Inconsistency-tolerant Reasoning with Classical Logic and Large Databases

Timothy L. Hinrichs
University of Chicago

Jui-Yi Kao*
Stanford University

Michael R. Genesereth
Stanford University

Abstract

Real-world automated reasoning systems must contend with inconsistencies and the vast amount of information stored in relational databases. In this paper, we introduce compilation techniques for inconsistency-tolerant reasoning over the combination of classical logic and a relational database. Our resolution-based algorithms address a quantifier-free, function-free fragment of first-order logic while leveraging off-the-shelf database technology for all data-intensive computation.

Introduction

Automated reasoning systems face two major challenges in real-world reasoning: logically inconsistent information and the vast amount of information stored in relational databases.

These two issues stand out because of their prevalence in the world today. Occasional errors and disagreements are unavoidable in real-world information. A customer may report an incorrect social security number; a technician may record an incorrect parameter; two scholars may disagree over the birth year of Julius Caeser; two organizations may disagree over the gross domestic product of a country. It is natural for errors and disagreements in our information to cause contradictions. Classical reasoning systems unable to cope with contradictions cannot differentiate one query from another, making them useless when inconsistencies arise. Orthogonally, relational databases are ubiquitous in our society. Companies, universities, and governments rely crucially on database technology to store and publish vast amounts of information. Automated reasoning systems unable to integrate relational databases cannot exploit that information and consequently fail to find many desirable conclusions.

Building a reasoning system that copes with inconsistencies is difficult because classical semantics is explosive (entails every sentence from an inconsistent premise set). Weakening this definition to one that captures our intuition as to which conclusions one can reasonably draw from unreasonable premises has been the subject of much study, and as of yet no single definition has become the universal standard. Relational databases compound this problem by the sheer quantity of information that they routinely store. Automated reasoning with premise sets that include gigabytes, terabytes, or even petabytes of relational data introduces memory and disk management problems that have not been adequately addressed by the automated reasoning community.

Recently, progress has been made toward semantics that cope with logical inconsistency, especially in the context of classical logic, *e.g.*, (Subrahmanian and Amgoud 2007; Everaere, Konieczny, and Marquis 2008; Besnard and Hunter 2008). Of the relatively few results concerned with implementation (da Costa et al. 1990; Sofronie-Stokkermans 2000; Besnard and Hunter 2006; Efstathiou and Hunter 2008; Binas and McIlraith 2008; Gomez, Chesnevar, and Simari 2008), none address the problems of massive data sets or interfacing with external databases. Standard techniques such as procedural attachments (Weyhrauch 1980; Myers 1990; Sikka 1996) and theory resolution (Stickel 1985) do not address inconsistency and only provide the infrastructure by which an automated reasoning system can pose queries to a relational database. When the answers to such queries are gigabytes large, the unaddressed memory and disk management issues become crucial for operational automated reasoning systems.

Our approach simultaneously addresses logical inconsistency and the data management problems caused by combining classical reasoning with database evaluation using novel, resolution-based compilation algorithms. Instead of building an inconsistency-tolerant automated reasoning system that operates on classical axioms while periodically consulting a database, our system compiles the classical axioms into database queries, invokes the database to answer those queries, and returns exactly the database's results. An important advantage of this approach is that the compilation layer deals directly with the classical premises but not the massive amount of data in the database. The key challenge of this approach is constructing database queries that encapsulate the classical premise set and implement an inconsistency-tolerant entailment relation. The compilation algorithms introduced here meet this challenge for premise sets in a finite version of function-free, quantifier-free first-order logic and terminate exactly when the resolution clo-

*This work was supported in part by the National Science Foundation under award number IIS-0841152.

sure of the premises is finite.

After formalizing the problem, we explain the compilation algorithms for one non-explosive entailment relation and soundness and completeness results for quantifier-free premise sets. Then we discuss another non-explosive entailment relation, which is popular in the literature, and the extent to which our algorithms implement it. Finally, we cite related work, and conclude with a discussion of future work.

Problem Statement

The inconsistency-tolerant reasoning system we designed for combinations of classical logic and relational databases first compiles the classical axioms into database queries, then invokes the database to answer those queries, and finally returns exactly the database's results. Since the second two steps utilize off-the-shelf database technology, the problem we address in this paper is the compilation of classical axioms into database queries that implement an inconsistency-tolerant entailment relation. Before formalizing this problem, we discuss languages for classical logic and relational databases, the semantics for combinations of classical logic and relational databases, and inconsistency-tolerant entailment relations.

Two Languages: FHL and DATALOG

The objects of study in this paper are two knowledge representation formalisms: classical logic and relational databases. The classical logic considered is first-order logic with equality and the following three restrictions: no function constants, a domain closure assumption (DCA), and a unique names assumption (UNA). We call this logic Finite Herbrand Logic (FHL) because the only models of interest are the finite Herbrand models. The relational database language studied in this paper is nonrecursive DATALOG with negation, which can be embedded in SQL and therefore used in off-the-shelf database systems. We use standard syntax and semantics for both languages and therefore only highlight the crucial definitions below.

The syntax for FHL is the same as for function-free first-order logic. A model for FHL is the same as for first-order logic but simplified to take advantage of the UNA and DCA. A model is a set of ground atoms from the language. A model satisfies a set of FHL sentences if it satisfies all ground instances of those sentences. A sentence set is satisfiable (or consistent) if there is some model that satisfies it. A sentence set is complete if it is satisfied by at most one model. $Mod[\Gamma]$ denotes the set of models that satisfy Γ.

