# Verification of Knowledge Bases based on Containment Checking

**Alon Y. Levy**
AT&T Laboratories
AI Principles Research Department
levy@research.att.com

**Marie-Christine Rousset**
L.R.I. U.R.A C.N.R.S
University of Paris-Sud, France
mcr@lri.lri.fr

## Abstract

Building complex knowledge based applications requires encoding large amounts of domain knowledge. After acquiring knowledge from domain experts, much of the effort in building a knowledge base goes into verifying that the knowledge is encoded correctly. We consider the problem of verifying hybrid knowledge bases that contain both Horn rules and a terminology in a description logic. Our approach to the verification problem is based on showing a close relationship to the problem of *query containment*. Our first contribution, based on this relationship, is presenting a thorough analysis of the decidability and complexity of the verification problem, for knowledge bases containing recursive rules and the interpreted predicates $=$, $\leq$, $<$ and $\neq$. Second, we show that important new classes of constraints on correct inputs and outputs can be expressed in a hybrid setting, in which a description logic class hierarchy is also considered, and we present the first complete algorithm for verifying such hybrid knowledge bases.

## Introduction

Building complex knowledge based applications requires encoding large amounts of domain knowledge. After acquiring this knowledge from domain experts, much of the effort in building a knowledge base goes into verifying that the knowledge is encoded correctly. A knowledge base is verified if, for any *correct* set of inputs, the knowledge base will entail a *correct* set of outputs. Verifying a knowledge base manually is both unwieldy and unlikely to find all the possible errors in the knowledge base. Therefore, several authors have considered the problem of building tools to assist knowledge base verification. The space of verification problems varies depending on the representation language used for the knowledge base, and the way we specify constraints defining correct inputs and outputs.

This paper considers the verification problem for knowledge bases containing Horn rules and hybrid knowledge bases containing terminologies in a KL-ONE style language [Brachman and Schmolze, 1985] in addition to Horn rules. We describe novel algorithms for verifying such knowledge bases, and obtain new results concerning the complexity and decidability of the verification problem. Our algorithms also handle a wider class of input and output constraints.

We begin by showing that the verification problem is closely related to the problem of *query containment*, that has been studied in the database literature. As a result, we obtain algorithms for deciding the verification problem for knowledge bases containing Horn rules, and *tight* complexity bounds on the problem. Our results consider the cases of function-free Horn rules that may be recursive and contain the interpreted predicates $=$, $\leq$, $<$ and $\neq$. In contrast, previous work has only considered non recursive Horn rule knowledge bases, and has not given any complexity or decidability results about the problem. Next, we describe the first complete algorithm for verifying hybrid knowledge bases that contain a terminology in the description logic $\mathcal{ALCNR}$ [Baader and Hollunder, 1991; Buchheit *et al.*, 1993] in addition to Horn rules.

In most works, the constraints defining correct inputs and outputs are specified also using Horn rules whose consequent is a *bad* predicate. An input (or output) is considered to be correct if the bad predicate is not entailed. In many domains Horn rules are not sufficient for describing correct inputs and outputs. In particular, it is often natural to express constraints using *tuple generating dependencies* (tgd's) [Ullman, 1989]. In such constraints, the right hand side of the implications may also be a conjunction in which some of the variables are existentially quantified. The connection between the query containment and the verification problem also shows that verifying knowledge bases is undecidable when input or output constraints are specified using tuple generating dependencies. This is because the problem of entailment between tgd's is undecidable.

We identify a novel class of *separable* tuple generating dependencies. We show that in the context of our hybrid knowledge representation language we can translate separable tgd's into Horn rules. In conjunction with our algorithm for verifying hybrid knowledge bases, we obtain a method for handling verification problems in which the input and output constraints are specified using separable tgd's. This result also entails an algorithm for query containment in the presence of separable tgd's.

## Definition of the verification problem

Informally, a knowledge base is intended to model a space of problems. Given a problem instance, the solution to the problem will be some set of facts that are entailed by the union of the knowledge base and the problem instance. We say that the knowledge base is verified if, for any set of *correct* input problem instances, we will only entail *correct* outputs.

In our discussion we consider knowledge bases that include a set of function-free Horn rules, i.e., logical sentences of the form: $p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \Rightarrow q(\bar{Y})$ where $\bar{X}_1, \ldots, \bar{X}_n, \bar{Y}$ are tuples of variables or constants. We require that the rules are safe, i.e., a variable that appears in $\bar{Y}$ appears also in $\bar{X}_1 \cup \ldots \cup \bar{X}_n$. We distinguish the set of *base predicates* as those predicates that do not appear in the consequents of the Horn rules. The problem instances will be specified as ground atomic facts for some of the base predicates. We also allow the *interpreted predicates* $\leq$, $<$, $=$ and $\neq$ to appear in the antecedents of the rules. Note that a ground atomic fact of the form $p(\bar{a})$ is also a (trivial) instance of a Horn rule.

