Knowl Inf Syst (2013) 34:335–371
DOI 10.1007/s10115-012-0478-9

REGULAR PAPER

Weight-based consistent query answering over inconsistent $\mathcal{SHIQ}$ knowledge bases

Jianfeng Du · Guilin Qi · Yi-Dong Shen

Received: 9 April 2009 / Revised: 18 October 2011 / Accepted: 27 February 2012 /
Published online: 15 March 2012

Abstract Non-standard query mechanisms that work under inconsistency are required in some important description logic (DL)-based applications, including those involving an inconsistent DL knowledge base (KB) whose intensional knowledge is consistent but is violated by its extensional knowledge. This paper proposes a *weight-based semantics* for querying such an inconsistent KB. This semantics defines an answer of a conjunctive query posed upon an inconsistent KB as a tuple of individuals whose substitution for the variables in the query head makes the query body entailed by any subbase of the KB consisting of the intensional knowledge and a *weight-maximally consistent* subset of the extensional knowledge. A novel computational method for this semantics is proposed, which works for *extensionally reduced* $\mathcal{SHIQ}$ KBs and conjunctive queries without *non-distinguished* variables. The method first compiles the given KB to a propositional program; then, for any given conjunctive query, it reduces the problem of computing all answers of the given query to a set of propositional satisfiability (SAT) problems with *PB-constraints*, which are then solved by SAT solvers. A decomposition-based framework for optimizing the method is also proposed. The feasibility of this method is demonstrated in our experiments.

Keywords Semantic Web · Description logics · Query answering · Inconsistency-tolerant reasoning · Weight-based semantics

J. Du (✉)
Guangdong University of Foreign Studies, Guangzhou 510006, China
e-mail: jfdu@mail.gdufs.edu.cn

J. Du · Y.-D. Shen
State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing 100190, China

G. Qi
School of Computer Science and Engineering, Southeast University, Nanjing 211189, China

G. Qi
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

1 Introduction

Description logics (DLs) [5] are mostly decidable fragments of first-order logic. They provide logical foundations for the highly evolving Semantic Web. In particular, among the three species of the standard Web Ontology Language (OWL) [40] for the Semantic Web, namely OWL Lite, OWL DL and OWL Full, the former two, respectively, correspond to DLs $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ [27]. A DL knowledge base (KB) consists of a TBox, an RBox and an ABox, where the TBox and the RBox store the intensional knowledge, while the ABox stores the extensional knowledge.

Logical contradictions can easily be introduced in a DL KB and make the KB *inconsistent* (i.e., have no models). The standard query mechanism in DLs returns anything that is satisfied by all models of an inconsistent KB, thus is meaningless. To solve this problem, one may repair the KB to render it consistent whenever some logical contradictions appear. But this approach is no panacea and some important applications, such as virtual data integration [45] and ontology population [14], require non-standard query mechanisms that work under inconsistency.

Data integration has been recognized as a crucial application in the Semantic Web [46]. In the scenario of virtual data integration [45], data distributed in different sources are mapped to an integrated view in a mediator. Since the data in different sources may often be heterogeneous, they are usually standardized using the terminology in the mediator. From the DL point of view, the knowledge stored in the mediator can be treated as the union of a TBox and an RBox, while the data in different sources as a whole can be treated as an ABox. However, the ABox may violate the constraints specified in the union of the TBox and the RBox, and it can hardly be repaired because its update reflects updates of the original data distributed in multiple sources. Hence, in this scenario, non-standard query mechanisms that work under inconsistency are needed.

In the scenario of ontology population [14], a DL KB is enriched by adding instance assertions. There are many research results on ontology population. To name a few, KIM [41] and Text2Onto [15] are frameworks that integrate algorithms for ontology population from textual data, including information extraction algorithms (e.g., [23]) that assign annotations carrying some semantics to regions of the data, as well as co-reference algorithms (e.g., [49]) that identify annotated individuals in multiple places. From the DL point of view, the course of information extraction adds concept or role assertions, whereas the course of co-reference adds (in)equality assertions. The populated DL KB, however, may become inconsistent due to potential conflicts between the new data and the original data. Directly repairing a populated DL KB may not be appropriate since choosing the best repair plan among a set of candidates may require massive human efforts. We had better use non-standard query mechanisms to directly work on the populated DL KB.

To provide a reasonable query mechanism that works under inconsistency, we propose a *weight-based semantics* for query answering over an inconsistent DL KB. We assume that the union of the TBox and the RBox of the KB is consistent, and every instance assertion in the ABox of the KB has a positive weight. We simply call a subset of the ABox *consistent* if it is consistent with the union of the TBox and the RBox and further call it *weight-maximally consistent* if the total weight of assertions in it is maximal among all consistent subsets of the ABox. The weight-based semantics defines an answer of a conjunctive query as a tuple of individuals whose substitution for the variables in the query head makes the query body entailed by any subbase of the given KB that consists of the TBox, the RBox and a weight-maximally consistent subset of the ABox. In a number of applications, such as virtual data integration and ontology population, the TBox and the RBox are fixed and well prepared,

so the union of the TBox and RBox is consistent. Weights for instance assertions can be obtained from different sources. For example, in Web-based scenarios, an instance assertion may redundantly exist in multiple data sources, so the redundancy of an assertion can be treated as the weight of the assertion.

We consider the problem of computing all answers to a conjunctive query under the weight-based semantics. We call this problem *weight-based consistent query answering* (*CQA*) problem and focus on an important class of it. For this class, the given KB is an *extensionally reduced* $\mathcal{SHIQ}$ KB and the given query is a conjunctive query without *non-distinguished* variables. $\mathcal{SHIQ}$ [28] is an expressive DL corresponding to OWL DL without nominals or datatypes. Since DL KBs can be easily extensionally reduced (see Sect. 2.1 for details), we simplify the problem by allowing only extensionally reduced KBs, in which the concepts in all concept assertions are atomic or negated atomic. Non-distinguished variables in a conjunctive query are existentially quantified variables. Disallowing these variables is commonly accepted. Many off-the-shelf DL query engines, such as Racer [26], KAON2 [30,31] and SHER [17], do not support conjunctive queries with non-distinguished variables.

We prove that the weight-based CQA problem for an *atomic query*, namely a query with a single atom and no variables, is $\Delta_2^p[O(\log n)]$-hard in *data complexity*, which is the complexity measured by the size of the ABox only. That is, the problem needs at least $O(\log n)$ calls to an NP oracle, where n is the number of instance assertions in the KB (also called the *size* of the ABox) and the input size of a call to the NP oracle is polynomial in n. We propose a two-phase method to solve the problem. The first phase is performed offline. It compiles the given $\mathcal{SHIQ}$ KB to a propositional program in time polynomial in n, the number of instance assertions in the KB, and then computes the maximum total weight of instance assertions in a consistent subbase of the KB by $O(\log n)$ calls to a SAT solver. The second phase is performed online. Given a conjunctive query, it reduces the weight-based CQA problem for this query into a set of weight-based CQA problems for atomic queries by using the output of the first phase and solves each of them by one call to a SAT solver. Hence, the proposed method solves the weight-based CQA problem for an atomic query by $O(\log n)$ calls to a SAT solver. Considering that a SAT solver is an NP oracle and that the weight-based CQA problem for an atomic query is $\Delta_2^p[O(\log n)]$-hard, the proposed method is actually time complexity optimal.

Since SAT solvers can hardly handle large propositional programs due to computational resource (such as memory) limitations, the proposed method may not work with large KBs. Hence, we also propose a decomposition-based framework to optimize it. The framework adds extra processes to both phases of the proposed method. In the first phase, the compiled propositional program is further decomposed into disjoint subsets. In the second phase, the weight-based CQA problem for an atomic query is solved by reducing to an extended SAT problem with *PB-constraints* [19], where the input of the SAT problem is a query-relevant propositional program extracted from some disjoint subsets obtained in the first phase. Since a query-relevant propositional program can be by orders of magnitude smaller than the compiled propositional program, the optimization can significantly improve the scalability.

We implemented the optimized method and conducted experiments on large KBs. All test KBs are obtained from publicly available ones by inserting *conflicts*, where a conflict is a set of instance assertions violating a functional role restriction or a disjointness constraint. Experimental results show that the decomposition-based framework is highly effective and the optimized method is promising in querying large KBs containing hundreds of conflicts and millions of instance assertions.

The remainder of this paper is organized as follows. After providing preliminaries in the next section, in Sect. 3, we present the weight-based semantics for consistent query answer-

ing (CQA) and analyze its computational complexity. In Sect. 4, we describe the proposed method for solving the weight-based CQA problem as well as the decomposition-based framework. In Sect. 5, we present our experimental evaluation. Before concluding the paper, we discuss related work in Sect. 6.

2 Preliminaries

In this section, we first introduce the description logic (DL) $\mathcal{SHIQ}$, conjunctive queries and disjunctive datalog. Afterwards, we introduce the KAON2 transformation method for reducing $\mathcal{SHIQ}$ to disjunctive datalog [30,31] and a well-known method for axiomatizing equality [24], both of which are fundamental to our proposed method.

2.1 Description logic $\mathcal{SHIQ}$

Thanks to some desirable properties, such as readability and decidability, DLs [5] have become logical foundations of the Semantic Web [27]. Among all DLs, $\mathcal{SHIQ}$ [28] is an expressive DL underlying the standard language OWL [40] in the Semantic Web. $\mathcal{SHIQ}$ has an expressive power comparable to OWL DL, with the exception that nominals and datatypes are disallowed and qualifying number restrictions are allowed. We use the same syntax of $\mathcal{SHIQ}$ as given in [30,31].

Let N_R be a set of role names. A $\mathcal{SHIQ}$ *role* is either some $r \in N_R$ (atomic role) or an *inverse role* r^- for $r \in N_R$. Let $\mathsf{Inv}(r) = r^-$ and $\mathsf{Inv}(r^-) = r$ for $r \in N_R$. A $\mathcal{SHIQ}$ *RBox* $\mathcal{R}$ is a finite set of *transitivity axioms* $\mathsf{Trans}(r)$ and *role inclusion axioms* $r \sqsubseteq s$, such that $r \sqsubseteq s \in \mathcal{R}$ implies $\mathsf{Inv}(r) \sqsubseteq \mathsf{Inv}(s) \in \mathcal{R}$ and $\mathsf{Trans}(r) \in \mathcal{R}$ implies $\mathsf{Trans}(\mathsf{Inv}(r)) \in \mathcal{R}$, for roles r and s. Let $\sqsubseteq^*$ denote the reflexive-transitive closure of $\sqsubseteq$. A role r is said to be *transitive* if $\mathsf{Trans}(s) \in \mathcal{R}$ for some s with $s \sqsubseteq^* r$ and $r \sqsubseteq^* s$. r is said to be *simple* if there is no transitive role s such that $s \sqsubseteq^* r$. r is said to be *complex* if it is not simple.

Let N_C be a set of concept names. The set of $\mathcal{SHIQ}$ concepts is the smallest set recursively defined as follows. Each $A \in N_C$ (atomic concept) is a $\mathcal{SHIQ}$ concept. For $\mathcal{SHIQ}$ concepts C and D, a role r, a simple role s and a non-negative integer n, the following concepts are also $\mathcal{SHIQ}$ concepts: $\top$ (top concept), $\bot$ (bottom concept), $\neg C$ (negation), $C \sqcap D$ (conjunction), $C \sqcup D$ (disjunction), $\exists r.C$ (existential restriction), $\forall r.C$ (value restriction), $\leq_n s.C$ and $\geq_n s.C$ (qualifying number restrictions). A $\mathcal{SHIQ}$ *TBox* $\mathcal{T}$ is a finite set of *concept inclusion axioms* $C \sqsubseteq D$, where C and D are $\mathcal{SHIQ}$ concepts.

Let N_I be a set of individual names. A $\mathcal{SHIQ}$ *ABox* $\mathcal{A}$ is a finite set of *instance assertions*, including *concept assertions* $C(a)$, *role assertions* $r(a, b)$ and $\neg s(a, b)$, *equality assertions* $a \approx b$ and *inequality assertions* $a \not\approx b$, where C is a $\mathcal{SHIQ}$ concept, r a role, s a simple role, and a and b individual names in N_I. The sets N_R, N_C and N_I are mutually disjoint.

A $\mathcal{SHIQ}$ knowledge base KB is a triple $(\mathcal{T}, \mathcal{R}, \mathcal{A})$, where $\mathcal{T}$ is a TBox, $\mathcal{R}$ an RBox and $\mathcal{A}$ an ABox.

An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty set $\Delta^{\mathcal{I}}$, called the domain of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$ that maps every concept name A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, every role name r to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and every individual name a to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation is extended to roles by defining $(r^-)^{\mathcal{I}} = \{(x, y) \mid (y, x) \in r^{\mathcal{I}}\}$ and to $\mathcal{SHIQ}$ concepts according to the left part of Table 1, where $|S|$ denotes the cardinality of a set S. An axiom/assertion α is said to be satisfied by $\mathcal{I}$, denoted by $\mathcal{I} \models \alpha$, if the corresponding condition given in the right part of Table 1 holds. An interpretation $\mathcal{I}$ is called a *model* of

Table 1 The syntax and semantics of $\mathcal{SHIQ}$

Constructors	Semantics	Axioms	Conditions
$\top$	$\Delta^{\mathcal{I}}$	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
$\bot$	$\emptyset$	$\mathsf{Trans}(r)$	$(r^{\mathcal{I}})^{+} = r^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	$r \sqsubseteq s$	$r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	Assertions	Conditions
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x, y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$	$r(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$
$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x, y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$	$\neg s(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin s^{\mathcal{I}}$
$\leq_n s.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \in C^{\mathcal{I}} \mid (x, y) \in s^{\mathcal{I}}\}\rvert \leq n\}$	$a \approx b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
$\geq_n s.C$	$\{x \in \Delta^{\mathcal{I}} \mid \lvert\{y \in C^{\mathcal{I}} \mid (x, y) \in s^{\mathcal{I}}\}\rvert \geq n\}$	$a \not\approx b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$

a $\mathcal{SHIQ}$ knowledge base KB, denoted by $\mathcal{I} \models KB$, if all axioms and assertions in KB are satisfied by $\mathcal{I}$. KB is said to be *consistent* if it admits at least one model. The union of the TBox $\mathcal{T}$ and the RBox $\mathcal{R}$ of KB is said to be *consistent* if $(\mathcal{T}, \mathcal{R}, \emptyset)$ is consistent. A subset S of the ABox of KB is said to be *consistent with* the union of $\mathcal{T}$ and $\mathcal{R}$ if $(\mathcal{T}, \mathcal{R}, S)$ is consistent. An axiom/assertion α is said to be *entailed* by KB, denoted by $KB \models \alpha$, if $\mathcal{I} \models \alpha$ for all models $\mathcal{I}$ of KB.

This work considers *extensionally reduced* $\mathcal{SHIQ}$ KBs only. A KB is said to be extensionally reduced if the concepts in all concept assertions are atomic or negated atomic, that is, all concept assertions are of the form $A(a)$ or $\neg A(a)$, where A is an atomic concept. A non-extensionally reduced $\mathcal{SHIQ}$ KB can be converted to an extensionally reduced one by the following steps: for each concept assertion $C(a) \in \mathcal{A}$ such that C is neither atomic nor negated atomic, introduce a new atomic concept Q_C, add $Q_C \sqsubseteq C$ to $\mathcal{T}$ and replace $C(a)$ with $Q_C(a)$ in $\mathcal{A}$. The models of the original KB coincide with those of the converted one on all concept names, role names and individual names in the original KB. Most of the publicly available OWL KBs, including those used in our experiments, are extensionally reduced.

2.2 Conjunctive query in DLs

We use letters (possibly with subscripts) x, y, z for variables and a, b, c for individual names or constants. Since this work treats a $\mathcal{SHIQ}$ KB as a disjunctive datalog program and individual names as constants, the terms "individual" and "constant" are used exchangeably in the remainder of this paper.

A *conjunctive query* is an expression of the form $\exists \vec{y}.\mathsf{conj}(\vec{x}, \vec{y}, \vec{c})$, where $\vec{x}$ is a vector of *distinguished variables*, $\vec{y}$ a vector of *non-distinguished variables* and $\vec{c}$ a vector of individuals. $\mathsf{conj}(\vec{x}, \vec{y}, \vec{c})$ denotes a conjunction of atoms of the form $A(v)$ or $r(v_1, v_2)$, where A is a concept name, r is a role name and v, v_1 and v_2 are variables in $\vec{x}$ and $\vec{y}$ or individuals in $\vec{c}$. A *concept name query* is a conjunctive query consisting of a single atom $A(x)$, where A is a concept name and x is a variable. A *Boolean conjunctive query* is a conjunctive query without distinguished variables. An *atomic query* is a conjunctive query with a single atom and no variables. It is clear that atomic queries are also Boolean conjunctive queries.

Given a consistent $\mathcal{SHIQ}$ knowledge base KB and a Boolean conjunctive query $Q = \exists \vec{y}.\mathsf{conj}(\vec{y}, \vec{c})$, an interpretation $\mathcal{I}$ of KB is said to *satisfy* Q if there exists a tuple of elements

in $\Delta^{\mathcal{I}}$ whose substitution for the variables in $\vec{y}$ makes every atom in $\mathsf{conj}(\vec{y}, \vec{c})$ satisfied by $\mathcal{I}$. Q is said to be *entailed* by KB, denoted by $KB \models Q$, if every model of KB satisfies Q. A tuple $\vec{t}$ of individuals is called an *answer* of a conjunctive query $Q(\vec{x}) = \exists \vec{y}.\mathsf{conj}(\vec{x}, \vec{y}, \vec{c})$ in KB if $KB \models Q(\vec{x})[\vec{x} \mapsto \vec{t}]$, where $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$ denotes a Boolean conjunctive query obtained from $Q(\vec{x})$ by replacing every variable in $\vec{x}$ with its corresponding individual in $\vec{t}$.

2.3 Disjunctive datalog

Disjunctive datalog [21] is an extension of Datalog, a query and rule language for deductive databases, in which disjunctions may appear in rule heads. Atoms and rules are elements of disjunctive datalog. An *atom* is of the form $T(v_1, \ldots, v_n)$, where T is a predicate and the arguments $v_1, \ldots, v_n$ are variables or constants. An *equational atom* is an atom composed of the equality predicate $\approx$ and two arguments v_1 and v_2, written as $v_1 \approx v_2$. A *rule* is of the form $\alpha_1 \vee \ldots \vee \alpha_n \leftarrow \beta_1, \ldots, \beta_m$, where α_i and β_i are atoms, possibly equational atoms; $\alpha_1, \ldots, \alpha_n$ constitute the *head* of the rule and they are called *head atoms*; $\beta_1, \ldots, \beta_m$ constitute the *body* of the rule and they are called *body atoms*. The set of head atoms of a rule R is denoted by $\mathsf{head}(R)$, whereas the set of body atoms of R is denoted by $\mathsf{body}(R)$. A rule R is called a *constraint* if $|\mathsf{head}(R)| = 0$; called a *fact* if $|\mathsf{body}(R)| = 0$; called *definite* if $|\mathsf{head}(R)| = 1$. A fact $\alpha_1 \vee \ldots \vee \alpha_n \leftarrow$ is simply written as $\alpha_1 \vee \ldots \vee \alpha_n$. A rule is said to be *safe* if every variable occurring in a head atom also occurs in some body atom.

A *disjunctive datalog program* [21] is a finite set of safe rules. A *disjunctive datalog program with equality* is a disjunctive datalog program in which some head atoms are equational atoms. An atom or a rule is *ground* if it has no variables. A *ground instance* of an atom α (resp. a rule R) is a ground atom (resp. a ground rule) obtained from α (resp. R) by replacing all variables with constants. Given a disjunctive datalog program with equality P, the set of all ground instances of atoms in P obtained by replacing all variables with constants occurring in P is called the *Herbrand base* of P, denoted by $\mathsf{HB}(P)$. The set of all ground instances of rules in P obtained by replacing all variables with constants occurring in P is called the *primary grounding* of P, denoted by $\mathcal{G}_p(P)$.

An *interpretation* M of P is a subset of $\mathsf{HB}(P)$. M is called a *model* of P if (i) $\mathsf{body}(r) \subseteq M$ implies $\mathsf{head}(r) \cap M \neq \emptyset$ for every ground rule $r \in \mathcal{G}_p(P)$, and (ii) the equality predicate $\approx$ can be interpreted as a congruence relation in M, that is, $\approx$ is *reflexive* (i.e., $a \approx a \in M$ for all constants a occurring in M), *symmetric* (i.e., $a \approx b \in M$ implies $b \approx a \in M$) and *transitive* (i.e., $a \approx b \in M$ and $b \approx c \in M$ imply $a \approx c \in M$), and $T(a_1, \ldots, a_i, \ldots, a_n) \in M$ and $a_i \approx b_i \in M$ imply $T(a_1, \ldots, b_i, \ldots, a_n) \in M$ for every predicate T occurring in P. P is said to be *satisfiable* if it admits at least one model. A model M of P is called *minimal* if none of the proper subsets of M is a model of P. A ground atom α is said to be *entailed* by P, denoted by $P \models \alpha$, if α is in all models of P. Note that α is in all models of P iff it is in all minimal models of P.

2.4 The KAON2 transformation method

An extensionally reduced $\mathcal{SHIQ}$ knowledge base $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ can be reduced to a disjunctive datalog problem with equality by applying the KAON2 transformation method [30,31]. The method consists of six steps.

