# 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

April 20, 2009

**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.

## 1. 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. 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 the reasonable conclusions that can be drawn 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.*, [31, 13, 5]. Of the relatively few results concerned with implementation [10, 28, 4, 12, 6, 15], none address the problems of massive data sets or interfacing with external databases. Standard techniques such as procedural attachments [33, 23, 27] and theory resolution [29] 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-

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.

# 2. 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 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 $\Gamma$.

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 [32].

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 |
|---|---|---|
| $\neg p(a)$ | $\neg p(b)$ | |
| $q(a, a)$ | $\neg q(a, b)$ | $q(a, a)$ |
| $q(b, a)$ | $\neg q(b, b)$ | $q(b, a)$ |

## Combinations of FHL and DATALOG

This paper focuses on compilation techniques for performing reasoning about the combination of FHL and relational databases. 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). In the definition that follows, the combination of an FHL axiom set $\Delta$ and a database $\Lambda$ is called a *parameterized theory instance*.

While the combination of FHL and a database is an important concept, the compilation algorithms we introduce in this paper never operate on such a combination. Rather, they operate on a set of FHL axioms, $\Delta$, and a database's schema (the set of predicates defined in the database), $Q$. In the definition below, the combination of $\Delta$ and $Q$ is called a *parameterized theory*.

To see why our compilation algorithms operate on a database schema instead of the database itself, recall that the underlying assumption of this work is that the database contains vast quantities of data. Since it is natural for a compiler to be built upon automated reasoning technology, a compiler

capable of examining a database's contents would require automated reasoning technology capable of managing vast amounts of data—the very problem our compilation work was meant to address. And while a database's contents is often vast, its schema is almost always relatively tiny, making its manipulation practically feasible.

**Definition 1 (Parameterized Theory).** *A parameterized theory $\langle \Delta, Q \rangle$ is an FHL sentence set $\Delta$ and a predicate set $Q$. An instance of the parameterized theory $\langle \Delta, Q \rangle$ is a two-tuple $\langle \Delta, \Lambda \rangle$ where $\Lambda$ 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: $\Gamma$ entails $\phi$ (denoted $\Gamma \models \phi$ for FHL and $\Gamma \models_D \phi$ for DATALOG) if every model that satisfies $\Gamma$ also satisfies $\phi$. 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.,* [15]).

For a parameterized theory instance $\langle \Delta, \Lambda \rangle$, *strict existential entailment* states that $\Lambda \models_E^{\Delta} \phi$ if there is a subset $\Lambda_0$ of $\Lambda$ such that $\Delta \cup \Lambda_0$ is satisfiable and entails $\phi$. 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. In practice, this definition 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. For example, the ontologies constructed by the description logic community are often verified or forced to be consistent and assumed to correctly describe the domain in question. 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.

### 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 $\Delta'$ is a parameterized compilation if for each predicate $p$ in $\Delta$, there are predicates $p^-$ and $p^+$ occurring in $\Delta'$ such that for every complete, consistent FHL theory $\Lambda$ over $Q$*

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

## 3. Compilation

Our approach to parameterized compilation relies on a standard automated reasoning technique: resolution. Ignoring inconsistency for the moment, when the classical axioms $\Delta$ from a parameterized theory instance $\langle \Delta, \Lambda \rangle$ are closed under resolution and $\Lambda$ is a set of literals, entailment admits an especially simple algorithm. To compute whether or not $\langle \Delta, \Lambda \rangle$ entails $p(a)$, we only need to find a clause $\delta \in \Delta$ such that $\{\delta\} \cup \Lambda \models p(a)$. Moreover, because $\Lambda$ is a set of literals, unit resolution is sufficient to check whether or not $\{\delta\} \cup \Lambda \models p(a)$.

These observations are important because DATALOG evaluation amounts to unit resolution where one of the literals in each clause is denoted as the "head" of the clause. Assuming $\Delta$ is closed under resolution and $\Lambda$ is a set of ground literals, to construct DATALOG rules that implement entailment for $\langle \Delta, \Lambda \rangle$, we construct one DATALOG query for each contrapositive of each clause in $\Delta$. The head of each DATALOG rule is the first literal in each contrapositive, and the body of each DATALOG rule is the negation of the remainder of the contrapositive (sometimes referred to as rule-form in the context of model-elimination).

For example, consider a simple theory closed under resolution, which constitutes the FHL axioms $\Delta$ in the parameterized theory $\langle \Delta, Q \rangle$.

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

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).

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

Given any database (complete, consistent theory) $\Lambda$ complete for $\{p, q, r\}$, the rules above together with $Ext[\Lambda]$ entails (under DATALOG semantics) all those atomic sentences entailed by the parameterized theory instance $\langle \Delta, \Lambda \rangle$ (under FHL semantics).

To handle inconsistency, the algorithm outlined above requires only a small change. Strict entailment can be implemented by appending a consistency check to the end of each DATALOG query. Details of this consistency check can be found after discussion of the core algorithm.

Algorithm 1 performs parameterized compilation for parameterized theories where the FHL axioms are quantifier-free using the algorithm outlined above. 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 that is a different one-line application of Algorithm 2. Note that we use RES[CNF[$\Delta$]] to denote the resolution and factoring closure of the clausal form of $\Delta$ and PREDS[$\phi$] to denote the set of predicates occurring in $\phi$.

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

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

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

**Assumes:** $\Delta$ and $\Sigma$ are clause sets.
**Assumes:** $\Delta$ is closed under resolution and factoring.
**Outputs:** A DATALOG theory

1: **for all** predicates $p \in Q \cup$ 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: $\Gamma$ **:=** $\{d\} \cup \Sigma$
6: **OUTPUT-CONSISTENCY-CHECK[**$\phi(\bar{x})$**,**$\Gamma$**]**
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 $\Theta$ 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 $\Theta$ 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 can be found in Appendix A.