Given a set of rules $\mathcal{R}$ and a predicate $P$ appearing in $\mathcal{R}$, we define the set of rules relevant to $P$ in $\mathcal{R}$, denoted by $Rules(P)$ as the minimal subset of $\mathcal{R}$ that satisfy the following conditions:

1. If $P$ is the predicate in the consequent of the rule $r$, then $r \in Rules(P)$, and
2. If the predicate $Q$ appears in a rule $r \in Rules(P)$, then any rule whose consequent has $Q$ is also in $Rules(P)$.

Given a set of rules, we can define a dependency graph, whose nodes are the predicates appearing in the rules. There is an arc from the node of predicate $Q$ to the node of predicate $P$ if $Q$ appears in the antecedent of a rule whose consequent predicate is $P$. The rules are said to be *recursive* if there is a cycle in the dependency graph.

When the rules are not recursive, we can *unfold* them. That is, obtain a logically equivalent set of rules such that the only predicates appearing in the antecedents of the rules are base predicates. It should be noted that the process of unfolding can result in an exponential number of rules. However, the exponent is only in the *depth* of the set of rules.

The semantics of our knowledge bases is the standard first-order logic semantics. An *interpretation* $I$ of a KB $\Delta$ contains a non-empty domain $\mathcal{O}^I$. It assigns a n-ary relation $P^I$ over the domain $\mathcal{O}^I$ to every n-ary predicate $P \in \Delta$, and an element $a^I \in \mathcal{O}^I$ to every constant $a \in \Delta$. An interpretation $I$ is a model of a Horn rule $r$ if, whenever $\alpha$ is a mapping from the variables of $r$ to the domain $\mathcal{O}^I$, such that $\alpha(\bar{X}_i) \in p_i^I$ for every atom of the antecedent of $r$, then $\alpha(\bar{Y}) \in q^I$, where $q(\bar{Y})$ is the consequent of $r$. Finally, $I$ is a model $\Delta$ if it is a model of every rule in $\Delta$.

## Correct problem instances

A problem instance $G$ is a set of ground atomic facts for the base predicates. Given a problem instance $G$, the solution to the problem is the set of ground atomic facts entailed from $\Delta \cup G$. The goal of verifying a knowledge base is to make sure that the knowledge base does not enable entailing facts that contradict integrity constraints that are known to hold in the domain. We express such *output constraints* by a set of Horn rules defining a special predicate of arity 0, $P_{out}$. We say that the output of $\Delta \cup G$ is *correct* if $\Delta \cup G \not\models P_{out}$. Similarly, we do not want to consider that all sets of ground facts are *valid* inputs (i.e., problem instances). Therefore, we assume that $\Delta$ also contains a set of rules defining *input constraints* by a special predicate of arity 0, $P_{in}$. A set of ground atomic facts $G$ is a valid input if $\Delta \cup G \not\models P_{in}$.

We can now formally define the knowledge base verification problem:

**Definition 1:** *Let $\Delta$ be a knowledge base containing the predicates $P_{in}$ and $P_{out}$ describing valid inputs and correct outputs. The knowledge base $\Delta$ is said to be verified if, for any set of ground atomic facts $G$, for which $\Delta \cup G \not\models P_{in}$, then $\Delta \cup G \not\models P_{out}$.* □

It should be noted that the verification problem is not equivalent to the unsatisfiability of the formula $\Delta \wedge P_{out} \wedge \neg P_{in}$.[1] In cases where all the rules are non recursive and unfolded, the verification problem can be formulated as a problem of logical entailment. In fact, the results we present in the subsequent sections can also be viewed as providing the complexity of these specialized forms of entailment.

Our definition of the verification problem differs slightly from previous definitions that were proposed in the literature (e.g., [Rousset, 1988; Ginsberg, 1988; Ginsberg and Williamson, 1993; Loiseau and Rousset, 1993]). The definition in those works did not distinguish between the predicates $P_{in}$ and $P_{out}$, and used a single **false** predicate for defining incorrect inputs and outputs. Neither formulation of the verification problem is more expressive than the other. As we see

[1]The formula $\Delta \wedge P_{out} \wedge \neg P_{in}$ is satisfiable if there is *some* model of the predicates that satisfies each rule in $\Delta$ and $P_{out}$ and $\neg P_{in}$. However, the knowledge base is not verified only if there is a *least fixed point* model of the formula that is obtained by applying the rules to an *initial* set of ground facts for the base predicates.

shortly, our formulation makes the connection with the query containment problem more explicit.

