathcal{K}}$-completion graph that the algorithm yields.

Theorem 1. *The algorithm is a decision procedure for the satisfiability of an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathbb{P})$, and decides the KB satisfiability problem in 3ExpTime w.r.t. the size of $\mathcal{K}$.*

4.4 The Query Entailment Problem

Before we address this problem, queries are formalized.

Definition 5. *A conjunctive query (CQ) q over an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K}$ is of the form being $\{p_1(\boldsymbol{w_1}), \cdots, p_n(\boldsymbol{w_n})\}$, where p_i is either a DL concept or a DL role, and $\boldsymbol{w_i}$ is a (unary, binary) vector of terms, for each $1 \leqslant i \leqslant n$. By l_q, we denote the length of a CQ $q = \{p_1(\boldsymbol{w_1}), \cdots, p_n(\boldsymbol{w_n})\}$, and $l_q = n$.*

A union of conjunctive queries (UCQ) q' over an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K}$ is of the form being $q_1 \vee \cdots \vee q_m$, where q_j is a CQ for each $1 \leqslant j \leqslant m$. By $l_{q'}$, we denote the length of an UCQ $q' = q_1 \vee \cdots \vee q_m$, and $l_{q'} = \mathtt{max}\{l_{q_j} | 1 \leqslant j \leqslant m\}$.

For simplification, Q is said to be a *query*, whether Q is a CQ or an UCQ. Queries are interpreted in a standard way. Given a query Q and an extended Herbrand structure $\mathcal{I} = \langle(\mathcal{D}, id), I\rangle$, the *variable substitution* w.r.t. Q and $\mathcal{I}$ is $\theta = \{x_1/t_1, \cdots, x_n/t_n\}$, which substitutes each variable $x_i \in \mathcal{V}$ appearing in Q with a (un)named individual $t_i \in \mathcal{D}$ and $1 \leqslant i \leqslant n$. We use the LP notation $\alpha\theta$ to apply θ to variables in an atom α. As for a CQ q, a structure $\mathcal{I}$ is a model of q, denoted by $\mathcal{I} \models q$, if there is a variable substitution θ w.r.t. q and $\mathcal{I}$ such that $\mathcal{I} \models p_i(\boldsymbol{w_i})\theta$ for each $p_i(\boldsymbol{w_i})$ in q, where $1 \leqslant i \leqslant n$. For an UCQ q', $\mathcal{I}$ is a model of q', denoted by $\mathcal{I} \models q'$, if $\mathcal{I} \models q_j$ for some q_j in q', where $1 \leqslant j \leqslant m$.

Given an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K}$ and a query Q, we say $\mathcal{K}$ *entails* Q, denoted by $\mathcal{K} \models Q$, if $\mathcal{I} \models Q$ for each model $\mathcal{I}$ of $\mathcal{K}$. The *query entailment* problem is to decide whether $\mathcal{K} \models Q$. We assume $\mathcal{K}$ being satisfiable, in this context; otherwise, everything can be entailed from a contradictory KB.

Now, for an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A}, \mathbb{P})$ and a query Q, we redefine that: (1) Σ_C is the closure of concepts occurring in $\mathcal{T}$, $\mathcal{A}$, $\mathbb{P}$ and Q; (2) Σ_R is the set of roles occurring in $\mathcal{T}$, $\mathcal{A}$, $\mathbb{P}$ and Q; (3) Σ_I is the set of named individual occurring in $\mathcal{A}, \mathbb{P}$ and Q. The parameter $l_{\mathcal{K},Q}$ is the maximal of $l_{\mathcal{K}}$ and l_Q, and the previous algorithm, as described in Section 4.2, will yield $l_{\mathcal{K},Q}$-completion graphs.

