Implication and Referential Constraints: A New Formal Reasoning

Xubo Zhang* Z. Meral Ozsoyoglu†

Abstract

In this paper, we address the issue of reasoning with two classes of commonly used semantic integrity constraints in database and knowledge–base systems: implication constraints and referential constraints. We first consider a central problem in this respect, the *IRC–refuting problem*, which is to decide whether a conjunctive query always produces an empty relation on (finite) database instances satisfying a given set of implication and referential constraints.

Since the general problem is undecidable, we only consider *acyclic* referential constraints. Under this assumption, we prove that the IRC–refuting problem is decidable, and give a novel necessary and sufficient condition for it. Under the same assumption, we also study several other problems encountered in semantic query optimization, such as the semantics–based query containment problem, redundant join problem, and redundant selection–condition problem, and show that they are polynomially equivalent or reducible to the IRC–refuting problem. Moreover, we give results on reducing the complexity for some special cases of the IRC–refuting problem.

Index terms

semantic integrity constraints, implication constraints, referential constraints, query containment, semantic query optimization, refutation, database system

1 Introduction

Semantic constraints are logic rules specifying semantically meaningful database and knowledge–base instances. They constitute an integral part of most database/knowledge–base systems. In this paper we consider two important and commonly used classes of semantic constraints called implication and referential constraints. Since early 1980's, there have been investigations on the utilization of such constraints to optimize relational queries ([15], [19], [35], [5], [23] etc.), i.e. semantic query optimization.

The main tasks in semantic query optimization involve finding inconsistent queries, eliminating redundant joins and redundant selection–conditions, and adding some redundant selection–conditions that may result in more efficient query execution by permitting use of existing indices.

More recently, there have been research studies investigating a much wider range of semantically constrained problems in relational, deductive, and object-oriented databases: for example, the update problem ([11]), the redundancy problem in Datalog ([22]); the recursive query optimization problem ([21], [14], [30]); and the problem of providing intensional answers ([31], [27], [12]). The maintenance of non–redundant and consistent constraint sets is also studied ([28]).

*Xubo Zhang is with Intel Corp, 2625 Walsh Avenue, MS SC4-103, Santa Clara, CA 95051. Email: Xubo_Zhang@ccm.sc.intel.com

†Z. Meral Ozsoyoglu is with The Dept. of Computer Eng. and Sci., Case Western Reserve University, Cleveland, OH 44106. Email:ozsoy@alpha.ces.cwru.edu

In this paper, we discuss one basic problem that often appears in these studies: the problem of deciding whether a conjunctive query always produces an empty relation on database instances satisfying a given set of implication and referential constraints. We call this problem the IRC–refuting problem (IRC stands for implication and referential constraints). One of the major contributions of this paper is to reveal the relationship between the IRC–refuting problem and some other important problems, for example, the constraint–based query containment problem. Before going into the technical development, let us consider an example.

1.1 An Example

Example 1.1

We will use the following simple *university database* as a running example to illustrate the results in this paper. It consists of the following components:

1. two relation schemes: **dept(dname, chair)** and **faculty(fname, salary, status)**;
2. an implication constraint stating that there can not be any faculty member who is a chair of two different departments:

$$ic_1: \ \forall(D_1, D_2, C, S, T)\ [(\mathbf{dept}(D_1, C),\ \mathbf{dept}(D_2, C),\ \mathbf{faculty}(C, S, T),\ D_1 \neq D_2) \rightarrow false]^1;$$

3. a referential constraint stating that the chair of a department is also a faculty member:

$$rc_1: \ \forall(D, C)\ [\mathbf{dept}(D, C)\ \rightarrow\ \exists(S, T)\ \mathbf{faculty}(C, S, T)].$$

Now, let us ask a query to "list the person who is a chair of both the physics department and the chemistry department", i.e.,

$$Q = \{\langle C\rangle :\ \mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C)\}.$$

Assuming the constraints have been enforced, i.e., only relation instances satisfying ic_1 and rc_1 can be stored in the database, the answer for Q is always empty, because if there is such a person, then he/she must be a faculty member chairing two different departments. This (empty) answer as we will see later, can be obtained without searching the actual database, but by *refuting* the query (formula) using the constraints, which is most likely to be time–saving with real world applications, where the size of the database is large (millions of records), queries are simple, and the size of the constraint base is very small in comparison to the size of the database.

We can look at the problem from another perspective: if we view Q as defining an implication constraint stating that "there is no person chairing both the physics and the chemistry department", i.e.,

$$ic: \ \forall C\ [\mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C)\ \rightarrow\ false].$$

We know that ic is always true if ic_1 and rc_1 are true, because ic_1 and rc_1 logically imply ic. Thus ic is *redundant* with respect to the existing constraints. Solving the constraint redundance problem helps us to properly maintain constraint bases.

[1]Note that ic_1 states that it is not satisfiable by any database instance which has one faculty member chairing two different departments.

More interestingly, we can also view ic_1 as defining a query Q_1, asking "the faculty members who are chairs of two different departments", i.e.,

$$Q_1 = \{\langle C\rangle : \ \mathbf{dept}(D_1, C),\ \mathbf{dept}(D_2, C),\ \mathbf{faculty}(C, S, T),\ D_1 \neq D_2\},$$

then we can see that the result of Q_1 always *contains* the result of Q, when applied to any database instance satisfying rc_1.

□

¿From the above example we can see the relationship between the IRC–refuting problem, the (semantics–based) query containment problems, and the redundant implication constraint problem.

1.2 Related Works and Our Contributions

Although the IRC–refuting problem has been addressed by several authors in different contexts ([5], [35], [39] etc.), our result differs from the previous ones in the following aspects:

1. Our result is a necessary and sufficient one;
2. Our problem definition is most general in that
   (a) we consider both implication constraints and referential constraints;
   (b) we allow multiple predicates with the same name in the implication constraints.

Solving the IRC–refuting problem allows us to decide unsatisfiable queries without actually searching the databases. Moreover, as will be shown, solving the IRC–refuting problem enables us to solve, in the context of semantically constrained database instances, the following problems: query containment problem for conjunctive queries ([4], [1], [2], [33], [32], [16], [17], [20], [37], [38], [9], [18], [40]); redundant join problem; redundant selection–condition problem, and redundant implication constraint problem ([5], [35], [39] etc.).

Note that referential constraints essentially express the *inclusion dependencies* (INDs), and implication constraints greatly generalize the *functional dependencies* (FDs) in that more than one relation and inequality may be involved. In [17], the query containment problem under FDs and INDs is studied. In this paper we propose a new method different than the *chase* process to solve the query containment problem under ICs and RCs, and prove it to be polynomially equivalent to the IRC-refuting problem.

Since the decision problem for INDs and FDs are known to be not k-ary axiomatizable ([3]), and undecidable ([7] [26]), we make the assumption that the referential constraints considered in this paper are *acyclic* ([34], [8]).

The main contributions of this paper are as follows: under the assumption that referential constraints are acyclic,

(1) We give novel (non-axiomatic) necessary and sufficient conditions for the IRC–refuting problem and the semantics–based query containment problem.

(2) We establish the polynomial equivalence relation between the IRC–refuting problem, (semantics–based) query containment problem, and the redundant implication constraint problem.

(3) We give polynomial reductions from the redundant selection–condition problem and the redundant join problem to the IRC–refuting problem.

(5) We also give novel characterizations on the reduced complexity for some special cases of the IRC–refuting problem.

The rest of the paper is organized as follows:

- In Section 2 we give preliminary notations.
- In Section 3 we study the IRC–refuting problem.
- In Section 4 we study the query containment problem, redundant implication constraint problem, redundant selection–condition problem, and the redundant join problem.
- In Section 5 we discuss a special case of the IRC–refuting problem, and give some sufficient conditions to reduced the complexity.
- In Section 6 we conclude our work and propose some future research topics.

2 Preliminaries

In this section we briefly present most of the concepts and notations that are used throughout the paper. First of all, we use a partially interpreted first order language. There is a finite set of types

$$\mathcal{T} = \{T_1, ..., T_d\} \quad (d \geq 1)$$

where each T_i has a fixed interpretation (*domain*, universe) which is either *the set of real numbers*, $\mathbb{R}$, or an unordered (finite or infinite) set of constants, Σ. Note $\mathbb{R}$ is a *dense and totally ordered* domain with the transitive and irreflexive ordering $<$.

For each type there is an infinite (distinct) set of variables and constants (we just use the constants in its domain). Variables and constants of different types are assumed to be incomparable in the same (in)equalities even if they have the same domain, for example, a department's name and an employee's name.

A *database schema* is a set of *relation schemes*. A relation scheme is denoted as

$$\mathbf{p}(\mathbf{att}_1, ..., \mathbf{att}_n),$$

where $\mathbf{p}$ is the name of the relation (predicate), $n \geq 1$ is the *arity*, and $\mathbf{att}_i$'s are *attributes*. Each $\mathbf{att}_i$ has a type in $\mathcal{T}$, denoted as $T(\mathbf{p}[i])$. By an *instance* of $\mathbf{p}$ we mean a subset of

$$Dom(\mathbf{p}[1]) \times ... \times Dom(\mathbf{p}[n]),$$

where $Dom(\mathbf{p}[i])$ is the domain (interpretation) of $T(\mathbf{p}[i])$. A database (instance) is a collection of instances for its relation schemes.

A *conjunct* (ordinary subgoal) is an atomic formula of the form

$$\mathbf{p}(u_1, ..., u_n),$$

where $\mathbf{p}$ is an n-ary relation name, and each u_i is either a variable or a constant of $T(\mathbf{p}[i])$.

An *(in)equality* (built-in subgoal) is a formula of the form $(u_1\ op\ u_2)$, where both u_1 and u_2 can be constants or variables, and are of the same type, called the type of the (in)equality, and $op \in \{<, \leq, >, \geq, =, \neq\}$ if the domain is $\mathbb{R}$, or $op \in \{=, \neq\}$ if the domain is Σ. We write op's, variables, constants, and function symbols without subscripts designating their types, as they can be understood from context.

A (conjunctive) *query* is a formula of the form

$$\{\langle o_1, ..., o_k\rangle : \exists Y_1, ..., \exists Y_l[t_1, ..., t_m, c_1, ..., c_n]\},$$

where $k, l, n \geq 0$, $m \geq 1$, and

1. $\langle o_1, ..., o_k\rangle$ is called *summary*, where each o_i is either a variable, called *distinguished variable* (dv), or a constant. We define the *type* of the query to be $T(o_1) \times ... \times T(o_k)$, where $T(o_i)$ is the type of o_i.
2. Y_i is called a *nondistinguished variable* (ndv) which is not a dv.
3. t_i is a conjunct; c_i is an (in)equality; commas between t_i's and c_j's denote logical "and".
4. Every variable (dv or ndv) must occur in a conjunct, or can be (transitively) equated either to a variable appearing in a conjunct or to a constant.

For example, $\{\langle C\rangle :\ \mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C)\}$ is a query, asking for the person who is a chair of both the physics department and the chemistry department. We only consider queries whose (in)equality subformulas (if any) are satisfiable, because otherwise they always produce empty results. The result of a query on a database instance $\mathbf{D}$, denoted as $Q(\mathbf{D})$, is the set of all the instantiations of the summary such that the query formula is satisfied.

Historically, conjunctive queries are called *equality queries* if their built–in subformulas consisting only of equalities; they are called *(in)equality queries* if both equalities and inequalities are present ([20]). Without other indications, the term *query* in this work refers to (in)equality queries.

A *union query* is a formula of the form

$$union(Q_1, ..., Q_r),$$

where $Q_1, ..., Q_r$ $(r \geq 1)$ are queries of the same type. The result of the above query on a database instance $\mathbf{D}$ is

$$Q_1(\mathbf{D}) \cup ... \cup Q_r(\mathbf{D}).$$

It is clear that $union(Q_1) = Q_1$.

An *implication constraint* (IC) is a formula of the form

$$\forall \mathbf{V}[(t_1, ..., t_m, c_1, ..., c_n)\ \rightarrow\ false],$$

or simply

$$t_1, ..., t_m, c_1, ..., c_n \rightarrow,$$

where $\mathbf{V}$ denotes the free variables (ndv's) in t_i's and c_i's, and $t_1, ..., t_m, c_1, ..., c_n$ is in the same format as the body of a query. For example,

$$ic_1 :\ \mathbf{dept}(D_1, C),\ \mathbf{dept}(D_2, C),\ \mathbf{faculty}(C, S, T), D_1 \neq D_2\ \rightarrow$$

is an implication constraint, stating that there is no faculty member who is a chair of two different departments. As with queries, we also assume that the (in)equality subformula (if any) is satisfiable.

A *referential constraint* (RC) is a formula of the form

$$\forall(X_1, ..., X_n)\ [\mathbf{p}(X_1, ..., X_n)\ \rightarrow\ \exists(Y_1, ..., Y_l)\mathbf{q}(u_1, ..., u_m)],$$

where $n, m \geq 1$, $l \geq 0$, $\mathbf{p}$ and $\mathbf{q}$ are different predicate names, $X_1, ..., X_n, Y_1, ..., Y_l$ are different variables, and each u_i is a variable from $\{X_1, ..., X_n, Y_1, ..., Y_l\}$ or a constant. For example,

$$rc_1:\ \forall(D, C)\ [\mathbf{dept}(D, C)\ \rightarrow\ \exists(S, T)\ \mathbf{faculty}(C, S, T)].$$

is a referential constraint, stating that the chair of a department is also a faculty member.