For DATALOG, the syntax is also standard and can be identified by the use of $:-$ for implication. Just as for FHL, a model is a set of ground atoms. Unlike FHL, every set of DATALOG sentences has exactly one model and thus represents a complete, consistent theory. Without negation, this model is the smallest one (ordered by subset) that satisfies the sentences under the FHL definition of satisfaction. With negation, this model is assigned according to the stratified semantics (Ullman 1989).

Since this paper introduces compilation algorithms for FHL and DATALOG, it is convenient to have conventions for discussing the canonical FHL and DATALOG representations of complete, consistent theories (every theory axiomatizable in DATALOG). A complete, consistent theory is said to be represented *explicitly* in FHL by a logically equivalent set of ground literals, denoted $Exp[T]$ for theory T. A complete, consistent theory is said to be represented *extensionally* in DATALOG by the non-negated elements of its explicit representation, denoted $Ext[T]$. To denote the set of predicates Q that occur in the explicit representation of a complete, consistent theory T, we will say that T is *complete over predicates* Q.

For example, the following sentences induce a consistent FHL theory complete over predicates $\{p, q\}$.

$$\neg p(a) \wedge \neg p(b)$$
$$\forall x.(\neg p(x) \Rightarrow q(x, a))$$
$$\forall x.\neg q(x, b)$$

Below are the explicit and extensional representations.

Explicit		Extensional
$\neg p(a)$	$\neg p(b)$	$q(a, a)$
$q(a, a)$	$\neg q(a, b)$	$q(b, a)$
$q(b, a)$	$\neg q(b, b)$	

Combinations of FHL and DATALOG

The algorithms introduced in this paper enable reasoning about the combination of FHL and relational databases; thus, we must define the semantics of such combinations. Since every database can be represented in DATALOG and every DATALOG theory is equivalent to a complete, consistent FHL theory, we can formalize a database as a complete, consistent FHL theory (hence our interest in FHL). The combination of FHL axioms Δ and database Λ (a complete, consistent FHL theory) will be written as $\langle \Delta, \Lambda \rangle$. Its semantics is the set of models that satisfy $\Delta \cup \Lambda$.

While the combination of FHL and databases is an important concept, it turns out that the compilation algorithms we introduce in this paper operate on the combination of FHL axioms and a database *schema*. To understand why, first notice that it is natural for the compiler to be built upon automated reasoning technology because it must manipulate sentences in classical logic. A compiler capable of examining the contents of the database Λ (as opposed to just its schema) would require automated reasoning technology capable of managing vast amounts of data—the very problem our compilation work was meant to address. Second, if the output of the compiler depends upon the database contents, then as the database contents change (often a frequent occurrence), the compilation process must be started anew. Because our compilation algorithms operate on FHL axioms and a database schema, they are both more practical to execute and need to be executed less frequently.

Definition 1 (Parameterized Theory). *A parameterized theory $\langle \Delta, Q \rangle$ is an FHL sentence set Δ and a predicate set Q. An instance of the parameterized theory $\langle \Delta, Q \rangle$ is a two-tuple $\langle \Delta, \Lambda \rangle$ where Λ is a complete, consistent FHL theory where all occurring predicates belong to Q.*

A model satisfies a parameterized theory instance $\langle \Delta, \Lambda \rangle$ if it satisfies $\Delta \cup \Lambda$.

Inconsistency

For both FHL and DATALOG, logical entailment is defined as usual: Γ entails ϕ (denoted $\Gamma \models \phi$ for FHL and $\Gamma \models_D \phi$ for DATALOG) if every model that satisfies Γ also satisfies ϕ. Since inconsistent theories trivially entail every sentence, traditional entailment fails to tolerate inconsistencies; consequently, we focus on a non-explosive notion of entailment specialized for parameterized theories and similar to those found in defeasible logic programming (*e.g.*, (Gomez, Chesnevar, and Simari 2008)).

For a parameterized theory instance $\langle \Delta, \Lambda \rangle$, *strict existential entailment* states that $\Lambda \models_E^\Delta \phi$ if there is a subset Λ_0 of Λ such that $\Delta \cup \Lambda_0$ is satisfiable and entails ϕ. Intuitively, a conclusion is strictly existentially entailed when there is some portion of the database consistent with all the FHL axioms that entails the conclusion. As compared to the more popular existential entailment, which allows all conclusions entailed by any satisfiable subset of premises, strict existential entailment requires the justification for every conclusion include the entire set of FHL axioms Δ. We call the set of all sentences strictly existentially entailed the *strict existential consequences* of a parameterized theory. For brevity, the word "existential" may be dropped.

In practice, strict existential entailment is useful when the FHL axioms correctly axiomatize some domain but the database contains erroneous information, *e.g.*, data acquisition errors, out-of-sync information, or genuine disagreements. Such axiom sets can be found, for example, in the description logic community, where ontologies are often verified or forced to be consistent.

Problem Statement

This work assumes that databases contain vast amounts of information, and the combination of a database and classical logic will often result in inconsistency. Our work addresses both issues simultaneously by developing compilation techniques that translate a parameterized theory $\langle \Delta, Q \rangle$ into a series of database queries that when evaluated over a database instance with schema Q answer a fixed set of strict entailment queries: $p(\bar{a})$ and $\neg p(\bar{a})$ for every predicate p and tuple of object constants $\bar{a}$. We call the compilation problem studied in this paper Parameterized Compilation to DATALOG.