**Theorem 1 (Soundness of COMPILATION).** *Consider a parameterized theory $\langle \Delta, Q \rangle$, where $\Delta$ is a clause set closed under resolution and factoring, and $\Sigma$ is a satisfiable set of FHL sentences. Use $\Delta'$ to denote the output of* COMPILATION*($\langle \Delta, Q \rangle$,$\Sigma$). Suppose $\Lambda$ 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 $\Sigma$ and entails $p(\bar{a})$.*

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

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

The corollaries below extend soundness and completeness to parameterized theories where the FHL axioms are arbitrary clause sets. 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 $\Delta$ 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 $\phi$. Because $Mod[\Delta] = Mod[f[\Delta]]$, $\Lambda_0 \cup f[\Delta]$ must be satisfiable and entail $\phi$ by the usual definition of satisfaction and entailment. Thus, $\Lambda_0$ witnesses the strict entailment $\Lambda \models^{f[\Delta]}_{E} \phi$. The other direction follows similarly. □

To prove the soundness 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 original premises are quantifier-free.

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

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

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

*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[$\Delta$]] and entails $p(\bar{a})$. Consequently, $Exp[\Lambda] \models_E^{\text{RES[CNF[}\Delta\text{]]}} p(\bar{a})$. Since $\Delta$ is quantifier-free, RES[CNF[$\Delta$]] has exactly the same models as $\Delta$. By Proposition 1, $Exp[\Lambda] \models_E^{\Delta} p(\bar{a})$. □

**Corollary 2 (Completeness of STRICT-COMPILATION).** *Suppose $\Delta'$ denotes the output of* STRICT-COMPILATION*$(\langle \Delta, Q \rangle)$, where $\Delta$ is quantifier-free. Further suppose $\Lambda$ 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 $\Delta$ is quantifier-free, $\Delta$ has the same models as RES[CNF[$\Delta$]], and by Proposition 1, $Exp[\Lambda] \models_E^{\text{RES[CNF[}\Delta\text{]]}} p(\bar{a})$. Theorem 2 can then be applied (where $\Sigma$ is RES[CNF[$\Delta$]]) 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 that operates on a conjunction of literals $\phi(\bar{x})$ and a sentence set $\Theta$. $consistent_{\phi}^{\Theta}(\bar{x})$ must be true for exactly those $a$ where $\phi(\bar{a})$ is consistent with $\Theta$. Intuitively, this means that the variable bindings are not made in such a way that a subset of the literals in $\phi$ unify with the negated literals of some clause in $\Theta$. The algorithm that follows implements this intuition using subsumption.

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

**Algorithm 3** OUTPUT-CONSISTENCY-CHECK($\phi$,$\Theta$)

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

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

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[$\Theta$] = $\{p(x,b) \vee r(x,b)$,
$\neg r(a,y) \vee \neg q(y,z)$,
$p(a,b) \vee \neg q(b,z)\}$.

No substitution into the variables of $\phi$ causes a contradiction on its own, so $T$ is empty. The only substitution into the variables of $\phi$ that contradicts $\Theta$ 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 $\Theta$ 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 $\Theta$ are consistent.*

In Algorithm 3, it's not immediately clear how to compute the set $\Sigma$, or even that the set is finite. It turns out that it is sufficient for computing $\Sigma$ to consider each $d \in \Theta$ and for each one run a slightly modified subsumption algorithm, e.g. Stillman's [30]. Similarly, the set $T$ is easy to compute: whenever a predicate occurs both positively and negatively in $\phi$, add to $T$ the most general unifier of the corresponding atoms.

# 4. 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

$\Lambda$ consistent with $\Delta$ that entails $p(\bar{a})$. When $\Delta$ correctly axiomatizes the domain of interest, strict entailment produces intuitive results, but when $\Delta$ includes mistakes or disagreements, strict entailment may produce fewer results than one would like. For example, when $\Delta$ is inconsistent, there are no strict consequences for any database, making it intolerant of inconsistencies within $\Delta$.

One related entailment relation that has received much attention in the literature is stronger (produces more consequences) than strict entailment but is weaker (produces fewer consequences) than traditional entailment. *Existential entailment* applies to a single premise set and a conclusion. $\Gamma$ existentially entails $\phi$ if there is a consistent subset of $\Gamma$ that entails $\phi$. When applied to a parameterized theory, existential entailment states that $\langle \Delta, \Lambda \rangle$ existentially entails $\phi$ if there is a consistent subset of $\Delta \cup \Lambda$ that entails $\phi$. Thus, existential entailment is tolerant of inconsistencies in $\Delta$ because only a portion of $\Delta$ need be used to witness the entailment of $\phi$.

As shown below in Algorithm 4, a simple application of our core compilation algorithm (Algorithm 2) can be used to perform 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.

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

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

**Corollary 3 (Soundness of EXISTENTIAL COMPILATION).** *Suppose $\Delta'$ denotes the output of* EXISTENTIAL-COMPILATION*$(\langle \Delta, Q \rangle)$, where $\Delta$ is a clause set closed under resolution and factoring. Further suppose $\Lambda$ 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 $\Delta$ is closed under resolution and factoring and $\Sigma$ 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 $\Delta'$ denotes the output of* EXISTENTIAL-COMPILATION*$(\langle \Delta, Q \rangle)$, where $\Delta$ is a clause set closed under resolution and factoring. Further suppose $\Lambda$ 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 $\Sigma$ is the empty set, we immediately get the desired result. □

## Negative Results

The important condition on the soundness and completeness results for existential entailment compilation is that the FHL axioms $\Delta$ must be closed under resolution. In contrast, the soundness and completeness results for strict entailment allow $\Delta$ to be any quantifier-free axiom set. It is tempting to forcibly close $\Delta$ 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 $\Gamma$ 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. □

| Sound | Unsound |
|---|---|
| Double Negation | Reflexivity |
| Conditional Proof | Modus Ponens |
| $\wedge$ Elimination | $\wedge$ Introduction |
| $\vee$ Introduction | $\vee$ Elimination |
| Monotonicity | Reductio-ad Absurdum |
| Excluded Middle | Cut |
| | Resolution |

Table 1: Inference rule soundness for existential entailment in propositional logic

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 $\phi$ and $\psi$, $R[\phi, \psi] \subseteq ECons[\{\phi, \psi\}]$. Use $R^*[\Gamma]$ to denote the closure of $\Gamma$ 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. Clearly, though, unsound inference rules will not preserve existential consequences. Table 1 catalogs common inference rules and their soundness with respect to existential entailment. Formal definitions of each rule and proofs can be found in Appendix A.