Note that we require X_1, ..., X_n to be distinct variables, for effectively dealing with the interaction between referential and implication constraints. This is slightly more restrictive than the inclusion dependencies.

Throughout this paper we will consider databases in which semantic integrity constraints are part of the system. We say a database instance $\mathbf{D}$ satisfies a set of semantic constraints (implication and referential constraints), if $\mathbf{D}$ is a (finite) *model* [2] for the constraints in the set. Note here that all database instances in reality are finite.

Notations. In our database language, we denote variables by upper–case letters, denote relation and attribute names by lower–case strings in boldface, and denote constants by lower–case strings. Symbol mappings (Section 3) are denoted by Greek letters, ρ, θ, etc. Skolem function symbols ([6]) are denoted by h, g, etc., possibly having subscripts.

Queries and implication constraints are often written in an abstracted way, like

$$\{\langle\mathbf{O}\rangle:\ F,\ I\}$$

and

$$F,\ I\ \rightarrow$$

respectively, in which $\langle\mathbf{O}\rangle$ is the summary, F is the conjunct subformula, and I is the (in)equality subformula. For RCs, we often use their Skolemized (unquantified) forms.

For example, the Skolemized form for rc_1 in Example 1.1 is:

$$\mathbf{dept}(D, C)\ \rightarrow\ \mathbf{faculty}(C, h_s(D, C), h_t(D, C)),$$

where h_s and h_t are unique Skolem function symbols.

3 The IRC–Refuting Problem

In this section we study our first major problem in reasoning with implication and referential constraints: the IRC–refuting problem. As shown in the first example, the IRC–refuting problem is to decide whether a query always produces an empty result when a given set of implication and referential constraints are enforced. We use the IRC–refuting problem as the basis of our formal treatment of semantic constraints.

[2] in the sense of the model in first order logic.

Definition 3.1 (IRC–Refuting)
Given a non–empty set $\mathcal{I}_\mathcal{C}$ of ICs, a set $\mathcal{R}_\mathcal{C}$ of RCs, and a query Q, the IRC–refuting problem is to decide whether

$$(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models_f (Q \equiv \{\}),$$

i.e., whether Q always produces an empty relation on any finite database instance satisfying $\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}$.

To solve the IRC–refuting problem we first *expand* the original query using referential constraints, and then look for certain *symbol mappings* ([20] and [39]) from the implication constraints to the query.

3.1 Referential Expansions

Definition 3.2 (Referential Expansion)
Let $\mathcal{R}_\mathcal{C}$ be a set of (Skolemized) RCs and $Q = \{\langle \mathbf{O} \rangle : F, I\}$. The referential expansion *of Q (by $\mathcal{R}_\mathcal{C}$) is a formula of the form $\{\langle \mathbf{O} \rangle : F^{re}, I\}$, where F^{re} is the set (conjunction) of atoms defined as follows:*

1. *$F \subseteq F^{re}$;*
2. *for any $t \in F^{re}$ and $(t_1 \rightarrow t_2) \in \mathcal{R}_\mathcal{C}$, if there exists a substitution ρ such that $\rho(t_1) = t$, then $\rho(t_2) \in F^{re}$.*

Example 3.1
Using referential constraint

$$rc_1 : \ \mathbf{dept}(D, C) \ \rightarrow \ \mathbf{faculty}(C, h_s(D, C), h_t(D, C)),$$

the referential expansion of query

$$Q = \{\langle C \rangle : \ \mathbf{dept}(physics, C), \ \mathbf{dept}(chemistry, C)\}$$

in Example 1.1 is

$$Q^{re} = \{\langle C \rangle : \mathbf{dept}(physics, C), \ \mathbf{dept}(chemistry, C), \ \mathbf{faculty}(C, h_s(physics, C),$$

$$h_t(physics, C)), \ \mathbf{faculty}(C, h_s(chemistry, C), h_t(chemistry, C))\}.$$

□

Following is the procedure to produce the referential expansions, where the input parameter Q gives the query to be expanded; input parameter $\mathcal{R}_\mathcal{C}$ gives the set of referential constraints; and output parameter Q^{re} gives the referential expansion.

Procedure Expand$(Q, Q^{re}, \mathcal{R}_\mathcal{C})$
begin
$F^{re} \leftarrow$ the set of conjuncts in Q
mark all elements in F^{re} as *new*
while there is a conjunct in F^{re} that is *new*
$t \leftarrow$ a *new* conjunct in F^{re}

for every $(t_1 \rightarrow t_2) \in \mathcal{R}_\mathcal{C}$ **do**

find a substitution ρ such that $\rho(t_1) = t$

if such ρ exists **and** $\rho(t_2) \notin F^{re}$

then mark $\rho(t_2)$ as *new*

$F^{re} \leftarrow F^{re} \cup \{\rho(t_2)\}$

mark t as *old*

$Q^{re} \leftarrow$ query Q with the F replaced by F^{re}

end

□

The above procedure may, however, not terminate. For example, if we have two referential constraints like

$$\mathbf{p}(X) \rightarrow \mathbf{q}(X, h(X))$$

and

$$\mathbf{q}(X, Y) \rightarrow \mathbf{p}(Y),$$

and a query $\{\langle X \rangle : \mathbf{p}(X)\}$, then infinitely many conjuncts of the form

$$\mathbf{p}(h^n(X)) \quad (n \geq 1)$$

are in the body of the referential expansion of this query.

In fact, the general termination problem for the rewriting system obtained from the RCs is undecidable, and there are many ways to put restrictions which guarantee the terminating property ([10]). The discussion of such choices is out of the scope of this paper. Instead, we make the assumption that the sets of referential constraints considered in this paper are acyclic ([34], [8]), which guarantees termination.

The above procedure has some similarity with the *chase* process used in [17]. There are, however, two major differences: 1) We do not use the ICs in the expansion, whereas in chase FDs are also used. The reason is while FD's can be naturally used in chase to equate variables, IC's in general reflect inequalities. 2) We use Skolem function symbols to keep track of the levels of expansions, which will enable us to use resolution–like proofs.

Definition 3.3 (Acyclicity)
A set $\mathcal{R}_\mathcal{C}$ of (Skolemized) RCs is acyclic, if there are no distinct predicate names $\mathbf{p}_1$, ..., $\mathbf{p}_n$ ($n > 1$) such that $\mathcal{R}_\mathcal{C}$ contains

$\mathbf{p}_1(\mathbf{u}_1) \rightarrow \mathbf{p}_2(\mathbf{v}_2)$, $\mathbf{p}_2(\mathbf{u}_2) \rightarrow \mathbf{p}_3(\mathbf{v}_3)$, ..., *and* $\mathbf{p}_n(\mathbf{u}_n) \rightarrow \mathbf{p}_1(\mathbf{v}_1)$.

The above condition can be polynomially checked (e.g. by a breadth-first-search like procedure). With this property, we can see that the number of conjuncts in a referential expansion will not exceed the number of the original conjuncts times the number of RCs.

In defining the IRC–refuting problem we use the notion of finite implication (denoted $\models_f$), instead of implication (denoted $\models$), as the former is suitable in database context.

These two notions generally do not coincide. However our next lemma shows that if the RCs are acyclic, then finite implication (finite unsatisfiability) and implication (unsatisfiability) do coincide and are decidable. In the subsequent proofs, we will construct the finite models (sets of ground facts that satisfy the formulas).

Lemma 3.1

Let Q be a query, and $\mathcal{IRC}$ be a set of ICs and RCs such that the RCs are acyclic. Then $\mathcal{IRC} \models_f (Q \equiv \{\})$ if and only if $\mathcal{IRC} \models (Q \equiv \{\})$.

Proof.

Suppose $Q = \{\langle \mathbf{O} \rangle : F, I\}$, and $\mathcal{IRC} = \{ic_1, ..., ic_r, rc_1, ..., rc_s\}$ $(r, s \geq 0)$. Let

$$\Psi = ic_1 \wedge ... \wedge ic_r \wedge rc_1 \wedge ... \wedge rc_s \wedge \exists \mathbf{V}[F, I],$$

where $\mathbf{V}$ are the dv's and ndv's in Q. $\mathcal{IRC} \models_f (Q \equiv \{\})$ is equivalent to saying that Ψ is finitely unsatisfiable, and $\mathcal{IRC} \models (Q \equiv \{\})$ is equivalent to saying that Ψ is unsatisfiable. Clearly if Ψ is unsatisfiable then it is finitely unsatisfiable.

Now suppose $\mathbf{M}$ is a model (finite or not) for Ψ, then it is a model for $\Psi' = rc_1 \wedge ... \wedge rc_s \wedge \exists \mathbf{V}[F, I]$. Since rc_i's are acyclic, there must exist a finite submodel $\mathbf{M}'$ of $\mathbf{M}$ for Ψ', which consists of a number of (ground) facts in $\mathbf{M}$ that is no more than the number of conjuncts in F times the number of RCs. Since any subset of $\mathbf{M}$ is a model for $ic_1 \wedge ... \wedge ic_r$, $\mathbf{M}'$ is a finite model for Ψ.

□

Due to the above lemma, from now on, we do not make distinction between finite implication and implication when discussing the IRC–refuting problem, and just use the notation $\models$ instead of $\models_f$.

3.2 Query Forms and Symbol Mappings

Before giving a necessary and sufficient condition for the IRC–refuting problem, we need to define the notions of compressed form, normal form, and symbol mappings.

We call a query $Q = \{\langle \mathbf{O} \rangle : F, I\}$ in *normal form* ([18]), if there are only distinct occurrences of variables in the subformula $\{\langle \mathbf{O} \rangle : F\}$. Q is said to be in *compressed form* if there are no explicit or implied equalities in I. The normal and compressed forms for implication constraints are defined similarly if we view IC's as yes/no–queries. Note that any query or IC can be polynomially rewritten into normal or compressed forms.

Example 3.2

We can view ic_1 in Example 1.1 as the following compressed yes/no–query,

$$\{\langle\rangle : \ \mathbf{dept}(D_1, C), \ \mathbf{dept}(D_2, C), \ \mathbf{faculty}(C, S, T), \ D_1 \neq D_2\},$$

and a normal form of the above query is

$$\{\langle\rangle : \ \mathbf{dept}(D_1, C_1), \ \mathbf{dept}(D_2, C_2), \ \mathbf{faculty}(C_3, S, T), \ D_1 \neq D_2, \ C_1 = C_2, \ C_1 = C_3\}.$$

□

Definition 3.4 (Symbol Mapping)

Let Q_1 be a query and Q be a query or a referential expansion of a query. A symbol mapping ρ from Q_1 to Q is a function *from the set of symbols (variables and constants) in Q_1 to the set of terms in Q that satisfies the following conditions:*

1. *it is an identity on constants and function symbols;*

2. *it induces a mapping that maps the summary of Q_1 to that of Q;*

3. *it induces a mapping from the set of conjuncts of Q_1 to that of Q.*

A symbol mapping (which is just a substitution satisfying the second and third conditions above) also induces a mapping on (in)equalities. For a symbol mapping ρ, and a conjunction of (in)equalities I, we write $\rho(I)$ to denote the formula obtained from I under the mapping induced from ρ. $\rho(I)$ is empty if I is empty (empty set of (in)equalities means *true*).

Note that $\rho(I)$ is a conjunction of (in)equalities of the form u_1 *op* u_2 where u_1 and u_2 are *terms* possibly having (Skolem) function symbols. Hereafter by (in)equalities we mean such generalized (in)equalities. Symbol mappings from implication constraints to queries are similarly defined, except the second condition above is not applicable.

Example 3.3
Consider queries Q_1 and Q^{re} from previous examples, where

$$Q_1 = \{\langle C\rangle : \ \mathbf{dept}(D_1, C),\ \mathbf{dept}(D_2, C),\ \mathbf{faculty}(C, S, T),\ D_1 \neq D_2\},$$

and

$$Q^{re} = \{\langle C\rangle : \mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C),\ \mathbf{faculty}(C, h_s(physics, C),$$
$$h_t(physics, C)),\ \mathbf{faculty}(C, h_s(chemistry, C), h_t(chemistry, C))\}.$$

Then one of the symbol mappings from Q_1 to Q^{re} is as follows:

$$\{C \mapsto C,\ D_1 \mapsto physics,\ D_2 \mapsto chemistry,\ S \mapsto h_s(physics, C),\ T \mapsto h_t(physics, C)\}$$

□

3.3 Main Result on the IRC–Refuting Problem

One of the main contributions of this paper is to give a necessary and sufficient condition for the IRC–refuting problem, which is stated in the following theorem.