Definition 2 (Parameterized Compilation to Datalog). *For a parameterized theory $\langle \Delta, Q \rangle$, the* DATALOG *sentence set Δ' is a parameterized compilation if for each predicate p in Δ, there are predicates p^- and p^+ occurring in Δ' such that for every consistent FHL theory Λ complete over Q*

$$Exp[\Lambda] \models_E^\Delta p(\bar{a}) \text{ iff } \Delta' \cup Ext[\Lambda] \models_D p^+(\bar{a})$$
$$Exp[\Lambda] \models_E^\Delta \neg p(\bar{a}) \text{ iff } \Delta' \cup Ext[\Lambda] \models_D p^-(\bar{a})$$

Compilation

Our approach to parameterized compilation relies on a standard automated reasoning technique: resolution. In particular, we will employ resolution to compute the resolution closure of the FHL axioms. This is the crucial step in the compilation process that bridges the gap between classical semantics and DATALOG semantics.

The utility of the resolution closure for compiling classical logic to DATALOG can be explained even without the presence of inconsistency. When the FHL axioms Δ are closed under resolution, answering an entailment query about $\Delta \cup \Lambda$ (for any Λ) is especially simple. To compute whether or not $\Delta \cup \Lambda$ entails $p(a)$, we must find a clause δ that belongs to Δ that together with Λ entails $p(a)$. In contrast, when Δ is not closed under resolution, one must find an entire *set* of statements belonging to Δ that together with Λ entail $p(a)$. The reduction of search from a set of statements to a singleton is especially useful when Λ is a set of literals (as it is for databases) because unit resolution applied to $\Delta \cup \Lambda$ is by itself sufficient to answer entailment queries.

These observations about the connections between binary resolution and unit resolution are important because DATALOG evaluation amounts to a form of unit resolution. Intuitively the problem of compiling classical logic to DATALOG is akin to the problem of making unit resolution a complete rule of inference.

DATALOG evaluation is not exactly unit resolution, however—it is an ordered form of unit resolution. To ensure completeness of DATALOG evaluation, we must construct from each clause in the resolution closure of Δ all of its contrapositives. Each of these contrapositives then becomes a different DATALOG query. This simple procedure (compute the resolution closure of Δ; compute the contrapositives of each clause in the result; translate each contrapositive to DATALOG syntax) turns out to be sound and complete for inconsistency-free theories $\langle \Delta, \Lambda$ when Δ is quantifier-free and Λ is represented explicitly.

For example, consider a simple quantifier-free theory closed under resolution, constituting Δ.

$$\begin{array}{l} p(x) \vee q(x) \\ \neg q(x) \vee \neg r(x) \\ p(x) \vee \neg r(x) \end{array} \quad (1)$$

Suppose we want to construct DATALOG queries so that for any database over predicates $Q = \{p, q, r\}$, we could compute whether or not $p(x)$, $q(x)$, and $r(x)$ are entailed (for any argument x). To do that, we consider all contrapositives of the clauses above, write each contrapositive in rule form, and change the head of each rule to break recursive paths. The result is a set of implications that can be interpreted as DATALOG rules (modulo safety).

$$\begin{array}{l} p^+(X) :- \neg q(X) \\ q^+(X) :- \neg p(X) \\ r^+(X) :- p(X) \\ p^+(X) :- r(X) \end{array} \quad (2)$$

For any database Λ complete for $\{p, q, r\}$, the rules above together with $Ext[\Lambda]$ entails (under DATALOG semantics) all those atomic sentences entailed by $\Delta \cup Exp[\Lambda]$ (under FHL semantics) as long as $\Delta \cup \Lambda$ is consistent.

The procedure outlined so far addresses one of the two motivations for this paper: reasoning about the combination of classical logic and large databases. The second motivation, inconsistency-tolerant reasoning, can be addressed

with a small change to the procedure above. In particular, to implement strict entailment one need only add a consistency check to the end of each of the constructed DATALOG queries. An algorithm for constructing such consistency checks in DATALOG will be discussed in the section that follows.

Algorithm 1 formalizes the procedure outlined above and performs parameterized compilation when the FHL axioms are quantifier-free. It is a one-line application of Algorithm 2, which in addition to a parameterized theory takes as input the set of sentences that must be included during the consistency check. For strict entailment, that set is always the FHL axiom set, but later we will discuss a variation of strict-entailment (the well-known existential entailment) that is a different one-line application of Algorithm 2. Note that we use RES[CNF[Δ]] to denote the resolution and factoring closure of the clausal form of Δ and PREDS[ϕ] to denote the set of predicates occurring in ϕ.

Algorithm 1 STRICT-COMPILATION[$\langle \Delta, Q \rangle$]

1: COMPILATION[$\langle$RES[CNF[Δ]], $Q\rangle$, Δ]

Algorithm 2 COMPILATION[$\langle \Delta, Q \rangle$, Σ]

Assumes: Δ and Σ are clause sets.
Assumes: Δ is closed under resolution and factoring.
Outputs: A DATALOG theory

1: **for all** predicates $p \in Q \cup \text{PREDS}[\Delta] - \{=\}$ **do**
2: **for all** clauses d in $\Delta \cup \{p(\bar{x}) \vee \neg p(\bar{x})\}$ including a literal $p(t)$ **do**
3: write d as $p(\bar{x}) \Leftarrow \phi(\bar{x})$
4: **when** PREDS[$\phi(\bar{x})$] $\not\subseteq Q$] **then goto next** d
5: Γ **:=** $\{d\} \cup \Sigma$
6: **OUTPUT-CONSISTENCY-CHECK[**$\phi(\bar{x})$**,**Γ**]**
7: **print** **MAKE-SAFE[**$p^+(\bar{x}) :- \phi(\bar{x}) \wedge consistent^{\Gamma}_{\phi}(\bar{x})$**]**
8: **end for**
9: Likewise for p^-.
10: **end for**

In Algorithm 2, MAKE-SAFE forces its argument to satisfy the DATALOG syntax requirements, and OUTPUT-CONSISTENCY-CHECK outputs an axiomatization of the consistency check inserted into each DATALOG rule in line (6): $consistent^{\Gamma}_{\phi}(\bar{x})$. The next section discusses how OUTPUT-CONSISTENCY-CHECK axiomatizes $consistent^{\Gamma}_{\phi}(\bar{x})$ in more detail, but for the moment it suffices to describe its semantics.