Despite the fact that soundness of an inference rule does not guarantee soundness of the closure of that rule, 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), 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 $\Delta$ be the sentences $\{p \vee q, \neg p\}$ and $\Lambda$ be $\{p\}$. Together these sentences are inconsistent. Since the sentences are already in clausal form, consider the resolution closure of $\Delta$: $\{p \vee q, \neg p, q\}$. 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 [18] 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 ($\Delta$), 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. It is worth mentioning that in the case where all the constraints are ground (essentially restricting the language to that of propositional logic) and the resolution closure removes tautologies, the entailment relation implemented agrees with quasi-classical logic on the set of queries considered. See Appendix A for details.

# 5. Related Work

The work reported here 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 [18, 21, 3, 20, 17, 34, 14, 31, 19, 13, 5]. In contrast to some

of that work, the problem addressed in this paper detects but does not attempt to repair inconsistencies [13, 21, 2, 31]. Second, our work focuses on a classical logic that is properly neither propositional, *e.g.*, [12], yet retains decidability nor first-order [3] yet retains a relational syntax, important for parameterized theories. Third, instead of building an argument tree to establish the relationships between all possible arguments, *e.g.*, [12, 5], we focus on finding exactly two arguments for each conclusion: one that supports it and one that undermines it.

In the context of knowledge compilation, most work does not address the combination of databases and classical logic [11, 26, 24, 8, 4] or does not consider inconsistency [16, 7, 24, 8]. 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 [14, 17].

The closest related work [15] 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, [15] 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 [15], 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.

# 6. 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 clauses 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 [24]. 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 [22] and SUMO [25]. These knowledge bases separate data from complex axioms, just as we do in this work, 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 data sets under the open world assumption.

Third, the compiler introduced here was designed to answer only (negated) atomic queries: $[\neg]p(\bar{a})$. In the future, we will investigate techniques for answering more complex queries, leveraging the abduction-based results reported in [16].

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 (strict) existential entailment queries studied here. In the future we will study how argument trees might be handled in the context of compilation.

# References


[1] Owen Astrachan and Mark Stickel. Caching and lemmaizing in model elimination theorem provers. In *Proceedings of the Conference on Automated Deduction*, 1992.

[2] Salem Benferhat, Sylvain Lagrue, and Julien Rossit. An egalitarist fusion of incommensurable ranked belief bases under constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 367–372, 2007.

[3] Philippe Besnard and Anthony Hunter. Practical first-order argumentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 590–595, 2005.

[4] Philippe Besnard and Anthony Hunter. Knowledgebase compilation for efficient logical argumentation. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 123–133, 2006.

[5] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. MIT Press, 2008.

[6] Arnold Binas and Sheila McIlraith. Peer-to-peer query answering with inconsistent knowledge. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 329–339, 2008.

[7] Marco Cadoli and Toni Mancini. Knowledge compilation = query rewriting + view synthesis. In *Proceedings of the ACM Symposium on Principles of Database Systems*, pages 199–208, 2002.

[8] Diego Calvanese, G. De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, and Marco Ruzzi. Data integration through dl-litea ontologies. In *Proceedings of the International Workshop on Semantics in Data and Knowledge Bases*, 2008.

[9] Chin-Liang Chang and Richard Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.

[10] Newton C. da Costa, Lawrence J. Henschen, James J. Lu, and V S. Subrahmanian. Automatic theorem proving in paraconsistent logics: Theory and implementation. In *Proceedings of the Conference on Automated Deduction*, pages 72–86, 1990.

[11] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[12] Vasiliki Efstathiou and Anthony Hunter. Algorithms for effective argumentation of classical propositional logic. In *Proceedings of the Symposium on Foundations of Information and Knowledge Systems*, 2008.

[13] Patricia Everaere, Sebastien Konieczny, and Pierre Marquis. Conflict-based merging operators. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 348–357, 2008.

[14] Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis, and Holger Wache. Inconsistencies, negations and changes in ontologies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1295–1300, 2006.

[15] Sergio A. Gomez, Carlos I. Chesnevar, and Guillermo R. Simari. An argumentative approach to reasoning with inconsistent ontologies. In *Proceedings of the KR Workshop on Knowledge Representation and Ontologies*, pages 11–20, 2008.

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

[17] Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Reasoning with inconsistent ontologies. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2005.

[18] Anthony Hunter. Paraconsistent logics. In *Handbook of Defeasible Reasoning and Uncertain Information*, pages 11–36. Kluwer Academic Publishers, 1998.

[19] Anthony Hunter and Sebastien Konieczny. Measuring inconsistency through minimal inconsistent sets. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 2008.

[20] Sebastien Konieczny, Jerome Lang, and Pierre Marquis. Reasoning under inconsistency: the forgotten connective. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 484–489, 2005.

[21] Jerome Lang and Pierre Marquis. Resolving inconsistencies by forgetting. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, pages 239–250, 2002.

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

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

[24] Zsolt Nagy, Gergely Lukacsy, and Peter Szeredi. Translating description logic queries to prolog. In *Proceedings of the Symposium on Practical Aspects of Declarative Languages*, pages 168–182, 2006.

[25] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the Formal Ontology in Information Systems*, pages 2–9, 2001.

[26] Bart Selman and Henry Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, 1996.

[27] Vishal Sikka. *Integrating Specialized Procedures into Proof Systems*. PhD thesis, Stanford University, 1996.

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

[29] Mark Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1:333–356, 1985.

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

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

[32] Jeffrey Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, 1989.

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

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

# A Proofs

## Parameterized Compilation

We start with completeness, showing that because the compilation starts with a set of clauses closed under resolution, it is sufficient to ignore the interactions between clauses when searching for a proof. While this lemma assumes consistency, the next lemma does not, confronting existential entailment directly.