In step 1, every transitivity axiom $\mathsf{Trans}(s) \in \mathcal{R}$ is removed and concept inclusion axioms of the form $\forall r.C \sqsubseteq \forall s.(\forall s.C)$ are added to $\mathcal{T}$, for all roles r such that $s \sqsubseteq^* r$ and all concepts C appearing in KB. This step is the standard method for eliminating transitivity axioms.

In step 2, $\mathcal{T} \cup \mathcal{R}$ is translated into a set of first-order clauses, using standard transformation methods from first-order logic. This step involves eliminating existential quantifiers by Skolemization and may introduce function symbols.

In step 3, the set of clauses obtained in step 2 is *saturated* by adding non-redundant logical consequences. This step takes up to exponential time w.r.t. the size of $\mathcal{T} \cup \mathcal{R}$. For an arbitrary atom in the saturated set of non-redundant clauses, its arguments can be variables or *functional terms* of the form $f(x)$, where f is a function symbol introduced in step 2.

In step 4, any functional term $f(x)$ occurring in the resulting set of clauses in step 3 is rewritten to a new variable x_f. The resulting set of clauses is then syntactically transformed to a set of disjunctive datalog rules. To make the resulting rules safe, auxiliary atoms of the form $HU(x)$, $HU(x_f)$ or $S_f(x, x_f)$ are added to rule bodies if necessary. For example, the rule $B(x_f) \leftarrow A(x), S_f(x, x_f)$ is rewritten from $B(f(x)) \leftarrow A(x)$, and the rule $A(x) \vee B(x) \leftarrow HU(x)$ is rewritten from $A(x) \vee B(x)$. We denote the set of rules computed in this step by $\Gamma(\mathcal{T} \cup \mathcal{R})$, which has no functional terms.

In step 5, a set of ground facts of the form $HU(a)$, $HU(a_f)$ or $S_f(a, a_f)$ is constructed, which are instantiated for all individual names a occurring in $\mathcal{A}$ and all function symbols f introduced in step 2. We denote this set by $\Delta(KB)$.

In the last step, $\mathcal{A}$ is directly translated to a set of ground facts or ground constraints. More precisely, instance assertions of the form $A(a)$ (resp. $r(a, b)$ or $a \approx b$) are translated to ground facts $A(a)$ (resp. $r(a, b)$ or $a \approx b$), while instance assertions of the form $\neg A(a)$ (resp. $\neg s(a, b)$ or $a \not\approx b$) are translated to ground constraints $\leftarrow A(a)$ (resp. $\leftarrow s(a, b)$ or $\leftarrow a \approx b$). We denote this set by $\Xi(\mathcal{A})$.

Let $\mathsf{DD}(KB)$ be defined as $\Gamma(\mathcal{T} \cup \mathcal{R}) \cup \Xi(\mathcal{A}) \cup \Delta(KB)$. We have the following theorem.

Theorem 2.1 ([31]) *Let KB be an extensionally reduced $\mathcal{SHIQ}$ KB, then KB is consistent iff* $\mathsf{DD}(KB)$ *is satisfiable.*

When data complexity is concerned, the size of $\mathcal{T} \cup \mathcal{R}$ is fixed and can be treated as a constant, so the number of rules in $\mathsf{DD}(KB)$ is at most polynomial in the size of $\mathcal{A}$, and $\mathsf{DD}(KB)$ is computed in time at most polynomial in the size of $\mathcal{A}$.

Example 2.1 This example is about an inconsistent knowledge base $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, where $\mathcal{T} = \{A \sqsubseteq \exists r.A \sqcap \leq_1 r.\top\}$, $\mathcal{R} = \emptyset$ and $\mathcal{A} = \{A(a), r(a, b), \neg A(b), a \not\approx b\}$. By applying the KAON2 transformation method to KB, $\Gamma(\mathcal{T} \cup \mathcal{R})$ consists of non-ground rules R_1, R_2 and R_3, $\Xi(\mathcal{A})$ consists of ground rules $R_4, \ldots, R_7$, and $\Delta(KB)$ consists of ground facts R_8 and R_9, where $R_1, \ldots, R_9$ are given below. We have $\mathsf{DD}(KB) = \Gamma(\mathcal{T} \cup \mathcal{R}) \cup \Xi(\mathcal{A}) \cup \Delta(KB) = \{R_1, \ldots, R_9\}$.

$R_1\colon\ r(x, x_f) \leftarrow A(x), S_f(x, x_f).$ $\quad R_2\colon\ A(x_f) \leftarrow A(x), S_f(x, x_f).$

$R_3\colon\ y_1 \approx y_2 \leftarrow A(x), r(x, y_1), r(x, y_2).$ $\quad R_4\colon\ A(a).$

$R_5\colon\ r(a, b).$ $\quad R_6\colon \leftarrow A(b).$ $\quad R_7\colon \leftarrow a \approx b.$ $\quad R_8\colon\ S_f(a, a_f).$ $\quad R_9\colon\ S_f(b, b_f).$

2.5 Equality axiomatization

A propositional program Π is a finite set of ground rules. By $\mathsf{atoms}(\Pi)$, we denote the set of ground atoms occurring in Π. A *model* M of Π is a subset of $\mathsf{atoms}(\Pi)$ such that for every rule $r \in \Pi$, $\mathsf{body}(r) \subseteq M$ implies $\mathsf{head}(r) \cap M \neq \emptyset$. This semantics for propositional programs is widely adopted in existing SAT solvers, but it does not distinguish the equality predicate $\approx$ from ordinary predicates. Hence, for a disjunctive datalog program with equality P, the propositional program $\mathcal{G}_p(P)$, namely the primary grounding of P, has not the same set of models as P has. To remove this semantical difference, we transform P to a disjunctive

datalog program without equality by using the well-known method for axiomatizing equality [24], described below.

Let $\pi(P)$ denote the logic program obtained from P by replacing the equality predicate $\approx$ with a new ordinary predicate eq, and $P_{\approx}$ denote the logic program consisting of the following rules.

$$\mathrm{eq}(a, a). \quad \text{(for every constant } a \text{ occurring in } P) \tag{1}$$

$$\mathrm{eq}(y, x) \leftarrow \mathrm{eq}(x, y). \tag{2}$$

$$\mathrm{eq}(x, z) \leftarrow \mathrm{eq}(x, y), \mathrm{eq}(y, z). \tag{3}$$

$$T(x_1, \ldots, y_i, \ldots, x_n) \leftarrow T(x_1, \ldots, x_i, \ldots, x_n), \mathrm{eq}(x_i, y_i).$$
$$\text{(for every predicate } T \text{ other than } \approx \text{ in } P \text{ and every position } i \text{ in } T) \tag{4}$$

The group of rules (1) ensures that eq is reflexive. Rule (2) ensures that eq is symmetric. Rule (3) ensures that eq is transitive. The group of rules (4) ensures that for every model M of $\pi(P)$ and every predicate T other than $\approx$ occurring in P, $T(a_1, \ldots, a_i, \ldots, a_n) \in M$ and $\mathrm{eq}(a_i, b_i) \in M$ imply $T(a_1, \ldots, b_i, \ldots, a_n) \in M$. It is clear that M is a model of P iff M is an interpretation of $\pi(P) \cup P_{\approx}$ such that $\mathsf{body}(r) \subseteq M$ implies $\mathsf{head}(r) \cap M \neq \emptyset$ for all rules $r \in \mathcal{G}_p(\pi(P) \cup P_{\approx})$. $\pi(P) \cup P_{\approx}$ is the disjunctive datalog program without equality obtained from P by axiomatizing equality.

3 A weight-based semantics for consistent query answering

The standard query mechanism for a $\mathcal{SHIQ}$ KB does not give meaningful answers to conjunctive queries when the KB is inconsistent, because the KB has no models and can thus entail any axiom. To provide a reasonable query mechanism for inconsistent $\mathcal{SHIQ}$ KBs, we intend to adapt the notion of *consistent query answering* (CQA) in the database field, which was first proposed by [1]. A database is *consistent* if it satisfies all integrity constraints specified over its schema, or *inconsistent* otherwise. The original CQA problem in the database field computes *consistent answers* of a given query that are satisfied by all *repairs* of the original database, where a repair is a consistent database that shares the schema of the original one, but differs from the latter by a subset-minimal (simply minimal) set of tuples. Computational methods and complexity results for the original CQA problem have been extensively studied [9,13]. To provide a less skeptical query mechanism, the cardinality-based CQA problem was also proposed by [2] , which computes consistent answers of a given query that are satisfied by all *C-repairs* of the original database, where a C-repair is a consistent database that shares the schema of the original one, but differs from the latter by a cardinality-minimal set of tuples. C-repairs are also repairs because cardinality-minimality is a special case of subset-minimality. The cardinality-based CQA problem has also been well studied [3,34].

The notion of repair or C-repair in the database field seems applicable to the DL context by regarding the schema together with integrity constraints as the union of the TBox and the RBox, the database as the ABox, and a repair or a C-repair as a model of the union of the TBox and the RBox that is minimally different from the ABox. However, a direct application does not work because there is generally not a closed domain for models of the union of the TBox and the RBox, but repairs or C-repairs in the database field are defined in closed domains. To adapt the notion of repair or C-repair to DLs, one may consider finite subsets of models of the union of the TBox and the RBox that are minimally different from the ABox. Based on this idea, [33] proposed an adaption of repairs to the DL context. They define a

repair of $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, where the union of $\mathcal{T}$ and $\mathcal{R}$ is consistent, as an interpretation $\mathcal{I}$ such that (1) $\mathcal{I} \models (\mathcal{T}, \mathcal{R}, \emptyset)$, and (2) there is not any other interpretation $\mathcal{I}'$ such that $\mathcal{I}' \models (\mathcal{T}, \mathcal{R}, \emptyset)$ and $\{\alpha \in \mathcal{A} \mid \mathcal{I} \models \alpha\} \subset \{\alpha \in \mathcal{A} \mid \mathcal{I}' \models \alpha\}$. Based on this definition, a tuple $\vec{t}$ of individuals is called a *consistent answer* of a conjunctive query $Q(\vec{x}) = \exists \vec{y}.\mathsf{conj}(\vec{x}, \vec{y}, \vec{c})$ in KB if every repair of KB satisfies $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$. We call this semantics *inclusion-based semantics*. Under this semantics, a tuple $\vec{t}$ of individuals is a consistent answer of a conjunctive query $Q(\vec{x})$ in KB iff every maximally consistent subbase (simply called *MC-subbase*) of KB satisfies $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$, where a *subbase* $KB' = (\mathcal{T}', \mathcal{R}', \mathcal{A}')$ of KB is a $\mathcal{SHIQ}$ KB such that $\mathcal{T}' = \mathcal{T}$, $\mathcal{R}' = \mathcal{R}$ and $\mathcal{A}' \subseteq \mathcal{A}$, and a *consistent subbase* of KB is a subbase of KB that is consistent.

Inclusion-based semantics may be too skeptical for Web-based scenarios where there are often inconsistent data. For example, consider $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ where $\mathcal{T} = \{\mathsf{Journal} \sqsubseteq \neg\mathsf{Conference}\}$, $\mathcal{R} = \emptyset$ and $\mathcal{A} = \{\mathsf{Conference}(\mathsf{ISWC}), \mathsf{Journal}(\mathsf{ISWC})\}$. There are two MC-subbases of KB, one does not contain the instance assertion $\mathsf{Conference}(\mathsf{ISWC})$, the other does not contain the instance assertion $\mathsf{Journal}(\mathsf{ISWC})$. Hence, neither $\mathsf{Conference}(\mathsf{ISWC})$ nor $\mathsf{Journal}(\mathsf{ISWC})$ can be inferred from KB under the inclusion-based semantics. At the same time, there is a well-known phenomenon in the World Wide Web called *information redundancy*, which says that true information is likely to redundantly exist. By considering information redundancy, we can define an interesting semantics that allows us to compute relevant answers. Consider the above example again. If we know that the instance assertion $\mathsf{Conference}(\mathsf{ISWC})$ holds in more data sources than the instance assertion $\mathsf{Journal}(\mathsf{ISWC})$, we would like to be able to infer that ISWC is a conference.

In this paper, we will exploit information redundancy to provide instance assertions with positive weights. One possible way to acquire these weights is to extract them from the output of an information extraction algorithm proposed, for example, in [37,47]. These algorithms exploit information redundancy to extract instance assertions and output instance assertions with weights that are directly proportional to the redundancy of the assertions. We do not focus on how weights are acquired but on how weights are exploited to deduce information from an inconsistent $\mathcal{SHIQ}$ KB. To this end, we propose the following notion of WOC-subbase of KBs.

Definition 3.1 (*WOC-Subbase*) For KB, a possibly inconsistent $\mathcal{SHIQ}$ KB where each instance assertion $\alpha \in KB_{\mathcal{A}}$ is given a positive weight $w(\alpha)$, a *weight-optimal consistent subbase* (*WOC-subbase* for short) KB' of KB is a consistent subbase of KB such that for all consistent subbases KB'' of KB, $\sum_{\alpha \in KB''_{\mathcal{A}}} w(\alpha) \leq \sum_{\alpha \in KB'_{\mathcal{A}}} w(\alpha)$.

A WOC-subbase of an inconsistent $\mathcal{SHIQ}$ KB can be viewed as a consistent subbase of the KB that has the most redundant information. To formalize a query mechanism based on WOC-subbases, we extend the notion of repair in the DL context [33] and call an interpretation of a WOC-subbase of KB a *W-repair* of KB, which is formally defined below.

Definition 3.2 (*W-Repair*) For $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, a possibly inconsistent $\mathcal{SHIQ}$ KB where $\mathcal{T} \cup \mathcal{R}$ is consistent and each instance assertion $\alpha \in \mathcal{A}$ is given a positive weight $w(\alpha)$, an interpretation $\mathcal{I}$ is called a *W-repair* of KB if (1) $\mathcal{I} \models (\mathcal{T}, \mathcal{R}, \emptyset)$, and (2) $\sum_{\alpha \in \mathcal{A}, \mathcal{I}' \models \alpha} w(\alpha) \leq \sum_{\alpha \in \mathcal{A}, \mathcal{I} \models \alpha} w(\alpha)$ for all interpretations $\mathcal{I}'$ such that $\mathcal{I}' \models (\mathcal{T}, \mathcal{R}, \emptyset)$.

The *weight-based semantics* for CQA is defined accordingly. A Boolean conjunctive query $Q = \exists \vec{y}.\mathsf{conj}(\vec{y}, \vec{c})$ is said to be *consistently entailed* by KB, denoted by $KB \overset{\mathrm{w}}{\approx} Q$, if every W-repair of KB satisfies Q. A tuple $\vec{t}$ of individuals is called a *consistent answer* of a conjunctive query $Q(\vec{x}) = \exists \vec{y}.\mathsf{conj}(\vec{x}, \vec{y}, \vec{c})$ in KB if $KB \overset{\mathrm{w}}{\approx} Q(\vec{x})[\vec{x} \mapsto \vec{t}]$. The problem of

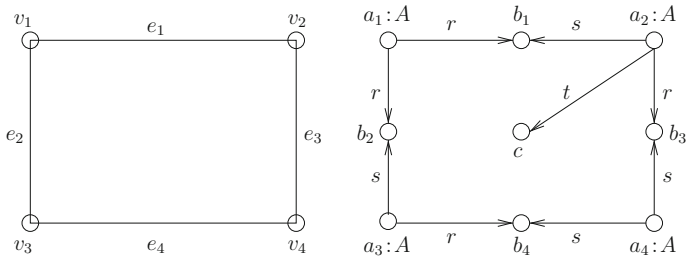

Fig. 1 The graph representation of the ABox of KB (*right*) constructed from a graph G (*left*) in the proof of Theorem 3.1, where o is set as 2, $a_i : A$ denotes a concept assertion $A(a_i)$ and $a_i \xrightarrow{r'} a'$ denotes a role assertion $r'(a_i, a')$ for $a' \in \{b_1, b_2, b_3, b_4, c\}$ and $r' \in \{r, s, t\}$

computing all consistent answers of a conjunctive query Q in KB is called the *weight-based CQA problem* for Q. When Q is a Boolean conjunctive query, the weight-based CQA problem for Q amounts to checking if Q is consistently entailed by KB, so it is also called the *entailment checking problem* for Q. The following lemma characterizes the weight-based semantics for CQA.

Lemma 3.1 *For KB, a possibly inconsistent $\mathcal{SHIQ}$ KB and Q a Boolean conjunctive query, $KB \overset{\text{w}}{\approx} Q$ iff $KB' \models Q$ for all WOC-subbases KB' of KB.*

Proof This lemma trivially holds since any model of a WOC-subbase of KB is a W-repair of KB, while any W-repair of KB is a model of some WOC-subbase of KB. □

In this paper, we assume that all positive weights of instance assertions have been scaled to positive integers by the same factor. Scaling weights by the same factor does not change the set of WOC-subbases of KB; thus, it does not change the solutions of weight-based CQA problems in KB.

Below we prove that under the weight-based semantics, the entailment checking problem for an atomic query in extensionally reduced $\mathcal{SHIQ}$ knowledge bases is $\Delta_2^p[O(\log n)]$-hard in data complexity, that is, the problem needs at least $O(\log n)$ calls to an NP oracle, where n is the number of instance assertions in the ABox and the input size of a call to the NP oracle is polynomial in n.

Theorem 3.1 *For $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, an extensionally reduced $\mathcal{SHIQ}$ KB, checking if $KB \overset{\text{w}}{\approx} A(a)$, where A is an atomic concept, is $\Delta_2^p[O(\log n)]$-hard in data complexity.*

Proof We show this by a reduction from the following $\Delta_2^p[O(\log n)]$-hard problem [34]: given an undirected graph $G = (V, E)$, decide if a vertex $v \in V$ belongs to every maximum independent set of G. An *independent set* S of G is a set of vertices in V that are not connected to each other by edges. S is called *maximum* if there is not an independent set S' of G such that $|S'| < |S|$. For example, a graph G is shown in the left part of Fig. 1; both $\{v_1, v_4\}$ and $\{v_2, v_3\}$ are maximum independent sets of G.

Given an undirected graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$, and an integer $o \in \{1, \ldots, n\}$, we construct a $\mathcal{SHIQ}$ knowledge base $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, where $\mathcal{T} = \{A \sqsubseteq \forall r.B, A \sqsubseteq \forall s.C, B \sqcap C \sqsubseteq \bot, A \sqsubseteq \forall t.W\}$, $\mathcal{R} = \emptyset$ and $\mathcal{A} = \{A(a_i) \mid 1 \le i \le n\} \cup \{r(a_i, b_k), s(a_j, b_k) \mid 1 \le k \le m, (v_i, v_j) = e_k \in E, i < j\} \cup \{t(a_o, c)\}$, and

where the weights of role assertions are all $n+1$ and the weights of concept assertions are all one. For example, an ABox is shown in the right part of Fig. 1, which is constructed from the graph G in the left part of Fig. 1 by setting $o = 2$.

It can be seen that KB is inconsistent, but it becomes consistent after all concept assertions $A(a_i)$ are removed from $\mathcal{A}$. This implies that every W-repair of KB satisfies all role assertions in $\mathcal{A}$. It can be seen that, if $\mathcal{I}$ is W-repair of KB, then $\{v_i \in V \mid \mathcal{I} \models A(a_i)\}$ is a maximum independent set of G; if S_V is a maximum independent set of G, then the interpretation $\mathcal{I}$, such that $a_i^{\mathcal{I}} = a_i$ for all $1 \le i \le n$, $b_k^{\mathcal{I}} = b_k$ for all $1 \le k \le m$, $c^{\mathcal{I}} = c$, $A^{\mathcal{I}} = \{a_i \mid v_i \in S_V\}$, $B^{\mathcal{I}} = \{b_k \mid v_i \in S_V, r(a_i, b_k) \in \mathcal{A}\}$, $C^{\mathcal{I}} = \{b_k \mid v_i \in S_V, s(a_i, b_k) \in \mathcal{A}\}$, $r^{\mathcal{I}} = \{(a_i, b_k) \mid 1 \le k \le m, (v_i, v_j) = e_k \in E, i < j\}$, $s^{\mathcal{I}} = \{(a_j, b_k) \mid 1 \le k \le m, (v_i, v_j) = e_k \in E, i < j\}$, $t^{\mathcal{I}} = \{(a_o, c)\}$, and $W^{\mathcal{I}} = \{c\}$ if $v_o \in S_V$ or $W^{\mathcal{I}} = \emptyset$ otherwise, is a W-repair of KB. For example, in Fig. 1, $\{v_2, v_3\}$ is a maximum independent set of G; correspondingly, the interpretation $\mathcal{I}$ such that $a_i^{\mathcal{I}} = a_i$ for all $1 \le i \le 4$, $b_k^{\mathcal{I}} = b_k$ for all $1 \le k \le 4$, $c^{\mathcal{I}} = c$, $A^{\mathcal{I}} = \{a_2, a_3\}$, $B^{\mathcal{I}} = \{b_3, b_4\}$, $C^{\mathcal{I}} = \{b_1, b_2\}$, $r^{\mathcal{I}} = \{(a_1, b_1), (a_1, b_2), (a_2, b_3), (a_3, b_4)\}$, $s^{\mathcal{I}} = \{(a_2, b_1), (a_3, b_2), (a_4, b_3), (a_4, b_4)\}$, $t^{\mathcal{I}} = \{(a_2, c)\}$ and $W^{\mathcal{I}} = \{c\}$ is a W-repair of KB.