Definition 3 ($consistent^{\Theta}_{\phi}$). *Suppose Θ is an FHL sentence set. For an FHL sentence $\phi(\bar{x})$, $consistent^{\Theta}_{\phi}(\bar{x})$ is completely and consistently defined to be true for exactly those $\bar{a}$ such that $\phi(\bar{a})$ and Θ are consistent. C_{con} denotes the complete theory defining $consistent^{\Theta}_{\phi}$ for all $\phi(\bar{x})$.*

The next theorems guarantee the soundness and completeness of our core compilation algorithm (Algorithm 2) under a very strong assumption: that the FHL axioms are closed under resolution and factoring. Proofs have been omitted for lack of space but can be found in the extended version of this paper (Hinrichs, Kao, and Genesereth 2009).

Theorem 1 (Soundness of COMPILATION). *Consider a parameterized theory $\langle \Delta, Q \rangle$, where Δ is a clause set closed under resolution and factoring, and Σ is a satisfiable set of FHL sentences. Use Δ' to denote the output of* COMPILATION*($\langle \Delta, Q \rangle$,$\Sigma$). Suppose Λ is a consistent theory complete over the predicates Q. If*

$$\Delta' \cup Ext[\Lambda] \cup Ext[C_{con}] \models_D p^+(\bar{a})$$

then there is a satisfiable subset of $\Delta \cup Exp[\Lambda]$ that is consistent with Σ and entails $p(\bar{a})$.

Theorem 2 (Completeness of COMPILATION). *Consider a parameterized FHL theory $\langle \Delta, Q \rangle$, where Δ is a clause set closed under resolution and factoring, and Σ is a satisfiable set of FHL sentences. Use Δ' to denote the output of* COMPILATION*($\langle \Delta, Q \rangle$,$\Sigma$). Suppose Λ is a consistent theory complete over the predicates Q. If there is a satisfiable subset of $\Delta \cup Exp[\Lambda]$ consistent with Σ that entails $p(\bar{a})$ then*

$$\Delta' \cup Ext[\Lambda] \cup Ext[C_{con}] \models_D p^+(\bar{a}).$$

The corollaries below guarantee soundness and completeness for our top-level compilation algorithm (Algorithm 1) and remove the restriction that the FHL clauses are closed under resolution. Those corollaries rely on the following proposition, which states that for strict entailment, arbitrary equivalence-preserving operations can be applied to the FHL axioms of a parameterized theory without changing the strict existential consequences.

Proposition 1. *Suppose f is an operation such that for all premise sets Δ in its domain, $Mod[\Delta] = Mod[f[\Delta]]$. For every strict entailment query,*

$$\Lambda \models^{\Delta}_{E} \phi \text{ if and only if } \Lambda \models^{f[\Delta]}_{E} \phi.$$

Proof. Suppose $\Lambda \models^{\Delta}_{E} \phi$. Then there is a $\Lambda_0 \subseteq \Lambda$ such that $\Lambda_0 \cup \Delta$ is satisfiable and entails ϕ. Because $Mod[\Delta] = Mod[f[\Delta]]$, $\Lambda_0 \cup f[\Delta]$ must be satisfiable and entail ϕ by the usual definition of satisfaction and entailment. Thus, Λ_0 witnesses the strict entailment $\Lambda \models^{f[\Delta]}_{E} \phi$. The other direction follows similarly. □

To prove the soundness and completeness of STRICT-COMPILATION, we leverage the fact that the conversion to clausal form (denoted CNF) and the resolution and factoring closure of a clause set (denoted RES) are well-known to be model-preserving when the premises are quantifier-free.

Corollary 1 (Soundness of STRICT-COMPILATION). *Suppose Δ' denotes the output of* STRICT-COMPILATION*($\langle \Delta, Q \rangle$), where Δ is quantifier-free. Further suppose Λ is a consistent theory, complete over the predicates Q. If*

$$\Delta' \cup Ext[\Lambda] \cup Ext[C_{con}] \models_D p^+(\bar{a})$$

then $Exp[\Lambda] \models^{\Delta}_{E} p(\bar{a})$.

Proof. Suppose $\Delta' \cup Ext[\Lambda] \cup Ext[C_{con}]) \models_D p^+(\bar{a})$. By the definition of STRICT-COMPILATION and Theorem 1, we see that there is a satisfiable subset of $Exp[\Lambda]$ that is consistent with RES[CNF[Δ]] and entails $p(\bar{a})$. Consequently, $Exp[\Lambda] \models_E^{\text{RES[CNF[}\Delta\text{]]}} p(\bar{a})$. Since Δ is quantifier-free, RES[CNF[Δ]] has exactly the same models as Δ. By Proposition 1, $Exp[\Lambda] \models_E^{\Delta} p(\bar{a})$. □