**Example 1:** We use the following illustrative example throughout the paper. Consider a domain of approving curricula for college students. The university has two disjoint types of students, engineering and humanities students, whose instances are described by the unary predicates $EngStud$ and $HumStud$. Courses are either basic or advanced, described by the predicates $Basic$ and $Adv$, and they are either engineering courses or humanities courses, described by $EngCourse$ and $HumCourse$. Inputs describe which courses the student *wants* to take, and which courses the student has already taken. The atom $Want(s,c)$ denotes that student $s$ wants to take course $c$ during the current year, and $Prev(s,c)$ denotes that $s$ has already taken $c$ in a previous year. The output is the set of courses that the student will take. The atom $Take(s,c)$ denotes that $s$ will take course $c$. The following rules describe our domain.

$r_1 : Want(s,c) \wedge Qualifies(s,c) \Rightarrow Take(s,c)$
$r_2 : PrereqOf(c_1,c_2) \wedge Prev(s,c_2) \Rightarrow Qualifies(s,c_1)$
$r_3 : Year(s,n) \wedge Mandatory(c,n) \Rightarrow Take(s,c)$

Rule $r_1$ says that students can take a course they want if they are qualified for it. Rule $r_2$ says that students are qualified for a course if they took one of its prerequisite courses. Finally, rule $r_3$ guarantees that students will take the courses that are mandatory for their year.

The following is the output constraint rule stating that humanities students cannot take advanced engineering courses:

$r_4 : HumStud(s) \wedge Adv(c) \wedge EngCourse(c) \wedge$
$\quad Take(s,c) \Rightarrow P_{out}$.

The following two rules describe the input constraints specifying that engineering students are disjoint from humanities students, and that students do not want to take courses they have already taken.

$r_5 : EngStud(s) \wedge HumStud(s) \Rightarrow P_{in}$
$r_6 : Want(s,c) \wedge Prev(s,c) \Rightarrow P_{in}$

Our knowledge base is *not* verified, because we can have a valid input for which we can derive a incorrect output. Specifically, consider the following valid input:

$\{Want(S_1,C_2), HumStud(S_1), Adv(C_2), Prev(S_1,C_1),$
$PrereqOf(C_2,C_1), EngCourse(C_2)\}$

The student $S_1$ wants to take the advanced engineering course $C_2$. $S_1$ qualifies for the course by having taken the prerequisite $C_1$. In this case, the knowledge base would entail $Take(S_1,C_2)$, which entails $P_{out}$, i.e., the output is incorrect.

The knowledge base designer can correct the problem by either modifying the knowledge base (e.g., refining the rule $r_2$), or by adding the (possibly very likely) input constraint that states that humanities students are never interested in advanced engineering courses.

### Verification and the containment problem

Our approach to solving the verification problem is based on showing a close connection to the problem of *query containment*, that has been considered in the database literature. In the next section, we show that the relationship between these two problems yields several new results about verifying Horn rule knowledge bases. In particular, it provides a set of core results about the complexity and decidability of the problem.

The query containment problem is to decide whether in any minimal fixed-point model of the knowledge base, the extension of one predicate contains the extension of another. Formally, given a knowledge base $\Delta$ and a set of ground facts $G$, we can entail a (finite) set of ground atomic facts for every predicate $P \in \Delta$. We denote by $P^{\Delta}(G)$ the set of tuples $\bar{a}$, such that $\Delta \cup G \models P(\bar{a})$. If $P$ is a proposition, i.e., a predicate of arity 0, then $P^{\Delta}(G)$ is the the set containing the empty list if $\Delta \cup G \models P$, and the empty set otherwise.

**Definition 2:** *Let $P_1$ and $P_2$ be two predicates of the same arity in the knowledge base $\Delta$. The predicate $P_1$ is contained in $P_2$, denoted by $P_1 \subseteq P_2$, if for any set of ground atomic facts $G$, $P_1^{\Delta}(G) \subseteq P_2^{\Delta}(G)$.*

Note that when $P_1$ and $P_2$ are propositions, the definition says that whenever $P_1$ is entailed by $\Delta \cup G$, then so is $P_2$.

The following theorem formalizes the connection between the verification and containment problems:

**Theorem 1:** *Let $\Delta$ be a knowledge base with predicates $P_{in}$ and $P_{out}$ describing correct inputs and outputs. The knowledge base $\Delta$ is verified if and only if $P_{out} \subseteq P_{in}$.*

## The complexity of verification

Previous work on the verification of knowledge bases did not consider the fundamental decidability and complexity of the problem. In contrast, the connection between the verification and containment problems yields several core decidability and complexity results. This section describes these results.

Most previous work on verification considered algorithms for verifying non recursive Horn rule knowledge bases without interpreted predicates. The following theorem establishes results about the inherent complexity of the problem. Furthermore, the theorem provides the first results concerning the verification problem for recursive Horn rules. We assume in our discussion that given a knowledge base $\Delta$, when the rules $Rules(P_{in})$ and $Rules(P_{out})$ are not recursive then they are unfolded.