Next, referring to [20], we establish mappings from the query to those obtained $l_{\mathcal{K},Q}$-completion graphs. For a CQ $q : \{p_1(\boldsymbol{w_1}), \cdots, p_n(\boldsymbol{w_n})\}$ and a graph G, a mapping δ maps named individuals and variables in q to nodes in G, s.t.
1. For each named individual $a \in \Sigma_I$ in q, $\delta(a) = u_a$ is the corresponding a-node;
2. For each $C(t)$ and $R(t_1, t_2)$ in q where t, t_1, t_2 are named individuals or variables, $C \in \mathcal{L}(\delta(t))$ and $R \in \mathcal{L}(\langle\delta(t_1), \delta(t_2)\rangle)$.

A CQ q is mappable to a graph G, denoted by $q \hookrightarrow G$, if there exists such a mapping δ. An UCQ $q' : q_1 \vee \cdots \vee q_m$ is mappable to a graph G, denoted by $q' \hookrightarrow G$, if $q_j \hookrightarrow G$ for some q_j in q' where $1 \leqslant j \leqslant m$.

Thus, the algorithm returns "Q is entailed by $\mathcal{K}$", denoted as $\mathcal{K} \vdash Q$, if for all complete and clash-free $l_{\mathcal{K},Q}$-completion graphs G that the algorithm yields, $Q \hookrightarrow G$ holds, otherwise the algorithm returns "Q is not entailed by $\mathcal{K}$".

We direct readers to the Online Appendix[1] for details on termination, soundness, completeness, and complexity etc. Below is the resulting theorem.

Theorem 2. *The updated algorithm is a decision procedure for the entailment of an $\mathcal{ALC}^u_{\mathbb{P}}$ KB $\mathcal{K}$ to a query Q, and decides the query entailment problem in 3ExpTime w.r.t. the size of $\mathcal{K}$.*

5 Conclusion

This paper presents an $\mathcal{ALC}^u_{\mathbb{P}}$ KB, consisting of a TBox of subsumptions and an ABox of assertions, augmented by a PBox of general rules sharing predicates with the DL concepts and roles. For its open answer set semantics, extended Herbrand structures are used to interpret DL concepts and roles, while open answer sets hold for general rules. To retain decidability, a well-known weak safeness [22] condition is employed. We extend DL tableaux-based algorithms to $\mathcal{ALC}^u_{\mathbb{P}}$ decision procedures for the KB satisfiability and query entailment problems.

By way of comparison, CARIN [16] builds completion trees, which are used to evaluate hybrid rules externally, so that the information flow is uni-directional, i.e., from DL to rules but not vice versa. Our $\mathcal{ALC}^u_{\mathbb{P}}$ constructs completion graphs shared homogeneously between the DL and rule components, which makes bi-directional information flow a characteristic of $\mathcal{ALC}^u_{\mathbb{P}}$.

Although existing DL reasoners and rule engines facilitate other related work (e.g., [8][18][22]), we believe that novel algorithms, specified for a homogeneous integration of DL and general rules, enable the newly envisioned unifying logic on top of ontologies and rules in the Semantic Web. The unary/binary $\mathcal{ALC}^u_{\mathbb{P}}$ logic could be extended for n-ary relations, which may be realized by decomposition into binary ones; this might also benefit from $\mathcal{DLR}$'s [4] extension of binary DL roles to n-ary DL relations; we currently explore which n-ary approach works best with our (extended) $\mathcal{ALC}^u_{\mathbb{P}}$ algorithms. On top of n-ary relations, further rule layers, such as undecidable (function-full) Horn rules, could be built. On the other hand, extensions of $\mathcal{ALC}^u_{\mathbb{P}}$ towards higher OWL layers, e.g., $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, deserve more investigation w.r.t. corresponding DL tableaux-based algorithms that integrate general rules.