Theorem 3.1 (IRC–Refuting)
Suppose we are given the following:

1. *a set of implication constraints $\mathcal{I}_\mathcal{C} = \{ic_1, ..., ic_r\}$ $(r \geq 1)$ such that the ic_i's are in normal form;*

2. *an acyclic set of referential constraints $\mathcal{R}_\mathcal{C}$;*

3. *a query $Q = \{\langle \mathbf{O}\rangle : F, I\}$, where $\langle \mathbf{O}\rangle$ is the summary, and F and I are the conjunct and the (in)equality subformulas respectively;*

4. *referential expansion Q^{re} of Q with respect to $\mathcal{R}_\mathcal{C}$.*

Then $(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models (Q \equiv \{\})$ *if and only if there exist* $(n_1 + ... + n_r) \geq 1$ *symbol mappings:*

$\rho_{i,1}, ..., \rho_{i,n_i}$ *from* ic_i *to* Q^{re} *(for* $i = 1, ..., r$*)*

such that I *implies* $\bigvee_{(1 \leq i \leq r,\ 1 \leq j \leq n_r)} \rho_{i,j}(I_i)$*, where* $I_1, ..., I_r$ *are the (in)equality subformulas of* $ic_1, ..., ic_r$ *respectively.*

Proof.

Let $Q = \{\langle \mathbf{O} \rangle : F, I\}$ and $Q^{re} = \{\langle \mathbf{O} \rangle : F^{re}, I\}$.

First we prove that

$$(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models (Q \equiv \{\})$$

if and only if

$$\mathcal{I}_\mathcal{C} \cup \{\exists \mathbf{V}[F^{re}, I]\}$$

is unsatisfiable, where $\mathbf{V}$ represents the variables (dv's and ndv's) of Q (or Q^{re}).

Since $\exists \mathbf{V}[F^{re}, I]$ is a logical consequence of the RCs in $\mathcal{R}_\mathcal{C}$ and $\exists \mathbf{V}[F, I]$, so if $\mathcal{I}_\mathcal{C} \cup \{\exists \mathbf{V}[F^{re}, I]\}$ is unsatisfiable, then $\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C} \cup \{\exists \mathbf{V}[F, I]\}$ is also unsatisfiable, which means that $(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models (Q \equiv \{\})$.

On the other hand, if $\mathcal{I}_\mathcal{C} \cup \{\exists \mathbf{V}[F^{re}, I]\}$ is satisfiable, then there exists a database instance $\mathbf{D}$ satisfying $\mathcal{I}_\mathcal{C}$, and an interpretation ϕ for the atoms in F^{re} making $\exists \mathbf{V}[F^{re}, I]$ true in $\mathbf{D}$. By keeping only those tuples in $\mathbf{D}$ that are the images of some atoms in F^{re} under ϕ, we get a model for

$$\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C} \cup \{\exists \mathbf{V}[F, I]\},$$

which means that $(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \not\models (Q \equiv \{\})$.

Let $ic_i = (F_i, I_i \rightarrow)$ for $i = 1, ..., r$. And let $\mathbf{V}$ be the variables (dv's and ndv's) in Q. By the above reasoning,

$$(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models (Q \equiv \{\})$$

if and only if the following (sets) of clauses are unsatisfiable:

$$F^{re}(\mathbf{b}),$$

$$I(\mathbf{b}),$$

and

$$(\neg F_i \ \vee \ \neg I_i) \ \text{ for } \ i = 1, ..., r,$$

where $F^{re}(\mathbf{b})$ and $I(\mathbf{b})$ are respectively F^{re} and I with $\mathbf{V}$ being replaced by a vector of Skolem constant symbols $\mathbf{b}$. Therefore we can use resolutions with paramodulation ([6]) on conjunct literals to get an empty clause or an unsatisfiable set of resolvents consisting only of (in)equalities. Paramodulation ("replace equal with equal") in general is needed so that unification between ordinary conjuncts using provable equalities can proceed. Notice that an equality $X = Y$ maybe obtained from two (single-literal) inequalities $X \geq Y$ and $Y \geq X$.

However, since here all ic_i's are in normal form, every possible unification between conjunct literals can be performed without paramodulations. Moreover, the unifications here actually are substitutions. The substitutions that produce the empty clause or unsatisfiable set of (in)equalities constitute the symbol mappings as specified in the theorem. This leads to the conclusion of the theorem.

□

Example 3.4

Continued from Example 1.1, where the referential expansion of query Q is

$$Q^{re} = \{\langle C\rangle : \mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C),\ \mathbf{faculty}(C, h_s(physics, C),$$

$$h_t(physics, C)),\ \mathbf{faculty}(C, h_s(chemistry, C), h_t(chemistry, C))\}.$$

We first put the implication constraint ic_1 into normal form:

$$ic_1' :\ \mathbf{dept}(D_1, C_1),\ \mathbf{dept}(D_2, C_2),\ \mathbf{faculty}(C, S, T),\ D_1 \neq D_2,\ C_1 = C,\ C_2 = C \rightarrow .$$

Then

$$\{D_1 \mapsto physics,\ C_1 \mapsto C,\ D_2 \mapsto chemistry,\ C_2 \mapsto C,\ C \mapsto C,$$

$$S \mapsto h_s(physics, C),\ T \mapsto h_t(physics, C)\}$$

is a symbol mapping from ic_1' to Q^{re}, such that the (in)equality subformula of ic_1' is mapped to

$$physics \neq chemistry,\ C = C,\ C = C,$$

which is *true*. So by Theorem 3.1, we have

$$\{ic_1,\ rc_1\} \models (Q \equiv \{\}).$$

In this example, the implication relation between (in)equality subformulas is quite obvious, we will, however, see more complicated cases in the following subsection, where the general problem (called subsumption problem) is discussed. □

3.4 The Subsumption Problem

In this subsection we discuss the subproblem specified in Theorem 3.1, i.e., the problem of deciding whether I implies $I_1 \vee ... \vee I_n$, for (in)equality subformulas I, I_1, ..., and I_n. It is called the subsumption problem in the literature (e.g.[36]). Notice here $I, I_1, ..., I_n$ are propositional clauses.

Clearly, I implies $I_1 \vee ... \vee I_n$ if and only if the set of (in)equality clauses

$$\{I, \neg I_1, ..., \neg I_n\}$$

is unsatisfiable. So the subsumption problem can be solved by testing the satisfiabilities of a number of conjunctions of (in)equalities. When there is no function symbol in the (in)equalities, the satisfiability of a conjunction of (in)equalities is polynomially solvable ([28], [20]). An $O(n^3)$ algorithm is given in [28] to decide the satisfiability, where n is the number of variables and constants appearing in the conjunction. In this subsection we discuss the satisfiability of a conjunction of (in)equalities having function symbols.

Let $\mathcal{C}$ be a set (conjunction) of (in)equalities (of different types), and $Term(\mathcal{C})$ be the set of terms appearing in $\mathcal{C}$. Without losing generality, we make two assumptions:

1) the (in)equality predicate names are from $\{<, \leq, =, \neq\}$, and

2) for any pair of terms u_1 and u_2 in $Term(\mathcal{C})$, there is at most one (in)equality (of the forms $u_1\ op\ u_2$ or $u_2\ op\ u_1$) in $\mathcal{C}$.

The second assumption is based on the observation that more than one (in)equalities between two terms are either mergeable, or inconsistent ([29]).

Next we define a binary relation $\preceq$ on $Term(\mathcal{C})$ as follows.

Definition 3.5

Let $\mathcal{C}$ be a set of (in)equalities. The binary relation $\preceq$ on $Term(\mathcal{C})$ is defined as follows:

1) $u_1 \preceq u_2$ if $u_1 \leq u_2$, $u_1 = u_2$, or *$u_2 = u_1$ is in $\mathcal{C}$;*

2) $u_1 \preceq u_2$ if $u_1 \preceq u$ and *$u \preceq u_2$.*

For $\mathcal{C}$ to be satisfiable, there must exist an assignment of the values in the corresponding domains for the terms in $\mathcal{C}$, such that the (in)equalities in $\mathcal{C}$ are made true. The next definition of a *congruence relation* $\cong$ on $Term(\mathcal{C})$ specifies what terms must be assigned identical values in any of such assignments.

Definition 3.6

Let $\mathcal{C}$ be a set of (in)equalities. The binary relation $\cong$ on $Term(\mathcal{C})$ is defined as follows:

1. *$u \cong u$ for any $u \in Term(\mathcal{C})$;*
2. *$u_2 \cong u_1$ if $u_1 \cong u_2$;*
3. *$u_1 \cong u_3$ if $u_1 \cong u_2$* and *$u_2 \cong u_3$;*
4. *$h(u_1, ..., u_n) \cong h(u'_1, ..., u'_n)$ if $u_i \cong u'_i$ for $i = 1, ..., n$;*
5. *$u_1 \cong u_2$ if $u_1 \preceq u_2$* and *$u_2 \preceq u_1$.*

Conditions 1, 2, and 3 make $\cong$ an equivalence (reflexive, symmetric, and transitive) relation. Condition 4 is the *compatibility property*. Condition 5 defines members of $\cong$ based on $\preceq$–relation. Since $\cong$ is an equivalence relation on $Term(\mathcal{C})$, we can partition $Term(\mathcal{C})$ into $\cong$–equivalence classes. Partitioning $Term(\mathcal{C})$ into $\cong$–equivalence classes is illustrated in Example 3.5. The following theorem gives the necessary and sufficient conditions for satisfiability of a set $\mathcal{C}$ of (in)equalities under the two assumptions discussed previously.

Theorem 3.2

A set $\mathcal{C}$ of (in)equalities (satisfying the two assumptions) is satisfiable if and only if the following conditions are true:

1. *there are no distinct constants b_1 and b_2 in $Term(\mathcal{C})$ such that $b_1 \cong b_2$;*
2. *there is no inequality of the form $(u_1 \neq u_2)$ or $(u_1 < u_2)$ in $\mathcal{C}$ such that $u_1 \cong u_2$;*
3. *there is no "cycle" consisting of (in)equalities of the forms $u\ op_1\ u_1$, $u_1\ op_2\ u_2$, ..., $u_n\ op_n\ u$ $(n \geq 1)$, such that at least one op_i is $<$. Here op_i's are in $\{<, \leq, =\}$, and each inequality is either in $\mathcal{C}$ or a strict inequality tautology between constants (like $3 < 5$).*

Remark. The (in)equalities in $\mathcal{C}$ can be from either ordered or unordered domains.

Proof.
If $\mathcal{C}$ is satisfiable, then there is a constant preserving function

$$\alpha : \ Term(\mathcal{C}) \rightarrow (\mathbb{R} \cup \Sigma)$$

such that $\mathcal{C}$ is made satisfied, and α has the compatibility property, i.e., if $\alpha(u_i) = \alpha(u_i')$ for $i = 1, ..., n$ $(n \geq 1)$, then $\alpha(h(u_1, ..., u_n)) = \alpha(h(u_1', ..., u_n'))$ (if both are defined). By structural induction, it is not difficult to prove that if $u_1 \cong u_2$ then $\alpha(u_1) = \alpha(u_2)$. Therefore, conditions 1), 2), and 3) must be true.

Reversely, suppose the conditions 1), 2), and 3) are true. Let $\mathcal{V}$ be the set of (all types of) variables, β be a constant preserving function from $Term(\mathcal{C})$ to $\mathcal{V} \cup \mathbb{R} \cup \Sigma$, such that

1. a non–constant term is mapped to a variable of the same type;
2. $\beta(u_1) = \beta(u_2)$ if $u_1 \cong u_2$, $\beta(u_1) \neq \beta(u_2)$ otherwise.

Let

$$\mathcal{C}_1 = \{(\beta(u_1) \ op \ \beta(u_2)) | (u_1 \ op \ u_2) \in \mathcal{C}\} \cup \{(\beta(u_1) \neq \beta(u_2)) | u_1 \in \mathcal{C}, u_2 \in \mathcal{C}, u_1 \ncong u_2\}.$$

By the result in [28], $\mathcal{C}_1$ is satisfiable. So there is a constant preserving function

$$\gamma : \ Term(\mathcal{C}_1) \rightarrow (\mathbb{R} \cup \Sigma)$$

such that $\mathcal{C}_1$ and thus $\mathcal{C}$ are made satisfied. By the constructions of β and γ, $\gamma\beta$ has the compatibility property.
□

Example 3.5
Let $\mathcal{C} = \{X = 2, \ Y = abc, \ h_1(X) = h_2(2, Y), \ h_1(2) < h_2(2, abc)\}$. Using only the "=" predicate, we can see there are three $\cong$–equivalence classes in $\mathcal{C}$:

$$\{X, \ 2\},$$

$$\{Y, \ abc\},$$

and

$$\{h_1(X), \ h_1(2), \ h_2(2, Y), \ h_2(2, abc)\}.$$

However from Theorem 3.2, we can immediately conclude that $\mathcal{C}$ is unsatisfiable, because $h_1(2) < h_2(2, abc)$ is in $\mathcal{C}$, which inequates two terms in the same $\cong$–equivalence class.
□

The satisfiability of a conjunction of (in)equalities can be decided in polynomial time. In fact, it is not difficult to construct an $O(n^4)$ algorithm for this problem by adapting the algorithm in [28], where n is the number of variables, constants, and function symbols in the conjunction.

4 Other Problems in Semantic Query Optimization and Constraint Maintenance

We have already discussed one important problem, i.e., the IRC–refuting problem, in semantic query processing. In this section, we will apply the method and results about the IRC–refuting problem to the following problems found in semantic query optimization and constraint maintenance:

1. the query containment problem;
2. the redundant implication constraint problem;
3. the redundant selection–conditions problem;
4. the redundant conjunct problem.