We show that v_o belongs to every maximum independent set of G iff $KB \overset{\text{w}}{\approx} W(c)$. ($\Rightarrow$) For every W-repair $\mathcal{I}$ of KB, let $S_V = \{v_i \mid \mathcal{I} \models A(a_i)\}$, then S_V is a maximum independent set of G. Since v_o belongs to every maximum independent set of G, we have $v_o \in S_V$. It follows that $\mathcal{I} \models A(a_o)$. Since $\mathcal{I} \models t(a_o, c)$ and $\mathcal{I} \models A \sqsubseteq \forall t.W$, we have $\mathcal{I} \models W(c)$. It follows that $KB \overset{\text{w}}{\approx} W(c)$. ($\Leftarrow$) For every maximum independent set S_V of G, let $\mathcal{I}$ be an interpretation such that $a_i^{\mathcal{I}} = a_i$ for all $1 \le i \le n$, $b_k^{\mathcal{I}} = b_k$ for all $1 \le k \le m$, $c^{\mathcal{I}} = c$, $A^{\mathcal{I}} = \{a_i \mid v_i \in S_V\}$, $B^{\mathcal{I}} = \{b_k \mid v_i \in S_V, r(a_i, b_k) \in \mathcal{A}\}$, $C^{\mathcal{I}} = \{b_k \mid v_i \in S_V, s(a_i, b_k) \in \mathcal{A}\}$, $r^{\mathcal{I}} = \{(a_i, b_k) \mid 1 \le k \le m, (v_i, v_j) = e_k \in E, i < j\}$, $s^{\mathcal{I}} = \{(a_j, b_k) \mid 1 \le k \le m, (v_i, v_j) = e_k \in E, i < j\}$, $t^{\mathcal{I}} = \{(a_o, c)\}$, and $W^{\mathcal{I}} = \{c\}$ if $v_o \in S_V$ or $W^{\mathcal{I}} = \emptyset$ otherwise, then $\mathcal{I}$ is a W-repair of KB. Since $KB \overset{\text{w}}{\approx} W(c)$, we have $\mathcal{I} \models W(c)$. It follows that $v_o \in S_V$, that is, v_o belongs to every maximum independent set of G.

Based on the above conclusion and the facts that KB is constructible from G in polynomial time and $\mathcal{T}$ has a constant size, the theorem follows. □

4 Computational methods for the weight-based semantics

Lemma 3.1 shows that the problem of computing all consistent answers of a conjunctive query in KB can be solved by considering all WOC-subbases of KB. A straightforward method for computing all consistent answers of a conjunctive query in KB is to first compute all WOC-subbases of KB, then compute the set of answers of the query in each WOC-subbase of KB, and finally return the intersection of these sets of answers. However, this method is hard to be applied in practice because computing a WOC-subbase of KB is already NP-hard in data complexity for many DLs, which are less expressive than $\mathcal{SHIQ}$ [18]; moreover, there can be exponentially many WOC-subbases of KB w.r.t. the number of instance assertions, as shown in the following example.

Example 4.1 Let $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, where $\mathcal{T} = \{A \sqcap B \sqsubseteq \bot\}$, $\mathcal{R} = \emptyset$, $\mathcal{A} = \{A(a_i), B(a_i) \mid 1 \le i \le n\}$, and every instance assertion in $\mathcal{A}$ is given the same positive weight. We can easily check that a subbase of KB is a WOC-subbase of KB iff it is of the form $\{C_i(a_i) \mid 1 \le i \le n\}$, where C_i is either A or B. Hence, the number of WOC-subbases of KB is 2^n.

In order to evaluate a conjunctive query in a practical way, we propose a novel method that need not compute any WOC-subbase of KB beforehand. The method works under the

restrictions that the given KB is an extensionally reduced $\mathcal{SHIQ}$ KB and the given conjunctive query has no non-distinguished variables. As mentioned in Sect. 1, these restrictions are not severe in practice.

The basic idea of the proposed method for evaluating a conjunctive query is to reduce the original problem to multiple entailment checking problems for Boolean conjunctive queries without variables, then further reduce latter problems to multiple entailment checking problems for atomic queries, and finally solve the reduced problems by applying SAT solvers. Given a conjunctive query without non-distinguished variables $Q(\vec{x}) = \mathsf{conj}(\vec{x}, \vec{c})$, all consistent answers of $Q(\vec{x})$ in KB can be computed by checking if $KB \overset{\text{w}}{\approx} Q(\vec{x})[\vec{x} \mapsto \vec{t}]$ for all possible tuples $\vec{t}$ of individuals, so the weight-based CQA problem for $Q(\vec{x})$ can be reduced to multiple entailment checking problems for Boolean conjunctive queries without variables. On the other hand, the entailment checking problem for a Boolean conjunctive query without variables can be reduced to multiple entailment checking problems for atomic queries, as shown in the following lemma.

Lemma 4.1 *For KB, a possibly inconsistent $\mathcal{SHIQ}$ KB and $Q = \alpha_1 \wedge \ldots \wedge \alpha_n$ a Boolean conjunctive query without variables, $KB \overset{\text{w}}{\approx} Q$ iff $KB \overset{\text{w}}{\approx} \alpha_i$ for all $1 \leq i \leq n$.*

Proof ($\Rightarrow$) When $KB \overset{\text{w}}{\approx} Q, \mathcal{I} \models \alpha_1 \wedge \ldots \wedge \alpha_n$ for all W-repairs $\mathcal{I}$ of KB. Thus, for any $1 \leq i \leq n, \mathcal{I} \models \alpha_i$ for all W-repairs $\mathcal{I}$ of KB, that is, $KB \overset{\text{w}}{\approx} \alpha_i$. ($\Leftarrow$) When $KB \overset{\text{w}}{\approx} \alpha_i$ for all $1 \leq i \leq n, \mathcal{I} \models \alpha_i$ for all W-repairs $\mathcal{I}$ of KB. Thus, $\mathcal{I} \models \alpha_1 \wedge \ldots \wedge \alpha_n$ for all W-repairs $\mathcal{I}$ of KB, that is, $KB \overset{\text{w}}{\approx} Q$. □

In the following Sects. 4.1 and 4.2, we address the entailment checking problem for an atomic query. Then, in Sect. 4.3, we address the problem of computing all consistent answers of a conjunctive query.

4.1 Entailment checking of atomic queries

The key idea for solving the entailment checking problem for an atomic query is to reduce the problem to an extended SAT problem with *pseudo-Boolean constraint* (*PB-constraint*). A PB-constraint [19] is a non-standard rule of the form $\sum_i c_i x_i \leq d$ with integer constants c_i, d and variables $x_i \in \{0, 1\}$. An extended SAT problem with PB-constraints can be either solved by standard SAT solvers after translating PB-constraints to standard SAT clauses [6,19] or solved by special SAT solvers, such as `GALENA` [12] and `PUEBLO` [48], which support PB-constraints natively.

The basic method without optimizations works in two phases.

The first phase (see Fig. 2a) is performed offline. In this phase, $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ is first transformed to a disjunctive datalog program $\mathsf{DD}(KB)$ using the KAON2 transformation method [30,31] (see Sect. 2.4), then to a *repair program* $\mathsf{RP}(KB)$ by adding a *decision atom* $\hbar_\alpha$ to the rule head of every ground rule translated from an instance assertion α in $\mathcal{A}$ (see Definition 4.1). There is a main result about the relationship between KB and $\mathsf{RP}(KB)$. That is, given an atomic query q whose only atom is not on complex roles, the problem of checking if $KB \overset{\text{w}}{\approx} q$ can be reduced to an extended SAT problem with PB-constraints, namely $\mathsf{RP}(KB) \cup \{\sum_{\alpha \in \mathcal{A}} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_{\min}\} \cup \{\leftarrow q\}$ (see Corollary 4.1), where $\mathsf{assign}(\hbar_\alpha)$ denotes the 0–1 truth value of the decision atom $\hbar_\alpha$ (0 for false and 1 for true) and $w_{\min}$ is the minimum value v such that $\mathsf{RP}(KB) \cup \{\sum_{\alpha \in \mathcal{A}} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq v\}$ is satisfiable.

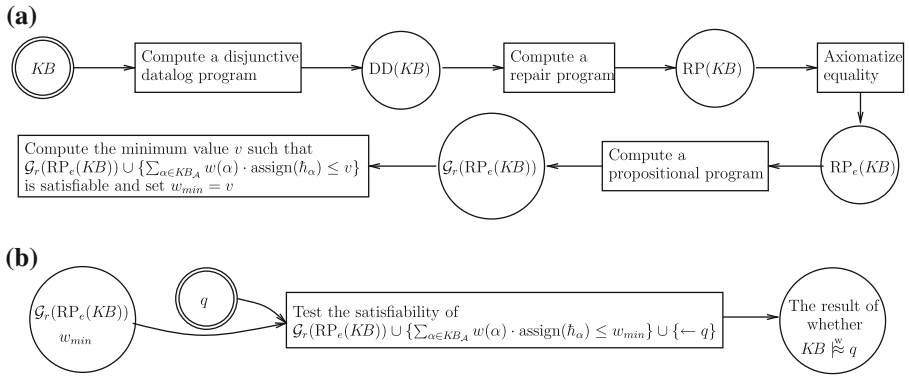

Fig. 2 The basic method for entailment checking of atomic queries. **a** The offline phase, **b** the online phase

Existing SAT solvers that support PB-constraints only work on propositional programs. In order to apply these solvers to tackle the extended SAT problem reduced from the entailment checking problem, we first transform $\mathsf{RP}(KB)$ to a disjunctive datalog program without equality $\mathsf{RP}_e(KB)$ by axiomatizing equality (see Sect. 2.5), then compile $\mathsf{RP}_e(KB)$ to a propositional program using the intelligent grounding technique [22]). Then, $w_{\min}$ is equal to the minimum value v such that $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\sum_{\alpha\in\mathcal{A}} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq v\}$ is satisfiable (see Theorem 4.2 (1)).

The second phase (see Fig. 2b) is performed online, that is, the computations are done for every given atomic query q. In this phase, the satisfiability of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\sum_{\alpha\in\mathcal{A}} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_{\min}\} \cup \{\leftarrow q\}$ is checked by applying a SAT solver that supports PB-constraints. The unsatisfiability of the above SAT problem implies $KB \overset{\text{w}}{\mid\!\approx} q$, when the only atom in q is not on complex roles (see Theorem 4.2 (2)).

In the following, more details of the basic method are provided. The notion of repair program is given below, and an example for repair programs is shown in Example 4.2.

Definition 4.1 (*Repair Program*) The *repair program* of KB, written as $\mathsf{RP}(KB)$, is a disjunctive datalog program with equality, obtained from $\mathsf{DD}(KB)$ as follows: for every instance assertion $\alpha \in \mathcal{A}$, introduce a corresponding nullary *decision atom* $\hbar_\alpha$ and add it as a head atom to the rule translated from α in $\mathsf{DD}(KB)$.

Example 4.2 (Example 2.1 continued) Consider KB given in Example 2.1, suppose the weights of instance assertions in $\mathcal{A}$ are, respectively, $w(A(a)) = 2, w(r(a,b)) = 1, w(\neg A(b)) = 1$ and $w(a \not\approx b) = 1$. By adding decision atoms to rule heads of ground rules translated from instance assertions in $\mathcal{A}$, we obtain the repair program of KB, namely $\mathsf{RP}(KB) = \{R_1, \ldots, R_3, R'_4, \ldots, R'_7, R_8, R_9\}$, where R_1, R_2, R_3, R_8 and R_9 are given in Example 2.1 and $R'_4, \ldots, R'_7$ are given below.

R'_4: $\hbar_{A(a)} \vee A(a)$. $\quad R'_5$: $\hbar_{r(a,b)} \vee r(a,b)$.

R'_6: $\hbar_{\neg A(b)} \leftarrow A(b)$. $\quad R'_7$: $\hbar_{a\not\approx b} \leftarrow a \approx b$.

The following lemma shows a relationship between consistent subsets of KB and the repair program $\mathsf{RP}(KB)$.

Lemma 4.2 *Let $X = \{\hbar_\alpha \mid \alpha \in \mathcal{A}\}$ be the set of decision atoms occurring in $\mathsf{RP}(KB)$. For any subset S of $\mathcal{A}$, $(\mathcal{T}, \mathcal{R}, S)$ is a consistent subbase of KB iff $\mathsf{RP}(KB)$ has a model M such that $M \cap X = \{\hbar_\alpha \mid \alpha \in \mathcal{A} \setminus S\}$.*

Proof Let $KB' = (\mathcal{T}, \mathcal{R}, S)$, then KB' is obviously a subbase of KB. Let ϕ_X denote an assignment on X and $\mathsf{RP}(KB) \downarrow \phi_X$ denote the disjunctive datalog program obtained from $\mathsf{RP}(KB)$ by deleting all rules that have decision atoms $\hbar_\alpha$ such that $\phi_X(\hbar_\alpha) =$ true and by removing all decision atoms from remaining rules.

($\Rightarrow$) When KB' is consistent, by Theorem 2.1, $\mathsf{DD}(KB')$ is satisfiable. For the assignment ϕ_X on X such that $\phi_X(\hbar_\alpha) =$ false for all $\alpha \in S$ and $\phi_X(\hbar_\alpha) =$ true for all $\alpha \in \mathcal{A} \setminus S$, clearly $\mathsf{RP}(KB) \downarrow \phi_X$ is a superset of $\mathsf{DD}(KB')$ and the difference set between $\mathsf{RP}(KB) \downarrow \phi_X$ and $\mathsf{DD}(KB')$ consists of only ground facts of the form $HU(a)$, $HU(a_f)$ or $S_f(a, a_f)$, where a occurs in KB but not in KB'. Hence, $\mathsf{RP}(KB) \downarrow \phi_X$ is satisfiable and admits a model M. It follows that $M' = M \cup \{\hbar_\alpha \in X \mid \phi_X(\hbar_\alpha) = \text{true}\}$ is a model of $\mathsf{RP}(KB)$ such that $M' \cap X = \{\hbar_\alpha \mid \alpha \in \mathcal{A} \setminus S\}$.

($\Leftarrow$) When $\mathsf{RP}(KB)$ has a model M such that $M \cap X = \{\hbar_\alpha \mid \alpha \in \mathcal{A} \setminus S\}$, $\mathsf{RP}(KB)$ is satisfiable under the assignment ϕ_X on X such that $\phi_X(\hbar_\alpha) =$ false for all $\alpha \in S$ and $\phi_X(\hbar_\alpha) =$ true for all $\alpha \in \mathcal{A} \setminus S$, that is, $\mathsf{RP}(KB) \downarrow \phi_X$ is satisfiable. Since $\mathsf{RP}(KB) \downarrow \phi_X$ is a superset of $\mathsf{DD}(KB')$, $\mathsf{DD}(KB')$ is also satisfiable. By Theorem 2.1, KB' is consistent. □

Lemma 4.2 implies that there should be some correspondence between WOC-subbases of KB and models of $\mathsf{RP}(KB)$. Before presenting this correspondence, we introduce the following definition.

Definition 4.2 ((X, w)-*Minimal Model*) For P, a disjunctive datalog program with equality, X a set of ground atoms occurring in P and $w : X \to \mathbb{R}$ a weight function, where $\mathbb{R}$ is the domain of real numbers. A (X, w)-minimal model M of P is a model of P such that for all models M' of P, $\sum_{\beta \in M \cap X} w(\beta) \leq \sum_{\beta \in M' \cap X} w(\beta)$.

In the remainder of this section, by X, we denote the set of decision atoms occurring in $\mathsf{RP}(KB)$ and by w' we denote a weight function defined over X such that $w'(\hbar_\alpha) = w(\alpha)$ for all $\alpha \in \mathcal{A}$. Lemma 4.2 actually implies a correspondence between WOC-subbases of KB and (X, w')-minimal models of $\mathsf{RP}(KB)$, as shown in the following lemma.

Lemma 4.3 (1) *For M, a (X, w')-minimal model of $\mathsf{RP}(KB)$, $(\mathcal{T}, \mathcal{R}, \{\alpha \mid \hbar_\alpha \in X \setminus M\})$ is a WOC-subbase of KB.* (2) *For $(\mathcal{T}, \mathcal{R}, S)$, a WOC-subbase of KB, there exists a (X, w')-minimal model M of $\mathsf{RP}(KB)$ such that $M \cap X = \{\hbar_\alpha \mid \alpha \in \mathcal{A} \setminus S\}$.*

Proof (1) Since M is a (X, w')-minimal model of $\mathsf{RP}(KB)$, by Lemma 4.2, $(\mathcal{T}, \mathcal{R}, S)$ is consistent where $S = \{\alpha \mid \hbar_\alpha \in X \setminus M\}$. Suppose $(\mathcal{T}, \mathcal{R}, S)$ is not a WOC-subbase of KB, then there exists a consistent subbase $(\mathcal{T}, \mathcal{R}, S')$ of KB such that $\sum_{\alpha \in S'} w(\alpha) > \sum_{\alpha \in S} w(\alpha)$. By Lemma 4.2, $\mathsf{RP}(KB)$ has a model M' such that $M' \cap X = \{\hbar_\alpha \mid \alpha \in \mathcal{A} \setminus S'\}$. But $\sum_{\hbar_\alpha \in M' \cap X} w'(\hbar_\alpha) = \sum_{\alpha \in \mathcal{A} \setminus S'} w(\alpha) < \sum_{\alpha \in \mathcal{A} \setminus S} w(\alpha) = \sum_{\hbar_\alpha \in M \cap X} w'(\hbar_\alpha)$, contradicting that M is a (X, w')-minimal model of $\mathsf{RP}(KB)$. Hence, $(\mathcal{T}, \mathcal{R}, S)$ is a WOC-subbase of KB.

(2) Since $(\mathcal{T}, \mathcal{R}, S)$ is a WOC-subbase of KB, by Lemma 4.2, $\mathsf{RP}(KB)$ has a model M such that $M \cap X = \{\hbar_\alpha \mid \alpha \in KB_{\mathcal{A}} \setminus S\}$. Suppose M is not a (X, w')-minimal model of $\mathsf{RP}(KB)$, then there exists a model M' of $\mathsf{RP}(KB)$ such that $\sum_{\beta \in M' \cap X} w'(\beta) < \sum_{\beta \in M \cap X} w'(\beta)$. Let $S' = \{\alpha \mid \hbar_\alpha \in X \setminus M'\}$, then by Lemma 4.2, $(\mathcal{T}, \mathcal{R}, S')$ is a consistent subbase of KB. But $\sum_{\alpha \in S} w(\alpha) = \sum_{\hbar_\alpha \in X \setminus M} w' hbar_\alpha) < \sum_{\hbar_\alpha \in X \setminus M'} w'(\hbar_\alpha) = \sum_{\alpha \in S'} w(\alpha)$, contradicting that $(\mathcal{T}, \mathcal{R}, S)$ is a WOC-subbase. Hence, M is a (X, w')-minimal model of $\mathsf{RP}(KB)$. □

The following theorem shows that, by considering all (X, w')-minimal models of $\mathsf{RP}(KB)$, we can solve the entailment checking problem for an atomic query.

Theorem 4.1 *For q, an atomic query whose only atom is not on complex roles, $KB \overset{\mathrm{w}}{\approx} q$ iff q is in all (X, w')-minimal models of $\mathsf{RP}(KB)$.*

Proof Let $KB' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{\neg q\})$, $X' = X \cup \{\hbar_{\neg q}\}$, and w'' be a weight function defined over X' such that $w''(\hbar_\alpha) = w'(\hbar_\alpha)$ for all $\hbar_\alpha \in X$ and $w''(\hbar_{\neg q}) = 1$.

($\Rightarrow$) Suppose $KB \overset{\mathrm{w}}{\approx} q$ but $q \notin M$ for some (X, w')-minimal model M of $\mathsf{RP}(KB)$. Note that $\mathsf{RP}(KB) \subseteq \mathsf{RP}(KB')$ and $\mathsf{RP}(KB') \setminus \mathsf{RP}(KB)$ only consists of a ground rule $\hbar_{\neg q} \leftarrow q$ and ground facts of the form $HU(a)$, $HU(a_f)$ or $S_f(a, a_f)$ where a occurs in q but not in KB, so there exists a (X', w'')-minimal model M' of $\mathsf{RP}(KB')$ such that $M' \setminus M$ consists of only ground facts of the form $HU(a)$, $HU(a_f)$ or $S_f(a, a_f)$ where a occurs in q but not in KB. Let $S = \{\alpha \mid \hbar_\alpha \in X' \setminus M'\}$. By Lemma 4.3, $(\mathcal{T}, \mathcal{R}, S)$ is a WOC-subbase of KB'. Since $q \notin M$, we have $q \notin M'$. It follows that $\hbar_{\neg q} \notin M'$ and thus $\neg q \in S$. Therefore, $(\mathcal{T}, \mathcal{R}, S \setminus \{\neg q\})$ is a WOC-subbase of KB. Since $KB \overset{\mathrm{w}}{\approx} q$, by Lemma 3.1, $(\mathcal{T}, \mathcal{R}, S \setminus \{\neg q\}) \models q$. But then $(\mathcal{T}, \mathcal{R}, S)$ is inconsistent, contradicting that it is a WOC-subbase of KB'. Hence, q is in all (X, w')-minimal models of $\mathsf{RP}(KB)$ when $KB \overset{\mathrm{w}}{\approx} q$.