**Lemma 1 (Singleton Sufficiency for Nonexplosive Entailment).** *Suppose $\Delta$ is a clause set that includes the clause $\{p(\bar{x}) \vee \neg p(\bar{x})\}$, and $L$ is a set of ground literals such that $\Delta \cup L$ is satisfiable. If $\Delta \cup L$ entails $p(\bar{a})$ then there is some clause $d$ in the resolution-factoring closure of $\Delta$ such that $d \cup L$ entails $p(\bar{a})$, and $\{d\} \cup L$ is satisfiable. Further, there is a proof of $p(\bar{t})$ using only unit resolution on $d$ and the elements of $L$, where $\bar{a}$ is an instance of $\bar{t}$.*

*Proof.* If $L$ by itself entails $p(\bar{a})$, then the conclusion holds with $p(\bar{x}) \vee \neg p(\bar{x})$ as $d$. So suppose $L$ does not entail $p(\bar{a})$ by itself. By the generative completeness of resolution (up to subsumption), the inference rules of resolution and factoring will generate a proof of $p(\bar{t})$ from $\Delta \cup L$. Consider any such proof that uses $n > 0$ clauses from $\Delta$ as premises. By the commutativity of resolution and factoring (assuming resolution is defined according to [9]), the proof can be rewritten so that all of the resolutions between the $n$ clauses from $\Delta$ are placed first, followed by factoring, followed by unit resolutions using literals from $L$. Notice that after the resolution and factoring phase, the proof will only use premises consisting of one clause $e$ and some number of literals. $e$ is

obviously in the resolution-factoring closure of $\Delta$, ensuring that $e$ together with $L$ entails $p(\bar{t})$ using only unit resolution.

All that remains is to argue that $e$ is consistent with $L$. Since $e$ is in the resolution-factoring closure, and that closure is sound, i.e. produces sentences with no fewer models than the original sentence set, $e$ has no fewer models than $\Delta$. Consequently, the model that satisfies $\Delta$ and $L$ must also be a model of $e$, which means $e$ is consistent with $L$. □

The basic resolution fact above requires a satisfiable premise set. The next lemma does away with that assumption and addresses existential entailment for a parameterized theory $\langle \Delta, Q \rangle$ and a theory $\Lambda$ complete over $Q$, represented explicitly. It guarantees that if $\Delta \cup \Lambda$ existentially entails $p(\bar{a})$, that there is some clause in the resolution closure of $\Delta$ that witnesses the existential entailment.

**Lemma 2 (Singleton Sufficiency for (Strict) Existential Entailment).** *Suppose $\Delta$ is a clause set that includes $\{p(\bar{x}) \vee \neg p(\bar{x})\}$ and is closed under resolution and factoring, $L$ is a set of ground literals, and $\Sigma$ is a satisfiable set of sentences over the same language. If there is a subset of $\Delta \cup L$ that is consistent with $\Sigma$ and entails $p(\bar{a})$, then there is a clause $d \in \Delta$ and a literal set $L' \subseteq L$ such that $\{d\} \cup L' \cup \Sigma$ is consistent and $\{d\} \cup L'$ admits a proof of $p(\bar{a})$ using only unit resolution.*

*Proof.* Suppose there the subset $\Delta_1 \cup L_1$ that is consistent with $\Sigma$ and entails $p(\bar{a})$. Assume $\Delta_1$ includes $p(\bar{x}) \vee \neg p(\bar{x})$, for if it does not, adding it preserves consistency and entailment. By the singleton sufficiency for nonexplosive entailment lemma, there is a clause $d$ in the resolution-factoring closure of $\Delta_1$ such that $d \cup L_1$ entails $p(\bar{a})$ and admits a proof using only unit resolution. Since $\Delta_1 \cup L_1 \cup \Sigma$ is satisfiable, $\{d\} \cup L_1 \cup \Sigma$ is also satisfiable. Since the resolution-factoring closure of $\Delta_1$ is a subset of $\Delta$, that clause $d$ exists in $\Delta$. □

Notice that this theorem does not guarantee that the witness from $\Delta$ is consistent with $L$ in its entirety. It could be that a subset of $L$ is inconsistent with the witness. Thus, each element of $\Delta$ must be augmented to check whether the subset of $L$ used to prove $p(\bar{a})$ is consistent with the witness clause. The following lemma guarantees this small augmentation is complete and relies on the notion of $C_{con}$ found in Definition 3.

**Lemma 3 (Basic Completeness).** *Suppose $\Delta$ is a clause set that contains $\{p(\bar{x}) \vee \neg p(\bar{x})\}$ and is closed under resolution and factoring, $L$ is a set of ground literals, and $\Sigma$ is a satisfiable set of sentences over the same language. Suppose further that there is a satisfiable subset of $\Delta \cup L$ that is consistent with $\Sigma$ and entails $p(\bar{a})$. There must be a subset of $L \cup C_{con}$ together with an element of $\Delta$ that after being augmented with an inconsistency check admits a unit resolution proof of some generalization of $p(\bar{a})$.*

*Proof.* Since the singleton sufficiency for (strict) existential entailment lemma applies, there must be some clause $e$ in $\Delta$ (written in rule form as $p(\bar{u}) \Leftarrow \phi$) that together with $C' \subseteq C$ nonexplosively proves $p(\bar{t})$ (a generalization of $p(\bar{a})$) using only unit resolution. The sequence of resolutions that produce that proof will apply just as readily to the augmentation of $e$, shown below, since the augmentation includes a superset of $e$'s literals.

$$p(\bar{u}) \vee d_1(\bar{t_1}) \vee \cdots \vee d_n(\bar{t_n}) \vee \neg consistent_{\phi}^{\Sigma \cup \{e\}}(\bar{t_1}, \ldots, \bar{t_n})$$

After the resolution sequence is applied, all that remains is

$$p(\bar{t}) \vee \neg consistent_{\phi}^{\Sigma \cup \{e\}}(\bar{a_1}, \ldots, \bar{a_n})$$

where $\bar{a_i}$ is an instance of $\bar{t_i}$ for every $i$. The elements of $L'$ used in resolution sequence must be exactly those that resolve away each of the disjuncts except for $p(\bar{t})$ and $consistent_{\phi}^{\Sigma \cup \{e\}}$. By assumption, these elements are consistent with $e$; thus, by the definition of $C_{con}$, $consistent_{\phi}^{\Sigma \cup \{e\}}(\bar{a_1}, \ldots, \bar{a_n})$ must be included in $C_{con}$. Consequently, unit resolution removes $consistent_{\phi}^{\Sigma \cup \{e\}}$. Thus, the proof without augmentation guarantees the existence of a proof with the augmentation, and both use just unit resolution. □