Again, the acyclic assumption is made to ensure the coincidence of finite and unrestricted implication and the decidability of the problems. We first give a theorem similar to Theorem 3.1 to solve the semantics–based query containment problem, and then show that it is polynomially equivalent to the IRC–refuting problem. After that we prove that the redundant IC problem is also polynomially equivalent to the IRC–refuting problem. In the last subsection we show that the redundant (in)equality and redundant conjunct problems are polynomially reducible to the IRC–query–containment (and thus to the IRC–refuting) problem.

The studies in this section enable us to have a new understanding of these problems. The basic observation we use is that queries and implication constraints can be viewed as playing complementary roles and can be transformed to one another.

4.1 Semantics–Based Query Containment Problem

Query optimization involves a series of *equivalent* transformations on queries, and the equivalence of transformations is guaranteed by the (set) *containment* in both directions. Therefore, Query containment problem for conjunctive queries is one of the fundamental problems in query optimization.

Given two queries Q and Q_1 of the same type, we say Q_1 contains Q, denoted $Q_1 \supseteq Q$, if

$$Q_1(\mathbf{D}) \supseteq Q(\mathbf{D})$$

holds for any database instance $\mathbf{D}$.

In this section we study the containment problem based on data semantics specified by implication and referential constraints, as shown in Example 1.1. Another generalization is that *union* queries are considered. Remember that a union query is a formula of the form

$$union(Q_1, ..., Q_r),$$

where $Q_1, ..., Q_r$ $(r \geq 1)$ are queries of the same type. The result of $union(Q_1, ..., Q_r)$ on a database instance $\mathbf{D}$ is

$$Q_1(\mathbf{D}) \cup ... \cup Q_r(\mathbf{D}).$$

Definition 4.1 (IRC–Query-Containment)
Given a set of ICs and RCs, $\mathcal{IR_C}$, and queries Q, Q_1, ..., Q_r ($r \geq 1$) of the same type, the IRC–query–containment problem *is to decide whether*

$$\mathcal{IR_C} \models_f (union(Q_1, ..., Q_r) \supseteq Q),$$

i.e., whether

$$union(Q_1, ..., Q_r)(\mathbf{D}) \supseteq Q(\mathbf{D})$$

for any finite database instance $\mathbf{D}$ *satisfying* $\mathcal{IR_C}$.

Among the works on query containment problem, Johnson and Klug ([17]) addressed the containment between conjunctive queries in the presence of functional and inclusion dependencies. Our results presented here generalize the previous works in two aspects:

1. we consider ICs that are more general than FDs and can not be used in the chase process proposed by Johnson and Klug.
2. we consider union queries.

As in the case of IRC–refuting problem, we first prove that finite implication and implication coincide due to the acyclic property.

Lemma 4.1
Let Q, Q_1, ..., Q_s ($s \geq 1$) be queries of the same type, and $\mathcal{IR_C}$ be a set of ICs and RCs such that the RCs are acyclic. Then $\mathcal{IR_C} \models_f (union(Q_1, ..., Q_s) \supseteq Q)$ if and only if $\mathcal{IR_C} \models (union(Q_1, ..., Q_s) \supseteq Q)$.

Proof.
Let $Q = \{\langle \mathbf{O}_i \rangle : F, I\}$ and $Q_i = \{\langle \mathbf{O}_i \rangle : F'_i, I'_i\}$ for $i = 1, ..., s$.

Assume the variables in Q_i's are properly renamed so that they share no variables among themselves and with Q. Let $\mathbf{V_d}$ be the dv's in Q. Let $\mathbf{V_n}$ and $\mathbf{V_n}_i$ be the ndv's of Q and Q_i respectively, for $i = 1, ..., s$.

Let $rc_1, ..., rc_t$ be the (acyclic) RCs and $ic_1, ..., ic_r$ be the ICs where $t, r \geq 0$. Logically,

$$(\mathcal{IR_C}) \models_f (union(Q_1, ...Q_s) \supseteq Q)$$

is equivalent to saying that formula $\Phi \rightarrow_f \Psi$ is true. Here "$\rightarrow_f$" is finite implication, and

$$\Phi = ic_i \wedge ... \wedge ic_r \wedge rc_1 \wedge ... \wedge rc_t,$$

and

$$\Psi = \forall \mathbf{V_d}[\exists \mathbf{V_n}[F, I] \rightarrow \bigvee_{(1 \leq i \leq s)} \exists(\mathbf{O}_i, \mathbf{V_n}_i) \ [\langle \mathbf{O}_i \rangle = \langle \mathbf{O} \rangle, F'_i, I'_i]],$$

where

$$\langle \mathbf{O}_i \rangle = \langle \mathbf{O} \rangle \quad (i = 1, \ ..., \ s)$$

is a conjunction of equalities between the corresponding components of $\langle \mathbf{O}_i \rangle$ and $\langle \mathbf{O} \rangle$.

It is not difficult to see that $\Phi \rightarrow_f \Psi$ is true if and only if $\Phi \wedge \Psi'$ is finitely unsatisfiable, where Ψ' is the Skolemized version of $\neg\Psi$:

$$F(\mathbf{b}, \mathbf{b}') \wedge I(\mathbf{b}, \mathbf{b}') \wedge \bigwedge_{(1 \leq i \leq s)} \forall(\mathbf{O}_i, \mathbf{V_n}_i)[[\langle \mathbf{O}_i \rangle = \langle \mathbf{O}[\mathbf{V_d} \leftarrow \mathbf{b}] \rangle], \ F_i', \ I_i' \rightarrow].$$

Here $F(\mathbf{b}, \mathbf{b}')$, and $I(\mathbf{b}, \mathbf{b}')$ are respectively F and I with $\mathbf{V_d}$ and $\mathbf{V_n}$ being replaced by vectors of Skolem constant symbols $\mathbf{b}$ and $\mathbf{b}'$ respectively, and $\mathbf{O}[\mathbf{V_d} \leftarrow \mathbf{b}]$ is $\mathbf{O}$ with $\mathbf{V_d}$ being replaced by $\mathbf{b}$.

By a reasoning similar to that in the proof of Lemma 3.1, we can prove that $\Phi \wedge \Psi'$ is finitely unsatisfiable if and only if $\Phi \wedge \Psi'$ is unsatisfiable, which means $(\mathcal{IR}_\mathcal{C}) \models (union(Q_1, ...Q_s) \supseteq Q)$.

□

Due to the above lemma, we will just use $\models$ when referring to the IRC–query–containment problem. The following theorem solves the IRC–query–containment problem. Like the theorem for the IRC–refuting problem, the condition is still given in terms of referential expansion and symbol mappings. The only difference is that here we also consider the symbol mappings from the implication constraints to the contained query. This theorem also makes it clear that IRC–query–containment problem is polynomially equivalent to IRC–refuting problem (under the acyclic assumption).

Theorem 4.1 (IRC–Query–Containment)
Suppose we are given the following:

1. *a set of implication constraints $\mathcal{I}_\mathcal{C} = \{ic_1, ..., ic_r\}$ $(r \geq 0)$ such that the ic_i's are in normal form;*
2. *an acyclic set of referential constraints $\mathcal{R}_\mathcal{C}$;*
3. *a query $Q = \{\langle \mathbf{O} \rangle : \ F, \ I\}$, where $\langle \mathbf{O} \rangle$ is the summary, and F and I are the conjunct and the (in)equality subformulas;*
4. *the referential expansion Q^{re} of Q with respect to $\mathcal{R}_\mathcal{C}$;*
5. *a union query $union(Q_1, ..., Q_s)$ $(s \geq 1)$ where the Q_i's are in normal form and of the same type as Q.*

Then $(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models (union(Q_1, ..., Q_s) \supseteq Q)$ if and only if there exist $(n_1 + ... + n_r + m_1 + ... + m_s) \geq 1$ symbol mappings:

$\rho_{i,1}, ..., \rho_{i,n_i}$ *from ic_i to Q^{re} (for $i = 1, ..., r$),*

and

$\theta_{i,1}, ..., \theta_{i,m_i}$ *from Q_i to Q^{re} (for $i = 1, ..., s$),*

such that I implies

$$\bigvee_{(1 \leq i \leq r, \ 1 \leq j \leq n_r)} \rho_{i,j}(I_i) \qquad \bigvee_{(1 \leq i \leq s, \ 1 \leq j \leq m_s)} \theta_{i,j}(I_i'),$$

where $I_1, ..., I_r$ are the (in)equality subformulas of $ic_1, ..., ic_r$ respectively, and $I_1', ..., I_s'$ are the (in)equality subformulas of $Q_1, ..., Q_s$ respectively.

Remark. In Example 1.1 we have seen that $\{rc_1\} \models_f (Q_1 \supseteq Q)$, where

$$\mathbf{dept}(D, C) \ \rightarrow \ \mathbf{faculty}(C, h_s(D, C), h_t(D, C)),$$

$$Q_1 = \{\langle C \rangle : \ \mathbf{dept}(D_1, C), \ \mathbf{dept}(D_2, C), \ \mathbf{faculty}(C, S, T), \ D_1 \neq D_2\}, \text{ and}$$

$$Q = \{\langle C\rangle :\ \mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C)\}.$$

The above theorem basically says that this containment holds due to the same symbol mapping given in Example 3.4 for proving the IRC–refuting problem $\{ic_1,\ rc_1\} \models_f (Q \equiv \{\})$, where ic_1 is $\mathbf{dept}(D_1, C),\ \mathbf{dept}(D_2, C),\ \mathbf{faculty}(C, S, T),\ D_1 \neq D_2 \rightarrow$. Below we give the proof of Theorem 4.1.

Proof.

Let $Q = \{\langle \mathbf{O}\rangle : F, I\}$, $Q^{re} = \{\langle \mathbf{O}\rangle : F^{re}, I\}$, and $Q_i = \{\langle \mathbf{O}_i\rangle : F_i', I_i'\}$ for $i = 1, ..., s$.

Assume the variables in Q_i's are renamed so that they share no variables among themselves and with Q. Let $\mathbf{V_d}$ be the dv's in Q. Let $\mathbf{V_n}$ and $\mathbf{V_{n}}_i$ be the ndv's of Q and Q_i respectively, for $i = 1, ..., s$. Let $\mathcal{R_C} = \{rc_1, ..., rc_t\}$ $(t \geq 0)$.

Using the same reasoning in the proof of the previous lemma, $(\mathcal{I_C} \cup \mathcal{R_C}) \models (union(Q_1, ...Q_r) \supseteq Q)$ if and only if $\Phi \wedge \Psi'$ is unsatisfiable, where

$$\Phi = ic_i \wedge ... \wedge ic_r \wedge rc_1 \wedge ... \wedge rc_t,$$

and

$$\Psi' = [F(\mathbf{b}, \mathbf{b}'), I(\mathbf{b}, \mathbf{b}')] \wedge \bigwedge_{(1 \leq i \leq s)} \forall(\mathbf{O}_i, \mathbf{V_n}_i)[[\langle \mathbf{O}_i\rangle = \langle \mathbf{O}[\mathbf{V_d} \leftarrow \mathbf{b}]\rangle],\ F_i',\ I_i' \rightarrow],$$

where $F(\mathbf{b}, \mathbf{b}')$, and $I(\mathbf{b}, \mathbf{b}')$ are respectively F and I with $\mathbf{V_d}$ and $\mathbf{V_n}$ being replaced by Skolem constant symbols $\mathbf{b}$ and $\mathbf{b}'$ respectively, and $\mathbf{O}[\mathbf{V_d} \leftarrow \mathbf{b}]$ is $\mathbf{O}$ with $\mathbf{V_d}$ being replaced by $\mathbf{b}$.

By a reasoning similar to that in the first two paragraphs of the proof of Theorem 3.1, we can prove that $\Phi \wedge \Psi'$ is unsatisfiable if and only if

$$ic_1 \wedge ... \wedge ic_r \bigwedge_{(1 \leq i \leq s)} \forall(\mathbf{O}_i, \mathbf{V_n}_i)[[\langle \mathbf{O}_i\rangle = \langle \mathbf{O}[\mathbf{V_d} \leftarrow \mathbf{b}]\rangle], F_i', I_i' \rightarrow] \wedge [F^{re}(\mathbf{b}, \mathbf{b}'), I(\mathbf{b}, \mathbf{b}')]$$

is unsatisfiable.

Let $ic_i = (F_i, I_i \rightarrow)$ for $i = 1, ..., r$. By the above reasoning, $(\mathcal{I_C} \cup \mathcal{R_C}) \models (union(Q_1, ...Q_r) \supseteq Q)$ if and only if the following (sets) of clauses are unsatisfiable:

$$cl_1 :\ F^{re}(\mathbf{b}, \mathbf{b}'),$$

$$cl_2 :\ I(\mathbf{b}, \mathbf{b}'),$$

$$cl_3 :\ \bigvee[\langle \mathbf{O}_i\rangle \neq \langle \mathbf{O}[\mathbf{V_d} \leftarrow \mathbf{b}]\rangle] \ \vee\ \neg F_i' \ \vee\ \neg I_i',\ \text{ for }\ i = 1,\ ...,\ s,$$

and

$$cl_4 :\ \neg F_i \ \vee\ \neg I_i,\ \text{ for }\ i = 1,\ ...,\ r,$$

where

$$\bigvee[\langle \mathbf{O}_i\rangle \neq \langle \mathbf{O}[\mathbf{V_d} \leftarrow \mathbf{s}]\rangle]$$

is a disjunction of $\neq$-inequalities between the corresponding components of $\langle \mathbf{O}_i\rangle$ and $\langle \mathbf{O}[\mathbf{V_d} \leftarrow \mathbf{s}]\rangle$. Therefore, as in the proof of Theorem 3.1, we can use resolutions with paramodulation on conjunct literals to get an empty clause or an unsatisfiable set of resolvents consisting only of (in)equalities. However, since here all ic_i's are in normal form, every possible unification between conjunct literals can be performed without paramodulations. Moreover, the unifications here actually are substitutions.