**Theorem 2:** *Let $\Delta$ be a Horn-rule knowledge base without interpreted predicates. Let $P_{in}$ and $P_{out}$ be predicates in $\Delta$ describing correct inputs and outputs, respectively.*

1. *If both $Rules(P_{in})$ and $Rules(P_{out})$ are not recursive, then the verification problem is NP-Complete in*

*the size of the rules in $Rules(P_{in})$ and $Rules(P_{out})$ and polynomial in the number of rules $Rules(P_{in})$ and $Rules(P_{out})$.*

2. *If one of $Rules(P_{in})$ or $Rules(P_{out})$ are recursive, but not both of them, then the verification problem is complete for doubly exponential time in the size of the rules in $Rules(P_{in})$ and $Rules(P_{out})$ and polynomial in the number of rules in $Rules(P_{in})$ and $Rules(P_{out})$.*
3. *If both $Rules(P_{in})$ and $Rules(P_{out})$ are recursive, then the verification problem is undecidable.*

The algorithm and complexity results for the first case of the theorem follow from [Sagiv and Yannakakis, 1981]. The results of the second case follow from [Chaudhuri and Vardi, 1992]. The undecidability result follows from [Shmueli, 1987].

The connection between the verification and containment also provides core complexity results for verifying Horn rule knowledge bases that include the interpreted order predicates $\leq$, $<$, $=$ and $\neq$ in the antecedents of the rules. The following theorem provides a precise characterization of the complexity of the verification problem in this case.

**Theorem 3:** *Let $\Delta$ be a Horn-rule knowledge base, possibly with the interpreted predicates $\leq$, $<$, $=$ and $\neq$ in the antecedents of the rules. Let $P_{in}$ and $P_{out}$ be predicates in $\Delta$ describing correct inputs and outputs respectively.*

1. *If both $Rules(P_{in})$ and $Rules(P_{out})$ are not recursive, then the verification problem is $\Pi_2^P$-Complete in the size of the rules in $Rules(P_{in})$ and $Rules(P_{out})$ and the number of constants appearing in the rules. It is polynomial time in the number of rules $Rules(P_{in})$ and $Rules(P_{out})$.*
2. *If the rules in $Rules(P_{in})$ are recursive and $Rules(P_{out})$ are not recursive, then the verification problem is decidable and it is complete for $\Pi_2^P$ in the size of the rules in $Rules(P_{in})$ and $Rules(P_{out})$ and the number of constants appearing in them, and it is polynomial in the number of rules in $Rules(P_{in})$ and $Rules(P_{out})$.*
3. *If the rules in $Rules(P_{out})$ are recursive, then the verification problem is undecidable.*

Note that in the above theorem there is an asymmetry between the rules defining $P_{in}$ and those defining $P_{out}$. An algorithm and the upper complexity bound for the first part of the theorem follow from [Klug, 1988]. The lower bound for the first part of the theorem and the undecidability result follow from [van der Meyden, 1992]. Finally, if the rules in $Rules(P_{in})$ do not contain interpreted predicates, but the rules in $P_{out}$ do contain interpreted predicates, then it follows from [Levy and Sagiv, 1995] that the verification problem is decidable also in the third case of the theorem.

## Specifying input and output constraints

An important aspect of the knowledge base verification problem is how input and output constraints are described. In our problem definition and in most previous work in the field the constraints were specified by Horn rules defining the *bad* predicates $P_{in}$ and $P_{out}$. However, Horn rules are not always expressive enough for describing constraints that arise in applications.

**Example 2:** Suppose we want to express the constraint on the domain of our example stating that engineering students who want to take an advanced humanities course *must* have previously taken a basic humanities course. Formally, we could state the constraint with the following formula which is not a Horn rule:

$$EngStud(s) \wedge Want(s,c) \wedge Adv(c) \wedge HumCourse(c) \\ \Rightarrow (\exists c_1) Prev(s,c_1) \wedge Basic(c_1) \wedge HumCourse(c_1).$$

The above example is an instance of a *tuple generating dependency* constraint (tgd) [Fagin, 1982; Beeri and Vardi, 1984; Yannakakis and Papadimitriou, 1980]. A tuple generating dependency constraint is a formula of the form

$$p_1(\bar{X}_1) \wedge \ldots \wedge p_n(\bar{X}_n) \Rightarrow (\exists \bar{Y}) q_1(\bar{Y}_1) \wedge \ldots \wedge q_m(\bar{Y}_m).$$

The tuple $\bar{Y}$ includes the variables that appear in the right hand side and not on the left hand side. All other variables are universally quantified. Such a formula states that whenever there are facts in the knowledge base such that the conjunction on the left hand side is satisfied, then the knowledge base must also *include* facts such that the conjunction on the right hand side is satisfied.