**Theorem 4 (Completeness of COMPILATION).** *Consider a parameterized theory $\langle \Delta, Q \rangle$, where $\Delta$ is a clause set closed under resolution and factoring, and $\Sigma$ is a satisfiable set of FHL sentences. Use $\Delta'$ to denote the output of* COMPILATION*$(\langle \Delta, Q \rangle, \Sigma)$. Suppose $\Lambda$ is a consistent theory complete over predicates $Q$. If there is a satisfiable subset of $\Delta \cup Exp[\Lambda]$ consistent with $\Sigma$ that entails $p(\bar{a})$ then*

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

*Proof.* The compilation algorithm adds the clause $p(\bar{x}) \vee \neg p(\bar{x})$ to $\Delta$. Suppose there is a satisfiable subset of $\Delta \cup Exp[\Lambda]$ consistent with $\Sigma$ that entails $p(\bar{a})$. The basic completeness lemma applies, and there is a subset of $Exp[\Lambda] \cup C_{con}$ together with an $e$ ($p(\bar{u}) \leftarrow \phi$) from $\Delta$ augmented with an inconsistency check that admits a unit resolution proof of a generalization of $p(\bar{a})$: $p(\bar{t})$.

$$p(\bar{u}) \vee d_1(\bar{t_1}) \vee \cdots \vee d_n(\bar{t_n}) \vee \neg consistent_{\phi}^{\Sigma \cup \{e\}}(\bar{t_1}, \ldots, \bar{t_n})$$

The rule form of this augmented clause belongs to $\Delta'$ by construction.

$$p^+(\bar{X}) :- \neg d_1(\bar{t_1}) \wedge \cdots \wedge \neg d_n(\bar{t_n}) \wedge consistent_{\phi}^{\Sigma \cup \{e\}}(\bar{t_1}, \ldots, \bar{t_n})$$

The elements belonging to $Exp[\Lambda] \cup C_{con}$ that resolved away all of the $d_i$ and $consistent_{\phi}^{\Sigma \cup \{e\}}$ can now be used

(applying the exact same unifiers) to the body of the rule above during DATALOG evaluation. Consequently,

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

□

Finally, we proceed to the soundness proof, broken into two pieces. The first argues for existential entailment given a unit resolution proof. The second shows the soundness of Compilation, subsequent to a lemma showing the relationship between DATALOG evaluation and unit resolution.

**Lemma 4 (Singleton Soundness).** *Suppose $C$ is an explicitly represented complete, consistent theory, $\Sigma$ is a satisfiable set of FHL sentences, and $e$ is a clause $p(\bar{u}) \vee d_1(\bar{t_1}) \vee \cdots \vee d_n(\bar{t_n})$ in FHL. Consider this clause augmented with an inconsistency check:*

$$p(\bar{u}) \vee d_1(\bar{t_1}) \vee \cdots \vee d_n(\bar{t_n}) \vee \neg consistent_{\phi}^{\Sigma \cup \{e\}}(\bar{t_1}, \ldots, \bar{t_n})$$

*If there is a unit resolution proof that ends with $p(\bar{t})$ from the augmented clause and $C \cup C_{con}$, then the $C' \subseteq C$ used in the proof is consistent with $\Sigma \cup \{e\}$ and $C' \cup \{e\}$ entails every instance of $p(\bar{t})$.*

*Proof.* By the soundness of resolution, we see that $C'$, the members of $C_{con}$ participating in the proof, and the augmented clause entail $p(\bar{a})$, and since the original clause is stronger than the augmented one, it with $C'$ also entails $p(\bar{t})$ (and consequently every instance). All that remains is showing that the original clause $e$ together with $C' \cup \Sigma$ is satisfiable. Consider the sequence of unit resolutions used to prove $p(\bar{t})$. The elements belonging to $C'$ must be exactly those that resolve away each of the $d_i$ disjuncts.

$$\{\neg d_1(\bar{a_1}), \ldots, \neg d_n(\bar{a_n})\}$$

where $\bar{a_i}$ is an instance of $\bar{t_i}$ for every $i$. In addition, to resolve away the negative $consistent_{\phi}^{\Sigma \cup \{e\}}$ literal, some atom $consistent_{\phi}^{\Sigma \cup \{e\}}(\bar{a_1}, \ldots, \bar{a_n})$ from $C_{cons}$ must have participated in the proof. The very presence of that literal ensures that $\{\neg d_1(\bar{a_1}), \ldots, \neg d_n(\bar{a_n})\}$ is consistent with $\Sigma \cup \{e\}$. Consequently, $C$' is consistent with $\Sigma \cup \{e\}$, and $C' \cup \{e\}$ entails every instance of $p(\bar{t})$. □

**Lemma 5 (Unit Resolution and DATALOG Evaluation).** *Suppose $\Delta$ is a* DATALOG *program where rule bodies only mention EDB predicates from $Q$. Suppose further that $C$ is an extensionally represented theory complete over $Q$. If $\Delta \cup C \models_D p(\bar{a})$ for an IDB predicate $p$ then there is a unit resolution proof of $p(\bar{t})$ (where $\bar{a}$ is an instance of $\bar{t}$) using the clausal form of one of the rules in $\Delta$ together with $Exp[C]$.*

*Proof.* Here we provide a proof sketch. Since $\Delta \cup C$ entails $p(\bar{a})$ under DATALOG semantics and the body of every rule mentions only EDB predicates, there must be some rule $d \in \Delta$ such that $\{d\} \cup C \models_D p(\bar{a})$. The rule must take the following form.

$$p(\bar{u}) :- b_1(\bar{t_1}) \wedge \cdots \wedge b_n(\bar{t_n}) \wedge \neg c_1(\bar{u_1}) \wedge \cdots \wedge \neg c_m(\bar{u_m})$$

We will show that top-down DATALOG evaluation of this rule concluding $p(\bar{t})$ guarantees the existence of a unit resolution proof of $p(\bar{t})$ using the explicit representation of $C$ and the clausal form of the above rule:

$$p(\bar{u}) \vee \neg b_1(\bar{t_1}) \vee \cdots \vee \neg b_n(\bar{t_n}) \vee c_1(\bar{u_1}) \vee \cdots \vee c_m(\bar{u_m}).$$