Corollary 2 (Completeness of STRICT-COMPILATION). *Suppose Δ' denotes the output of* STRICT-COMPILATION*($\langle \Delta, Q \rangle$), where Δ is quantifier-free. Further suppose Λ is a consistent theory complete over the predicates Q. If $Exp[\Lambda] \models_E^{\Delta} p(\bar{a})$ then*

$$\Delta' \cup Ext[\Lambda] \cup Ext[C_{con}] \models_D p^+(\bar{a}).$$

Proof. Suppose $Exp[\Lambda] \models_E^{\Delta} p(\bar{a})$. Then because Δ is quantifier-free, Δ has the same models as RES[CNF[Δ]], and by Proposition 1, $Exp[\Lambda] \models_E^{\text{RES[CNF[}\Delta\text{]]}} p(\bar{a})$. Theorem 2 can then be applied (where Σ is RES[CNF[Δ]]) to produce the desired result:

$$\Delta' \cup Ext[\Lambda] \cup Ext[C_{con}] \models_D p^+(\bar{a}).$$

□

Axiomatizing the Consistency Check

Here we introduce an algorithm for axiomatizing the consistency check of Definition 3 in DATALOG. It operates on a conjunction of literals $\phi(\bar{x})$ and a sentence set Θ. $consistent_{\phi}^{\Theta}(\bar{x})$ must be true for exactly those a where $\phi(\bar{a})$ is consistent with Θ. Intuitively, this means that the variable bindings are not made in such a way that a subset of the literals in ϕ unify with the negated literals of some clause in Θ. The algorithm that follows implements this intuition using subsumption.

Algorithm 3 OUTPUT-CONSISTENCY-CHECK(ϕ,Θ)

Assumes: ϕ is a conjunction of literals with variables $\bar{x}$.
Assumes: Θ is a clause set.
Outputs: A DATALOG axiomatization of $consistent_{\phi}^{\Theta}$.

1: Θ := RES[Θ]
2: $\bar{\phi}$:= $\neg\phi$ in clausal form // a single clause
3: T := $\{\sigma | \sigma$ is a maximally general substitution such that $\phi\sigma$ is inconsistent$\}$
4: Σ := $\{\sigma | \sigma$ is a maximally general substitution such that some $d \in \Theta$ subsumes $\bar{\phi}\sigma\}$
5: **print** MAKE-SAFE$\Big[consistent_{\phi}^{\Theta}(\bar{x}) :- \neg \bigvee_{\sigma \in \Sigma \cup T} \bigwedge_{t \leftarrow u \in \sigma} t = u \Big]$

For illustration, here is an example of the input and output of OUTPUT-CONSISTENCY-CHECK.

Input: $\phi : \neg p(x,y) \wedge q(y,a)$,
$\Theta : \{p(x,b) \vee r(x,b), \neg r(a,y) \vee \neg q(y,z)\}$
RES[Θ] = $\{p(x,b) \vee r(x,b)$,
$\neg r(a,y) \vee \neg q(y,z)$,
$p(a,b) \vee \neg q(b,z)\}$.

No substitution into the variables of ϕ causes a contradiction on its own, so T is empty. The only substitution into the variables of ϕ that contradicts Θ is $\sigma = \{a \rightarrow x, b \rightarrow y\}$. $\phi\sigma = \neg p(a,b) \wedge q(b,a)$, contradicting $p(a,b) \vee \neg q(b,z)$.

Output: $consistent_{\phi}^{\Theta}(x,y) :- universe(x,y) \wedge x \neq a$
$consistent_{\phi}^{\Theta}(x,y) :- universe(x,y) \wedge y \neq b$

In practice, the rule bodies produced by OUTPUT-CONSISTENCY-CHECK can be inlined in the DATALOG queries produced by COMPILATION for the sake of efficiency. In a bottom-up evaluation of the resulting DATALOG program, inlining the consistency checks avoid the need to first build these consistency relations.

Theorem 3 (Consistency Axiomatization). *Given an FHL clause set Θ closed under resolution and factoring and an FHL conjunction of literals $\phi(\bar{x})$ with variables $\bar{x}$, Algorithm 3 produces a* DATALOG *definition of $consistent_{\phi}^{\Theta}(\bar{x})$ such that $consistent_{\phi}^{\Theta}(\bar{a})$ evaluates to true if and only if $\phi(\bar{a})$ and Θ are consistent.*

In Algorithm 3, it's not immediately clear how to compute the set Σ, or even that the set is finite. It turns out that it is sufficient for computing Σ to consider each $d \in \Theta$ and for each one run a slightly modified subsumption algorithm, e.g. Stillman's (Stillman 1973). Similarly, the set T is easy to compute: whenever a predicate occurs both positively and negatively in ϕ, add to T the most general unifier of the corresponding atoms.

Non-strict Existential Entailment

Recall that for a parameterized theory instance $\langle \Delta, \Lambda \rangle$ to strictly entail a conclusion $p(\bar{a})$, there must be a portion of Λ consistent with Δ that entails $p(\bar{a})$. In particular, this means that all the premises used to prove $p(\bar{a})$ must be consistent with Δ in its entirety. When Δ correctly axiomatizes the domain of interest, strict entailment produces intuitive results, but when Δ includes mistakes, strict entailment may produce fewer results than one would like. For example, if Δ is inconsistent, there are no strict consequences for any database.

One entailment relation, popular in the literature, that is tolerant of inconsistencies both within Δ as well as across Δ and Λ is existential entailment. Γ existentially entails ϕ if there is a consistent subset of Γ that entails ϕ. Likewise, for a parameterized theory, $\Delta \cup \Lambda$ existentially entails ϕ if there is a consistent subset of $\Delta \cup \Lambda$ that entails ϕ. Thus, unlike strict entailment, existential entailment does not allow an inconsistency in Δ to block all conclusions. Existential entailment is therefore stronger (produces more consequences) than strict entailment but is weaker (produces fewer consequences) than traditional entailment.