An example of the usage of such constraints is to express constraints that describe *test cases*, which are often a natural way for an expert to describe domain constraints. That is, the expert can specify what needs to hold on the output (the right hand side of a tgd) given a certain input (the left hand side).

In [Vardi, 1984; Gurevich and Lewis, 1982] it is shown that the problem of deciding whether one tgd entails another is undecidable. Consequently, it follows that the verification problem is undecidable if we were able to express input and output constraints using arbitrary tgds.

In the next section we show how to verify hybrid knowledge bases that contain a set of *extended* Horn rules. Extended Horn rules contain predicates that are defined in a description logic terminology in addition to ordinary predicates. In this section we identify the class of *separable* tgd's, and show they can be *translated* to extended Horn rules whose consequents are the predicates $P_{in}$ and $P_{out}$. As a result, the algorithm presented in the next section provides a method for handling verification problems in which the input and output constraints are specified by separable tgd's.

**Example 3:** We first illustrate how separable tgd's are translated to extended Horn rules using Example 2.

Informally, description logics will enable us to define the class of students that *do not* satisfy the right hand side of the tgd. We begin by considering the predicates $Basic$ and $HumCourse$ as *primitive classes*, and the predicate $Prev$ as a property of objects. In a description logic we can define *complex classes*. The description $Basic \sqcap HumCourse$ denotes the class of objects that are basic humanities courses. We define the class $C_{tgd}$ by the description $\forall Prev.\neg(Basic \sqcap HumCourse)$ which denotes precisely the class of objects, such that fillers of the property $Prev$ do not belong to the class $Basic \sqcap HumCourse$. We now use the class $C_{tgd}$ as a predicate in an extended Horn rule. Specifically, the tgd can be translated into the following rule:

$$EngStud(s) \wedge Want(s,c) \wedge Adv(c) \wedge$$
$$HumCourse(c) \wedge C_{tgd}(s) \Rightarrow P_{in}.$$

We begin by formally defining description logic terminologies and extended rules. We then describe the algorithm for translating separable tgd's to extended Horn rules.

## Hybrid knowledge bases

A description logic is a subset of first order logic, which is especially designed to describe rich hierarchical structures. A description logic contains unary relations (called concepts) which represent classes of objects in the domain and binary relations (called roles) which describe relationships between objects. A description logic uses a set of *constructors* to build complex concept and role descriptions. The set of *constructors* varies from one language to another. In our discussion we consider the rather expressive description logic $\mathcal{ALCNR}$ (which has formed the basis for the KRIS system [Baader and Hollunder, 1991]), in which descriptions can be built using the following grammar ($A$ denotes a concept name, $P_i$'s denote role names, $C$ and $D$ represent concept descriptions and $R$ denotes a role description):

| | | |
|---|---|---|
| $C, D \rightarrow$ | $A \mid$ | (primitive concept) |
| | $\top \mid \bot \mid$ | (top, bottom) |
| | $C \sqcap D \mid C \sqcup D \mid$ | (conjunction, disjunction) |
| | $\neg C \mid$ | (complement) |
| | $\forall R.C \mid$ | (universal quantification) |
| | $\exists R.C \mid$ | (existential quantification) |
| | $(\geq n\ R) \mid (\leq n\ R)$ | (number restrictions) |
| $R \rightarrow$ | $P_1 \sqcap \ldots \sqcap P_m$ | (role conjunction) |

A terminology $\mathcal{T}$ is a set of *inclusion statements*, which are of the form $C \sqsubseteq D$, where $C$ and $D$ are concept descriptions. Intuitively, an inclusion states that every instance of the concept $C$ must be an instance of $D$. Formally, the semantics of a terminology is given via *interpretations*, that assign a unary relation $C^I$ to every concept name in $\mathcal{T}$ and a binary relation $R^I$ over $\mathcal{O}^I \times \mathcal{O}^I$ to every role name in $\mathcal{T}$. The extensions of concept and role descriptions are given by the following equations: ($\sharp\{S\}$ denotes the cardinality of a set $S$):

$\top^I = \mathcal{O}^I$, $\bot^I = \emptyset$, $(C \sqcap D)^I = C^I \cap D^I$,
$(C \sqcup D)^I = C^I \cup D^I$, $(\neg C)^I = \mathcal{O}^I \setminus C^I$,
$(\forall R.C)^I = \{d \in \mathcal{O}^I \mid \forall e : (d,e) \in R^I \rightarrow e \in C^I\}$
$(\exists R.C)^I = \{d \in \mathcal{O}^I \mid \exists e : (d,e) \in R^I \wedge e \in C^I\}$
$(\geq n\ R)^I = \{d \in \mathcal{O}^I \mid \sharp\{e \mid (d,e) \in R^I\} \geq n\}$
$(\leq n\ R)^I = \{d \in \mathcal{O}^I \mid \sharp\{e \mid (d,e) \in R^I\} \leq n\}$
$(P_1 \sqcap \ldots \sqcap P_m)^I = P_1^I \cap \ldots \cap P_m^I$

An interpretation $I$ is a model of $\mathcal{T}$ if $C^I \subseteq D^I$ for every inclusion $C \sqsubseteq D$ in $\mathcal{T}$.