References

1. Grigoris Antoniou, Carlos Viegas Damasio, Benjamin Grosof, Ian Horrocks, Michael Kifer, Jan Maluszynski, and Peter F. Patel-Schneider. Combining Rules and Ontologies - A survey. Deliverables I3-D3, REWERSE, http://rewerse.net/deliverables/m12/i3-d3.pdf, March 2005.
2. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
3. Tim Berners-Lee. AAAI-06 Invited Talk: Artificial Intelligence and the Semantic Web. http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html.
4. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Conjunctive Query Containment in Description Logics with n-ary Relations. In *Proceedings of International Workshop on Description Logics*, volume 410 of *URA-CNRS*, 1997.
5. Jos de Bruijn, Thomas Eiter, Axel Polleres, and Hans Tompits. On Representational Issues About Combinations of Classical Theories with Nonmonotonic Rules. In *Proceedings of the 1st International Conference on Knowledge Science, Engineering and Management (KSEM)*, LNCS 4092, pages 1–22. Springer, 2006.
6. Jos de Bruijn, David Pearce, Axel Polleres, and Agustín Valverde. A Logic for Hybrid Rules. In *Proceedings of RuleML 2006 Workshop: Ontology and Rule Integration*, November 2006.

7. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. AL-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems (JIIS)*, 10(3):227–252, 1998.
8. Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. Effective Integration of Declarative Rules with External Evaluations for Semantic-Web Reasoning. In *Proceedings of 3rd European Semantic Web Conference (ESWC)*, LNCS 4011, pages 273–287. Springer, 2006.
9. Thomas Eiter, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. In *Proceedings of the 9th KR*, pages 141–151. AAAI Press, 2004.
10. Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In *Proceedings of the 5th International Conference and Symposium on Logic Programming (ICLP/SLP)*, pages 1070–1080. MIT Press, 1988.
11. Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description Logic Programs: Combining Logic Programs with Description Logic. In *Proceedings of the 12th International World Wide Web Conference*, pages 48–57. ACM, 2003.
12. Stijn Heymans. *Decidable Open Answer Set Programming*. Phd thesis, Theoretical Computer Science Lab (TINF), Vrije Universiteit Brussel, February 2006.
13. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. A Description Logic with Transitive and Converse Roles, Role Hierarchies and Qualifying Number Restrictions. LTCS-Report 99-08, RWTH Aachen, Germany, 1999.
14. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In *Proceedings of the 17th International Conference on Automated Deduction (CADE)*, LNCS 1831, pages 482–496. Springer, 2000.
15. Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing SHIQ-Description Logic to Disjunctive Datalog Programs. In *Proceedings of the 9th KR*, pages 152–162. AAAI Press, 2004.
16. Alon Y. Levy and Marie-Christine Rousset. Combining Horn Rules and Description Logics in CARIN. *Artifician Intelligence*, 104(1-2):165–209, 1998.
17. John W. Lloyd. *Foundations of Logic Programming (second, extended edition)*. Springer-Verlag, 1987.
18. Boris Motik, Ian Horrocks, Riccardo Rosati, and Ulrike Sattler. Can OWL and Logic Programming Live Together Happily Ever After? In *Proceedings of the 5th ISWC*, LNCS 4273, pages 501–514. Springer, 2006.
19. Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with rules. *Journal of Web Semantics*, 3(1):41–60, 2005.
20. Magdalena Ortiz, Diego Calvanese, and Thomas Eiter. Data complexity of Answering Unions of Conjunctive Queries in $\mathcal{SHIQ}$. Technical report, Faculty of Computer Science, Free University of Bozen-Bolzano, 2006.
21. Riccardo Rosati. On the Decidability and Complexity of Integrating Ontologies and Rules. *Journal of Web Semantics*, 3(1):61–73, 2005.
22. Riccardo Rosati. DL+log: Tight Integration of Description Logics and Disjunctive Datalog. In *Proceedings of the 10th KR*, pages 68–78. AAAI Press, 2006.
23. W3C. OWL: Web Ontology Language Semantics and Abstract Syntax. http://www.w3.org/TR/owl-absyn/.
24. W3C. Rule Interchange Format Working Group. http://www.w3.org/2005/rules/.
25. W3C. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. http://www.w3.org/Submission/SWRL/.