Algorithm 4, a simple application of our core compilation algorithm (Algorithm 2), performs parameterized theory compilation while preserving the existential consequences (as opposed to the strict consequences) of a parameterized theory. Soundness and completeness hold when the FHL axioms are clauses closed under resolution and factoring.

Corollary 3 (Soundness of EXISTENTIAL COMPILATION). *Suppose Δ' denotes the output of* EXISTENTIAL-

Algorithm 4 EXISTENTIAL-COMPILATION$[\langle \Delta, Q \rangle]$

1: COMPILATION$[\langle \Delta, Q \rangle, \emptyset]$

COMPILATION*$(\langle \Delta, Q \rangle)$, where Δ is a clause set closed under resolution and factoring. Further suppose Λ is a consistent theory, complete over the predicates Q. If*

$$\Delta' \cup Ext[\Lambda] \cup Ext[C_{con}] \models_D p^+(\bar{a})$$

then $\Delta \cup Exp[\Lambda] \models_E p(\bar{a})$.

Proof. Suppose $\Delta' \cup Ext[\Lambda] \cup Ext[C_{con}]) \models_D p^+(\bar{a})$. By the definition of EXISTENTIAL-COMPILATION, Theorem 1 applies because Δ is closed under resolution and factoring and Σ is the empty set. Thus we see that there is a satisfiable subset of $\Delta \cup Exp[\Lambda]$ that entails $p(\bar{a})$. Consequently, $\Delta \cup Exp[\Lambda] \models_E p(\bar{a})$. □

Corollary 4 (Completeness of EXISTENTIAL COMPILATION). *Suppose Δ' denotes the output of* EXISTENTIAL-COMPILATION*$(\langle \Delta, Q \rangle)$, where Δ is a clause set closed under resolution and factoring. Further suppose Λ is a consistent theory complete over the predicates Q. If $\Delta \cup Exp[\Lambda] \models_E p(\bar{a})$ then*

$$\Delta' \cup Ext[\Lambda] \cup Ext[C_{con}] \models_D p^+(\bar{a}).$$

Proof. Suppose $\Delta \cup Exp[\Lambda] \models_E p(\bar{a})$. Then applying Theorem 2 where Σ is the empty set, we immediately get the desired result. □

Negative Results

The important condition on the soundness and completeness results of the previous section (for non-strict existential entailment) is that the FHL axioms Δ must be closed under resolution. In contrast, the soundness and completeness results for strict entailment allow Δ to be any quantifier-free axiom set. It is tempting to forcibly close Δ under resolution, just as we did with strict entailment; however, it turns out that the resolution closure of an axiom set does not give the same existential consequences as the original. Here we illustrate this property, denoting the existential consequences of a sentence set Γ with $ECons[\Gamma]$.

First we show that neither clausal form conversion nor resolution preserve the existential consequences of a sentence set.

Proposition 2 (CNF Failure). *Suppose* CNF *denotes clausal form conversion. It is not necessarily the case that* $ECons[\Gamma] = ECons[\text{CNF}[\Gamma]]$.

Proof. The sentence $p \wedge \neg p$ has no existential consequences (except tautologies), but its clausal form has the existential consequences p and $\neg p$. □

Proposition 3 (Resolution and Subsumption Failures). *Suppose* RES *denotes the resolution closure and* SUBSUMP *denotes the subsumption deletion strategy. It is not necessarily the case that* $ECons[\Gamma] = ECons[\text{RES}[\Gamma]]$ *nor that* $ECons[\Gamma] = ECons[\text{SUBSUMP}[\Gamma]]$.

Proof. Consider the following four sentences.

$$p$$
$$\neg p \vee q \vee r$$
$$\neg p$$
$$p \vee \neg q \vee r$$

Perform resolution on the first two clauses to produce $q \vee r$. Perform resolution on the last two clauses to produce $\neg q \vee r$. Perform resolution on these two to produce r. Since r is included in the resolution closure, it is an existential consequence of the closure.

Notice though that no satisfiable subset of the original premises entail r; thus, the resolution closure does not preserve existential consequences. Also, notice that if subsumption is applied to the original premises, the result is two unit clauses: p and $\neg p$. Neither of these clauses entails $q \vee r$, which is an existential consequence of the original sentences; thus, subsumption does not preserve existential consequences. □

Resolution is not the only inference rule for which computing the closure fails to preserve existential consequences; rather, failure can occur when closing any sound set of inference rules.

Proposition 4 (Closure Failure). *Suppose R is a binary inference rule that is sound for existential entailment,* i.e., *for every two premises ϕ and ψ, $R[\phi, \psi] \subseteq ECons[\{\phi, \psi\}]$. Use $R^*[\Gamma]$ to denote the closure of Γ under R. It is not necessarily the case that $R^*[\Gamma] \subseteq ECons[\Gamma]$.*

Proof. The proof of Proposition 3 provides the counterexample. All three applications of resolution are sound (produce existential consequences of the sentences that were resolved together), but the last application produces a sentence not existentially entailed by the original premise set. □

Since soundness is insufficient to guarantee the preservation of existential consequences, it is as of yet unclear how to predict the effects of inference rules on the existential consequences of a premise set. Despite this, certain types of closures are known to have certain effects. If the closure only adds sentences to the premise set, like resolution, the existential consequences must monotonically increase, and if the closure only removes sentences, like subsumption, the consequences must monotonically decrease. Closures that apply both additive and subtractive inference rules, like resolution with subsumption, neither monotonically increase nor decrease the existential consequences.