Also observe that in every (in)equality resolvent obtained from cl_1 and cl_3, there exists a subformula of the form

$$\bigvee[\langle \mathbf{O}_i \rangle \neq \langle \mathbf{O}[\mathbf{V_d} \leftarrow \mathbf{s}] \rangle],$$

which is always satisfiable unless $\langle \mathbf{O}_i \rangle$ is mapped to $\langle \mathbf{O}[\mathbf{V_d} \leftarrow \mathbf{s}] \rangle$, because O_i does not appear elsewhere. Thus, the substitutions that produce the empty clause or unsatisfiable set of (in)equalities constitute the symbol mappings as specified in the theorem. This leads to the conclusion of the theorem.

□

Proposition 4.1

IRC–query–containment problem is polynomially equivalent to IRC–refuting problem (for acyclic RCs).

Proof.

First the reduction from IRC–query–containment problem to IRC–refuting problem. Consider a general IRC–query–containment problem as specified in Theorem 4.1, then $(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models (union(Q_1, ..., Q_s) \supseteq Q)$ if and only if there exist $(n_1 + ... + n_r + m_1 + ... + m_s) \geq 1$ symbol mappings:

$\rho_{i,1}, ..., \rho_{i,n_i}$ from ic_i to Q^{re} (for $i = 1, ..., r$),

and

$\theta_{i,1}, ..., \theta_{i,m_i}$ from Q_i to Q^{re} (for $i = 1, ..., s$),

such that I implies

$$\bigvee_{(1 \leq i \leq r,\ 1 \leq j \leq n_r)} \rho_{i,j}(I_i) \bigvee_{(1 \leq i \leq s,\ 1 \leq j \leq m_s)} \theta_{i,j}(I'_i),$$

where $I_1, ..., I_r$ are the (in)equality subformulas of $ic_1, ..., ic_r$ respectively, and $I, I'_1, ..., I'_s$ are the (in)equality subformulas of $Q, Q_1, ..., Q_s$ respectively.

Let $Q = \{\langle \mathbf{O} \rangle : F, I\}$, $Q_i = \{\langle \mathbf{O} \rangle : F'_i, I'_i\}$, and **out** be a "new" predicate name (not used in any queries or constraints). Observe that if Q^{re} is of the form $\{\langle \mathbf{O} \rangle : F^{re}, I\}$, then the referential expansion (wrt. the RCs) of $\{\langle\rangle : \mathbf{out}(\mathbf{O}), F, I\}$ is $\{\langle\rangle : \mathbf{out}(\mathbf{O}), F^{re}, I\}$. Since any symbol mapping from Q_i to Q^{re} is also a symbol mapping from the $\mathbf{out}(\mathbf{O}_i), F'_i, I'_i \rightarrow$ to $\{\langle\rangle : \mathbf{out}(\mathbf{O}), F^{re}, I\}$, by Theorem 3.1 we have

$$\mathcal{I}\mathcal{R}_\mathcal{C} \models (union(Q_1, ..., Q_r) \supseteq Q)$$

if and only

$$(\mathcal{I}\mathcal{R}_\mathcal{C} \cup \{ic'_1, ..., ic'_s\}) \models (\{\langle\rangle : \mathbf{out}(\mathbf{O}), F, I\} \equiv \{\}),$$

where $ic'_i = (\mathbf{out}(\mathbf{O}_i), F'_i, I'_i \rightarrow)$ for $i = 1, ..., s$.

Next the reduction from IRC–refuting problem to IRC–query–containment problem. Consider a general IRC-refuting problem as specified in Theorem 3.1. $(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models (Q \equiv \{\})$ if and only if there exist $(n_1 + ... + n_r) \geq 1$ symbol mappings:

$\rho_{i,1}, ..., \rho_{i,n_i}$ from ic_i to Q^{re} (for $i = 1, ..., r$)

such that I implies $\bigvee_{(1 \leq i \leq r,\ 1 \leq j \leq n_r)} \rho_{i,j}(I_i)$, where $I, I_1, ..., I_r$ are the (in)equality subformulas of $Q, ic_1, ..., ic_r$ respectively.

Let $Q = \{\langle \mathbf{O} \rangle : F, I\}$ and let $ic_i = (F_i, I_i \rightarrow)$ for $i = 1, ..., r$. Since any symbol mapping from ic_i to Q^{re} is also a symbol mapping from $\{\langle\rangle : F_i, I_i\})$ to Q^{re}, by Theorem 4.1 we have

$$(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models (Q \equiv \{\})$$

if and only if

$$\mathcal{R}_{\mathcal{C}} \models (union(Q_1, ..., Q_r) \supseteq \{\langle\rangle : F, I\}),$$

where $Q_i = \{\langle\rangle : F_i, I_i\}$ for $i = 1, ..., r$.

□

Example 4.1

Let us reconsider Example 1.1, where the implication constraint is

$$ic_1 : \ \mathbf{dept}(D_1, C),\ \mathbf{dept}(D_2, C),\ \mathbf{faculty}(C, S, T),\ D_1 \neq D_2\ \rightarrow,$$

the referential constraint is

$$rc_1 : \ \mathbf{dept}(D, C)\ \rightarrow\ \exists(S, T)\ \mathbf{faculty}(C, S, T).$$

and the queries are

$$Q = \{\langle C\rangle : \ \mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C)\},$$

and

$$Q_1 = \{\langle C\rangle : \ \mathbf{dept}(D_1, C),\ \mathbf{dept}(D_2, C),\ \mathbf{faculty}(C, S, T),\ D_1 \neq D_2\}.$$

The reduction from the refuting problem $\{ic_1, rc_1\} \models (Q \equiv \{\})$ to the containment problem is:

$$\{ic_1, rc_1\} \models (Q \equiv \{\})$$

if and only if

$$\{rc_1\} \models (Q' \supseteq \{\langle\rangle : \ \mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C)\},$$

where $Q' = \{\langle\rangle : \ \mathbf{dept}(D_1, C),\ \mathbf{dept}(D_2, C),\ \mathbf{faculty}(C, S, T),\ D_1 \neq D_2\}$.

On the other hand, the reduction from the containment problem $\{rc_1\} \models (Q_1 \supseteq Q)$ to the refuting problem is

$$\{rc_1\} \models (Q_1 \supseteq Q)$$

if and only if

$$\{ic_1',\ rc_1\} \models (\{\langle\rangle : \ \mathbf{out}(C),\ \mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C)\} \equiv \{\}),$$

where the implication constraint ic_1' transformed from Q_1 is

$$ic_1' : \ \mathbf{out}(C),\ \mathbf{dept}(D_1, C),\ \mathbf{dept}(D_2, C),\ \mathbf{faculty}(C, S, T),\ D_1 \neq D_2\ \rightarrow.$$

□

Query containment problem (without considering constraints) has recently been proved to be Π_2^p–complete ([24]) for the class of queries having (in)equalities over densely ordered domains, so the IRC–query–containment problem and the IRC–refuting problem we are considering are at least as hard. Π_2^p is a class higher than NP in the polynomial structure ([13]).

Although we are dealing with high complexity here, the complexity is, as pointed out by many other authors, not in terms of the size of database instance, but in terms of the size of query statement. A closer inspection of Theorem 3.1

and 4.1 reveals that the exponentiality arises from two sources, the number of symbol mappings and the number of conjunctions that need satisfiability test for solving the subsumption subproblem.

The number of symbol mappings is kept small if there are not many conjuncts with same predicate name in the queries or constraints, as is often the case in many applications. The size of the subsumption problem is generally very small, since usually there is not a large number of constraints referring to the predicates in a query. In Section 5 we will give sufficient conditions for reducing the asymptotic complexity of the subsumption problem in some special cases.

4.2 Redundant Implication Constraints

In maintaining a constraint–base (a set of constraints), we need to decide whether a new constraint is redundant before it is added into the base. [3], [8], and [25] give axiomatic and graph–theoretic characterizations for the implication problem of INDs and FDs, and recently [28] investigated the problem of deciding redundant implication constraints with respect to a set of ICs. In this subsection, we consider the problem of deciding redundant ICs with respect to a set of ICs and RCs, as illustrated in the very first example. The problem of deciding redundant RCs with respect to ICs and RCs is still open.

Definition 4.2 (IRC–Redundant IC Problem)
Given a non–empty set $\mathcal{I}_\mathcal{C}$ of ICs, a set $\mathcal{R}_\mathcal{C}$ of RCs, and an implication constraint ic, the IRC–redundant IC problem *is to decide whether*

$$(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models_f ic,$$

i.e., whether ic is satisfied by any finite database instance satisfying $\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}$.

Proposition 4.2
IRC–Redundant IC problem is polynomially equivalent to IRC–refuting problem.

Remark. Note that this proposition does not require the acyclicity of the RCs, and when the RCs are acyclic, both problems are decidable.

Proof.
We only give the reductions, the proof is straightforward from the formal definitions of the two problems.

Let $\mathcal{I}_\mathcal{C}$ be a non–empty set of implication constraints, $\mathcal{R}_\mathcal{C}$ be a set of referential constraints. Then the reduction from the refuting problem to the redundant IC problem is as follows: given a $Q = \{\langle \mathbf{O} \rangle : F, I\}$,

$$(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models_f (Q \equiv \{\})$$

if and only if

$$(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models_f (F, I \rightarrow).$$

The reduction from the redundant IC problem to the refuting problem is as follows: given an implication constraint $ic: \ (F, I \rightarrow)$,

$$(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models_f ic$$

if and only if

$$(\mathcal{I}_\mathcal{C} \cup \mathcal{R}_\mathcal{C}) \models_f (\{\langle\rangle : F, I\} \equiv \{\}).$$

□

Example 4.2

In Example 1.1 we have reasoned informally that

$$\{ic_1,\ rc_1\} \models (Q \equiv \{\})$$

if and only if

$$\{ic_1,\ rc_1\} \models (\mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C)\ \rightarrow),$$

where ic_1 is

$$\mathbf{dept}(D_1, C),\ \mathbf{dept}(D_2, C),\ \mathbf{faculty}(C, S, T),\ D_1 \neq D_2\ \rightarrow,$$

rc_1 is

$$\mathbf{dept}(D, C)\ \rightarrow\ \exists(S, T)\ \mathbf{faculty}(C, S, T),$$

and Q is

$$\{\langle C\rangle :\ \mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C)\}.$$

¿From the proof of the above proposition we can see that indeed this is just the reduction from the refuting problem

$$\{ic_1,\ rc_1\} \models (Q \equiv \{\})$$

to the redundant IC problem

$$\{ic_1,\ rc_1\} \models (\mathbf{dept}(physics, C),\ \mathbf{dept}(chemistry, C)\ \rightarrow)$$

□

4.3 Redundant Selection–Conditions and Redundant Joins

Another major issue in semantic query optimization is to determine redundant selection–conditions ((in)equalities) and redundant conjuncts ([19], [5]). It is often beneficial to eliminate certain redundant selection conditions, e.g., those on unindexed attributes or across relations. To remove redundant conjuncts is even more important, because conjuncts represent the join operation, which is the most expensive relational operation. Although the two problems are not known to be polynomially equivalent to the IRC–refuting problem, we can easily show that they are polynomially reducible to the IRC–refuting problem.

Definition 4.3 (IRC–Redundant (In)equality Problem)
Let $Q = \{\langle \mathbf{O}\rangle : F, I, c\}$ where c is an (in)equality, and $\mathcal{IR}_\mathcal{C}$ be a set of ICs and RCs, the IRC–redundant (in)equality problem *is to decide whether*

$$\mathcal{IR}_\mathcal{C} \models_f (Q \equiv \{\langle \mathbf{O}\rangle : F, I\}),$$

i.e., whether Q and $\{\langle \mathbf{O}\rangle : F, I\}$ always produce the same answer on any finite database instance satisfying $\mathcal{IR}_\mathcal{C}$.

Indeed, $\mathcal{IR}_\mathcal{C} \models_f (Q \equiv \{\langle \mathbf{O}\rangle : F, I\})$ if and only if $\mathcal{IR}_\mathcal{C} \models_f (Q \supseteq \{\langle \mathbf{O}\rangle : F, I\})$, since it is always true that $\{\langle \mathbf{O}\rangle : F, I\} \supseteq Q$. Now, we have reduced the redundant (in)equality problem to the query containment problem, which in turn can be reduced to the IRC–refuting problem by Proposition 4.1. This analysis leads to the following proposition, which also does not require the acyclicity of the RCs.

Proposition 4.3

IRC–Redundant (in)equality problem is (polynomially) reducible to IRC–refuting problem.