($\Leftarrow$) Suppose q is in all (X, w')-minimal models of $\mathsf{RP}(KB)$ but $KB \overset{\mathrm{w}}{\not\approx} q$. By Lemma 3.1, there exists a subset S of $\mathcal{A}$ such that $(\mathcal{T}, \mathcal{R}, S)$ is a WOC-subbase of KB but $(\mathcal{T}, \mathcal{R}, S) \not\models q$. Let $S' = S \cup \{\neg q\}$, then $(\mathcal{T}, \mathcal{R}, S')$ is consistent. It follows that $(\mathcal{T}, \mathcal{R}, S')$ is a WOC-subbase of KB'. By Lemma 4.3, there exists a (X', w'')-minimal model M of $\mathsf{RP}(KB')$ such that $M \cap X' = \{\hbar_\alpha \mid \alpha \in (\mathcal{A} \cup \{\neg q\}) \setminus S'\} = \{\hbar_\alpha \mid \alpha \in \mathcal{A} \setminus S\}$. Thus, $\hbar_{\neg q} \notin M$ and $q \notin M$. Note that $\mathsf{RP}(KB) \subseteq \mathsf{RP}(KB')$ and $\mathsf{RP}(KB') \setminus \mathsf{RP}(KB)$ only consists of a ground rule $\hbar_{\neg q} \leftarrow q$ and ground facts of the form $HU(a)$, $HU(a_f)$ or $S_f(a, a_f)$ where a occurs in q but not in KB, so $M' = M \cap \mathsf{HB}(\mathsf{RP}(KB))$ is a (X, w')-minimal model of $\mathsf{RP}(KB)$. But then $q \notin M'$, contradicting that q is in all (X, w')-minimal models of $\mathsf{RP}(KB)$. Hence, $KB \overset{\mathrm{w}}{\approx} q$ when q is in all (X, w')-minimal models of $\mathsf{RP}(KB)$. □

The above theorem may not hold for an atomic query $r(a, b)$ where r is a complex role, because $KB' = (\mathcal{T}, \mathcal{R}, \mathcal{A} \cup \{\neg r(a, b)\})$ is not a $\mathcal{SHIQ}$ KB defined in Sect. 2.1 and KB' may not be equisatisfiable with $\mathsf{DD}(KB')$ [31].

We can see that for all (X, w')-minimal models M of $\mathsf{RP}(KB)$, the sum of weights $\sum_{\hbar_\alpha \in M \cap X} w'(\hbar_\alpha)$ is the same and is equal to the minimum value v such that $\mathsf{RP}(KB) \cup \{\sum_{\alpha \in \mathcal{A}} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq v\}$ is satisfiable. In the remainder of this section, we denote this minimum value by $w_{\min}$. We can see that a (X, w')-minimal model of $\mathsf{RP}(KB)$ is a model of $\mathsf{RP}(KB) \cup \{\sum_{\alpha \in \mathcal{A}} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_{\min}\}$, and vice versa. It follows that q is in all (X, w')-minimal models of $\mathsf{RP}(KB)$ iff q is in all models of $\mathsf{RP}(KB) \cup \{\sum_{\alpha \in \mathcal{A}} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_{\min}\}$. Hence, we have the following corollary from Theorem 4.1.

Corollary 4.1 *For q an atomic query whose only atom is not on complex roles, $KB \overset{\mathrm{w}}{\approx} q$ iff $\mathsf{RP}(KB) \cup \{\sum_{\alpha \in \mathcal{A}} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_{\min}\} \cup \{\leftarrow q\}$ is unsatisfiable.*

We intend to use SAT solvers that support PB-constraints to solve the extended SAT problem given in Corollary 4.1. Although most of these SAT solvers are highly optimized, they only work on propositional programs, so we need to ground $\mathsf{RP}(KB)$, that is, compile

$\mathsf{RP}(KB)$ to a semantically equivalent propositional program, before checking if $KB \overset{\text{w}}{\approx} q$. In propositional programs, the equality predicate $\approx$ is not distinguished from ordinary predicates, so we axiomatize equality in $\mathsf{RP}(KB)$ before grounding $\mathsf{RP}(KB)$. By $\mathsf{RP}_e(KB)$, we denote the disjunctive datalog program without equality obtained from $\mathsf{RP}(KB)$ by axiomatizing equality, that is, if the equality predicate $\approx$ occurs in some rule heads in $\mathsf{RP}(KB)$, $\mathsf{RP}_e(KB)$ is defined as $\pi(\mathsf{RP}(KB)) \cup \mathsf{RP}_{\approx}(KB)$ (see Sect. 2.5), or defined as $\mathsf{RP}(KB)$ otherwise.

The primary grounding of $\mathsf{RP}_e(KB)$, namely $\mathcal{G}_p(\mathsf{RP}_e(KB))$ (see Sect. 2.4), may be too large to be handled by SAT solvers. Hence, we apply the well-known intelligent grounding (IG) technique [22] to ground $\mathsf{RP}_e(KB)$, which only yields a subset of $\mathcal{G}_p(\mathsf{RP}_e(KB))$. Given a disjunctive datalog program without equality P, the IG technique essentially computes the least fixpoint of $\Pi^{(n)}$ such that $\Pi^{(0)} = \emptyset$ and for $n \geq 1$, $\Pi^{(n)} = \{R\sigma \mid R \in P,\ \sigma$ is a ground substitution such that $\mathsf{body}(R\sigma) \subseteq \mathsf{atoms}(\Pi^{(n-1)})\}$. The resulting least fixpoint, denoted by $\mathcal{G}_{IG}(P)$, consists of all relevant rules for preserving the semantics of P. That is, $\mathcal{G}_{IG}(P)$ has the same set of minimal models as P has [22]. Based on this semantic equivalence between $\mathsf{RP}_e(KB)$ and $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, we have the following theorem.

Theorem 4.2 (1) $w_{\min}$ *is equal to the minimum value* v *such that* $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\sum_{\alpha\in\mathcal{A}} w(\alpha)\cdot\mathsf{assign}(\hbar_\alpha) \leq v\}$ *is satisfiable.* (2) *For* q*, an atomic query whose only atom is not on complex roles,* $KB \overset{\text{w}}{\approx} q$ *iff* $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))\cup\{\sum_{\alpha\in\mathcal{A}} w(\alpha)\cdot\mathsf{assign}(\hbar_\alpha) \leq w_{\min}\}\cup\{\leftarrow q\}$ *is unsatisfiable.*

Proof (1) It suffices to show that for any v, $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))\cup\{\sum_{\alpha\in\mathcal{A}} w(\alpha)\cdot\mathsf{assign}(\hbar_\alpha) \leq v\}$ is satisfiable iff $\mathsf{RP}(KB) \cup \{\sum_{\alpha\in\mathcal{A}} w(\alpha)\cdot\mathsf{assign}(\hbar_\alpha) \leq v\}$ is satisfiable.

($\Rightarrow$) When $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))\cup\{\sum_{\alpha\in\mathcal{A}} w(\alpha)\cdot\mathsf{assign}(\hbar_\alpha) \leq v\}$ is satisfiable, $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ has a minimal model M such that $\sum_{\hbar_\alpha\in M\cap X} w'(\hbar_\alpha) \leq v$ holds. Since $\mathsf{RP}_e(KB)$ and $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ have the same set of minimal models, M is also a minimal model of $\mathsf{RP}_e(KB)$ such that $\sum_{\hbar_\alpha\in M\cap X} w'(\hbar_\alpha) \leq v$ holds. Hence, $\mathsf{RP}(KB) \cup \{\sum_{\alpha\in\mathcal{A}} w(\alpha)\cdot\mathsf{assign}(\hbar_\alpha) \leq v\}$ is satisfiable.

($\Leftarrow$) When $\mathsf{RP}(KB)\cup\{\sum_{\alpha\in\mathcal{A}} w(\alpha)\cdot\mathsf{assign}(\hbar_\alpha) \leq v\}$ is satisfiable, $\mathsf{RP}_e(KB)$ has a minimal model M such that $\sum_{\hbar_\alpha\in M\cap X} w'(\hbar_\alpha) \leq v$ holds. Since $\mathsf{RP}_e(KB)$ and $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ have the same set of minimal models, M is also a minimal model of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ such that $\sum_{\hbar_\alpha\in M\cap X} w'(\hbar_\alpha) \leq v$ holds. Hence, $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))\cup\{\sum_{\alpha\in\mathcal{A}} w(\alpha)\cdot\mathsf{assign}(\hbar_\alpha) \leq v\}$ is satisfiable.

(2) It can be proved analogously as (1) that $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))\cup\{\sum_{\alpha\in\mathcal{A}} w(\alpha)\cdot\mathsf{assign}(\hbar_\alpha) \leq w_{\min}\}\cup\{\leftarrow q\}$ is satisfiable iff $\mathsf{RP}(KB)\cup\{\sum_{\alpha\in\mathcal{A}} w(\alpha)\cdot\mathsf{assign}(\hbar_\alpha) \leq w_{\min}\}\cup\{\leftarrow q\}$ is satisfiable. By Corollary 4.1, we have $KB \overset{\text{w}}{\approx} q$ iff $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))\cup\{\sum_{\alpha\in\mathcal{A}} w(\alpha)\cdot\mathsf{assign}(\hbar_\alpha) \leq w_{\min}\} \cup \{\leftarrow q\}$ is unsatisfiable. □

Example 4.3 (Example 4.2 continued) Since the equality predicate $\approx$ occurs in some rule heads in $\mathsf{RP}(KB)$, we have $\mathsf{RP}_e(KB) = \pi(\mathsf{RP}(KB)) \cup \mathsf{RP}_{\approx}(KB)$, where $\pi(\mathsf{RP}(KB)) = \{R_1, \ldots, R_9\}$ and $\mathsf{RP}_{\approx}(KB) = \{R_{10}, \ldots, R_{20}\}$. The rules $R_1, \ldots, \mathcal{R}_{20}$ are given below.

$R_1\colon\ r(x, x_f) \leftarrow A(x), S_f(x, x_f).$ $\quad R_2\colon\ A(x_f) \leftarrow A(x), S_f(x, x_f).$

$R_3\colon\ \mathrm{eq}(y_1, y_2) \leftarrow A(x), r(x, y_1), r(x, y_2).$ $\quad R_4\colon\ \hbar_{A(a)} \vee A(a).$

$R_5\colon\ \hbar_{r(a,b)} \vee r(a, b).$ $\quad R_6\colon\ \hbar_{\neg A(b)} \leftarrow A(b).$ $\quad R_7\colon\ \hbar_{a\not\approx b} \leftarrow \mathrm{eq}(a, b).$

$R_8\colon\ S_f(a, a_f).$ $\quad R_9\colon\ S_f(b, b_f).$ $\quad R_{10}\colon\ \mathrm{eq}(a, a).$ $\quad R_{11}\colon\ \mathrm{eq}(b, b).$

$R_{12}\colon\ \mathrm{eq}(a_f, a_f).$ $\quad R_{13}\colon\ \mathrm{eq}(b_f, b_f).$ $\quad R_{14}\colon\ \mathrm{eq}(y, x) \leftarrow \mathrm{eq}(x, y).$

$R_{15}\colon\ \mathrm{eq}(x,z) \leftarrow \mathrm{eq}(x,y), \mathrm{eq}(y,z).$ $\quad R_{16}\colon\ A(y) \leftarrow A(x), \mathrm{eq}(x,y).$
$R_{17}\colon\ r(z,y) \leftarrow r(x,y), \mathrm{eq}(x,z).$ $\quad R_{18}\colon\ r(x,z) \leftarrow r(x,y), \mathrm{eq}(y,z).$
$R_{19}\colon\ S_f(z,y) \leftarrow S_f(x,y), \mathrm{eq}(x,z).$ $\quad R_{20}\colon\ S_f(x,z) \leftarrow S_f(x,y), \mathrm{eq}(y,z).$

By applying the IG technique and deleting instantiated rules that have body atoms of the form $\mathrm{eq}(c,c)$ or $\mathrm{eq}(c_f,c_f)$, we have $\Pi^{(0)} = \emptyset$, $\Pi^{(1)} = \{r_1,\ldots,r_{10}\}$, $\Pi^{(2)} = \{r_1,\ldots,r_{12}\}$, $\Pi^{(3)} = \{r_1,\ldots,r_{14}\}$, $\Pi^{(4)} = \{r_1,\ldots,r_{20}\}$, $\Pi^{(5)} = \{r_1,\ldots,r_{22}\}$, and $\Pi^{(6)} = \Pi^{(5)}$, so $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) = \Pi^{(5)} = \{r_1,\ldots,r_{22}\}$, where $r_1,\ldots,r_{22}$ are given below.

$r_1\colon\ \hbar_{A(a)} \vee A(a).$ $\quad r_2\colon\ \hbar_{r(a,b)} \vee r(a,b).$ $\quad r_3\colon\ \hbar_{\neg A(b)} \leftarrow A(b).$
$r_4\colon\ \hbar_{a\not\approx b} \leftarrow \mathrm{eq}(a,b).$ $\quad r_5\colon\ S_f(a,a_f).$ $\quad r_6\colon\ S_f(b,b_f).$ $\quad r_7\colon\ \mathrm{eq}(a,a).$
$r_8\colon\ \mathrm{eq}(b,b).$ $\quad r_9\colon\ \mathrm{eq}(a_f,a_f).$ $\quad r_{10}\colon\ \mathrm{eq}(b_f,b_f).$
$r_{11}\colon\ r(a,a_f) \leftarrow A(a), S_f(a,a_f).$ $\quad r_{12}\colon\ A(a_f) \leftarrow A(a), S_f(a,a_f).$
$r_{13}\colon\ \mathrm{eq}(b,a_f) \leftarrow A(a), r(a,b), r(a,a_f).$ $\quad r_{14}\colon\ \mathrm{eq}(a_f,b) \leftarrow A(a), r(a,a_f), r(a,b).$
$r_{15}\colon\ \mathrm{eq}(b,a_f) \leftarrow \mathrm{eq}(a_f,b).$ $\quad r_{16}\colon\ \mathrm{eq}(a_f,b) \leftarrow \mathrm{eq}(b,a_f).$
$r_{17}\colon\ A(b) \leftarrow A(a_f), \mathrm{eq}(a_f,b).$ $\quad r_{18}\colon\ r(a,b) \leftarrow r(a,a_f), \mathrm{eq}(a_f,b).$
$r_{19}\colon\ r(a,a_f) \leftarrow r(a,b), \mathrm{eq}(b,a_f).$ $\quad r_{20}\colon\ S_f(a,b) \leftarrow S_f(a,a_f), \mathrm{eq}(a_f,b).$
$r_{21}\colon\ A(a_f) \leftarrow A(b), \mathrm{eq}(b,a_f).$ $\quad r_{22}\colon\ S_f(a,a_f) \leftarrow S_f(a,b), \mathrm{eq}(b,a_f).$

According to Theorem 4.2, we have $w_{\min} = 1$ and, for example, $KB \overset{\mathrm{w}}{\models\!\!\approx} A(a)$, $KB \overset{\mathrm{w}}{\not\models\!\!\approx} A(b)$ and $KB \overset{\mathrm{w}}{\not\models\!\!\approx} r(a,b)$.

Consider the time complexity of the basic method for entailment checking of atomic queries in terms of data complexity. Let n be the number of instance assertions in $\mathcal{A}$. As analyzed in Sect. 2.4, the number of rules in $\mathsf{DD}(KB)$ is at most polynomial in n, and the number of variables occurring in any rule in $\mathsf{DD}(KB)$ is bounded by some constant. Hence, the number of rules in $\mathsf{RP}_e(KB)$ is also polynomial in n, and the number of variables occurring in any rule in $\mathsf{RP}_e(KB)$ is also bounded by some constant. It follows that the number of ground rules in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ is polynomial in n. Under the assumption that every instance assertion in $\mathcal{A}$ has a positive weight polynomial in n, by Theorem 4.2 (1), $w_{\min}$ can be computed by binary search for the minimum value v between 1 and $\sum_{\alpha\in\mathcal{A}} w(\alpha)$, making $O(\log n)$ calls to a SAT solver that supports PB-constraints. In addition, to check whether $KB \overset{\mathrm{w}}{\models\!\!\approx} q$ for an atomic query q, a further call to the SAT solver is needed. Since the input size of each call to the SAT solver is polynomial in n, checking if $KB \overset{\mathrm{w}}{\models\!\!\approx} q$ is in $\Delta_2^p[O(\log n)]$ in data complexity. This time complexity is guaranteed to be optimal by Theorem 3.1, which says that the problem of checking if $KB \overset{\mathrm{w}}{\models\!\!\approx} q$ is $\Delta_2^p[O(\log n)]$-hard in data complexity.

4.2 Optimizations in a decomposition-based framework

Although the basic method for entailment checking of atomic queries is time complexity optimal, it is still hard to work with large $\mathcal{SHIQ}$ KBs. This is because the method transforms KB to a propositional program $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, which is probably large and can hardly be handled by SAT solvers. To address this issue, we propose a decomposition-based framework to make easier the satisfiability tests required in Theorem 4.2.

The decomposition-based framework adds extra processes to both phases of the basic method. In the first phase, $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ is decomposed into a set of disjoint subsets, and then a set of weights whose sum is $w_{\min}$ is separately computed from these subsets. In the second phase, for every given atomic query q, a query-relevant extended SAT problem with PB-constraints is constructed from some disjoint subsets computed in the first phase. When the only atom in q is not on complex roles, this SAT problem is unsatisfiable iff $KB \overset{\mathrm{w}}{\models\!\!\approx} q$.

The rationality of the framework is that if we can compute many small disjoint subsets of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ and can construct a query-relevant SAT problem from a small number of these disjoint subsets, then we may efficiently solve the entailment checking problem for an atomic query no matter how large $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ is.

4.2.1 The key optimization in the first phase: decomposition

The aim of decomposing $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ is to compute a set of disjoint subsets of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ such that $w_{\min}$ can be separately computed from these subsets. In order to compute small disjoint subsets of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, we propose an optimized method that first removes some rules from $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ which are irrelevant to computing $w_{\min}$, then computes *maximal connected components* of the set of remaining rules in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$. The notion of *connected component*, formally defined in Definition 4.3, has been widely used in solving SAT problems since some seminal work such as [4]. Simply speaking, a connected component of a propositional program Π is a subset of Π such that any two clauses in it are connected via common ground atoms; a connected component is maximal if it has no proper subsets that are also connected components. Maximal connected components of Π are disjoint subsets of Π that preserve unsatisfiability since some maximal connected component of Π must be unsatisfiable when Π is unsatisfiable.

Definition 4.3 (*Connected Component*) Two ground rules r and r' are called *connected* in a propositional program Π if there exists a sequence of rules $r_0 = r, r_1, \ldots, r_n = r'$ in Π such that r_{i-1} and r_i have common ground atoms for any $1 \leq i \leq n$. A *connected component* Π_c of Π is a subset of Π such that any two rules r and r' in Π_c are connected in Π_c. Π_c is called *maximal* if there is no connected component Π'_c of Π such that $\Pi_c \subset \Pi'_c$.

The first step in the proposed method for decomposing $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ is to find a subset Π_0 of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ such that $w_{\min}$ can still be computed in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0$. We call a subset Π_0 of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ an *X-irrelevant* subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ if $w_{\min}$ is equal to the minimum value v such that $(\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0) \cup \{\sum_{\beta \in X'} w'(\beta) \cdot \mathsf{assign}(\beta) \leq v\}$ is satisfiable, where X' is the set of decision atoms occurring in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0$. It is clear that when Π_0 is an X-irrelevant subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, $w_{\min}$ can be computed in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0$. The notion of X-irrelevance is independent of any weight function $w' : X \to \mathbb{R}$.

In the following, an example is provided to motivate our proposed method for computing an X-irrelevant subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$.

Example 4.4 This example is about an inconsistent knowledge base $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, where $\mathcal{T} = \{A \sqcap \exists r.\top \sqsubseteq C, \top \sqsubseteq \forall s.A, \top \sqsubseteq \forall t.B, \exists r.C \sqsubseteq \bot\}, \mathcal{R} = \{s \sqsubseteq r, t \sqsubseteq r\}$, and $\mathcal{A} = \{A(a), C(b), C(c), s(a,b), t(a,c)\}$. We have $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) = \{r_1, \ldots, r_{13}\}$ shown below.