A New Entailment Relation

The negative results of the last section demonstrate the difficulties of implementing existential entailment using our core compilation algorithm (Algorithm 2) for premise sets not closed under resolution. The obvious approach, computing the resolution closure before compiling (Algorithm 5, shown below), turns out not to preserve the existential consequences of the original theory; however, that algorithm

does address the two concerns of this paper: tolerating inconsistency and reasoning about the combination of classical logic and a relational database. Since there is currently no standard definition for drawing conclusions from inconsistent premise sets (non-explosively), the community is well-served when people identify easy-to-implement, non-explosive entailment relations. Algorithm 5 is easy to implement, and below we show that it implements a non-explosive entailment relation.

Algorithm 5 EXISTENTIALRES-COMPILATION$[\langle \Delta, Q \rangle]$

1: COMPILATION$[\langle$RES$[$CNF$[\Delta]], Q\rangle, \emptyset]$

Proposition 5. *The entailment relation $\models'$ is non-explosive when defined as follows.*

$$\Delta \cup \Lambda \models' \phi \text{ if } \text{RES}[\text{CNF}[\Delta]] \cup \Lambda \models_E \phi$$

Proof. Let Δ be the sentences $\{p \vee q, \neg p\}$ and Λ be $\{p\}$. Together these sentences are inconsistent. Since the sentences are already in clausal form, consider the resolution closure of Δ: $\{p \vee q, \neg p, q\}$. When combined with $\{p\}$, $\neg q$ is not an existential consequence; hence, $\models'$ is non-explosive. □

It turns out a very similar entailment relation was defined by Besnard and Hunter (Hunter 1998) for propositional logic and was called quasi-classical logic. Its proof theory converts a premise set to clausal form, computes the resolution closure, and entails exactly the nonempty disjunctions in that closure (where each nonempty disjunction may be augmented with arbitrary new literals via disjunction introduction). The entailment relation discussed above differs in that resolution is only applied to a portion of the premise set (Δ), and a conclusion is entailed only if it is existentially entailed by the database and the closure; thus, there is a consistency requirement imposed by $\models'$ not imposed by quasi-classical entailment. Consequently, the $\models'$ consequences are a strict subset of the quasi-classical consequences.

Related Work

The related work involves two issues studied in the literature: reasoning about inconsistency and knowledge compilation. Recently, much attention has been paid to inconsistency tolerance in the context of classical logic (Hunter 1998; Besnard and Hunter 2005; Konieczny, Lang, and Marquis 2005; Huang, van Harmelen, and ten Teije 2005; Zamansky and Avron 2006; Flouris et al. 2006; Subrahmanian and Amgoud 2007; Hunter and Konieczny 2008; Everaere, Konieczny, and Marquis 2008; Besnard and Hunter 2008). In contrast to some of that work, the problem addressed in this paper involves detecting but not repairing inconsistencies (Everaere, Konieczny, and Marquis 2008; Benferhat, Lagrue, and Rossit 2007; Subrahmanian and Amgoud 2007). Second, our work focuses on a classical logic that is properly neither propositional, *e.g.*, (Efstathiou and Hunter 2008), yet retains decidability nor first-order (Besnard and Hunter 2005) yet retains a relational syntax. Third, instead of establishing the relationships between all possible arguments with an argument tree, *e.g.*, (Efstathiou and Hunter 2008; Besnard and Hunter 2008), we find exactly two arguments for each conclusion: one supporting and one undermining.

In the context of knowledge compilation, most work does not address the combination of databases and classical logic (Darwiche and Marquis 2002; Selman and Kautz 1996; Nagy, Lukacsy, and Szeredi 2006; Calvanese et al. 2008; Besnard and Hunter 2006) or does not consider inconsistency (Hinrichs and Genesereth 2008; Cadoli and Mancini 2002; Nagy, Lukacsy, and Szeredi 2006; Calvanese et al. 2008). With the exception discussed below, the compilation work we are aware of that addresses inconsistency in the context of data separate from complex axioms does not focus on large data sets (Flouris et al. 2006; Huang, van Harmelen, and ten Teije 2005).

The closest related work (Gomez, Chesnevar, and Simari 2008) translates a description logic, which naturally separates the data (Abox) and constraints (Tbox), into defeasible logic programming (DeLP) for the purpose of reasoning about inconsistent premises. Several distinctions are worth mentioning. First, (Gomez, Chesnevar, and Simari 2008) defines entailment with respect to the constructed DeLP program, *i.e.*, a sentence is entailed by the description logic premises if it is entailed by the translation of the premises to DeLP; thus, it is unclear which paraconsistency semantics are being computed by the translation. Second, the translation to DeLP applies only to a specific fragment of all sentences expressible in the given description logic; the same holds true of our work, but the fragments are incomparable. Lastly, in (Gomez, Chesnevar, and Simari 2008), the premises for every argument must be consistent with the entire Abox; our work does just the opposite for the case of strict entailment. The data is assumed less trustworthy than the constraints, and arguments never need to be consistent with all of the data.

Conclusion and Future Work

This paper describes compilation algorithms that implement non-explosive entailment relations for inconsistent combinations of Finite Herbrand Logic (a decidable fragment of first-order logic) and relational databases. For strict existential entailment, our algorithms are sound and complete as long as the FHL axioms are quantifier-free. For non-strict existential entailment, the same basic compilation algorithm is sound and complete as long as the FHL axioms are closed under resolution and factoring. Termination is guaranteed when the resolution closure of the FHL axioms is finite.