**Example 4:** In Example 3 our terminology would contain the following two inclusion statements that define precisely the concept named $C_{tgd}$.

$C_{tgd} \sqsubseteq \forall Prev.\neg(Basic \sqcap HumCourse)$
$\forall Prev.\neg(Basic \sqcap HumCourse) \sqsubseteq C_{tgd}$

Traditionally, terminologies are given as part of the knowledge base. In our case we will automatically construct part of the terminology in our algorithm for translating separable tgd constraints to extended Horn rules.

We consider hybrid knowledge bases that contain a terminology and a set of extended Horn rules [Levy and Rousset, 1996a]. An extended Horn rule can contain in its antecedent unary and binary predicates which are concepts and roles defined in the terminology. An interpretation $I$ is a model of $\Delta$ if it is a model of both the terminology and the extended Horn rules. Algorithms for reasoning in this language are described in [Levy and Rousset, 1996a; Levy and Rousset, 1996b].

## Translating separable tgd's

Informally, the class of separable tgd's can be translated into a conjunction of concepts in $\mathcal{ALCNR}$. Formally, let $T$ be a tgd of the form $\psi \Rightarrow \phi$. Given such a conjunction $\phi$, we can define a graph $g_\phi$ as follows. The nodes in the graph are the variables of $\phi$ and there is an arc from a variable $X$ to a variable $Y$ if there is an atom of the form $R(X,Y)$, where $R$ is a binary predicate. A *maximal path* in $g_\phi$ is a path $X_1, \ldots, X_n$, such that there is no arc emanating from $X_n$ and no arcs coming into $X_1$. A prefix $p_1$ of a path $p$ is a subpath of $p$ that has the same initial point.

**Definition 3:** *Let $T$ be a tgd of the form $\psi \Rightarrow \phi$. $T$ is a separable tgd if:*

1. *$\phi$ involves only unary and binary predicates,*
2. *$g_\phi$ is acyclic,*
3. *a variable that appears in $\psi$ can only appear in the beginning of a maximal path in $g_\phi$, and*
4. *if two maximal paths share a variable $X$, then $X$ appears only in their common prefix path.*

The algorithm shown in Figure 1 creates extended Horn rules and an $\mathcal{ALCNR}$ terminology that are equivalent to a separable tgd. If the given tgd describes input constraints, then the predicate in the consequent of the rules will be $P_{in}$, and otherwise, it will be $P_{out}$.

**procedure tgd-to-horn**$(T, P)$
/* $T$ is a separable tgd of the form $\psi \Rightarrow \phi$. */
/* $P$ is either $P_{in}$ or $P_{out}$ */

**for** every variable $X \in \phi$ define a concept $C_X$ as follows:
  Let $C_1, \ldots, C_l$ be the literals appearing in unary atoms in $\phi$ containing $X$.
  **if** $X$ appears only in the end of maximal paths **then**
    $C_X = C_1 \sqcap \ldots \sqcap C_l$ (or $\top$ if $l = 0$).
  **else**
  Let $Y_1, \ldots, Y_k$ be the variables in $\{Y \mid R(X,Y) \in \phi\}$.
  **for** every $Y \in \{Y_1, \ldots, Y_k\}$ **do:**
    Let $Role_{X,Y}$ be the conjunction of the roles in the set $\{R \mid R(X,Y) \in \phi\}$.
  $C_X = (\exists\ Role_{X,Y_1}.C_{Y_1}) \sqcap \ldots \sqcap (\exists\ Role_{X,Y_k}.C_{Y_k}) \sqcap C_1 \sqcap \ldots \sqcap C_l$.

**return** the terminology $D_i \sqsubseteq \neg C_{X_i}$, $\neg C_{X_i} \sqsubseteq D_i$,
  and the rules $\psi \wedge D_i(X_i) \Rightarrow P$,
for $i = 1, \ldots, n$, where $X_1, \ldots, X_n$ are the variables that appear in the beginning of maximal paths in $\phi$.
**end tgd-to-horn**.

Figure 1: Algorithm for translating tgd constraints to extended Horn rules with a terminology.