$r_1 : A(a) \vee \hbar_{A(a)}.$ $r_2 : C(b) \vee \hbar_{C(b)}.$ $r_3 : C(c) \vee \hbar_{C(c)}.$
$r_4 : s(a,b) \vee \hbar_{s(a,b)}.$ $r_5 : t(a,c) \vee \hbar_{t(a,c)}.$ $r_6 : C(a) \leftarrow A(a), r(a,b).$
$r_7 : C(a) \leftarrow A(a), r(a,c).$ $r_8 : A(b) \leftarrow s(a,b).$ $r_9 : B(c) \leftarrow t(a,c).$
$r_{10} : \leftarrow C(b), r(a,b).$ $r_{11} : \leftarrow C(c), r(a,c).$ $r_{12} : r(a,b) \leftarrow s(a,b).$
$r_{13} : r(a,c) \leftarrow t(a,c).$

Consider Example 4.4 for computing an X-irrelevant subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$. It can be seen that $A(a), C(a), A(b)$ and $B(c)$ occur in the heads of $r_1, r_6, \ldots, r_9$, but do not occur anywhere in rules $r_2, \ldots, r_5, r_{10}, \ldots, r_{13}$. Let $w'_{\min}$ be the minimum value v such that $\{r_2, \ldots,$

$r_5, r_{10}, \ldots, r_{13}\} \cup \{\sum_{\beta \in X'} w'(\beta) \cdot \mathsf{assign}(\beta) \leq v\}$ is satisfiable, where X' is the set of decision atoms occurring in $\{r_2, \ldots, r_5, r_{10}, \ldots, r_{13}\}$. Let M' be a model of $\{r_2, \ldots, r_5, r_{10}, \ldots, r_{13}\}$ such that $\sum_{\beta \in M' \cap X'} w'(\beta) \leq w'_{\min}$ holds, then $M' \cup \{A(a), C(a), A(b), B(c)\}$ is a model of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ such that $\sum_{\beta \in M' \cap X} w'(\beta) \leq w'_{\min}$ holds. By the minimality of $w_{\min}$, it follows that $w'_{\min} = w_{\min}$. Thus, $\{r_1, r_6, \ldots, r_9\}$ is an X-irrelevant subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$. By generalizing this example, we conclude that a subset S of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, such that every rule in S has at least one head atom not in X nor occurring in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus S$, is an X-irrelevant subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$. This conclusion is formally shown in the following lemma.

Lemma 4.4 *Let Π_0 be a subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ such that $\mathsf{head}(r) \not\subseteq X \cup \mathsf{atoms}(\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0)$ for all rules $r \in \Pi_0$. Let $\Pi = \mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0$ and $X' = \mathsf{atoms}(\Pi) \cap X$, then $w_{\min}$ is equal to the minimum value v such that $\Pi \cup \{\sum_{\beta \in X'} w'(\beta) \cdot \mathsf{assign}(\beta) \leq v\}$ is satisfiable.*

Proof Let $M_0 = \bigcup_{r \in \Pi_0} \mathsf{head}(r) \setminus (X \cup \mathsf{atoms}(\Pi))$, then $M_0 \cap \mathsf{head}(r) \neq \emptyset$ for all rules $r \in \Pi_0$. Hence, M_0 is a model of Π_0. Let $w'_{\min}$ be the minimum value v such that $\Pi \cup \{\sum_{\beta \in X'} w'(\beta) \cdot \mathsf{assign}(\beta) \leq v\}$ is satisfiable, and M' be a model of $\Pi \cup \{\sum_{\beta \in X'} w'(\beta) \cdot \mathsf{assign}(\beta) \leq w'_{\min}\}$. Since $M_0 \cap \mathsf{atoms}(\Pi) = \emptyset$ and $M' \subseteq \mathsf{atoms}(\Pi)$, $M_0 \cup M'$ is a model of $\Pi \cup \{\sum_{\beta \in X} w'(\beta) \cdot \mathsf{assign}(\beta) \leq w'_{\min}\}$. By the minimality of $w_{\min}$, that is, $w_{\min}$ is the minimum value v such that $\Pi \cup \{\sum_{\beta \in X} w'(\beta) \cdot \mathsf{assign}(\beta) \leq v\}$ is satisfiable, we have $w_{\min} \leq w'_{\min}$. Suppose $w_{\min} < w'_{\min}$, then for M a model of $\Pi \cup \{\sum_{\beta \in X} w'(\beta) \cdot \mathsf{assign}(\beta) \leq w_{\min}\}$, $M \cap \mathsf{atoms}(\Pi)$ is a model of Π and $\sum_{\beta \in M \cap X'} w'(\beta) < w'_{\min}$ holds, contradicting the minimality of $w'_{\min}$. Hence, $w_{\min} = w'_{\min}$. □

The last step of the proposed method for decomposing $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ is to compute the set of maximal connected components of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0$. The following lemma shows that $w_{\min}$ can be separately computed from maximal connected components of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0$.

Lemma 4.5 *Let Π_0 be a subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ such that $\mathsf{head}(r) \not\subseteq X \cup \mathsf{atoms}(\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0)$ for all rules $r \in \Pi_0$, $\{\Pi_1, \ldots, \Pi_m\}$ be the set of maximal connected components of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0$, and for $1 \leq i \leq m$, $X_i = X \cap \mathsf{atoms}(\Pi_i)$, w_i be the minimum value v such that $\Pi_i \cup \{\sum_{\beta \in X_i} w'(\beta) \cdot \mathsf{assign}(\beta) \leq v\}$ is satisfiable. Then, $w_{\min} = \sum_{i=1}^{m} w_i$.*

Proof Let $\Pi = \mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0$ and M_i be a model of $\Pi_i \cup \{\sum_{\beta \in X_i} w'(\beta) \cdot \mathsf{assign}(\beta) \leq w_i\}$ for $1 \leq i \leq m$. For any $1 \leq j < k \leq m$, since $\mathsf{atoms}(\Pi_j) \cap \mathsf{atoms}(\Pi_k) = \emptyset$, we have $M_j \cap M_k = \emptyset$. Thus, $\bigcup_{i=1}^{m} M_i$ is a model of $\Pi \cup \{\sum_{\beta \in X} w'(\beta) \cdot \mathsf{assign}(\beta) \leq \sum_{i=1}^{m} w_i\}$. By Lemma 4.4 and the minimality of $w_{\min}$, that is, $w_{\min}$ is the minimum value v such that $\Pi \cup \{\sum_{\beta \in X} w'(\beta) \cdot \mathsf{assign}(\beta) \leq v\}$ is satisfiable, we have $w_{\min} \leq \sum_{i=1}^{m} w_i$. Suppose $w_{\min} < \sum_{i=1}^{m} w_i$, then for M a model of $\Pi \cup \{\sum_{\beta \in X} w'(\beta) \cdot \mathsf{assign}(\beta) \leq w_{\min}\}$, there exists $k \in \{1, \ldots, m\}$ such that $M \cap \mathsf{atoms}(\Pi_k)$ is a model of Π_k and $\sum_{\beta \in M \cap X_k} w'(\beta) < w_k$ holds, contradicting the minimality of w_k. Hence, $w_{\min} = \sum_{i=1}^{m} w_i$. □

The maximal connected components of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0$ are usually smaller than the maximal connected components of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$. Consider Example 4.4, $\Pi_0 = \{r_1, r_6, \ldots, r_9\}$ is an X-irrelevant subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, and $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0$ has two maximal connected components, namely $\{r_2, r_4, r_{10}, r_{12}\}$ and $\{r_3, r_5, r_{11}, r_{13}\}$. But $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ has only one maximal connected component, namely itself.

Input A propositional program Π and a set X of ground atoms occurring in Π.
Output A set of subsets of Π.
1. **for** each ground atom α occurring in Π **do** $cid(\alpha) := 0$;
2. $k := 0$; $fixed :=$ false;
3. **while** $fixed =$ false **do**
4. $fixed :=$ true;
5. **for** each rule $r \in \Pi$ such that $\mathsf{head}(r) = \emptyset$ or $cid(\alpha) > 0$ for all $\alpha \in \mathsf{head}(r) \setminus X$ **do**
6. **for** each ground atom α occurring in r such that $cid(\alpha) = 0$ **do**
7. $k := k + 1$; $cid(\alpha) := k$;
8. **if** $|cid(r)| > 1$ **then**
9. $fixed :=$ false; $min_{id} := \min(cid(r))$;
10. **for** each ground atom α occurring in r **do** $cid(\alpha) := min_{id}$;
11. **for** $i := 1, \ldots, k$ **do** $\Pi_i := \{r \in \Pi \mid cid(r) = \{i\}\}$;
12. **return** $(\{\Pi \setminus \bigcup_{i=1}^{k} \Pi_i\}, \{\Pi_i \neq \emptyset \mid 1 \leq i \leq k\})$;

Fig. 3 An algorithm for decomposing Π into disjoint subsets based on X

We develop an efficient algorithm, shown in Fig. 3, for computing an X-irrelevant subset Π_0 of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ and the set of maximal connected components $\{\Pi_i\}_{1 \leq i \leq m}$ of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \setminus \Pi_0$ at the same time. The basic idea is as follows. Initially, the algorithm sets Π_0 as Π and connected components of $\Pi \setminus \Pi_0$ as empty. Then, it shrinks Π_0 and expands connected components of $\Pi \setminus \Pi_0$ in an iterative manner until a fixpoint is reached.

To efficiently manipulate Π_0 and connected components of $\Pi \setminus \Pi_0$, the algorithm introduces a component identifier $cid(\alpha)$ for every ground atom α occurring in Π. In the algorithm, $cid(r)$ simply denotes the set $\{cid(\alpha) \mid \alpha \in \mathsf{head}(r) \cup \mathsf{body}(r)\}$ for a ground rule r. The set of rules $r \in \Pi$ such that $0 \in \{cid(\alpha) \mid \alpha \in \mathsf{head}(r) \setminus X\}$ is treated as Π_0. The algorithm always guarantees that, for any two rules r_1 and r_2 in $\Pi \setminus \Pi_0$, $cid(r_1) \cap cid(r_2) \neq \emptyset$ iff r_1 and r_2 are connected in $\Pi \setminus \Pi_0$.

Details are given as follows. Initially, the algorithm sets $cid(\alpha)$ as 0 for all ground atoms α occurring Π (line 1). Intuitively, this line sets Π_0 as Π. Then the algorithm enters a loop (lines 3–10). Intuitively, this loop iteratively shrinks Π_0 and expands connected components of $\Pi \setminus \Pi_0$. In each iteration (lines 5–10), each rule $r \in \Pi \setminus \Pi_0$, which is a rule in Π such that $\mathsf{head}(r) = \emptyset$ or $0 \notin \{cid(\alpha) \mid \alpha \in \mathsf{head}(r) \setminus X\}$, is processed as follows: first, for all ground atoms α occurring in r, $cid(\alpha)$ is set as the incremental value k if $cid(\alpha) = 0$ (lines 6–7); then, for every ground atom α occurring in r, $cid(\alpha)$ is set as the minimum $cid(\alpha)$ among all ground atoms α occurring in r (lines 8–10). Lines 6–7 give every ground atom α occurring in $\Pi \setminus \Pi_0$ a unique positive $cid(\alpha)$, ensuring that $cid(r_1) \cap cid(r_2) = \emptyset$ for any two rules r_1 and r_2 that are unconnected in $\Pi \setminus \Pi_0$. Lines 8–10 ensure that $cid(r_1) \cap cid(r_2) \neq \emptyset$ for any two rules r_1 and r_2 that are connected in $\Pi \setminus \Pi_0$, and also ensure that $cid(\alpha)$ does not increase after being changed from 0 to a positive integer for any ground atom α occurring in Π. Since there must be some $cid(\alpha)$ decreasing to another positive integer in every iteration except the last one, the set $\{cid(r) \mid r \in \Pi\}$ will be stable after a finite number of iterations. When $\{cid(r) \mid r \in \Pi\}$ is stable, Π_0 consists of all rules $r \in \Pi$ such that $\mathsf{head}(r) \not\subseteq X \cup \mathsf{atoms}(\Pi \setminus \Pi_0)$ and becomes an X-irrelevant subset of Π, and meanwhile, all rules r in $\Pi \setminus \Pi_0$ having the same $cid(r)$ constitute a maximal connected component of $\Pi \setminus \Pi_0$. Hence, the algorithm puts together all rules r having the same singleton set $cid(r) \neq \{0\}$ to form a subset Π_i $(1 \leq i \leq m)$ of Π (line 11) and sets Π_0 as $\Pi \setminus \bigcup_{i=1}^{m} \Pi_i$ (line 12).

Example 4.5 (Example 4.4 continued) We demonstrate how the algorithm `Decompose`$(\mathcal{G}_{IG}(\mathsf{RP}_e(KB)), X)$ works for $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ given in Example 4.4, where X is the

set of decision atoms in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, that is, $X = \{\hbar_{A(a)}, \hbar_{C(b)}, \hbar_{C(c)}, \hbar_{s(a,b)}, \hbar_{t(a,c)}\}$. We assume that all rules in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ are processed in turn in each iteration (lines 3–10) of the algorithm.

	k	$A(a)$	$\hbar_{A(a)}$	$C(b)$	$\hbar_{C(b)}$	$C(c)$	$\hbar_{C(c)}$	$s(a,b)$	$\hbar_{s(a,b)}$	$t(a,c)$	$\hbar_{t(a,c)}$	$C(a)$	$r(a,b)$	$r(a,c)$	$A(b)$	$B(c)$
r_{10}	2	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
r_{11}	4	0	0	1	0	3	0	0	0	0	0	0	1	3	0	0
r_2	5	0	0	1	1	3	0	0	0	0	0	0	1	3	0	0
r_3	6	0	0	1	1	3	3	0	0	0	0	0	1	3	0	0
r_{12}	7	0	0	1	1	3	3	1	0	0	0	0	1	3	0	0
r_{13}	8	0	0	1	1	3	3	1	0	3	0	0	1	3	0	0
r_4	9	0	0	1	1	3	3	1	1	3	0	0	1	3	0	0
r_5	10	0	0	1	1	3	3	1	1	3	3	0	1	3	0	0

The above table shows the result of the first three iterations, where the first column shows which rule r is processed in lines 6–10, the second column shows the incremental value k after r is processed, and the other columns show the component identifiers of all ground atoms occurring in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ after r is processed. It can be seen that in the fourth iteration the set $\{cid(r) \mid r \in \mathcal{G}_{IG}(\mathsf{RP}_e(KB))\}$ becomes stable and is not changed anymore. Thus, the algorithm returns $(\Pi_0, \{\Pi_1, \Pi_2\})$, where $\Pi_1 = \{r \in \mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \mid cid(r) = \{1\}\} = \{r_2, r_4, r_{10}, r_{12}\}$, $\Pi_2 = \{r \in \mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \mid cid(r) = \{3\}\} = \{r_3, r_5, r_{11}, r_{13}\}$, and $\Pi_0 = \Pi \setminus \{\Pi_1, \Pi_2\} = \{r_1, r_6, \ldots, r_9\}$.

The following lemma shows the computational complexity and correctness of `Decompose`(Π, X).

Lemma 4.6 *Let* $(\Pi_0, \{\Pi_i\}_{1 \le i \le m})$ *be returned from* `Decompose`(Π, X)*, and* N_{at} *be the number of ground atoms occurring in* Π. (1) `Decompose`(Π, X) *works in time polynomial in* N_{at}. (2) Π_0 *is the largest subset of* Π *such that* $\mathsf{head}(r) \not\subseteq X \cup \mathsf{atoms}(\Pi \setminus \Pi_0)$ *for all rules* $r \in \Pi_0$. (3) $\{\Pi_i\}_{1 \le i \le m}$ *is the set of maximal connected components of* $\Pi \setminus \Pi_0$.

Proof (1) For every ground atom α occurring in Π, if $cid(\alpha)$ is changed during the execution of `Decompose`(Π, X), then it is first changed from 0 to a positive integer not greater than N_{at} and continuously decreases to other positive integers in the subsequent iterations. Hence, $cid(\alpha)$ can be changed at most N_{at} times for all ground atoms α occurring in Π. Since in all iterations (lines 3–10) except the last one some $cid(\alpha)$ must decrease, there are at most $N_{\text{at}}^2 + 1$ iterations. Since each iteration works in time polynomial in N_{at}, `Decompose`(Π, X) also works in time polynomial in N_{at}.

(2) Suppose Π_0 is not the largest subset of Π such that $\mathsf{head}(r) \not\subseteq X \cup \mathsf{atoms}(\Pi \setminus \Pi_0)$ for all rules $r \in \Pi_0$, then there exists a subset Π_0' of Π such that $\Pi_0' \not\subseteq \Pi_0$ and $\mathsf{head}(r) \not\subseteq X \cup \mathsf{atoms}(\Pi \setminus \Pi_0')$ for all rules $r \in \Pi_0'$. Let $\Delta_0 = \Pi_0' \setminus \Pi_0$ and $\Pi_0^+ = \Pi_0 \cup \Pi_0'$, then $\Delta_0 \neq \emptyset$ and $\mathsf{atoms}(\Pi \setminus \Pi_0^+) \subseteq \mathsf{atoms}(\Pi \setminus \Pi_0')$. For all rules $r \in \Pi_0'$, since $\mathsf{head}(r) \not\subseteq X \cup \mathsf{atoms}(\Pi \setminus \Pi_0')$, we have $\mathsf{head}(r) \not\subseteq X \cup \mathsf{atoms}(\Pi \setminus \Pi_0^+)$. Since $\Delta_0 \subseteq \Pi_0'$, we have $\mathsf{head}(r) \not\subseteq X \cup \mathsf{atoms}(\Pi \setminus \Pi_0^+)$ for all rules $r \in \Delta_0$. On the other hand, since $\Delta_0 \cap \Pi_0 = \emptyset$, all rules $r \in \Delta_0$ must be processed at lines 6–10 during `Decompose`(Π, X) is executed. Let r_f be the first rule in Δ_0 that is processed at lines 6–10 during `Decompose`(Π, X) is executed. Consider the moment just before r_f is processed at line 6 for the first time. Since $cid(\alpha) > 0$ for all ground atoms $\alpha \in \mathsf{head}(r_f) \setminus X$, and since all ground atoms α such that $cid(\alpha) > 0$ occur in rules in $\Pi \setminus \Pi_0^+$, $\mathsf{head}(r_f) \subseteq X \cup \mathsf{atoms}(\Pi \setminus \Pi_0^+)$, contradicting that $\mathsf{head}(r) \not\subseteq X \cup \mathsf{atoms}(\Pi \setminus \Pi_0^+)$ for all rules $r \in \Delta_0$.

(3) Note that after the loop (lines 3–10) ends, for any rule $r \in \Pi \setminus \Pi_0$, $cid(r) = \{c\}$ for some $c > 0$, and for any two rules r_1 and r_2 in $\Pi \setminus \Pi_0$, if r_1 and r_2 are connected in $\Pi \setminus \Pi_0$, $cid(r_1)$ and $cid(r_2)$ are the same singleton set, otherwise $cid(r_1)$ and $cid(r_2)$ are two different singleton sets. Hence, all rules r in $\Pi \setminus \Pi_0$ that have

the same $cid(r)$ constitute a maximal connected component of $\Pi \setminus \Pi_0$. Since every Π_i $(1 \leq i \leq m)$ is formed by putting together all rules having the same $cid(r)$ with $cid(r) = \{c\}$ for some $c > 0$, Π_i is a maximal connected component of $\Pi \setminus \Pi_0$. Since $\bigcup_{i=1}^{m} \Pi_i = \Pi \setminus \Pi_0$, $\{\Pi\}_{1 \leq i \leq m}$ is the set of maximal connected components of $\Pi \setminus \Pi_0$. □

It follows from the above lemma that, `Decompose`($\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, X) works in polynomial time in data complexity, since the number of ground atoms occurring in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ is at most polynomial in the number of instance assertions in KB. In our experiments, `Decompose`($\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, X) is always finished in a few iterations and shows a high efficiency.

In the remainder of this section, let $(\Pi_0, \{\Pi_i\}_{1 \leq i \leq m})$ be the result returned from `Decompose`($\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, X), and $X_i = X \cap \mathsf{atoms}(\Pi_i)$ for all $0 \leq i \leq m$; moreover, for all $1 \leq i \leq m$, let w_i be the minimum value v such that $\Pi_i \cup \{\sum_{\beta \in X_i} w'(\beta) \cdot \mathsf{assign}(\beta) \leq v\}$ is satisfiable.

4.2.2 The key optimization in the second phase: query-relevant SAT problem construction

We now propose an optimized method for constructing a query-relevant SAT problem, which is an extended SAT problem with PB-constraints, to check whether $KB \overset{\mathrm{w}}{\approx} q$ for an atomic query q. In this method, we assume that the given atomic query q is a ground atom occurring in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$. When q does not occur in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, it is trivial that $KB \overset{\mathrm{w}}{\not\approx} q$ by Theorem 4.2.

By considering that $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\sum_{\hbar_\alpha \in X} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_{\min}\} \cup \{\leftarrow q\}$ is unsatisfiable iff $KB \overset{\mathrm{w}}{\approx} q$ (see Theorem 4.2), we simply call $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\sum_{\hbar_\alpha \in X} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_{\min}\} \cup \{\leftarrow q\}$ the *checking program* of q. Let S be an X-irrelevant subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\leftarrow q\}$. Since $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\sum_{\hbar_\alpha \in X} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_{\min}\}$ is satisfiable, the maximal connected component of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\leftarrow q\} \setminus S$ that contains the constraint $\leftarrow q$ must be unsatisfiable with some PB-constraints, iff the checking program of q is unsatisfiable. Based on this observation, we develop a method to compute an query-relevant SAT problem for checking if $KB \overset{\mathrm{w}}{\approx} q$. It first identifies an X-irrelevant subset S of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\leftarrow q\}$ in Π_0, then extracts from $\{\Pi_i\}_{1 \leq i \leq m}$ the maximal connected component Π of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\leftarrow q\} \setminus S$ that contains the constraint $\leftarrow q$, and meanwhile adds PB-constraints to Π to yield an query-relevant SAT problem for checking if $KB \overset{\mathrm{w}}{\approx} q$.