In the future, we plan to address several issues. First is the possibility that the compilation procedure does not terminate because the resolution closure is infinite. It is feasible that by targeting a more expressive database query language (stratified DATALOG), we could construct recursive database queries that simulate the effects of resolution, similar to (Nagy, Lukacsy, and Szeredi 2006). The benefit is that whereas the compiler does not know the size of the data (and hence cannot bound the size of the resolution closure), the database does have access to that information and can avoid a non-terminating computation.

Second, this work addressed the combination of databases and classical logic, which was motivated in part by today's largest knowledge bases: Cyc (Lenat and Guha 1990) and SUMO (Niles and Pease 2001). These knowledge bases separate data from complex axioms, but instead of applying the closed world assumption to the data, they employ the open world assumption. In the future we will investigate techniques for handling the open world assumption.

Finally, in the context of reasoning about inconsistency, substantial energy has been devoted to constructing argument trees that represent the relationships among the sentences existentially entailed by a premise set. These trees differentiate conclusions that are undermined (entailed by premises whose negations are also entailed) and those that are not. Such information gives a better understanding of the inconsistencies and their implications than the simple queries studied in this paper. In the future we will extend our compilation techniques to handle argument trees.

References

Benferhat, S.; Lagrue, S.; and Rossit, J. 2007. An egalitarist fusion of incommensurable ranked belief bases under constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 367–372.

Besnard, P., and Hunter, A. 2005. Practical first-order argumentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 590–595.

Besnard, P., and Hunter, A. 2006. Knowledgebase compilation for efficient logical argumentation. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 123–133.

Besnard, P., and Hunter, A. 2008. *Elements of Argumentation*. MIT Press.

Binas, A., and McIlraith, S. 2008. Peer-to-peer query answering with inconsistent knowledge. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 329–339.

Cadoli, M., and Mancini, T. 2002. Knowledge compilation = query rewriting + view synthesis. In *Proceedings of the ACM Symposium on Principles of Database Systems*, 199–208.

Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rosati, R.; and Ruzzi, M. 2008. Data integration through dl-litea ontologies. In *Proceedings of the International Workshop on Semantics in Data and Knowledge Bases*.

da Costa, N. C.; Henschen, L. J.; Lu, J. J.; and Subrahmanian, V. S. 1990. Automatic theorem proving in paraconsistent logics: Theory and implementation. In *Proceedings of the Conference on Automated Deduction*, 72–86.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research* 17:229–264.

Efstathiou, V., and Hunter, A. 2008. Algorithms for effective argumentation of classical propositional logic. In *Proceedings of the Symposium on Foundations of Information and Knowledge Systems*.

Everaere, P.; Konieczny, S.; and Marquis, P. 2008. Conflict-based merging operators. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 348–357.

Flouris, G.; Huang, Z.; Pan, J. Z.; Plexousakis, D.; and Wache, H. 2006. Inconsistencies, negations and changes in ontologies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1295–1300.

Gomez, S. A.; Chesnevar, C. I.; and Simari, G. R. 2008. An argumentative approach to reasoning with inconsistent ontologies. In *Proceedings of the KR Workshop on Knowledge Representation and Ontologies*, 11–20.

Hinrichs, T. L., and Genesereth, M. R. 2008. Injecting the how into the what: Investigating a finite classical logic. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*.

Hinrichs, T. L.; Kao, J.-Y.; and Genesereth, M. R. 2009. Inconsistency-tolerant reasoning with classical logic and large databases. Technical report, University of Chicago.

Huang, Z.; van Harmelen, F.; and ten Teije, A. 2005. Reasoning with inconsistent ontologies. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Hunter, A., and Konieczny, S. 2008. Measuring inconsistency through minimal inconsistent sets. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*.

Hunter, A. 1998. Paraconsistent logics. In *Handbook of Defeasible Reasoning and Uncertain Information*. Kluwer Academic Publishers. 11–36.

Konieczny, S.; Lang, J.; and Marquis, P. 2005. Reasoning under inconsistency: the forgotten connective. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 484–489.

Lenat, D. B., and Guha, R. V. 1990. *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project*. Addison-Wesley.

Myers, K. 1990. Automatically generating universal attachments through compilation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Nagy, Z.; Lukacsy, G.; and Szeredi, P. 2006. Translating description logic queries to prolog. In *Proceedings of the Symposium on Practical Aspects of Declarative Languages*, 168–182.

Niles, I., and Pease, A. 2001. Towards a standard upper ontology. In *Proceedings of the Formal Ontology in Information Systems*, 2–9.

Selman, B., and Kautz, H. 1996. Knowledge compilation and theory approximation. *Journal of the ACM* 43(2):193–224.

Sikka, V. 1996. *Integrating Specialized Procedures into Proof Systems*. Ph.D. Dissertation, Stanford University.

Sofronie-Stokkermans, V. 2000. Automated theorem proving by resolution for finitely-valued logics based on distributive lattices with operators. *Multiple-Valued Logic* 6:289–344.

Stickel, M. 1985. Automated deduction by theory resolution. *Journal of Automated Reasoning* 1:333–356.

Stillman, R. B. 1973. The concept of weak substitution in theorem-proving. *J. ACM* 20(4):648–667.

Subrahmanian, V. S., and Amgoud, L. 2007. A general framework for reasoning about inconsistency. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 599–604.

Ullman, J. 1989. *Principles of Database and Knowledge-Base Systems*. Computer Science Press.

Weyhrauch, R. 1980. Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence* 13:133–170.

Zamansky, A., and Avron, A. 2006. Non-deterministic semantics for first-order paraconsistent logics. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 431–439.