**Example 5:** Considering our example tgd

$$EngStud(s) \wedge Want(s,c) \wedge Adv(c) \wedge HumCourse(c) \Rightarrow (\exists c_1) Prev(s,c_1) \wedge Basic(c_1) \wedge HumCourse(c_1).$$

The right hand side of the tgd contains one maximal path $s \rightarrow c_1$. The algorithm will compute $C_{c_1} = Basic \sqcap HumCourse$. The concept for $s$ is $C_s = \exists\ Prev.(Basic \sqcap HumCourse)$. Procedure **tgd-to-horn** will return the terminology

$$D_1 \sqsubseteq \neg\exists\ Prev.(Basic \sqcap HumCourse)$$
$$\neg\exists\ Prev.(Basic \sqcap HumCourse) \sqsubseteq D_1,$$

and the rule

$$EngStud(s) \wedge Want(s,c) \wedge Adv(c) \wedge HumCourse(c) \wedge D_1(s) \Rightarrow P_{in}.$$

The following theorem shows that the terminology and the extended Horn rules returned by our algorithm are equivalent for the purpose of verification.

**Theorem 4:** *Let $\Delta$ be a hybrid knowledge base, and let $T$ be a separable tgd. Suppose that $\Delta_1$ is the set of extended Horn rules and terminology returned by procedure* **tgd-to-horn***$(T, P)$. Then, for any set of inputs $G$, $\Delta \cup G \models \neg T$ if and only if $\Delta \cup \Delta_1 \cup G \models P$.*

## Verifying hybrid knowledge bases

We now describe an algorithm that checks whether a hybrid knowledge base is verified. As described in the previous section, one of the contributions of this algorithm is that we obtain a method for verifying knowledge bases when the input and output constraints are described using separable tgd's.

Given a knowledge base $\Delta$ that contains a terminology $\Delta_{\mathcal{T}}$, a set of extended Horn rules and the predicates $P_{in}$ and $P_{out}$ the algorithm considers each rule $r \in Rules(P_{out})$. For each rule $r$, we consider the knowledge base $\Delta_r$ that is formed as follows:

- $\Delta_r$ includes $\Delta_{\mathcal{T}}$ and $Rules(P_{in})$, and
- $\Delta_r$ includes each of the conjuncts in the antecedent of $r$ as a ground fact, where the variables in the conjunct are viewed as constants.

For each rule, we check whether $\Delta_r \models P_{in}$. The entailment check is done using the *existential entailment* algorithm described in [Levy and Rousset, 1996a], which is guaranteed to be sound and complete. Note that simple application of Horn rule reasoning techniques (e.g., SLD resolution) is not complete for hybrid knowledge bases.

The algorithm returns that the knowledge base $\Delta$ is verified if and only if $\Delta_r \models P_{in}$ for every rule $r \in Rules(P_{out})$. If there is some $\Delta_r$ for which $\Delta_r \not\models P_{in}$, then the ground facts in $\Delta_r$ provide a counterexample for the verification of the KB. That is, they provide an example input that satisfies the input constraints, but does not satisfy the output constraints.

The correctness of our algorithm is established by the following theorem:

**Theorem 5:** *Let $\Delta$ be a knowledge base with an $\mathcal{ALCNR}$ terminology $\Delta_{\mathcal{T}}$. Assume that the Horn rules do not have the interpreted predicates $\leq, <, =$ and $\neq$. Let $P_{in}$ and $P_{out}$ be predicates in $\Delta$ describing correct inputs and outputs, respectively.*

*If both $Rules(P_{in})$ and $Rules(P_{out})$ are not recursive then the verification problem is decidable in time that is doubly exponential in the size of the rules in $Rules(P_{in})$ and $Rules(P_{out})$ and the size of $\Delta_{\mathcal{T}}$, and polynomial in the number of rules in $Rules(P_{in})$ and $Rules(P_{out})$.*

This verification method generalizes the one proposed in [Rousset, 1994]. In that work, hybrid knowledge bases were also considered, but the rule and terminology components were considered in isolation, and therefore the algorithm was not guaranteed to be complete.

## Conclusions

We described several new results concerning the verification problem for hybrid knowledge bases combining logical Horn rules and class hierarchies in a KL-ONE style terminology. We gained insight into the verification problem by showing that it is closely related to the problem of query containment. In particular we established the first complexity results for the verification problem of non recursive Horn rules, and we have shown the exact points at which the problem becomes undecidable when the rules are recursive. We have also presented the first complete algorithm for verifying hybrid knowledge bases. Finally, we have shown that with hybrid knowledge bases we are able to handle verification problems in which the output and input constraints are expressed via the class of separable tuple generating dependencies. Such dependencies provide more expressive power than constraints that have been declaratively specified in previous work on verification.

It should be noted that our current work considers only rules whose semantics are given within first order logic. Several works have considered the verification of OPS5-style production rules (e.g, [Schmolze and Snyder, 1995], [Ginsberg and Williamson, 1993]). In such rules, the right hand side of the rules is an *action* that may delete facts from the knowledge base. Verification of non-recursive logical rule knowledge bases has originally been considered by Rousset [Rousset, 1988] and Ginsberg [Ginsberg, 1988], and has been extended to handle interpreted constraints ( [Loiseau and Rousset, 1993] [Williamson and Dahl, 1993]).