The algorithm `Construct-SAT-Problem`(q), given in Fig. 4, shows more details of this method. In the case where the given atomic query q occurs in Π_k for some $1 \leq k \leq m$, Π_0 is an X-irrelevant subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\leftarrow q\}$, and $\Pi_k \cup \{\leftarrow q\}$ is the maximal connected component of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\leftarrow q\} \setminus \Pi_0$ that contains $\leftarrow q$. Hence, $\Pi_k \cup \{\sum_{\hbar_\alpha \in X_k} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_k\} \cup \{\leftarrow q\}$ preserves the unsatisfiability of the checking program of q and is returned (line 2). In other cases, q must occur in Π_0, and Π_0 may not be an X-irrelevant subset of $\mathcal{G}_{IG}(\mathsf{RP}_e(KB)) \cup \{\leftarrow q\}$. We use the following steps to compute a query-relevant SAT problem Π for checking if $KB \overset{\mathrm{w}}{\approx} q$. Since the decision atoms in Π_0 should be assigned the truth value 0 in any model of the checking program of q (by Lemma 4.5) and they occur only in rule heads, Π_0 can be simplified by removing all decision atoms without impacting the unsatisfiability of the checking program of q. We first simplify Π_0 to Π_0' by removing all decision atoms (line 3). Then, we initialize Π as the singleton set $\{\leftarrow q\}$ (line 4) and iteratively add to Π all rules in Π_0' whose head atoms all occur in $\Pi \cup \bigcup_{i=1}^{m} \Pi_i$ (lines 5–6). After Π becomes stable, $\Pi_0' \setminus \Pi$ is an X-irrelevant subset

Input An atomic query q occurring in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$.
Output A SAT problem that includes some PB-constraints.
Precondition $\{\Pi_i\}_{0\leq i\leq m}$, $\{X_i\}_{0\leq i\leq m}$ and $\{w_i\}_{1\leq i\leq m}$ have been computed.
1. **if** $q \in \Pi_k$ for some $1 \leq k \leq m$ **then**
2. **return** $\Pi_k \cup \{\sum_{\hbar_\alpha \in X_k} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_k\} \cup \{\leftarrow q\}$;
3. $\Pi'_0 := \{\bigvee(\mathsf{head}(r) \setminus X_0) \leftarrow \bigwedge \mathsf{body}(r) \mid r \in \Pi_0\}$;
4. $\Pi := \{\leftarrow q\}$; $\Pi' := \emptyset$;
5. **while** $\Pi \neq \Pi'$ **do**
6. $\Pi' := \Pi$; $\Pi := \Pi \cup \{r \in \Pi'_0 \mid \mathsf{head}(r) \subseteq \mathsf{atoms}(\Pi \cup \bigcup_{i=1}^m \Pi_i)\}$;
7. **for** each integer i from 1 to m such that $\mathsf{atoms}(\Pi_i) \cap \mathsf{atoms}(\Pi) \neq \emptyset$ **do**
8. $\Pi := \Pi \cup \Pi_i \cup \{\sum_{\hbar_\alpha \in X_i} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_i\}$;
9. **return** Π;

Fig. 4 The algorithm for constructing an extended SAT problem for entailment checking of an atomic query

of $\Pi'_0 \cup \{\leftarrow q\} \cup \bigcup_{i=1}^m \Pi_i$. Finally, we append $\Pi_i \cup \{\sum_{\hbar_\alpha \in X_i} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_i\}$ to Π for every Π_i ($1 \leq i \leq m$) that has ground atoms occurring in Π (lines 7–8). At this moment, the set of rules in Π excluding PB-constraints is the maximal connected component of $(\Pi'_0 \cup \{\leftarrow q\} \cup \bigcup_{i=1}^m \Pi_i) \setminus \Pi''_0$ that contains $\leftarrow q$, where $\Pi''_0 = \Pi'_0 \setminus \Pi$ is an X-irrelevant subset of $\Pi'_0 \cup \{\leftarrow q\} \cup \bigcup_{i=1}^m \Pi_i$. Hence, Π preserves the unsatisfiability of the checking program of q and is returned (line 9). The correctness of the algorithm is formally shown by the following lemma.

Lemma 4.7 *Let q be an atomic query whose only atom occurs in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ and is not on complex roles, and Π be returned by* `Construct-SAT-Problem(q)`*, then $KB \overset{\mathrm{w}}{\approx} q$ iff Π is unsatisfiable.*

Proof Let $\Pi'_0 = \{\bigvee(\mathsf{head}(r) \setminus X_0) \leftarrow \bigwedge \mathsf{body}(r) \mid r \in \Pi_0\}$, and $\Pi' = \Pi'_0 \cup \{\leftarrow q\} \cup \bigcup_{i=1}^m (\Pi_i \cup \{\sum_{\hbar_\alpha \in X_i} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_i\})$. We first show that the checking program of q is unsatisfiable $\iff$ Π' is unsatisfiable. For the ($\Rightarrow$) direction, suppose the checking program of q is unsatisfiable but Π' has a model M, then $\sum_{\hbar_\alpha \in M \cap X_i} w(\alpha) \leq w_i$ for all $1 \leq i \leq m$. By Lemma 4.5, $\sum_{\hbar_\alpha \in M \cap X} w(\alpha) \leq \sum_{i=1}^m w_i = w_{\min}$. For any rule r in Π_0 such that $\mathsf{body}(r) \subseteq M$, there exists a rule r' in Π'_0 such that $\mathsf{body}(r') = \mathsf{body}(r)$ and $\mathsf{head}(r') = \mathsf{head}(r) \setminus X_0$. Since M is a model of Π', $\mathsf{head}(r') \cap M \neq \emptyset$ and thus $\mathsf{head}(r) \cap M \neq \emptyset$ too. For any rule r in $\{\leftarrow q\} \cup \bigcup_{i=1}^m \Pi_i$ such that $\mathsf{body}(r) \subseteq M$, since M is a model of Π', $\mathsf{head}(r) \cap M \neq \emptyset$. Hence, M is also a model of $\{\sum_{\hbar_\alpha \in X} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_{\min}\} \cup \Pi_0 \cup \{\leftarrow q\} \cup \bigcup_{i=1}^m \Pi_i$, that is, a model of the checking program of q, contradiction. For the ($\Leftarrow$) direction, suppose Π' is unsatisfiable but the checking program of q has a model M, then for any $1 \leq i \leq m$, $M \cap \mathsf{atoms}(\Pi_i)$ is a model of Π_i. By the minimality of w_i, we have $\sum_{\hbar_\alpha \in M \cap X_i} w(\alpha) \geq w_i$. Thus, $\sum_{\hbar_\alpha \in M \cap X} w(\alpha) \geq \sum_{i=1}^m w_i$. By Lemma 4.5, $\sum_{\hbar_\alpha \in M \cap X} w(\alpha) \geq w_{\min}$ and the equality relation holds only when $M \cap X_0 = \emptyset$ and $\sum_{\hbar_\alpha \in M \cap X_i} w(\alpha) = w_i$ for all $1 \leq i \leq m$. Since M is a model of the checking program of q, $\sum_{\hbar_\alpha \in M \cap X} w(\alpha) \leq w_{\min}$. Hence, $\sum_{\hbar_\alpha \in M \cap X} w(\alpha) = w_{\min}$. It follows that $M \cap X_0 = \emptyset$ and $\sum_{\hbar_\alpha \in M \cap X_i} w(\alpha) = w_i$ for all $1 \leq i \leq m$. For any rule r in Π'_0 such that $\mathsf{body}(r) \subseteq M$, there exists a rule r' in Π_0 such that $\mathsf{body}(r') = \mathsf{body}(r)$ and $\mathsf{head}(r') \setminus X_0 = \mathsf{head}(r)$. Since M is a model of the checking program of q and $M \cap X_0 = \emptyset$, $\mathsf{head}(r') \cap M \neq \emptyset$ and thus $\mathsf{head}(r) \cap M \neq \emptyset$. For any rule r in $\{\leftarrow q\} \cup \bigcup_{i=1}^m \Pi_i$ such that $\mathsf{body}(r) \subseteq M$, since M is a model of the checking program of q, $\mathsf{head}(r) \cap M \neq \emptyset$. Hence, M is a model of $\Pi'_0 \cup \{\leftarrow q\} \cup \bigcup_{i=1}^m (\Pi_i \cup \{\sum_{\hbar_\alpha \in X_i} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_i\}) = \Pi'$, contradiction.

We then show that Π' is unsatisfiable $\iff$ Π is unsatisfiable. The ($\Leftarrow$) direction is trivial because Π is just a subset of Π'. For the ($\Rightarrow$) direction, suppose Π' is unsatisfiable but Π has a model M. Let $\Pi_0'' = \Pi_0' \setminus \Pi$ and $M_0 = \bigcup_{r \in \Pi_0''} \mathsf{head}(r) \setminus \mathsf{atoms}(\Pi \cup \bigcup_{i=1}^{m} \Pi_i)$. For any rule r in Π'', we have $M_0 \cap \mathsf{head}(r) \neq \emptyset$, otherwise r will be added to Π in `Construct-SAT-Problem`(q). Hence, M_0 is a model of Π_0''. Let $\{\Pi_{c_i}\}_{1 \leq i \leq k}$ be the subset of $\{\Pi_i\}_{1 \leq i \leq m}$ such that $\mathsf{atoms}(\Pi_{c_i}) \cap \mathsf{atoms}(\Pi) = \emptyset$, where $1 \leq c_1 \leq \ldots \leq c_k \leq m$. Let M_{c_i} be a model of $\Pi_{c_i} \cup \{\sum_{\hbar_\alpha \in X_{c_i}} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_{c_i}\}$ for $1 \leq i \leq k$. Let $M' = M_0 \cup M \cup \bigcup_{i=1}^{k} M_{c_i}$. Since M does not contain any decision atom occurring in $\bigcup_{i=1}^{k} \Pi_{c_i}$ and M_{c_i} does not contain any decision atom occurring in $\Pi \cup \bigcup_{j=1, j \neq i}^{k} \Pi_{c_j}$ for all $1 \leq i \leq k$, we have $\sum_{\hbar_\alpha \in M' \cap X_i} w(\alpha) \leq w_i$ for all $1 \leq i \leq m$. For any rule r in Π_0'', since $\mathsf{head}(r) \cap M_0 \neq \emptyset$, $\mathsf{head}(r) \cap M' \neq \emptyset$. For any rule r in Π which is not a PB-constraint, such that $\mathsf{body}(r) \subseteq M'$, since $M_0 \cup \bigcup_{i=1}^{k} M_{c_i}$ has no ground atoms occurring in Π, $\mathsf{body}(r) \subseteq M$. Since M is a model of Π, $\mathsf{head}(r) \cap M \neq \emptyset$ and thus $\mathsf{head}(r) \cap M' \neq \emptyset$. For any rule r in Π_{c_i}, where i is some integer between 1 and k, such that $\mathsf{body}(r) \subseteq M'$, since $M_0 \cup M \cup \bigcup_{j=1, j \neq i}^{k} M_{c_j}$ has no ground atoms occurring in Π_{c_i}, $\mathsf{body}(r) \subseteq M_{c_i}$. Since M_{c_i} is a model of Π_{c_i}, $\mathsf{head}(r) \cap M_{c_i} \neq \emptyset$ and thus $\mathsf{head}(r) \cap M' \neq \emptyset$. Hence, M' is a model of $\Pi_0'' \cup \Pi \cup \bigcup_{i=1}^{k} \Pi_{c_i} \cup \{\sum_{\hbar_\alpha \in X_{c_i}} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_{c_i}\} = \Pi'$, contradiction.

By Theorem 4.2 (2), we have $KB \overset{\mathrm{w}}{\approx} q$ iff the checking program of q is unsatisfiable. Hence, $KB \overset{\mathrm{w}}{\approx} q$ iff Π' is unsatisfiable, that is, Π is unsatisfiable. □

In the following, an example is provided to illustrate the proposed method for entailment checking of atomic queries.

Example 4.6 (Example 4.5 continued) Recall that $\Pi_0 = \{r_1, r_6, \ldots, r_9\}$, $\Pi_1 = \{r_2, r_4, r_{10}, r_{12}\}$ and $\Pi_2 = \{r_3, r_5, r_{11}, r_{13}\}$. Suppose $w(A(a)) = w(C(b)) = w(C(c)) = w(s(a,b)) = w(t(a,c)) = 1$, then $w_1 = w_2 = 1$ because 1 is both the minimum value v_1 such that $\Pi_1 \cup \{\mathsf{assign}(\hbar_{C(b)}) + \mathsf{assign}(\hbar_{s(a,b)}) \leq v_1\}$ is satisfiable and the minimum value v_2 such that $\Pi_2 \cup \{\mathsf{assign}(\hbar_{C(c)}) + \mathsf{assign}(\hbar_{t(a,c)}) \leq v_2\}$ is satisfiable.

Consider the atomic query $r(a,b)$. Since $r(a,b)$ occurs in Π_1, the algorithm `Construct-SAT- Problem`($r(a,b)$) returns $\Pi_1 \cup \{\mathsf{assign}(\hbar_{C(b)}) + \mathsf{assign}(\hbar_{s(a,b)}) \leq 1\} \cup \{\leftarrow r(a,b)\}$. It is satisfiable, thus $KB \overset{\mathrm{w}}{\not\approx} r(a,b)$.

Consider the atomic query $A(b)$. Since $A(b)$ occurs only in Π_0, the algorithm `Construct-SAT- Problem`($A(b)$) returns $\{\leftarrow A(b)\} \cup \{r_8\} \cup \Pi_1 \cup \{\mathsf{assign}(\hbar_{C(b)}) + \mathsf{assign}(\hbar_{s(a,b)}) \leq 1\}$. It is satisfiable, thus $KB \overset{\mathrm{w}}{\not\approx} A(b)$.

Consider the atomic query $A(a)$. Since $A(a)$ occurs only in Π_0, the algorithm `Construct-SAT- Problem`($A(a)$) returns $\{\leftarrow A(a)\} \cup \{A(a)\}$. Note that the ground fact $A(a)$ is simplified from r_1 by removing all decision atoms. Since $\{\leftarrow A(a)\} \cup \{A(a)\}$ is unsatisfiable, we have $KB \overset{\mathrm{w}}{\approx} A(a)$.

It can be seen from the above example that the entailment checking of some atomic queries, such as $A(a)$, does not require the computation of any w_i beforehand. We can prove that $KB \overset{\mathrm{w}}{\approx} q$ for any atomic query q which is a ground atom in every model of Π_0', where Π_0' is the propositional program simplified from Π_0 by removing all decision atoms (see the proof of Theorem 4.3 (2) for more details). A simple way to compute ground atoms in every model of Π_0' is to compute the smallest model M_0^{def} of $\{r \in \Pi_0' \mid |\mathsf{head}(r)| = 1\}$, the set of definite rules in Π_0', because any ground atom in M_0^{def} is in every model of Π_0'. The smallest

model of $\{r \in \Pi'_0 \mid |\mathsf{head}(r)| = 1\}$ is unique and can be constructed as the least fixpoint of $M^{(n)}$ such that $M^{(0)} = \emptyset$, and for $n > 0$, $M^{(n)} = \bigcup_{r \in \Pi'_0, |\mathsf{head}(r)|=1, \mathsf{body}(r) \subseteq M^{(n-1)}} \mathsf{head}(r)$.

Based on the above ideas, we further optimize the proposed method for entailment checking of an atomic query q: if q does not occur in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, then $KB \overset{\mathrm{w}}{\not\approx} q$; otherwise, if q is in M_0^{def}, then $KB \overset{\mathrm{w}}{\approx} q$; otherwise, $KB \overset{\mathrm{w}}{\approx} q$ iff the propositional program returned by `Construct-SAT-Problem`(q) is unsatisfiable. Consider Example 4.6 again, Π'_0 is $\{A(a)\} \cup \{r_6, \ldots, r_9\}$ (note that $A(a)$ is a ground fact here) and $M_0^{\mathrm{def}} = \{A(a)\}$ (note that $A(a)$ is a ground atom here). Since $A(a) \in M_0^{\mathrm{def}}$, we directly obtain $KB \overset{\mathrm{w}}{\approx} A(a)$ without calling `Construct-SAT-Problem`$(A(a))$. The correctness of this method is shown in the following theorem.

Theorem 4.3 *Let* $\Pi'_0 = \{\bigvee(\mathsf{head}(r) \setminus X_0) \leftarrow \bigwedge \mathsf{body}(r) \mid r \in \Pi_0\}$ *and* M_0^{def} *be the unique smallest model of* $\{r \in \Pi'_0 \mid |\mathsf{head}(r)| = 1\}$. *For* q *be an atomic query whose only atom is not on complex roles,* (1) *if* q *does not occur in* $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, *then* $KB \overset{\mathrm{w}}{\not\approx} q$, *else* (2) *if* $q \in M_0^{\mathrm{def}}$, *then* $KB \overset{\mathrm{w}}{\approx} q$, *else* (3) $KB \overset{\mathrm{w}}{\approx} q$ *iff* Π *is unsatisfiable, where* Π *is returned by* `Construct-SAT-Problem`(q).

Proof (1) When q does not occur in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, the checking program of q is obviously satisfiable. By Theorem 4.2 (2), $KB \overset{\mathrm{w}}{\not\approx} q$.

(2) When $q \in M_0^{\mathrm{def}}$, since $M_0^{\mathrm{def}} \subseteq M$ for all models of Π'_0, q is in all models of Π'_0. Let $\Pi' = \Pi'_0 \cup \{\leftarrow q\} \cup \bigcup_{i=1}^{m}(\Pi_i \cup \{\sum_{\hbar_\alpha \in X_i} w(\alpha) \cdot \mathsf{assign}(\hbar_\alpha) \leq w_i\})$. Since Π'_0 is a subset of Π', q is in all models of Π' too. Since the constraint $\leftarrow q$ is in Π', Π' must have no models, that is, Π' is unsatisfiable. As was proved in Lemma 4.7, $KB \overset{\mathrm{w}}{\approx} q$ iff Π' is unsatisfiable. Hence, $KB \overset{\mathrm{w}}{\approx} q$.

(3) This result has been proved in Lemma 4.7. □

4.3 Evaluating queries without non-distinguished variables

In this subsection, we propose a method for computing all consistent answers of a conjunctive query without non-distinguished variables $Q(\vec{x}) = \mathsf{conj}(\vec{x}, \vec{c})$. The basic idea comes from the fact that a tuple of individuals $\vec{t}$ is a consistent answer of $Q(\vec{x})$, only when all ground atoms occurring in $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$ (which is a conjunction of ground atoms) also occur in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$. This is a consequence of Theorem 4.3 (1) and Lemma 4.1. Hence, we define a *candidate answer* of $Q(\vec{x})$ as a tuple of individuals $\vec{t}$ such that all ground atoms occurring in $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$ also occur in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$. The method first retrieves all candidate answers of $Q(\vec{x})$ from $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$, then keeps only candidate answers $\vec{t}$ such that all ground atoms occurring $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$ are consistently entailed by KB. In the method, all ground atoms occurring in $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$ are collected and treated as atomic queries. The entailment checking of these ground atoms is done one by one before filtering any candidate answer.

The algorithm `Weight-based-CQA`$(KB, Q(\vec{x}))$, given in Fig. 5, shows more details of the method. First, the set $\mathcal{A}_{\mathrm{cand}}$ of candidate answers of $Q(\vec{x})$ in KB is computed, where a candidate answer of $Q(\vec{x})$ in KB is a tuple of individuals in KB such that $\mathsf{atoms}(Q(\vec{x})[\vec{x} \mapsto \vec{t}]) \subseteq \mathsf{atoms}(\mathcal{G}_{IG}(\mathsf{RP}_e(KB)))$ (line 1). Afterwards, all ground atoms occurring in $\mathcal{A}_{\mathrm{cand}}$ are collected in a set S_{atm}, then for every ground atom α in S_{atm}, if $KB \overset{\mathrm{w}}{\approx} \alpha$ (checked by the optimized method given above Theorem 4.3), α is added to a set S_{ent} (line 2). Finally, the set of consistent answers of $Q(\vec{x})$ in KB is retrieved from $\mathcal{A}_{\mathrm{cand}}$, where a consistent answer $\vec{t}$ of

Input An extensionally reduced $\mathcal{SHIQ}$ knowledge base KB and a conjunctive query without non-distinguished variables $Q(\vec{x}) = \mathsf{conj}(\vec{x}, \vec{c})$.
Output The set of all consistent answers of KB.
1. $\mathcal{A}_{cand} := \{\vec{t} \mid \vec{t}$ is a tuple of individuals in KB such that $\mathsf{atoms}(Q(\vec{x})[\vec{x} \mapsto \vec{t}]) \subseteq \mathsf{atoms}(\mathcal{G}_{IG}(\mathsf{RP}_e(KB)))\}$;
2. $S_{atm} := \bigcup_{\vec{t} \in \mathcal{A}_{cand}} \mathsf{atoms}(Q(\vec{x})[\vec{x} \mapsto \vec{t}])$; $S_{ent} := \{\alpha \in S_{atm} \mid KB \overset{\mathrm{w}}{\approx} \alpha\}$;
3. **return** $\{\vec{t} \in \mathcal{A}_{cand} \mid \mathsf{atoms}(Q(\vec{x})[\vec{x} \mapsto \vec{t}]) \subseteq S_{ent}\}$;

Fig. 5 The algorithm for evaluating a conjunctive query without non-distinguished variables

$Q(\vec{x})$ in KB is a candidate answer in $\mathcal{A}_{\text{cand}}$ such that $\mathsf{atoms}(Q(\vec{x})[\vec{x} \mapsto \vec{t}]) \subseteq S_{\text{ent}}$ (line 3), which means that $KB \overset{\mathrm{w}}{\approx} \alpha$ for all $\alpha \in \mathsf{atoms}(Q(\vec{x})[\vec{x} \mapsto \vec{t}])$.