Top-down DATALOG evaluation amounts to a series of extension operations (using the jargon from model elimination [1]) and negation as failure (NAF) applications. An extension operation applied to $b_i(\bar{t_i})$ requires some $b_i(\bar{a_i})$ to belong to $C$. This same $b_i(\bar{a_i})$ belongs to $Exp[C]$, ensuring that a unit resolution will remove $\neg b_i(\bar{t_i})$ from the clause above using the exact same unifier as during DATALOG evaluation. An NAF application applied to $\neg c_i(\bar{a_i})$ (which is guaranteed to always be ground) ensures that $c_i(\bar{a_i})$ does not belong to $C$. Since $c_i$ belongs to $Q$ and $C$ is complete over $Q$, $\neg c_i(\bar{a_i})$ belongs to $Exp[C]$. Thus, $c_i(\bar{u_i})$ in the clause can be resolved away with $\neg c_i(\bar{a_i})$, using the exact same unifier as during DATALOG evaluation. Since all of the $b$ and $c$ literals can be resolved away from the clause above using exactly the same unifiers as during DATALOG evaluation, and the DATALOG evaluation resulted in $p(\bar{t})$, so too does the unit resolution proof result in $p(\bar{t})$. Thus, the clause above together with $Exp[C]$ admits a unit resolution proof of $p(\bar{t})$. □

**Theorem 5 (Soundness of COMPILATION).** *Consider a parameterized theory $\langle \Delta, Q \rangle$, where $\Delta$ is a clause set closed under resolution and factoring, and $\Sigma$ is a satisfiable set of FHL sentences. Use $\Delta'$ to denote the output of* COMPILATION*$(\langle \Delta, Q \rangle, \Sigma)$. Suppose $\Lambda$ is a consistent theory complete over 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 $\Sigma$ and entails $p(\bar{a})$.*

*Proof.* Since COMPILATION generates no DATALOG rules from $\Delta$ where the predicates in the body are not a subset of $Q$, every DATALOG rule in $\Delta'$ mentions only EDB predicates in the body. Thus, by the unit resolution and DATALOG evaluation lemma, since $\Delta' \cup Ext[\Lambda] \cup Ext[C_{con}] \models_D p^+(\bar{a})$ there must be a unit resolution proof of $p^+(\bar{t})$ (where $\bar{a}$ is an instance of $\bar{t}$) using the clausal form of one of the rules in $\Delta'$ together with $Exp[\Lambda \cup C_{con}] = Exp[\Lambda] \cup Exp[C_{con}]$:

$$p^+(\bar{u}) \vee d_1(\bar{t_1}) \vee \cdots \vee d_n(\bar{t_n}) \vee \neg consistent_{\phi}^{\Sigma \cup \{e\}}(\bar{t_1}, \ldots, \bar{t_n}) \quad (3)$$

If we replace $p^+(\bar{u})$ with $p(\bar{u})$, that same proof can be used to conclude $p(\bar{t})$. By the Singleton Soundness Lemma, the proof of $p(\bar{t})$ using $Exp[\Lambda] \cup Exp[C_{con}]$ ensures that there is a $\Lambda_0 \subset Exp[\Lambda]$ consistent with $\Sigma$ and the clause $e$

$$p(\bar{u}) \vee \neg d_1(\bar{t_1}) \vee \cdots \vee \neg d_n(\bar{t_n}) \tag{4}$$

that entails $p(\bar{a})$. Because the rule corresponding to (3) belongs to $\Delta'$, clause (4) must belong to $\Delta$. Hence, there is a satisfiable subset of $\Delta \cup Exp[\Lambda]$ that is consistent with $\Sigma$ and entails $p(\bar{a})$. □

**Theorem 6 (Consistency Axiomatization).** *Given an FHL clause set $\Theta$ 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 $consistent^{\Theta}_{\phi}(\bar{x})$ such that $consistent^{\Theta}_{\phi}(\bar{a})$ evaluates to true if and only if $\phi(\bar{a})$ and $\Theta$ are consistent.*

*Proof.* We show that $consistent^{\Theta}_{\phi}(\bar{a})$ evaluates to false if and only if $\phi(\bar{a})$ and $\Theta$ are inconsistent.

($\Leftarrow$) Consider constants $\bar{a}$ such that $\phi(\bar{a})$ is inconsistent with $\Theta$. Write $\phi(\bar{a})$ as $l_1 \wedge \cdots \wedge l_n$, where each $l_i$ is a ground literal. Let $C$ be the set of clauses $\{\{l_1\}, \ldots, \{l_n\}\}$. The resolution closure of $C \cup \Theta$ must contain the empty clause. Since every clause in $C$ is a ground singleton and $\Theta$ is closed under resolution and factoring, there exists an $d \in \Theta$ such that the resolution closure of $C \cup \{d\}$ includes the empty clause.

Case 1: Assume that $\phi(\bar{a})$ is consistent by itself. So $C$ itself does not produce the empty clause. It must be the case that there is an instantiation of $d$ such that for each literal $l$ in $d$, $\neg l \in C$. In other words, $d$ subsumes $\neg\phi(\bar{a})$. Then $\bar{x} \leftarrow \bar{a}$ or some generalization belongs to $\Sigma$ and $\bigvee_{\sigma \in \Sigma} \bigwedge_{t \leftarrow u \in \sigma} t = u$ evaluates to true.

Case 2: $\phi(\bar{a})$ is itself inconsistent. Then $\bar{x} \leftarrow \bar{a}$ or some generalization belongs to $T$ and $\bigvee_{\tau \in T} \bigwedge_{t \leftarrow u \in \tau} t = u$ evalutes to true.

In both cases, $consistent^{\Theta}_{\phi}(\bar{a})$ evalutes to false.

($\Rightarrow$) Assume $consistent^{\Theta}_{\phi}(\bar{a})$ evalutes to false. Then there are two cases:

Case 1: $\bigvee_{\tau \in T} \bigwedge_{t \leftarrow u \in \tau} t = u$ is true. Then by construction $\phi(\bar{a})$ is itself inconsistent.