To eliminate a redundant conjunct we have to make sure to eliminate all the relevant (in)equalities, i.e., (in)equalities that involve variables which do not appear in any conjuncts except the ones that are removed. The resulting query must still be *safe* ([Ull89a]).

Definition 4.4 (Redundant Join Problem)

Let $\mathcal{IR}_\mathcal{C}$ be a set of ICs and RCs, and $Q = \{\langle \mathbf{O} \rangle : F, t, I, I_t\}$ be a compressed query such that:

1. *t is a conjunct, such that any dv in t appears in at least one other conjunct;*
2. *I_t is the conjunction of (in)equalities each having at least one variable that does not appear in any conjunct other than t.*

The IRC–redundant join problem *is to decide whether $\mathcal{IR}_\mathcal{C} \models_f (Q \equiv \{\langle \mathbf{O} \rangle : F, I\})$.*

Proposition 4.4

IRC–Redundant join problem is (polynomially) reducible to IRC–refuting problem.

Proof.

Indeed, $\mathcal{IR}_\mathcal{C} \models_f (Q \equiv \{\langle \mathbf{O} \rangle : F, I\})$ if and only if $\mathcal{IR}_\mathcal{C} \models_f (Q \supseteq \{\langle \mathbf{O} \rangle : F, I\})$, since it is always true that $\{\langle \mathbf{O} \rangle : F, I\} \supseteq Q$. Now, we have reduced the redundant (in)equality problem to the query containment problem, which in turn can be reduced to the IRC–refuting problem by Proposition 4.1. Like the previous propositions, this one also does not require the acyclicity of the RCs.

□

The above proposition can be proved similarly as the Proposition 4.3. We further explain the results in this section through the following example.

Example 4.3

Continued from Example 1.1. Suppose we have another implication constraint ic_2 saying that if a faculty member is a chair of a department, then his/her salary must be greater than \$80,000, i.e.,

$$ic_2 : \ \mathbf{dept}(D, C), \ \mathbf{faculty}(C, S, T), \ S \leq 80,000 \ \rightarrow .$$

Now, for a query like "list the department that has a chair earning \$65,000 or more", i.e.,

$$Q_2 = \{\langle D \rangle : \ \mathbf{dept}(D, C), \ \mathbf{faculty}(C, S, T), \ S \geq 65,000\},$$

it is clear that

$$Q_2 \subseteq \{\langle D \rangle : \ \mathbf{dept}(D, C)\}.$$

On the other hand, by Theorem 4.1, we have

$$\{ic_2, \ rc_1\} \models_f (Q_2 \supseteq \{\langle D \rangle : \ \mathbf{dept}(D, C)\}).$$

Therefore

$$\{ic_2,\ rc_1\} \models_f (Q_2 \equiv \{\langle D\rangle :\ \mathbf{dept}(D, C)\}),$$

i.e., conjunct $\mathbf{faculty}(C, S, T)$ and inequality $S \geq 65,000$ are redundant.

□

5 On Efficient Reasoning With Implication Constraints

In this section we study a special case of the IRC–refuting problem, in which the set of referential constraints is empty. We call the problem the IC–refuting problem. We are going to give some criteria for reducing the complexity of the subsumption problem for this case. The process we use is *units–refuting*. The tool we use is called the *units–refuting process*.

5.1 IC–Refuting Problem

Definition 5.1 (IC–Refuting Problem)
Given a non–empty set of ICs, $\mathcal{I}_\mathcal{C}$, and a query Q, the IC–refuting problem is to decide whether $\mathcal{I}_\mathcal{C} \models_f (Q \equiv \{\})$.

IC–refuting problem can be solved by the following corollary of Theorem 3.1.

Corollary 5.1
Let $\mathcal{I}_\mathcal{C} = \{ic_1, ..., ic_r\}$ be a non–empty set of ICs in which all constraints are in normal forms, and $Q = \{\langle \mathbf{O}\rangle : F, I\}$. Then $\mathcal{I}_\mathcal{C} \models_f (Q \equiv \{\})$ if and only if there are symbol mappings:

$\rho_{1,1}, ..., \rho_{1,n_1} (n_1 > 0)$ *from* ic_1 *to* Q,

$\vdots$

$\rho_{r,1}, ..., \rho_{r,n_r} (n_r > 0)$ *from* ic_r *to* Q

such that I implies $\bigvee_{(1\leq i\leq r,\ 1\leq j\leq n_r)} \rho_{i,j}(I_i)$.

Correspondingly, we can consider the problems of query containment, redundant implication constraints, redundant (in)equalities and joins all with respect to only implication constraints. The similar polynomial equivalences and the reductions still hold as in the case where both ICs and RCs are present.

5.2 Units–Refuting IC–Bases

The subsumption problem of deciding whether I implies $I_1 \vee ... \vee I_n$ for conjunctions of (in)equalities is equivalent to deciding the unsatisfiability of $\{I,\ \neg I_1,\ ...,\ \neg I_n\}$.

Definition 5.2 ((Units–Refutation))
Given a set of (propositional) (in)equality clauses $\mathcal{C}_\mathcal{L}$, the units–refutation *is a process defined as follows: use several single–literal clauses to eliminate a contradictory literal ((in)equality) in another clause cl, and replace the resulting clause for cl in $\mathcal{C}_\mathcal{L}$. Repeat the above steps, until an empty clause is obtained or no more literals can be eliminated.*

Notice that units–refutation is similar to the "unit resolution" in [6], where there are only two parent clauses involved in each resolution step. Here we may use more than one (in)equalities to refute another (in)equality.

Example 5.1
Consider the following set of propositional (in)equality clauses:

$$\{(F \neq F') \vee (T = permanent), \quad (T \neq permanent) \vee (S > 40,000), \quad (F = F')\}.$$

the units–refutation process proceeds as follows:

$$\{(F \neq F') \vee (T = permanent), \quad (T \neq permanent) \vee (S > 40,000), \quad (F = F')\}$$

$$\Longrightarrow$$

$$\{(T = permanent), \quad (T \neq permanent) \vee (S > 40,000), \quad (F = F')\}$$

$$\Longrightarrow$$

$$\{(T = permanent), \quad (S > 40,000), \quad (F = F')\}.$$

□

When the units–refutation process terminates with an empty clause, the set of clauses is unsatisfiable. Units–refutation process only takes polynomial time (with respect to the number of variables and constants in the clause set), since whether a conjunction of (in)equalities is satisfiable can be decided in polynomial time (see Section 3).

Generally speaking (unless P=NP), however, an exponential algorithm is needed to determine the satisfiability of a set of (in)equality clauses. Following is an example showing that units–refutation it is not sufficient for the IC–refuting problem.

Example 5.2 Suppose we have another implication constraint stating that there are only two possible status for a faculty member — "temporary" and "permanent":

$$ic_3: \ \mathbf{faculty}(F, S, T), \ T \neq temporary, \ T \neq permanent \ \rightarrow .$$

With ic_3 enforced, we know the following query will always produce an empty answer, as it asks for the faculties having three different status:

$$Q = \{\langle F \rangle : \ \mathbf{faculty}(F, S_1, T_1), \ \mathbf{faculty}(F, S_2, T_2), \ \mathbf{faculty}(F, S_3, T_3),$$

$$T_1 \neq T_2, \ T_1 \neq T_3, \ T_2 \neq T_3\}.$$

Using Theorem 5.1 we can show that $\{ic_3\} \models (Q \equiv \{\})$ if and only if the following clauses are unsatisfiable:

$$(T_1 = temporary) \vee (T_1 = permanent),$$

$$(T_2 = temporary) \vee (T_2 = permanent),$$

$$(T_3 = temporary) \vee (T_3 = permanent),$$

$$T_1 \neq T_2,$$

$$T_1 \neq T_3,$$

$$T_2 \neq T_3.$$

These clauses are indeed unsatisfiable, but the units–refutation process can not get us the empty clause. It is easily seen that we can eliminate no more literals using single–literal clauses $T_1 \neq T_2$, $T_1 \neq T_3$, and $T_2 \neq T_3$.
□

Next we define a type of IC–bases (sets of ICs) for which the units–refutation process is *sufficient* to solve the subsumption subproblem of the IC–refuting problem. We call them *units–refuting* bases.

Definition 5.3 (Units–Refuting Bases)
Let $\mathcal{I}_\mathcal{C} = \{ic_1, ..., ic_r\}$ $(r \geq 1)$ be a set of ICs in normal form, and $Q = \{\langle \mathbf{O} \rangle : F, I\}$ be a query, where $\langle \mathbf{O} \rangle$ is the summary, and F and I are the conjunct and the (in)equality subformulas respectively. $\mathcal{I}_\mathcal{C}$ is called a units–refuting base if whenever $\mathcal{I}_\mathcal{C} \models (Q \equiv \{\})$ for a query Q, then there exist $(n_1 + ... + n_r) \geq 1$ symbol mappings:

$\rho_{i,1}, ..., \rho_{i,n_i}$ from ic_i to Q (for $i = 1, ..., r$),

such that the units–refutation process can be used to obtain an empty clause from I, $\neg\rho_{1,1}(I_1)$, ..., $\neg\rho_{1,n_1}(I_1)$, ..., $\neg\rho_{r,1}(I_r)$, ..., and $\neg\rho_{r,n_r}(I_r)$, where $I_1, ..., I_r$ are the (in)equality subformulas of $ic_1, ..., ic_r$ respectively.

We will give examples of units–refuting bases after we give a criterion for it. First of all, we want to prove that the IC–refuting problem for units–refuting bases has the same complexity as the well known containment problem for *equality* queries ([4]).

Theorem 5.1
For any units–refuting IC–base, the IC–refuting problem is NP–complete (with respect to the number of variables and constants in the constraint base and the query).

Proof.
First we prove that the IC–refuting problem for units–refuting bases is NP. Let $\mathcal{I}_\mathcal{C}$ be a units–refuting base, and Q be any query such that $\mathcal{I}_\mathcal{C} \models (Q \equiv \{\})$. Let N_q be the number of variables and constants in Q. Since $\mathcal{I}_\mathcal{C} \models (Q \equiv \{\})$ and $\mathcal{I}_\mathcal{C}$ is units–refuting, there must exist a minimal number of symbol mappings corresponding to which the set of (in)equality clauses is unsatisfiable provable by the units–refutation process. This number is no more than $6(N_q)^2$ (which is the number of all possible (in)equalities that can be formed from the variables, constants and (in)equality predicate names). The reason is that in the worst case the units–refutation process will produce a literal ((in)equality) from every original (in)equality clause. Now, this number of symbol mappings can be *guessed* polynomially; and by units–refutation process, we can *check* the unsatisfiability of the corresponding set of clauses also in polynomial time.

To complete the proof, the NP–hardness can be shown by reducing the NP–complete containment problem for *equality queries* ([4]) to the IC-refuting problem (see the discussion after Theorem 5.2).
□

5.3 Conflicting (In)equalities

In order to find conditions for an IC–base to be units–refuting, we first consider the relationships between a pair of (in)equalities appearing in implication constraints that are in clausal form. Note that the (in)equalities considered in

this section have no function symbols, because we only consider the case where there is no referential constraint.

Definition 5.4 (Potentially–Conflicting)
Let c_1 and c_2 be two (not necessarily different) (in)equalities of the same type (and having no function symbols), and c_1' and c_2' be obtained from c_1 and c_2 respectively, by renaming the variables so that c_1' and c_2' have no common variables. Then c_1 and c_2 are called potentially–conflicting (or simply conflicting), if there exists a (finite) set of (in)equalities I (having no function symbols), such that both $I \cup \{c_1'\}$ and $I \cup \{c_2'\}$ are satisfiable, but $I \cup \{c_1', c_2'\}$ is not.

For example, $X \geq 3$ and $Y = 1$ are conflicting, because $\{X \geq 3,\ Y = 1,\ X = Y\}$ is unsatisfiable; but $X \geq 3$ and $Y \neq 5$ are not conflicting. An (in)equality may also be (potentially) conflicting to itself, for example, $T = temporary$ and $T = temporary$: after we rename T to T_1 in one of the literals, we get a pair of conflicting literals: $T = temporary$ and $T_1 = temporary$, with respect to, for example, $T \neq T_1$.

One important property of a pair of non–conflicting (in)equalities c_1 and c_2 is that $\rho_1(c_1)$ and $\rho_2(c_2)$ are non–conflicting for any symbol mappings ρ_1 and ρ_2, since a symbol mapping can be expressed by a set of equalities. In refuting a query, different instances of ICs may be used, hence we need this property. Also notice that if c_1 and c_2 are not conflicting, then $\{c_1, c_2\}$ is always satisfiable.

Since we are considering the IC–refuting problem where no function symbols are involved, (in)equalities of different types are not conflicting. Using Theorem 3.2, we can prove the conflicting relations between any pair of (in)equalities, as listed in Table 1 and Table 2.

Table 1 is for ordered domains, and Table 2 is for unordered domains. Entries marked with "NC" means the pair is not conflicting; other non–blank entries specify the necessary and sufficient conditions for the pair not to be conflicting; all the blank entries indicate the pair is conflicting.