In the following, an example is provided to illustrate the proposed method.

Example 4.7 (Example 4.6 continued) Given a conjunctive query $Q(\langle x, y\rangle) = A(x) \wedge A(y)$, the algorithm `Weight-based-CQA`$(KB, Q(\langle x, y\rangle))$ for KB given in Example 4.4 works as follows. Since $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ (shown in Example 4.4) contains only two ground atoms on A, namely $A(a)$ and $A(b)$, the set $\mathcal{A}_{\text{cand}}$ of candidate answers is $\{\langle a, a\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle b, b\rangle\}$. Thus, S_{atm} is $\{A(a), A(b)\}$. As was shown in Example 4.6, $KB \overset{\mathrm{w}}{\approx} A(a)$ and $KB \overset{\mathrm{w}}{\not\approx} A(b)$. Hence, S_{ent} is $\{A(a)\}$ and thus the set of consistent answers of $Q(\langle x, y\rangle)$ in KB is $\{\langle s, t\rangle \in \mathcal{A}_{\text{cand}} \mid \{A(s), A(t)\} \subseteq \{A(a)\}\} = \{\langle a, a\rangle\}$.

It can be seen from the above example that the number of candidate answers can be exponential in the number of variables occurring in the given conjunctive query. However, the number of reduced atomic queries that need to be checked is bounded by $n_c n_i + n_r n_i^2$, where n_c, n_r and n_i are, respectively, the number of concept names, the number of role names and the number of individual names in KB. Hence, in order to compute all consistent answers of a conjunctive query without non-distinguished variables, we only need to solve at most polynomially many entailment checking problems for atomic queries. Solving these entailment checking problems is the most time-consuming part of the method.

The following theorem shows the correctness of the proposed method.

Theorem 4.4 *For $Q(\vec{x})$, a conjunctive query which has no non-distinguished variables or atoms on complex roles, the algorithm* `Weight-based-CQA`$(KB, Q(\vec{x}))$ *returns the set of consistent answers of $Q(\vec{x})$ in KB.*

Proof Let $\mathcal{A}_{\text{cons}}$ be the set of consistent answers of $Q(\vec{x})$ in KB and $\mathcal{A}_{\text{cand}}$ be the set of candidate answers of $Q(\vec{x})$ in KB. For any tuple $\vec{t}$ in $\mathcal{A}_{\text{cons}}$, $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$ is a conjunction of ground atoms. By Lemma 4.1, $KB \overset{\mathrm{w}}{\approx} \alpha$ for all ground atoms α occurring in $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$. By Theorem 4.3 (1), α must occur in $\mathcal{G}_{IG}(\mathsf{RP}_e(KB))$ for all ground atoms α occurring in $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$. Hence, $\mathcal{A}_{\text{cons}} \subseteq \mathcal{A}_{\text{cand}}$. Let S_{ent} be the set of ground atoms α occurring in $\mathcal{A}_{\text{cand}}$ such that $KB \overset{\mathrm{w}}{\approx} \alpha$. For any tuple $\vec{t}$ in $\mathcal{A}_{\text{cand}}$, by Lemma 4.1, $\vec{t}$ is a consistent answer of $Q(\vec{x})$ in KB iff $KB \overset{\mathrm{w}}{\approx} \alpha$ for all ground atoms α occurring in $Q(\vec{x})[\vec{x} \mapsto \vec{t}]$, that is, $\mathsf{atoms}(Q(\vec{x})[\vec{x} \mapsto \vec{t}]) \subseteq S_{\text{ent}}$. Hence, $\mathcal{A}_{\text{cons}} = \{\vec{t} \in \mathcal{A}_{\text{cand}} \mid \mathsf{atoms}(Q(\vec{x})[\vec{x} \mapsto \vec{t}]) \subseteq S_{\text{ent}}\}$. □

The proposed method can be further optimized, for example, by decreasing the number of candidate answers, which is up to exponential in the number of variables occurring in the query. However, since we focus on optimizing entailment checking of atomic queries in this paper, we do not consider these optimizations here and leave optimizing the evaluation of conjunctive queries as our future work.

5 Experimental evaluation

We implemented the proposed method with decomposition-based optimizations in GNU C++. The implemented system works on disjunctive datalog programs output by the KAON2 OWL reasoner (simply called KAON2) and supports basic datatypes, such as string and integer. In this system,[1] MySQL is used as the back-end SQL engine; all instance assertions in ABoxes and derived ground atoms in the grounding process are maintained in MySQL databases; all instantiated rules in the grounding process are retrieved by SQL statements, maintained on disk and are sequentially accessed during the decomposition process; the SAT solver `MiniSat`$^+$ [19], which supports PB-constraints, is applied to perform satisfiability tests in Theorem 4.3. All our experiments were conducted on a 2.0 GHz Pentium Dual CPU with 2GB RAM PC running Windows XP and Cygwin.

5.1 Experimental setup

We collected seven publicly available KBs. The first one is Semintec,[2] which is about financial services. The other six are University Benchmark (UOBM) [35] KBs. UOBM enriches the well-known Lehigh University Benchmark (LUBM) [25] with more OWL constructors and more links between instance assertions. There are two species of UOBM. One is enriched from LUBM by adding OWL Lite constructors and is denoted by UOBM-Lite. The other is enriched from LUBM by adding OWL DL constructors and is denoted by UOBM-DL. Since the proposed method applies KAON2 and cannot deal with nominals, we removed all nominals from UOBM-DL. In addition, since KAON2 does not work well for number restrictions that are not functional,[3] we also replaced number restrictions in UOBM-DL with functional restrictions. We denote the weaken version of UOBM-DL by UOBM-DL$^-$. We also use UOBM-Lite$n$ (resp. UOBM-DL^-n) to denote the KB instance of UOBM-Lite (resp. UOBM-DL$^-$) that contains data from n universities. The collected UOBM KBs are UOBM-Lite$n$ and UOBM-DL^-n for $n = 1, 5$ and 10. They were originally downloaded from the UOBM Web site.[4] Table 2 summarizes the characteristics of each collected KB and the disjunctive datalog program transformed from its TBox and RBox by applying KAON2.

Since all collected KBs are consistent, we developed a tool, called `Injector`, to insert *conflicts*. A conflict is a set of instance assertions violating a functional role restriction or a disjointness constraint. Given a consistent knowledge base $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ and a number m of conflicts to be inserted, `Injector` inserts m conflicts to KB one by one by generating a set S_{cn} of instance assertions. Let S_{FR} denote the set of functional or inverse functional roles and S_{DC} the set of atomic concepts that have disjoint atomic concepts in KB. To insert a conflict, `Injector` randomly selects an entity in $S_{\text{FR}} \cup S_{\text{DC}}$. In case a functional role r is selected, if there exist role assertions on r that are entailed by KB, `Injector` randomly selects one, say $r(a, b)$, and adds $r(a, c)$ and $b \not\approx c$ to S_{cn}, where c is a new individual; otherwise, `Injector` adds $r(a, b)$, $r(a, c)$ and $b \not\approx c$ to S_{cn}, where a, b, c are new individuals. In case an inverse functional role r is selected, `Injector` does in the same way as for r^-. In case an atomic concept C is selected, if there exist concept assertions on C that are entailed by KB, `Injector` randomly selects one, say $C(a)$, and adds $D(a)$ to S_{cn}

[1] The implemented system, accessory tools and test KBs can be downloaded from the Web site http://jfdu.limewebs.com/wb-cqa/.

[2] http://www.cs.put.poznan.pl/alawrynowicz/semintec.htm.

[3] See the open issues listed at the KAON2 Web site http://kaon2.semanticweb.org/.

[4] http://www.alphaworks.ibm.com/tech/semanticstk/.

Table 2 The characteristics of test KBs and their transformed disjunctive datalog programs

	n_C	n_R	n_{ax}	n_{fr}	n_{dc}	n_I	n_{as}	T_{trans} (ms)	n_{rl}
Semintec	59	16	219	16	113	17,941	65,240	654	230
UOBM-Lite1						95,010	245,864		
UOBM-Lite5	51	43	145	2	0	420,149	1,075,060	510	171
UOBM-Lite10						820,208	2,096,973		
UOBM-DL$^-$1						96,081	260,540		
UOBM-DL$^-$5	68	44	205	2	1	420,702	1,132,376	567	217
UOBM-DL$^-$10						825,455	2,217,302		

n_C (resp. n_R) is the number of concept (resp. role) names. n_{ax} is the number of axioms in the union of the TBox and the RBox. n_{fr} is the number of (inverse) functional roles. n_{dc} is the number of axioms that declare disjointness constraints. n_I is the number of individual names. n_{as} is the number of instance assertions in the ABox. T_{trans} is the time (in milliseconds) for transforming the union of the TBox and the RBox to a disjunctive datalog program. n_{rl} is the number of rules in the transformed disjunctive datalog program

for a randomly selected disjoint atomic concept D of C; otherwise, `Injector` adds $C(a)$ and $D(a)$ to S_{cn}, where a is a new individual and D a randomly selected disjoint atomic concept of C. The `Injector` tool was implemented in JAVA, using the KAON2 API to find disjoint concepts and compute entailments. By KB_{+m}, we denote $(\mathcal{T}, \mathcal{R}, \mathcal{A} \cup S_{\text{cn}})$ in which m conflicts are inserted.

We generated 42 test KBs from the collected KBs. They are Semintec$_{+m}$, UOBM-Liteu_{+m} and UOBM-DL$^-u_{+m}$ for $u = 1, 5, 10$ and $m = 50, 100, 150, 200, 250, 300$. The expressivity of all test KBs is up to $\mathcal{SHIF}$ with datatypes.

5.2 Experimental results

In our experiments, all instance assertions were given the same weight (one). We performed two groups of experiments for each test KB. The experiments in the first group compute consistent answers of all concept name queries. Recall that a concept name query is a query of the form $A(x)$, where A is a concept name and x is a variable. These experiments can roughly show the performance of the proposed method in evaluating a conjunctive query in which every variable occurs in some atoms on concept names. This is because most atomic queries reduced from this kind of conjunctive queries are about concept names and the execution time is dominated by the execution time for evaluating all reduced atomic queries. The experiments in the second group compute consistent answers of benchmark queries. These experiments can show the performance of the proposed method in evaluating a commonly used conjunctive query.

We used the following five benchmark queries for Semintec$_{+m}$ KBs. They, respectively, enquire all clients with different types of loans and all clients with different types of credit cards. They are common queries about financial services.

$$Q_{S1}(\langle x, y, z, w\rangle) = \mathsf{Client}(x), \mathsf{isOwnerOf}(x, y), \mathsf{hasLoan}(y, z), \mathsf{hasLoanStatusValue}(z, w), \mathsf{OKStatus}(w)$$

$$Q_{S2}(\langle x, y, z, w\rangle) = \mathsf{Client}(x), \mathsf{isOwnerOf}(x, y), \mathsf{hasLoan}(y, z), \mathsf{hasLoanStatusValue}(z, w), \mathsf{ProblemStatus}(w)$$

$$Q_{S3}(\langle x, y\rangle) = \mathsf{Client}(x), \mathsf{hasCreditCard}(x, y), \mathsf{Classic}(y)$$

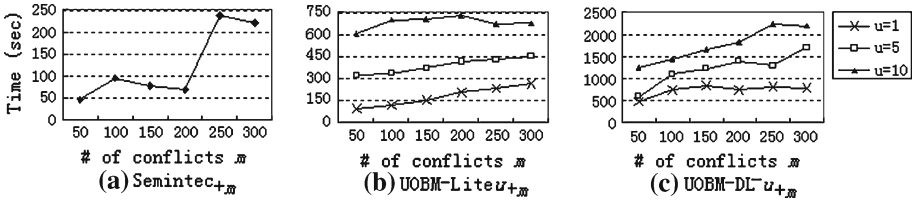

Fig. 6 The execution time of the offline phase

$$Q_{S4}(\langle x, y\rangle) = \mathsf{Client}(x), \mathsf{hasCreditCard}(x, y), \mathsf{Gold}(y)$$
$$Q_{S5}(\langle x, y\rangle) = \mathsf{Client}(x), \mathsf{hasCreditCard}(x, y), \mathsf{Junior}(y)$$

We used the following seven benchmark queries for both UOBM-Liteu_{+m} KBs and UOBM-DL$^{-}u_{+m}$ KBs. They are all the benchmark queries in the UOBM package which are not atomic queries and have neither non-distinguished variables nor atoms on complex roles.

$$Q_{U1}(\langle x\rangle) = \mathsf{UndergraduateStudent}(x),$$
$$\mathsf{takesCourse}(x, \text{http://www.Department0.University0.edu/Course0})$$
$$Q_{U2}(\langle x\rangle) = \mathsf{Student}(x), \mathsf{isMemberOf}(x, \text{http://www.Department0.University0.edu})$$
$$Q_{U3}(\langle x\rangle) = \mathsf{Person}(x), \mathsf{hasAlumnus}(\text{http://www.University0.edu}, x)$$
$$Q_{U4}(\langle x\rangle) = \mathsf{Person}(x), \mathsf{hasSameHomeTownWith}(x,$$
$$\text{http://www.Department0.University0.edu/FullProfessor0})$$
$$Q_{U5}(\langle x\rangle) = \mathsf{SportsLover}(x), \mathsf{hasMember}(\text{http://www.Department0.University0.edu}, x)$$
$$Q_{U6}(\langle x\rangle) = \mathsf{isFriendOf}(x, \text{http://www.Department0.University0.edu/FullProfessor0})$$
$$Q_{U7}(\langle x\rangle) = \mathsf{PeopleWithHobby}(x),$$
$$\mathsf{isMemberOf}(x, \text{http://www.Department0.University0.edu})$$

Figure 6 shows the execution time of the offline phase, namely compiling and decomposing a test KB to a set of propositional programs $\{\Pi_i\}_{0\le i\le m}$ as well as computing the minimum total weight w_i associated with each Π_i (where $1 \le i \le m$). It shows that the offline phase finishes in less than 40 min for each test KB even when the KB has hundreds of conflicts and millions of instance assertions. The execution time is roughly proportional to the number of instance assertions and the number of conflicts. To verify the effectiveness of the decomposition-based optimization in this phase, we also tested whether $\sum_{i=1}^{m} w_i$ (namely $w_{\min}$) can be directly computed from a propositional program which is compiled from a test KB without being decomposed. For all test KBs, the direct computation of $w_{\min}$ fails because the applied SAT solver `MiniSat`$^{+}$ runs out of memory. It shows that directly computing $w_{\min}$ from a compiled propositional program is probably infeasible.

Since the evaluation of a test query (i.e., a concept name query or a benchmark query) is reduced to the evaluation of atomic queries, we collected statistics in the online phase to show how the method works on atomic queries. We focus on atomic queries that need to be checked by satisfiability tests, called *normal atomic queries*, because the percentage of these queries in all reduced atomic queries determines whether the optimization presented above Theorem 4.3 is effective. Figure 7 shows that the percentage of normal atomic queries is less than 45 % for Semintec$_{+m}$ KBs and less than 15 % for other test KBs. This means that computing some ground atoms consistently entailed by a test KB before evaluating a conjunctive query can significantly improve the efficiency in evaluating the query. Figure 8 shows that evaluating a normal atomic query is efficient: for Semintec$_{+m}$ KBs, the average

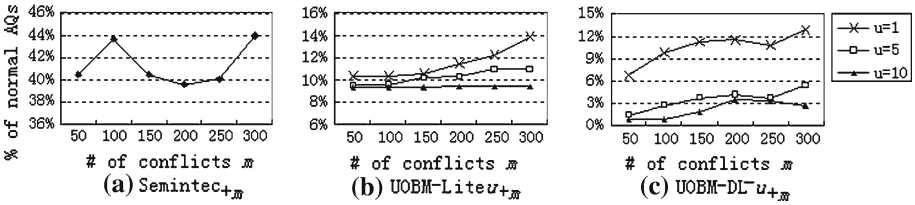

Fig. 7 The percentage of normal atomic queries in all reduced atomic queries

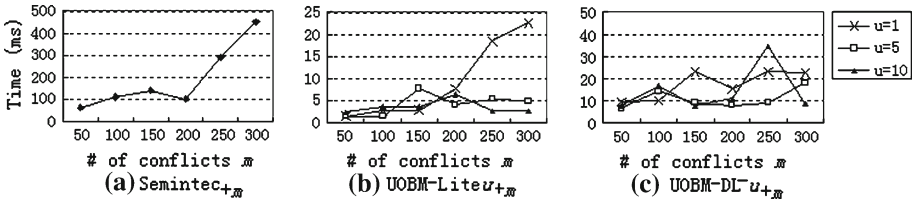

Fig. 8 The average execution time for evaluating a normal atomic query

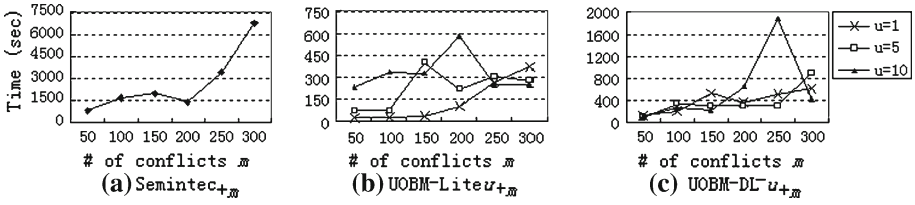

Fig. 9 The average execution time for evaluating a concept name query in the online phase

execution time is less than half a second; for other KBs, the average execution time is at most a few dozens of milliseconds. The execution time for evaluating a normal atomic query is often not proportional to the number of conflicts because conflicts are randomly inserted and have a random distribution in a test KB. It is also not proportional to the number of instance assertions because for larger test KBs the decomposition-based optimization tends to yield more smaller propositional programs.

The statistics for evaluating a test query is not as optimistic as the statistics for evaluating an atomic query. Figure 9 shows that the average execution time for evaluating a concept name query is often not proportional to the number of conflicts. This is because the number of normal atomic queries reduced from a concept name query may not be proportional to the number of conflicts. The figure also shows that the evaluation of a concept name query is not always easy. However, by analyzing the execution time for single concept name queries, we found that most concept name queries are evaluated quickly and the average execution time is dominated by only a small portion of hard queries. Figure 10 shows that for all test KBs, at most 30 % (15/51 or 20/68) concept name queries cost more than 1 min in the online phase. Table 3 shows that the maximum execution time for evaluating a concept name query in Semintec$_{+m}$ (resp. UOBM-Liteu_{+m} or UOBM-DL$^{-}u_{+m}$) KBs can be up to 42 h (resp. 8 or 16 h). It also shows that for the hardest concept name query Q, whose evaluation costs the most execution time, the average execution time for evaluating a normal atomic query reduced from Q is less than 1 min (resp. less than 6 or 7 s) for Semintec$_{+m}$ (resp. UOBM-Liteu_{+m} or UOBM-DL$^{-}u_{+m}$) KBs. This implies that the main reason why evaluating a concept name query Q is so hard is that there are too many normal atomic queries reduced from Q.