Case 2: $\bigvee_{\sigma \in \Sigma} \bigwedge_{t \leftarrow u \in \sigma} t = u$ is true. Then $\neg\phi(\bar{a})$ is subsumed by some $d \in \Theta$. Then $\Theta \models \neg\phi(\bar{a})$; hence $\Theta \cup \{\phi(\bar{a})\}$ is inconsistent. □

## Propositional Inference Rules and Existential Entailment

**Proposition 6 (Reflexivity is unsound).** *The rule $A \vdash A$ is unsound for existential entailment.*

*Proof.* $p \wedge \neg p \not\models_E p \wedge \neg p$ □

**Proposition 7 (Double Negation is sound).** *The following inference rule is sound for existential entailment.* $\dfrac{\Gamma \vdash \neg\neg p}{\Gamma \vdash p}$

*Proof.* The satisfiable subset of $\Gamma$ that entails $\neg\neg p$ must also entail $p$ by double-negation elimination of classical logic. This satisfiable subset therefore witnesses the existential entailment of $p$. □

**Proposition 8 (Modus Ponens is unsound).** *The following inference rule is unsound for existential entailment.* $\dfrac{\Gamma \vdash A \Rightarrow B \qquad \Sigma \vdash A}{\Gamma, \Sigma \vdash B}$

*Proof.* $p \wedge (r \Rightarrow s) \models_E r \Rightarrow s$ and $\neg p \wedge r \models_E r$ but neither $p \wedge (r \Rightarrow s)$ nor $\neg p \wedge r$ entails $d$ alone. Because the conjunction is inconsistent and $d$ is not a tautology, there is no satisfiable subset of

$$p \wedge (r \Rightarrow s)$$
$$\neg p \wedge r$$

that entails $d$. Hence, modus ponens is unsound for existential entailment. □

**Proposition 9 (Reductio Ad Absurdum is unsound).** *The following inference rule is unsound for existential entailment.* $\dfrac{\Gamma, \neg A \vdash B \qquad \Gamma, \neg A \vdash \neg B}{\Gamma \vdash A}$

*Proof.* $p, \neg p, \neg q \models_E p$ and $p, \neg p, \neg q \models_E \neg p$ but $p, \neg p \not\models_E \neg q$. □

**Proposition 10 (Conditional Proof is sound).** *The following inference rule is sound for existential entailment.* $\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}$

*Proof.* If $\Gamma \models_E B$ (i.e. $\Gamma$ without $A$ existentially entails $B$) then the satisfiable subset of $\Gamma$ that entails $B$ also entails $A \Rightarrow B$ since the consequent of the implication is known to be true in all models. Thus this subset is satisfiable and entails $A \Rightarrow B$. So suppose the subset of $\Gamma, A$ that entails $B$ requires $A$; denote the subset of $\Gamma$ as $\Gamma_0$. Since all models of $\Gamma_0, A$ satisfy $B$, then all models of $\Gamma_0$ that also satisfy $A$ satisfy $B$, i.e. if a model of $\Gamma_0$ satisfies $A$ then it satisfies $B$; otherwise it fails to satisfy $A$. Thus, every model of $\Gamma_0$ satisfies $\neg A \vee B$, which ensures that every model of $\Gamma_0$ entails $A \Rightarrow B$, and since the subset is guaranteed satisfiable, the existential entailment of $A \Rightarrow B$ holds. □

**Proposition 11 (Conjunction Introduction is unsound).** *The following inference rule is unsound for existential entailment.* $\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$

*Proof.* $p, \neg p \models_E p$ and $p, \neg p \models_E \neg p$ but $p, \neg p \not\models_E p \wedge \neg p$. □

**Proposition 12 (Conjunction Elimination is sound).** *The following inference rule is sound for existential entailment.*

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A \qquad \Gamma \vdash B}$$

*Proof.* If $\Gamma$ existentially entails $A \wedge B$ then there is a satisfiable subset of $\Gamma$ that entails $A \wedge B$. Every model of that subset satisfies $A \wedge B$; consequently, every model satisfies $A$ and $B$ individually. Thus, that subset of sentences entails $A$ and $B$ individually, and therefore each is existentially entailed. □

**Proposition 13 (Disjunction Introduction is sound).** *The following inference rule is sound for existential entailment.*

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}$$

*Proof.* If $\Gamma$ existentially entails $A$ then there is a satisfiable subset of $\Gamma$ that entails $A$. Every model of that subset satisfies $A$; consequently, every model satisfies $A \vee B$ as well. Thus, the subset of $\Gamma$ that witnesses the existential entailment of $A$ also witnesses the existential entailment of $A \vee B$. □

**Proposition 14 (Disjunction Elimination is unsound).** *The following inference rule is unsound for existential entailment.*

$$\frac{\Gamma \vdash A \vee B \qquad \Sigma \vdash A \Rightarrow C \qquad \Delta \vdash B \Rightarrow C}{\Gamma \vdash C}$$

*Proof.* Consider three premise sets and conclusions.

- $p \vee q \models_E p \vee q$
- $\neg p \models_E p \Rightarrow r$
- $\neg q \models_E q \Rightarrow r$

Disjunction elimination would conclude that the union of the three premise sets must conclude $r$. For existential entailment, this is clearly not the case since $r$ is not mentioned in any of the three premise sets, and neither is it mentioned in any satisfiable subset. Since $r$ is not a tautology, no satisfiable premise set that fails to mention $r$ entails $r$. Thus, the three premise sets do not existentially entail $r$. □

**Proposition 15 (Monotonicity).** *Existential entailment is monotonic, i.e. the following inference rule is sound.*

$$\frac{\Gamma \vdash A}{\Gamma, B \vdash A}$$

*Proof.* If there is a satisfiable subset of $\Gamma$ that entails $A$, those sentences are also a subset of $\Gamma, B$; consequently $\Gamma, B$ existentially entails $A$. □

**Proposition 16 (Cut is unsound).** *The following inference rule is unsound for existential entailment.*

$$\frac{\Gamma \vdash A \qquad \Sigma, A \vdash B}{\Gamma, \Sigma \vdash B}$$

*Proof.* Consider two premise sets and conclusions: $p \Rightarrow q, p \models_E q$ and $q, \neg p, q \wedge \neg p \Rightarrow r \models_E r$. We must show that the following premise set does not existentially entail $r$.