The algorithms we presented are designed to answer the question of whether the knowledge base is verified or not. However, when the knowledge base is *not* verified, it is important to tell the user the possible causes of the problem, and to suggest corrective actions. The algorithms we described can be easily modified to return a *counter example* set of inputs in cases in which the knowledge base is not verified. Finally, we are considering extending trace-based debugging methods (as described in [Rousset and Hors, 1996] for terminological knowledge bases) to hybrid knowledge bases.

## References


Baader, F. and Hollunder, B. 1991. A terminological knowledge representation system with complete inference algorithm. In *In Proceedings of the Workshop on Processing Declarative Knowledge, PDK-91, Lecture Notes in Artificial Intelligence*. Springer-Verlag. 67–86.

Beeri, Catriel and Vardi, Moshe 1984. A proof procedure for data dependencies. *Journal of the ACM* 31(4):718–741.

Brachman, Ronald J. and Schmolze, J. G. 1985. An overview of the KL-ONE knowledge representation system. *Cognitive Science* 9(2):171–216.

Buchheit, Martin; Donini, Francesco M.; and Schaerf, Andrea 1993. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research* 1:109–138.

Chaudhuri, Surajit and Vardi, Moshe 1992. On the equivalence of recursive and nonrecursive datalog programs. In *Proceedings of PODS-92*. 55–66.

Fagin, R. 1982. Horn clauses and database dependencies. *Journal of the ACM* 29(4):952–983.

Ginsberg, Allen and Williamson, Keith 1993. Inconsistency and redundancy checking for quasi-first-order-logic knowledge bases. *International Journal of Expert Systems: Research and Applications* 6.

Ginsberg, Allen 1988. Knowledge base reduction: A new approach to checking knowledge bases for inconsistency and redundancy. In *Proceedings of AAAI-88*.

Gurevich, Y. and Lewis, H. R. 1982. The inference problem for template dependencies. In *Proceedings of PODS-82*. 221–229.

Klug, A. 1988. On conjunctive queries containing inequalities. *Journal of the ACM* 35(1): 146–160.

Levy, Alon Y. and Rousset, Marie-Christine 1996a. CARIN: a representation language integrating rules and description logics. In *Proceedings of ECAI-96*.

Levy, Alon Y. and Rousset, Marie-Christine 1996b. The limits on combining recursive horn rules and description logics. In *Proceedings of AAAI-96*.

Levy, Alon Y. and Sagiv, Yehoshua 1995. Semantic query optimization in datalog programs. In *Proceedings PODS-95*.

Loiseau, Stephane and Rousset, Marie-Christine 1993. Formal verification of knowledge bases focused on consistency: Two experiments based on ATMS techniques. *International Journal of Expert Systems: Research and Applications* 6.

Rousset, Marie-Christine and Hors, Pascale 1996. Modeling and verifying complex objects: A declarative approach based on description logics. In *Proceedings of ECAI-96*.

Rousset, Marie-Christine 1988. On the consistency of knowledge bases: the COVADIS system. In *Proceedings ECAI-88*.

Rousset, Marie-Christine 1994. Knowledge formal specifications for formal verification: a proposal based on the integration of different logical formalisms. In *Proceedings of ECAI-94*.

Sagiv, Y. and Yannakakis, M. 1981. Equivalence among relational expressions with the union and difference operators. In *J. ACM 27:4 pp. 633-655*.

Schmolze, James G. and Snyder, Wayne 1995. A tool for testing confluence of production rules. In *Proceedings of the European Symposium on Validation and Verification of KBS, EUROVAV-95*.

Shmueli, Oded 1987. Decidability and expressiveness aspects of logic queries. In *Proceedings of the 6th ACM Symposium on Principles of Database Systems*. 237–249.

Ullman, Jeffrey D. 1989. *Principles of Database and Knowledge-base Systems, Volumes I, II.* Computer Science Press, Rockville MD.

van der Meyden, Ron 1992. *The Complexity of Querying Indefinite Information: Defined Relations Recursion and Linear Order*. Ph.D. Dissertation, Rutgers University, New Brunswick, New Jersey.

Vardi, Moshe 1984. The implication and finite implication problems for typed template dependencies. *Journal of Computer and System Sciences* 28(1):3–28.

Williamson, Keith and Dahl, Mark 1993. Knowledge base reduction for verifying rule bases containing equations. In *Proceedings of the AAAI-93 workshop on Validation and Verification of KBS*.

Yannakakis, M. and Papadimitriou, C. H. 1980. Algebric dependencies. *Journal of Computer and System Sciences* 25(1):2–41.