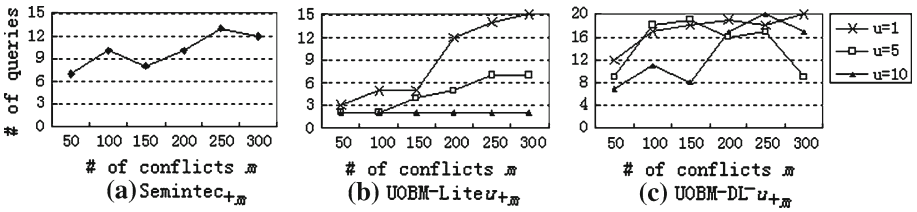

Fig. 10 The number of concept name queries that cost more than 1 min. *Note*: the total number of concept name queries for a Semintec$_{+m}$ KB (resp. a UOBM-Liteu_{+m} KB or a UOBM-DL$^{-}u_{+m}$ KB) is 59 (resp. 51 or 68)

Table 3 The statistics for evaluating the hardest concept name query

KB	$m = 50$	$m = 100$	$m = 150$	$m = 200$	$m = 250$	$m = 300$
Semintec$_{+m}$	PO/4.8	PO/11.9	W/18.2	PO/8.0	PO/18.0	M/55.8
	24,489	77,321	48,183	40,572	116,788	152,088
UOBM-Lite1$_{+m}$	GS/0.1	GS/0.1	GS/0.1	SL/2.9	SL/5.8	SL/5.5
	720	717	721	2,184	7,815	9,233
UOBM-Lite5$_{+m}$	GS/0.1	GS/0.1	GS/0.4	GS/0.1	SL/1.4	GS/0.1
	3,083	3,078	18,457	5,263	6,268	5,284
UOBM-Lite10$_{+m}$	GS/0.1	GS/0.2	GS/0.2	GS/0.3	GS/0.1	GS/0.1
	10,200	14,873	14,383	27,251	10,674	10,635
UOBM-DL^{-}1$_{+m}$	US/0.7	US/0.8	US/1.8	US/1.3	US/1.9	US/1.9
	3,433	6,021	15,417	11,278	15,462	18,418
UOBM-DL^{-}5$_{+m}$	S/0.4	US/0.8	US/0.6	US/0.6	US/0.7	S/1.0
	1,878	7,639	7,764	9,026	9,180	24,692
UOBM-DL^{-}10$_{+m}$	S/0.4	S/1.0	S/0.5	SL/1.0	SL/6.9	S/0.5
	3,857	6,472	7,576	17,892	57,654	12,240

$Q/^{t_2}_{t_1}$ means that the hardest concept name query (i.e., the concept name query whose evaluation costs the most execution time) is Q, the execution time for evaluating Q is t_1 seconds, and the average execution time for evaluating a normal atomic query reduced from Q is t_2 seconds. Q is shown as abbreviations: PO for PermanentOrder, W for Woman, M for Man, GS for GraduateStudent, SL for SportsLover, US for UndergraduateStudent and S for Student

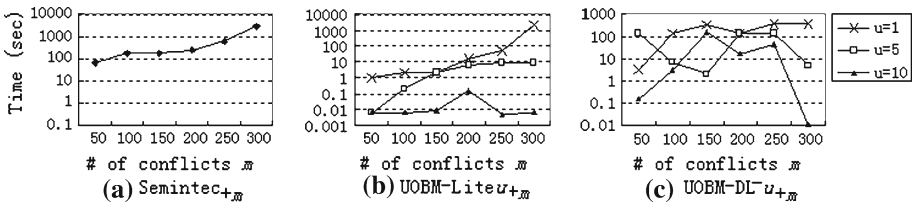

Fig. 11 The average execution time for evaluating a benchmark query in the online phase

Figure 11 shows that evaluating a benchmark query is not always easy either. Again, by analyzing the execution time for single benchmark queries, we found that the average execution time is dominated by a small portion of hard queries. Figure 12 shows that, for 36 test KBs at most two benchmark queries cost more than 1 min, and for other six test KBs three benchmark queries cost more than 1 min. Table 4 shows that the maximum execution

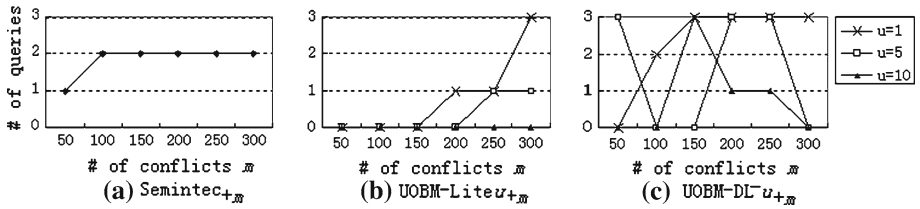

Fig. 12 The number of benchmark queries that cost more than 1 min

Table 4 The statistics for evaluating the hardest benchmark query

KB	$m = 50$	$m = 100$	$m = 150$	$m = 200$	$m = 250$	$m = 300$
Semintec$_{+m}$	Q_{S1}/0.5	Q_{S2}/0.6	Q_{S2}/0.6	Q_{S1}/0.8	Q_{S2}/22.3	Q_{S2}/9.7
	331	445	427	590	1,942	7,252
UOBM-Lite1$_{+m}$	Q_{U5}/0.2	Q_{U5}/1.1	Q_{U5}/1.6	Q_{U5}/2.8	Q_{U5}/5.5	Q_{U2}/10.3
	5	16	16	117	368	6,861
UOBM-Lite5$_{+m}$	Q_{U2}/0.1	Q_{U6}/0.1	Q_{U5}/0.6	Q_{U5}/1.3	Q_{U5}/1.3	Q_{U5}/1.2
	0.1	0.5	14	39	64	62
UOBM-Lite10$_{+m}$	Q_{U2}/0.1	Q_{U6}/0.1	Q_{U5}/0.1	Q_{U2}/0.2	Q_{U2}/0.1	Q_{U2}/0.1
	0.1	0.1	0.1	0.2	0.1	0.1
UOBM-DL$^-$1$_{+m}$	Q_{U5}/0.5	Q_{U2}/0.5	Q_{U2}/1.2	Q_{U2}/0.5	Q_{U2}/1.4	Q_{U2}/1.2
	19	678	1,571	689	1,816	1,584
UOBM-DL$^-$5$_{+m}$	Q_{U2}/0.5	Q_{U5}/3.4	Q_{U5}/0.6	Q_{U2}/0.5	Q_{U2}/0.5	Q_{U5}/2.5
	687	51	17	707	702	33
UOBM-DL$^-$10$_{+m}$	Q_{U1}/0.5	Q_{U5}/1.0	Q_{U2}/0.6	Q_{U5}/1.0	Q_{U5}/7.0	Q_{U1}/0.1
	1	23	733	112	314	0.1

$Q/^{t_2}_{t_1}$ means that the hardest benchmark query (i.e., the benchmark query whose evaluation costs the most execution time) is Q, the execution time for evaluating Q is t_1 seconds, and the average execution time for evaluating a normal atomic query reduced from Q is t_2 seconds

time for evaluating a benchmark query is up to 2 h for all test KBs. It also shows that, for the hardest benchmark query Q, whose evaluation costs the most execution time, the average execution time for evaluating a normal atomic query reduced from Q is <2 s for 81 % (34/42) test KBs. This also implies that the main reason why evaluating a benchmark query Q is so hard is that there are too many normal atomic queries reduced from Q.

6 Related work

As the Semantic Web era is coming, there are more and more proposals about providing non-standard reasoning mechanisms for inconsistent DL KBs.

Most of the existing proposals [29,33,38,42,43], like ours, follow an approach based on syntactically weakening a DL KB. This approach defines reasoning over an inconsistent KB as reasoning over all its preferred consistent subbases. [29] proposed a method which first selects a consistent subbase based on a selection function defined on the syntactic or semantic relevance, then reasons over the selected one using standard methods. This method does not adhere to the principle of minimal change, which is widely adopted in belief revision [44], because the selected subbase may not be minimally different from the original KB.

[38] adapted lexicographic inference in propositional logic [7] to DLs, which adheres to the principle of minimal change. Their proposed method first compiles a *disjunctive DL knowledge base* (DKB for short) in which every disjunct is a DL KB, then checks the consistent answers in all disjuncts of the DKB. However, the method is computationally hard because the compilation of a DKB usually needs exponentially many calls to a DL reasoner and the compiled DKB may have exponentially many disjuncts. Moreover, the method has not been empirically verified. [43] also proposed two adaptations of lexicographic inference to DLs. Their proposed method also needs to compile a DKB and has not been empirically verified. In another work, [42] adapted *possibilistic inference* and *linear order inference* in propositional logic [8] to DLs. However, possibilistic inference and linear order inference do not adhere to the principle of minimal change. [33] proposed an inclusion-based semantics for querying about inconsistent DL KBs as mentioned in Sect. 3. They provided a special method for DL-Lite [10,11] KBs. The method has not been empirically verified and cannot be extended to more expressive DLs such as $\mathcal{SHIQ}$. In contrast to all the above methods, our proposed method is the first one that works for the expressive DL $\mathcal{SHIQ}$, adheres to the principle of minimal change, and has been empirically verified on large $\mathcal{SHIF}$ KBs.

There is another approach to reasoning over inconsistent DL KBs, which is based on the four-valued semantics [36,39]. The basic idea is to weaken an interpretation from two truth values to four truth values so as to treat the traditional inconsistency as consistency. It results in a different reasoning mechanism for a DL KB even when the KB is traditionally consistent. Therefore, the methods following this approach are incomparable with ours.

As mentioned in Sect. 3, our proposed weight-based semantics originates from the database field. The computational methods for CQA in the database field are also based on the notion of repair program [1,2,9,13,34]. But this notion of repair program is different from ours: it is used to compute variants of a database, where these variants are models of the given integrity constants; in contrast, our notion of repair program is used to compute subsets of the ABox, where these subsets are consistent with, and may not be models of, the union of the TBox and the RBox. This means that our proposed method is not a simple adaptation of existing methods for CQA in the database field.

This work is also related to penalty logic [16], which extends propositional logic by attaching every formula with a penalty. The cautious inference problem in penalty logic can be regarded as the weight-based CQA problem studied in this paper by treating an axiom/assertion as a formula and a weight as a penalty. Although this correspondence shows possibility to apply computational methods in penalty logic to solve the weight-based CQA problem, we do not apply those methods because the computational complexity for the cautious inference problem in penalty logic is up to Δ_3^p [20], while those methods work for propositional KBs but a $\mathcal{SHIQ}$ KB cannot be directly translated to a finite and semantically equivalent propositional KB. In other words, we propose a tailored method for a variant of the cautious inference problem in penalty logic, which has a lower computational complexity.

7 Conclusions and future work

To provide a query mechanism for DL KBs that works under inconsistency, we have proposed a weight-based semantics for conjunctive query answering. This semantics works for an inconsistent $\mathcal{SHIQ}$ KB in which the union of the TBox and the RBox is consistent. We showed that the computational complexity of the proposed semantics is $\Delta_2^p[O(\log n)]$-complete in data complexity for extensionally reduced $\mathcal{SHIQ}$ KBs and atomic queries. We proposed a novel method for evaluating atomic queries in extensionally reduced $\mathcal{SHIQ}$ KBs, which is time complexity optimal and can be applied to evaluate conjunctive queries without

non-distinguished variables. We also integrated a novel decomposition-based framework into the method to make the method more scalable.

We conducted experiments on a number of inconsistent KBs. The experimental results have two implications. On the one hand, the decomposition-based framework is crucial in solving large-scale weight-based CQA problems. With this framework, the weight-based CQA problem for an atomic query can be efficiently solved in a large $\mathcal{SHIF}$ KB with hundreds of conflicts and millions of instance assertions. On the other hand, the weight-based CQA problem for a conjunctive query can still be efficiently solved for most test queries. The main reason why evaluating a small portion of test queries is so hard is that there are too many atomic queries reduced from some of the test queries.

To further improve the current method, in future work, we will investigate the possibility for adapting optimization techniques used in conjunctive query answering, such as those ones for fast identification of answers and non-answers, to decrease the number of atomic queries reduced from a given conjunctive query. Furthermore, the current method only supports $\mathcal{SHIQ}$ and cannot deal with nominals. We will also study the adaptation of the resolution-based decision procedure for $\mathcal{SHOIQ}$ [32] to solve weight-based CQA problems for DL KBs that contain nominals.

Acknowledgments We thank anonymous reviewers for their very useful comments and suggestions. Jianfeng Du is partially supported by the National Natural Science Foundation of China (NSFC) grant 61005043. Guilin Qi is partially supported by Excellent Youth Scholars Program of Southeast University under grant 4009001011, Doctoral Discipline Foundation for Young Teachers in the Higher Education Institutions of Ministry of Education (No. 20100092120029), NSFC (61003157), and the Key Laboratory of Computer Network and Information Integration (Southeast University). Yi-Dong Shen is partially supported by the National Natural Science Foundation of China (NSFC) grant 60970045.

References

1. Arenas M, Bertossi LE Chomicki J (1999) Consistent query answers in inconsistent databases. In: Proceedings of the 18th ACM symposium on principles of database systems (PODS), pp 68–79
2. Arenas M, Bertossi LE, Chomicki J (2003) Answer sets for consistent query answering in inconsistent databases. Theory Pract Logic Program 3(4–5):393–424
3. Arieli O, Denecker M, NuffelenBV Bruynooghe M (2004) Coherent integration of databases by abductive logic programming. J Artif Intell Res 21:245–286
4. Aspvall B, Plass MF, Tarjan RE (1979) A linear-time algorithm for testing the truth of certain quantified Boolean formulas. Inf Process Lett 8(3):121–123
5. Baader F, Calvanese D, McGuinness DL, Nardi D, Patel-Schneider PF (eds) (2003) The description logic handbook: theory, implementation, and applications. Cambridge University Press, Cambridge
6. Bailleux O, Boufkhad Y, Roussel O (2006) A translation of pseudo boolean constraints to SAT. J Satisf Boolean Model Comput 2:191–200
7. Benferhat S, Cayrol C, Dubois D, Lang J, Prade H (1993) Inconsistency management and prioritized syntax-based entailment. In: Bajcsy R (eds) Proceedings of the 13th international joint conference on artificial intelligence (IJCAI), pp 640–647
8. Benferhat S, Dubois D, Prade H (1995) How to infer from inconsisent beliefs without revising? In: Proceedings of the 14th international joint conference on artificial intelligence (IJCAI), pp 1449–1457
9. Bertossi LE, Chomicki J (2003) Query answering in inconsistent databases. In: Chomicki J, Meyden R, Saake G (eds) Logics for emerging applications of databases. Springer, pp 43–83
10. Calvanese D, Giacomo G, Lembo D, Lenzerini M, Rosati R (2005) DL-Lite: tractable description logics for ontologies. In: Veloso M, Kambhampati S (eds) Proceedings of the 20th national conference on artificial intelligence (AAAI), pp 602–607
11. Calvanese D, Giacomo G, Lembo D, Lenzerini M, Rosati R (2007) Tractable reasoning and efficient query answering in description logics: the DL-Lite family. J Autom Reason 39(3):385–429
12. Chai D, Kuehlmann A (2003) A fast pseudo-boolean constraint solver. In: Proceedings of the 40th design automation conference (DAC), pp 830–835

13. Chomicki J (2007) Consistent query answering: five easy pieces. In: Schwentick T, Suciu D (eds) Proceedings of the 11th international conference on database theory (ICDT), pp 1–17
14. Cimiano P (2006) Ontology learning and population from text algorithms evaluation and applications. Springer, Berlin
15. Cimiano P, Völker J (2005) Text2onto—a framework for ontology learning and data-driven change discovery. In: Montoyo A, Muñoz R, Métais E (eds) Proceedings of the 10th international conference on applications of natural language to information systems (NLDB), pp 227–238
16. de Saint-Cyr F, Lang J, Schiex T (1994) Penalty logic and its link with dempster-shafer theory. In: Mántaras R, Poole D (eds) Proceedings of the 10th annual conference on uncertainty in artificial intelligence (UAI), pp 204–211
17. Dolby J, Fokoue A, Kalyanpur A, Ma L, Schonberg E, Srinivas K, Sun X (2008) Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In: Sheth AP, Staab S, Dean M, Paolucci M, Maynard D, Finin TW, Thirunarayan K (eds) Proceedings of the 7th international semantic web conference (ISWC), pp 403–418
18. Du J, Shen Y (2008) Computing minimum cost diagnoses to repair populated DL-based ontologies. In: Huai J, Chen R, Hon H, Liu Y, Ma W, Tomkins A, Zhang X (eds) Proceedings of the 17th international world wide web conference (WWW), pp 575–584
19. Eén N, Sörensson N (2006) Translating pseudo-boolean constraints into SAT. J Satisf Boolean Model Comput 2:1–26
20. Eiter T, Gottlob G (1995) The complexity of logic-based abduction. J ACM 42(1):3–42
21. Eiter T, Gottlob G, Mannila H (1997) Disjunctive datalog. ACM Trans Database Syst 22(3):364–418
22. Eiter T, Leone N, Mateis C, Pfeifer G, Scarcello F (1997) A deductive system for non-monotonic reasoning. In: Dix J, Furbach U, Nerode A (eds) Proceedings of the 4th international conference on logic programming and nonmonotonic reasoning (LPNMR), pp 364–375
23. Feldman R, Rosenfeld B, Fresko M (2006) TEG—a hybrid approach to information extraction. Knowl Inf Syst 9(1):1–18
24. Fitting M (1996) First-order logic and automated theorem proving. 2. Springer, Secaucus
25. Guo Y, Pan Z, Heflin J (2005) LUBM: a benchmark for OWL knowledge base systems. J Web Semant 3(2–3):158–182
26. Haarslev V, Möller R (2001) Racer system description. In: Goré R, Leitsch A, Nipkow T (eds) Proceedings of the 1st international joint conference on automated reasoning (IJCAR), pp 701–706
27. Horrocks I, Patel-Schneider PF, van Harmelen F (2003) From $\mathcal{SHIQ}$ and RDF to OWL: the making of a web ontology language. J Web Semant 1(1):7–26
28. Horrocks I, Sattler U, Tobies S (2000) Practical reasoning for very expressive description logics. Logic J IGPL 8(3):239–263
29. Huang Z, van Harmelen F, ten Teije A (2005) Reasoning with inconsistent ontologies. In: Kaelbling LP, Saffiotti A (eds) Proceedings of the 19th international joint conference on artificial intelligence (IJCAI), pp 454–459
30. Hustadt U, Motik B, Sattler U (2004) Reducing $\mathcal{SHIQ}^-$ description logic to disjunctive datalog programs. In: Proceedings of the 9th international conference on principles of knowledge representation and reasoning (KR), pp 152–162
31. Hustadt U, Motik B, Sattler U (2007) Reasoning in description logics by a reduction to disjunctive datalog. J Autom Reason 39(3):351–384
32. Kazakov Y, Motik B (2008) A resolution-based decision procedure for $\mathcal{SHOIQ}$. J Autom Reason 40(2–3):89–116
33. Lembo D, Ruzzi M (2007) Consistent query answering over description logic ontologies. In: Marchiori M, Pan JZ, Marie C (eds) Proceedings of the 1st international conference on web reasoning and rule systems (RR), pp 194–208
34. Lopatenko A, Bertossi LE (2007) Complexity of consistent query answering in databases under cardinality-based and incremental repair semantics. In: Schwentick T, Suciu D (eds) Proceedings of the 11th international conference on database theory (ICDT), pp 179–193
35. Ma L, Yang Y, Qiu Z, Xie G, Pan Y, Liu S (2006) Towards a complete OWL ontology benchmark. In: Sure Y, Domingue J (eds) Proceedings of the 3rd European semantic web conference (ESWC), pp 125–139
36. Ma Y, Hitzler P, Lin Z (2007) Algorithms for paraconsistent reasoning with OWL. In: Franconi E, Kifer M, May W (eds) Proceedings of the 4th European semantic web conference (ESWC), pp 399–413
37. McDowell L, Cafarella MJ (2008) Ontology-driven, unsupervised instance population. J Web Semant 6(3):218–236
38. Meyer T, Lee K, Booth R (2005) Knowledge integration for description logics. In: Veloso M, Kambhampati S (eds) Proceedings of the 20th national conference on artificial intelligence (AAAI), pp 645–650
39. Odintsov SP, Wansing H (2003) Inconsistency-tolerant description logic: motivation and basic systems. Kluwer, Dordrecht 301–335

40. Patel-Schneider PF, Hayes P, Horrocks I (eds) (2004) OWL web ontology language semantics and abstract syntax. W3C recommendation. http://www.w3.org/TR/owl-semantics/
41. Popov B, Kiryakov A, Kirilov A, Manov D, Ognyanoff D, Goranov M (2003) Kim—semantic annotation platform. In: Fensel D, Sycara KP, Mylopoulos J (eds) Proceedings of the 2nd international semantic web conference (ISWC), pp 834–849
42. Qi G, Ji Q, Pan JZ, Du J (2011) Extending description logics with uncertainty reasoning in possibilistic logic. Int J Intell Syst 26(4):353–381
43. Qi G, Liu W, Bell D (2006) A revision-based approach to handling inconsistency in description logics. Artif Intell Rev 26(1–2):115–128
44. Rao AS, Foo NY (1989) Minimal change and maximal coherence: a basis for belief revision and reasoning about actions. In: Sridharan NS (eds) Proceedings of the 11th international joint conference on artificial intelligence (IJCAI), pp 966–971
45. Sattler K, Geist I, Schallehn E (2005) Concept-based querying in mediator systems. VLDB J 14(1): 97–111
46. Shadbolt N, Berners-Lee T, Hall W (2006) The semantic web revisited. IEEE Intell Syst 21(3):96–101
47. Shchekotykhin K, Jannach D, Friedrich G (2010) xCrawl: a high-recall crawling method for web mining. Knowl Inf Syst 25(2):303–326
48. Sheini HM, Sakallah KA (2006) Pueblo: a hybrid pseudo-boolean SAT solver. J Satisf Boolean Model Comput 2:157–181
49. Song M, Rudniy A (2010) Detecting duplicate biological entities using Markov random field-based edit distance. Knowl Inf Syst 25(2):371–387

Author Biographies

Jianfeng Du is currently an Associate Professor in Guangdong University of Foreign Studies. He received the PhD degree from the State Key Laboratory of computer science, Institute of Software, Chinese Academy of Sciences, and both the Master degree and the Bachelor degree from Sun Yet-Sen University in P.R. China. His main research interests include data mining, Semantic Web and business intelligence.

Guilin Qi is currently a full Professor in the School of Computer Science and Engineering at Southeast University. His research interests include knowledge representation and reasoning, uncertainty reasoning and Semantic Web. He got his PhD degree from Queen's University Belfast and was a postdoctoral researcher working at University of Karlsruhe. He is on the editorial board of *Journal of Web Semantics* and *Journal of Advances in Artificial Intelligence*. He has organized special issues in *Annals of Mathematics and Artificial Intelligence* and *Web Intelligence and Agent Systems*. He has involved in organization of many international conferences and workshops.

Yi-Dong Shen is a Professor of Computer Science in the State Key Laboratory of Computer Science at Institute of Software, the Chinese Academy of Sciences, China. Prior to joining this laboratory, he was a Professor at Chongqing University, China. His main research interests include knowledge representation and reasoning, Semantic Web, and data mining.