	$X > a$	$X \geq a$	$X = a$	$X \neq a$	$X < a$	$X \leq a$	$X > Y$	$X \geq Y$	$X = Y$	$X \neq Y$
$U > b$	NC	NC	$a > b$	NC	$a > b$	$a > b$				NC
$U \geq b$	NC	NC	$a > b$	$a \neq b$	$a > b$	$a > b$				
$U = b$	$a < b$	$a < b$		$a \neq b$	$a > b$	$a > b$				
$U \neq b$	NC	$a \neq b$	$a \neq b$	NC	NC	$a \neq b$				NC
$U < b$	$a < b$	$a < b$	$a < b$	NC	NC	NC				NC
$U \leq b$	$a < b$	$a < b$	$a < b$	$a \neq b$	NC	NC				
$U > V$										NC
$U \geq V$										
$U = V$										
$U \neq V$	NC			NC	NC		NC			NC

Table 1. Conflict table for (in)equalities of ordered domains

NC : non−conflicting;
blank : conflicting;
others : iff conditions for non−conflicting

	$X = a$	$X \neq a$	$X = Y$	$X \neq Y$
$U = b$		$a \neq b$		
$U \neq b$	$a \neq b$	NC		NC
$U = V$				
$U \neq V$		NC		NC

Table 2. Conflict table for (in)equalities of unordered domains

NC : non−conflicting;
blank : conflicting;
others : iff conditions for non−conflicting

Proposition 5.1 *The conflicting relation listed in Table 1 and Table 2 is correct.*

Proof.
First notice that the conflicting relation is symmetric, so we only have to prove the entries above and on the diagonal. Since the proofs of the entries are similar, we only proof one of them. The reasoning is applicable to other proofs. We proof that $X = a$ and $U > b$ is non–conflicting if and only if $a > b$, from Table 1.

First, it is easy to see that when $a \leq b$ then $X = a$ and $U > b$ are conflicting, with respect to, say $\{X \geq U\}$.

On the other hand, suppose $a > b$. Let I be a set of (in)equalities such that $I \cup \{X = a\}$ and $I \cup \{U > b\}$ are satisfiable. Let C_1 and C_2 be the $\cong$–equivalence classes of $I \cup \{X = a\}$ and $I \cup \{U > b\}$ respectively. It is easy to see that there will be no new $\cong$–equivalence class of $I \cup \{X = a,\ U > b\}$ other than those in C_1 and C_2. So the first two conditions in Theorem 3.2 are held. Also, since $a > b$, $X = a$, $U > b$, and $a > b$ can not be in a "cycle" specified by condition 3. Therefore, $I \cup \{X = a,\ U > b\}$ is satisfiable.
□

Definition 5.5 (Single–Conflicting (in)equalities)
Let $\mathcal{C} = \{\neg I_1,\ ...,\ \neg I_n\}$ be a set of (in)equality clauses. A literal in $\mathcal{C}$ is called a conflicting literal, *if it is conflicting with itself or some other literals in $\mathcal{C}$. A clause which has at most one conflicting literal is called a* single–conflicting *clause.*

Notice that conflicting literals and single–conflicting clauses are always defined with respect to a specified set of clauses.

Example 5.3
In the set with one clause

$$\{(T = temporary) \vee (T = permanent)\},$$

both literals are conflicting literals, because each literal is conflicting with, say, itself and the other literal. Thus the clause is not a single–conflicting clause.
□

5.4 Units–Refuting Bases and Implementation

In this subsection we give a sufficient condition for an IC–base to be units–refuting. After that, we will discuss how to use our result to lower the complexity of solving the IC–refuting problem for units–refuting bases, and non–units–refuting bases as well.

First of all, we give some lemmas about properties of single conflicting clauses. Our first lemma states that single–conflictingness is preserved under symbol mappings.

Lemma 5.1

Let $\{ic_1 : \neg F_1 \vee \neg I_1,\ ...,\ ic_r : \neg F_r \vee \neg I_r\}$ *be a set of ICs. Let* $\rho_{i,1}, ..., \rho_{i,n_i}$ *be symbol mappings from* ic_i *to some query, for* $i = 1, ..., r$. *Then* $\neg\rho_{i,j}(I_i)$ $(1 \leq i \leq r,\ 1 \leq j \leq n_i)$ *is a single-conflicting clause in*

$$\{\neg\rho_{1,1}(I_1),\ ...,\ \neg\rho_{r,n_r}(I_r)\},$$

if $\neg I_i$ *is a single–conflicting clause in* $\{\neg I_1,\ ...,\ \neg I_r\}$.

Proof.

If $\neg I_i$ is a single–conflicting clause in $\{\neg I_1,\ ...,\ \neg I_r\}$, then there is at most one literal in $\neg I_i$ which is conflicting to some literals in $\{\neg I_1,\ ...,\ \neg I_r\}$. By definition, if a pair of literals c and c' are not conflicting, then $\theta(c)$ and $\rho(c')$ are also not conflicting for any symbol mappings θ and ρ. Therefore $\neg\rho_{i,j}(I_i)$ is also a single–conflicting clause in $\{\neg I_1,\ ...,\ \neg I_r\}$.

□

The following lemma tells us that single–conflictingness is preserved when subset of subclauses are considered.

Lemma 5.2

Let $\mathcal{C}_\mathcal{L} = \{cl_1, ..., cl_n\}$ *and* $\mathcal{C}_\mathcal{L}{}' = \{cl'_{i_1}, ..., cl'_{i_m}\}$ *be two sets of (in)equality clauses, such that* $1 \leq i_1 < ... < i_m \leq n$, *and every* cl'_{i_j} *is a subclause of* cl_{i_j}. *Then* cl'_{i_j} $(j = 1, ..., m)$ *is a single-conflicting clause in* $\mathcal{C}_\mathcal{L}{}'$, *if* cl_{i_j} *is a single conflicting clause in* $\mathcal{C}_\mathcal{L}$.

Proof.

Since each clause in $\mathcal{C}_\mathcal{L}{}'$ is subclause of some clause in $\mathcal{C}_\mathcal{L}$, the single–conflictingness is preserved.

□

Our last lemma tells us that after the units–refutation process, if a clause having two or more literals is a single–conflicting clause, then it is always satisfiable together with other clauses.

Lemma 5.3

Let $\mathcal{C}_{\mathcal{L}1}$ *be a set of (in)equalities, and* $\mathcal{C}_{\mathcal{L}2}$ *be a set of multi–literal (in)equality clauses (ones having two or more literals), such that* $\mathcal{C}_{\mathcal{L}1} \cup \{c_0\}$ *is satisfiable for every literal* c_0 *in* $\mathcal{C}_{\mathcal{L}2}$. *Suppose* $\mathcal{C}_{\mathcal{L}1} \cup \mathcal{C}_{\mathcal{L}2}$ *is unsatisfiable. Then no minimal unsatisfiable subset of* $\mathcal{C}_{\mathcal{L}1} \cup \mathcal{C}_{\mathcal{L}2}$ *contains a single–conflicting clause of* $\mathcal{C}_{\mathcal{L}2}$.

Proof.

Let $\mathcal{C}_\mathcal{L}$ be a minimal unsatisfiable subset of $\mathcal{C}_{\mathcal{L}1} \cup \mathcal{C}_{\mathcal{L}2}$ and cl be a single-conflicting clause in $\mathcal{C}_{\mathcal{L}2}$. Suppose $cl \in \mathcal{C}_\mathcal{L}$, we construct a contradiction as follows.

Since $\mathcal{C}_{\mathcal{L}} - \{cl\}$ is satisfiable, we can form a set $\mathcal{C}$ of literals each from a different clause in $\mathcal{C}_{\mathcal{L}} - \{cl\}$, such that $\mathcal{C}$ is satisfiable. Let c be a non–conflicting literal of cl in $\mathcal{C}_{\mathcal{L}2}$. We know $\mathcal{C} \cup \{c\}$ is unsatisfiable. Let $\mathcal{C}_1$ be a minimal unsatisfiable subset of $\mathcal{C} \cup \{c\}$. $\mathcal{C}_1$ must contain c, and at least one other literal c' from $\mathcal{C}_{\mathcal{L}2}$, as $\mathcal{C}_{\mathcal{L}1} \cup \{c_0\}$ is satisfiable for every literal c_0 in $\mathcal{C}_{\mathcal{L}2}$. Because $\mathcal{C}_1$ is a minimal unsatisfiable set, $\mathcal{C}_1 - \{c\}$ and $\mathcal{C}_1 - \{c'\}$ are both satisfiable. But since c and c' are not conflicting, we know $\mathcal{C}_1$ is also satisfiable. A contradiction.
□

¿From the above three lemmas, we can easily prove the following theorem, which gives a criterion for units–refuting bases.

Theorem 5.2
Let $\mathcal{I}_{\mathcal{C}}$ be a set of ICs in normal form. $\mathcal{I}_{\mathcal{C}}$ is units–refuting, if, starting with its set of (in)equality subclauses, and by repeatedly determining a single–conflicting clause and deleting it from the set, we can get an empty set.

As a very special case, if there are only *equalities* in the (in)equality subformulas of the ICs in $\mathcal{I}_{\mathcal{C}}$, then $\mathcal{I}_{\mathcal{C}}$ is units–refuting. The reason is that equalities become $\neq$–inequalities in the disjunctive clauses, and any pair of $\neq$–inequalities are not conflicting. We have a more interesting example as follows.

Example 5.4
Now we supply some relevant type information for the attributes in the following relation schemes in Example 1.1, **dept(dname, chair)** and **faculty(fname, salary, status)**:

1. **fname** and **chair** have the same type;
2. **fname**, **dname**, and **status** are of different types, but all those types are interpreted in Σ;
3. **salary** has a type interpreted in $\mathbb{R}$.

Now we show that the set of the implication constraints $\mathcal{I}_{\mathcal{C}} = \{ic_1,\ ic_2,\ ic_4\}$ is units–refuting, where ic_1 and ic_2 from Example 1.1 and Example 4.3 are:

$$ic_1:\ \mathbf{dept}(D_1, F_1),\ \mathbf{dept}(D_2, F_2),\ \mathbf{faculty}(F, S, T),\ D_1 \neq D_2,\ F_1 = F,\ F_2 = F\ \rightarrow,$$

$$ic_2:\ \mathbf{dept}(D, F_1),\ \mathbf{faculty}(F, S, T),\ F_1 = F,\ S \leq 80,000\ \rightarrow,$$

and ic_4 is a new implication constraint saying that a temporary faculty member can not have a salary greater than $50,000:

$$ic_4:\ \mathbf{faculty}(F, S, T),\ T = temporary,\ S > 50,000\ \rightarrow,$$

The set of the (in)equality subclauses of $\{ic_1, ic_2, ic_4\}$ consists of the following clauses:

$$cl_1:\ (D_1 = D_2) \vee (F_1 \neq F) \vee (F_2 \neq F),$$

$$cl_2:\ (F_1 \neq F) \vee (S > 80,000),$$

$$cl_4:\ (T \neq temporary) \vee (S \leq 50,000).$$

It is easy to see that here cl_1, cl_2 and cl_4 are all single–conflicting clauses: in cl_1 only $D_1 = D_2$ is a conflicting literal (with itself); in cl_2 only $S > 80,000$ is a conflicting literal (with $S \leq 50,000$); and in cl_3 only $S \leq 50,000$ is a conflicting

literal. So the process of determining single–conflicting clauses needs only one round (to get an empty set of clauses). By Theorem 5.2, $\mathcal{I}_\mathcal{C}$ is a units–refuting base.

However, the IC–base $\{ic_1,\ ic_2,\ ic_3,\ ic_4\}$ is not units–refuting, where ic_3 was given in the previous example:

$$\textbf{faculty}(F, S, T),\ T \neq \mathit{temporary},\ T \neq \mathit{permanent}\ \rightarrow .$$

This is because $(T = \mathit{temporary}) \vee (T = \mathit{permanent})$ is not a single–conflicting clause even without any other clauses. In fact, Example 5.1 shows that units–refutation is not sufficient to prove $\{ic_3\} \models (Q \equiv \{\})$, for $Q = \{\langle F\rangle : \textbf{faculty}(F, S_1, T_1), \textbf{faculty}(F, S_2, T_2), \textbf{faculty}(F, S_3, T_3), T_1 \neq T_2, T_1 \neq T_3, T_2 \neq T_3\}$.

In the next section we will show that $\{ic_1,\ ic_2,\ ic_3,\ ic_4\}$ is units–refuting for queries having distinct predicate names. □

Now that we have given a sufficient condition for units–refuting bases, we discuss the issue of implementing an efficient algorithm to solve the IC–refuting problem. In fact, the three lemmas above (Lemma 5.1, 5.2, 5.3) provide some heuristics for the IC–refuting problem even for non–units–refuting bases.

Given an IC–base $\mathcal{I}_\mathcal{C}$ and a query Q, first we find out all the symbol mappings from the ICs to Q. In the worst case, there are exponentially many symbol mappings. But in practical cases, this number is usually very restricted due to the following factors: 1) only those ICs whose predicate names also appear in the query can be mapped to the query; 2) the number of mappings also depends on the number of times a predicate name appears in the query. More specifically, if the query only has distinct predicate names (more discussion in next section), there will be polynomial number of symbol mappings.

Suppose $\mathcal{C}_\mathcal{L} = \{I,\ cl_1, ...,\ cl_r\}$ is the set of (in)equalities of Q and the (in)equality subclauses of the ICs after the symbol mappings. We know that $\mathcal{I}_\mathcal{C} \models Q \equiv \{\}$ if and only if $\mathcal{C}_\mathcal{L}$ is unsatisfiable. First we do units–refutation on $\mathcal{C}_\mathcal{L}$. Upon termination, we get the updated $\mathcal{C}_\mathcal{L}$.

If $\mathcal{I}_\mathcal{C}$ is units–refuting, we can tell at once whether $\mathcal{I}_\mathcal{C} \models (Q \equiv \{\})$ by looking for the empty clause. If $\mathcal{I}_\mathcal{C}$ is not units–refuting, we can repeatedly delete the multi–literal single–conflicting clauses from $\mathcal{C}_\mathcal{L}$, and do exponential checking only on the remaining clauses. Although the asymptotic complexity is not changed, this method is usually more efficient, because it reduces the search space for the unsatisfiable subset of (in)equality clauses.

5.5 Units–Refuting Bases for DPN–Queries

In practice, we often encounter queries which do not have multiple conjuncts with the same name. We will call such queries *distinct–predicate–name* (DPN) queries. In this section, we are going to discuss the criterion of the units–refuting bases for DPN–queries. In this case, the IC–refuting problem is polynomial, because there are only polynomial number of symbol mappings from the implication constraints to the query. In fact, given a DPN–query, there is at most one instance of each implication constraint involved in the refutation. So, in searching for single–conflicting clauses, we don't need to consider conflicting literals within the same clause. This is formally defined as follows:

Definition 5.6 (Inter–Conflicting)
Given a set of (in)equality clauses, a literal is called a inter–conflicting literal *if it is conflicting with a literal in some other clause. A clause which has at most one inter–conflicting literal is called a* single–inter–conflicting *clause.*

Single–inter–conflicting clauses play a similar role as single–conflicting clauses. In fact, if we replace the phrase "single–conflicting" by "single–inter–conflicting" in Lemma 5.2 and Lemma 5.3, the results are still true.

The following lemma and theorem tells us that we can allow some kind of clauses *other than* the single–inter–conflicting clauses and still have a units–refuting base for DPN–queries.

Lemma 5.4

Let $\mathcal{C}_{\mathcal{L}1}$ be a set of (in)equalities, and $\mathcal{C}_{\mathcal{L}2}$ be a set of multi–literal (in)equality clauses, such that $\mathcal{C}_{\mathcal{L}1} \cup \{c_0\}$ is satisfiable for every literal c_0 in $\mathcal{C}_{\mathcal{L}2}$. Then $\mathcal{C}_{\mathcal{L}1} \cup \mathcal{C}_{\mathcal{L}2}$ is satisfiable if $\mathcal{C}_{\mathcal{L}2}$ has the following property (*):

Every literal c in $\mathcal{C}_{\mathcal{L}2}$ is conflicting with at most one literal in some other clause in $\mathcal{C}_{\mathcal{L}2}$.

Proof.

We can build a satisfiable set, $\mathcal{C}$, of (in)equalities each from a different clause in $\mathcal{C}_{\mathcal{L}2}$. Initially, $\mathcal{C}$ contains a literal c from a clause cl of $\mathcal{C}_{\mathcal{L}2}$, and we delete cl from $\mathcal{C}_{\mathcal{L}2}$. While $\mathcal{C}_{\mathcal{L}2}$ is not empty we repeat the following steps:

1. choose a clause from $\mathcal{C}_{\mathcal{L}2}$ which has a conflicting literal c' with the last inserted literal of $\mathcal{C}$ (if there is no such clause then pick any clause);
2. pick a literal other than c' from the clause, put it into $\mathcal{C}$, and delete the clause from $\mathcal{C}_{\mathcal{L}2}$.

It is easy to see that $\mathcal{C}$ thus formed consists of literals pairwisely non–conflicting, so $\mathcal{C}_{\mathcal{L}1} \cup \mathcal{C}$ is satisfiable. Hence $\mathcal{C}_{\mathcal{L}1} \cup \mathcal{C}_{\mathcal{L}2}$ is satisfiable.

□

Theorem 5.3

An IC–base $\mathcal{I}_{\mathcal{C}}$ is DPN–units–refuting, if, starting with the set of its (in)equality subclauses, and by iteratively determining a single–inter–conflicting clause and deleting it from the set, we can get a set of clauses satisfying the property (*) in Lemma 5.4.

Proof.

Straightforward from Lemma 5.4.

□

Example 5.5

Continued from the last example. We show $\{ic_1,\ ic_2,\ ic_3,\ ic_4\}$ is a DPN–units–refuting base. The corresponding set of (in)equality clauses is

$$\{cl_1,\ cl_2,\ cl_3 : (T = temporary) \vee (T = permanent),\ cl_4\}.$$

Since a single–conflicting clause is also a single–inter–conflicting clause, so by Theorem 5.3, we get $\{cl_3\}$ after deleting cl_1, cl_2, and cl_4. Obviously this set satisfies the property (*) of Lemma 5.4, as it has only one clause.

□

If an IC–base is not a DPN–units–refuting base, the implementation principle discussed at the end of the previous section is still applicable: after the units–refutation process, a multi–literal single–inter–conflicting clause can not be in any minimal unsatisfiable set of clauses. Hence we can delete the multi–literal single–inter–conflicting clauses, and do the exponential checking only on the remaining clauses.

6 Conclusion and Future Work

In this paper, we have given a necessary and sufficient condition for the IRC–refuting problem, which is a central problem in reasoning with implication and referential constraints. Solving the IRC–refuting problem allows us to answer inconsistent queries without actually searching the databases.

Next we have studied, based on the result of the IRC–refuting problem, the following problems found in semantic query optimization and constraint maintenance: the query containment problem; the redundant implication constraint problem; the redundant selection–conditions problem; and the redundant conjunct problem.

We have shown that they are either (polynomially) equivalent or reducible to the IRC–refuting problem. The reductions are obtained on the observation that the implication constraints and queries are playing complementary roles and can be transformed to one another.

As the last but not the least contribution, we have addressed the complexity issue of a special case of the IRC–refuting problem, i.e., the IC–refuting problem. We have given two criteria for designing a set of implication constraints so that an efficient "units–refutation" process can be used to solve the IC–refuting problem.

For this type of IC–bases, the complexity of the IC–refuting problem can be reduced from Π_2^p–complete to NP–complete, and even to polynomial in a more practical case. For IC–bases not totally satisfying our characterization, we can still use our results as heuristics.

To extend the present work, here we suggest the following research directions:

1. To find new efficient algorithms for the subsumption problem, as it can also be used in the areas of *constraint logic programming* and *constraint query language* processings.

2. To extend the results about the units–refuting IC–base to IRC–base, so that the complexity of the IRC–refuting problem can be reduced;

Semantic integrity constraints have been playing more and more important roles in databases and knowledge–base systems. The major new research trend in this area concerning the constraints would be to extend the existing results in constraint enforcement and applications to new generation of database systems, namely the object–oriented and deductive database systems.

References

[1] A.V. Aho, Y. Sagiv, and J.D. Ullman, "Equivalences among relational expressions", SIAM J. Comput. 8,2(May 1979), 218–246

[2] A.V. Aho, Y. Sagiv, and J.D. Ullman, "Efficient optimization of a class of relational expressions", ACM TODS 4, 4(Dec 1979), 435–454.

[3] M.A.Casanova, R. Fagin, and C. H. Papadimitriou, "Inclusion Dependencies and Their Interactions with Functional Dependencies", JCSS 28,1984.

[4] A.K. Chandra, and P.M. Merlin, "Optimal implementation of conjunctive queries in relational databases", Proc. ACM STOC, 77-90, 1976.

[5] U.S. Chakravarthy, J.Grant, and J. Minker, "Logic-Based Approach To Semantic Query Optimization", ACM TODS, Vol.15, 1990, pp 162-207.

[6] C-L Chang and R. C. Lee, Symbolic Logic and Mechanical Theorem Proving, Academic Press, 1973.

[7] A.K. Chandra, and M. Y. Vardi, "The Implication Problem for Functional and Inclusion Dependencies is Undecidable", 1983.

[8] S. Cosmadakis and P. C. Kanellakis, "Functional and Inclusion Dependencies: A Graph Theoretic Approach", Proc. of ACM Symp. on PODS, 1984.

[9] P.Dublish, J.Biskup and Y.Sagiv, "Optimization of a Subclass of Conjunctive Queries", LNCS 470(ICDT'90). pp455–469.

[10] N. Dershowitz, "Termination", Proc. 1st Int'l Conf. on RTA, Dijon, France, 1985, LNCS 202, pp 180-224.

[11] C. Elkan, "Independence of Logic Database Queries and Updates", Proc. of 9th ACM Symp. on PODS, pp 154-160, 1990.

[12] M. M. Fonkam and W. A. Gray, "Employing Integrity Constraints for Query Modification and Intensional Answer Generation in Multi-database systems", LNCS 618, pp 244-260, 1992.

[13] C.M. Garey and D.S. Johnson, Computers and Intractability: A Guide to the theory of NP-completeness. W.Freeman and Co., San Francisco, 1979.

[14] J. Han, "Constraint-Based Reasoning in Deductive Databases", Proc. of 7th Data Engineering, 1991, pp 257-265.

[15] M.M. Hammer and S.B. Zdonik, "Knowledge Based Query Processing", Proc. 6th VLDB, 1980, pp 137-147.

[16] D. Johnson, and A.Klug, "Optimizing conjunctive queries that contain untyped variables", SIAM J. comput. 12,4(Nov. 1983), 616-640.

[17] D.Johnson and A. Klug, "Testing containment of conjunctive queries under functional and inclusion dependencies", JCSS. 28, 1, 1984. 167-189.

[18] P.C.Kanellakis, G.M. Kuper, and P.Z. Revesz, "Constraint Query Languages", Proc. of 9th ACM Symp. on PODS, pp 299-313, 1990.

[19] J.J. King, "QUIST: A System for Semantic Query Optimization in Relational Databases", Proc. 7th VLDB, 1981, pp 510-517.

[20] A.Klug, "On Conjunctive Queries Containing Inequalities", JACM vol 35:1,pp 147-160,1988.

[21] S. Lee and J. Han, "Semantic Query Optimization in Recursive Databases", Proc. of 4th Data Engineering, 1988, pp 444-451.

[22] A. Levy and Y. Sagiv, "Constraints and Redundancy in Datalog", Proc. of 11th ACM Symp. on PODS, pp 67-80, 1992.

[23] A. Levy and Y. Sagiv, "Semantic Query Optimization in Datalog Programs", Proc. of 14th ACM Symp. on PODS, pp 163-173, 1995.

[24] R. van der Meyden, "The Complexity of Querying Infinite Data About Linearly Ordered Domains", Proc. of the 11th ACM Symp. on PODS, pp331-345, 1992.

[25] J. C. Mitchell, "Inference Rules for Functional and Inclusion Dependencies", Proc. of ACM Symp. on PODS, 1983.

[26] J. C. Mitchell, "The Implication Problem for Functional and Inclusion Dependencies", Rep. MIT/LCS/TM-235, 1983.

[27] A. Motro, "Using Integrity Constraints to Provide Intensional Answers to Relational Queries", Proc. 15th VLDB, 1989, pp 237-246.

[28] N.S. Ishakbeyoglu and Z.M.Ozsoyoglu, "On The Maintenance of Implication Integrity Constraints", Proc. of DEXA'93, pp 221-232.

[29] Z. M. Ozsoyoglu, "Query Optimization in Distributed Databases", Ph.D. Thesis, Dept. of CS, Univ. of Alberta, Canada, 1980.

[30] H.H. Pang, H.J. Lu, and B.C. Ooi, "An Efficient Query Optimization Algorithm", Proc. 7th Data Engineering, 1991, pp 326-335.

[31] A. Pirotte and D. Roelants, "Constraints for Improving the Generation of Intensional Answers in a Deductive Database", Proc. 5th Data Engineering, 1989, pp 652-659.

[32] Y. Sagiv, "Quadratic algorithms for minimizing joins in restricted relational expressions" SIAM J.comput. 12,2(May 1983), pp316–329.

[33] Y. Sagiv and M. Yannakakis, "Equivalences among relational expressions with union and difference operators", JACM 27,4(Oct 80), pp633–655.

[34] E. Sciore, "Inclusion Dependencies and the Universal Instance", Proc. of ACM Symp. on PODS, 1983.

[35] S.T. Shenoy and Z.M. Ozsoyoglu, "A System for Semantic Query Optimization", Proc. SIGMOD, 1987.

[36] Srivastava, "Subsumption and Indexing in Constraint Query Languages with Linear arithmetic Constraints", Annals of Mathematics and Artificial Intelligence, vol. 8, 1993, pp315-343.

[37] J.D. Ullman, Principles of Database and Knowledge-Base systems, volume I., Computer Science Press, 1989.

[38] J.D. Ullman, Principles of Database and Knowledge-Base systems, volume II., Computer Science Press, 1989.

[39] Xubo Zhang and Z.M.Ozsoyoglu, "On Efficient Reasoning with Implication Constraints", Proc. of the 3rd Int'l Conf. on Deductive and Object–Oriented Databases, 1993.

[40] Xubo Zhang and Z.M.Ozsoyoglu, "Some Results on the Containment and Minimization of (In)equality Queries", Information Processing Letters, 50(5), 1994.