1. $p \Rightarrow q$
2. $p$
3. $\neg p$
4. $\neg p \wedge q \Rightarrow r$

Consider the maximal satisfiable subsets. First, note that every satisfiable subset must leave out (2), (3) or both. Leaving out both is unnecessary because the remaining sentences are consistent with (2) and (3) individually. So consider the first maximal satisfiable subset. It consists of sentences (1), (2), and (4) and is satisfied by the model $\{p, q, \neg r\}$. This model fails to satisfy $r$; hence, this subset does not entail $r$. The only other maximal satisfiable subset consists of (1), (3), and (4). It is satisfied by the model $\{\neg p, \neg q, \neg r\}$. Because this model fails to satisfy $r$, the second satisfiable subset does not entail $r$ either. Since neither subset entails $r$ and these subsets constitute all of the maximal satisfiable subsets, by the monotonicity of classical entailment, none of the satisfiable subsets entails $r$; hence, the premise set above fails to existentially entail $r$. □

**Proposition 17 (Excluded Middle is sound).** *The inference rule $\Gamma \vdash p \vee \neg p$ is sound for existential entailment.*

*Proof.* For any premise set $\Gamma$, there is a satisfiable subset that entails $p \vee \neg p$. That subset is the empty set. An empty set of premises is satisfiable and entails all tautologies in classical logic. □

Constructing a sound and complete set of inference rules for existential entailment is in some sense trivial. Any sound and complete system for traditional entailment is sound and complete for existential entailment as long as the premises are satisfiable. An existential proof system that applies a traditional proof system to each satisfiable subset of the premises will therefore be sound and complete. The real question is whether or not there is a proof system that is sound and complete without checking satisfiability explicitly. While there are some obvious positive cases, i.e. whenever the consistency check is trivial, we leave this investigation to future work. It is noteworthy, however, that the soundness of a very basic inference rule, reflexivity, is sound *exactly* when the premise is satisfiable. This guarantees that determining whether or not a set of sentences existentially entails one of the sentences in the set requires a satisfiability check (at least implicitly).

**Proposition 18 (Satisfiable Reflexivity is sound).** *The inference rule $A \vdash A$ is sound for existential entailment exactly when A is satisfiable.*

*Proof.* ($\Leftarrow$) If $A$ is satisfiable, then clearly $A$ is existentially entailed by the satisfiable subset $\{A\}$. ($\Rightarrow$) If the inference rule is sound, then $A$ existentially entails $A$. There are only two possible witnesses of the entailment: $\{A\}$ and the empty set. For $\{A\}$ to be a witness, $A$ must itself be satisfiable. If the empty set is a witness, it classically entails $A$, meaning that $A$ must be a tautology and again satisfiable. □

**Proposition 19 (Resolution is unsound).** *The following inference rule is unsound for existential entailment.*

$$\frac{\Gamma \vdash A \vee B \qquad \Sigma \vdash \neg A \vee C}{\Gamma, \Sigma \vdash B \vee C}$$

*Proof.* Consider the four sentences $\{p, \neg p \vee q \vee r, \neg p, p \vee \neg q \vee r\}$. The first two existentially entail $q \vee r$; the last two existentially entail $\neg q \vee r$. Resolution applied to $q \vee r$ and $\neg q \vee r$ produces $r$, which is not existentially entailed by the original four sentences. □

While resolution applied to arbitrary existential consequences of a sentence set does not always produce additional existential consequences of that set, if resolution is applied to any two clauses that are members of the premise set, the result is guaranteed to be an existential consequence. In other words, resolution is a sound inference rule, but the closure of resolution is unsound.

**Proposition 20 (Resolution is initially sound).** *The following inference rule is sound for existential entailment as long as $n > 0$ or $m > 0$ and repeated literals do not appear in any disjunction.*

$$\frac{A \vee B_1 \vee \cdots \vee B_n \in \Gamma \qquad \neg A \vee C_1 \vee \cdots \vee C_m \in \Sigma}{\Gamma, \Sigma \vdash B_1 \vee \cdots \vee B_n \vee C_1 \vee \cdots \vee C_m}$$

*Proof.* Clearly $A \vee B_1 \vee \cdots \vee B_n$ and $\neg A \vee C_1 \vee \cdots \vee C_m$ entail $B_1 \vee \cdots \vee B_n \vee C_1 \vee \cdots \vee C_m$ by the soundness of resolution under traditional semantics. We need only demonstrate that $A \vee B_1 \vee \cdots \vee B_n$ and $\neg A \vee C_1 \vee \cdots \vee C_m$ are consistent, thereby witnessing the required satisfiable subset of $\Gamma \cup \Sigma$.

If either clause is a tautology, then satisfiability holds immediately, so suppose neither clause is a tautology. By the following lemma, the number of models satisfying a non-tautologous, non-repeating clause with $k$ literals is given by a recurrence that is easily rewritten in closed form as

$$2^n(\frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^k}).$$

The number of models not satisfying such a clause is then $2^n$ minus that expression. If the total number of models falsifying two clauses is less than the total number of models, then clearly the two clauses are satisfiable. Thus we show that

$$\begin{aligned}(2^n - 2^n(\frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^m}) \\ + (2^n - 2^n(\frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^n}) \\ < 2^n\end{aligned}$$

After simple algebraic manipulation, we get

$$1 < \frac{1}{2} + \cdots + \frac{1}{2^m} + \frac{1}{2} + \cdots + \frac{1}{2^n}.$$

Since we know that either $m > 0$ or $n > 0$, this expression is always true because the right hand side always includes two $\frac{1}{2}$ terms and at least one other term. □

**Lemma 6.** *In a propositional language with $n$ constants, a non-tautologous clause with $m$ literals, none of which are repeated, is satisfied by exactly $f(m)$ models, where*

$$\begin{aligned}f(0) &= 0 \\ f(m) &= \frac{1}{2}2^n + \frac{1}{2}f(m-1).\end{aligned}$$

*Proof.* By induction on the length of the clause. Base case. The empty clause is satisfied by no models, so f(0) = 0, as it should.

Inductive step. Assume that every non-tautologous, non-repeating clause with $k$ literals is satisfied by $f(k)$ models. Show for $k+1$. Consider an arbitrary literal in the clause, and without loss of generality, assume the literal is positive: $p$. Exactly half the models satisfy the clause because they satisfy $p$, and some portion of the remaining models satisfy the remainder of the clause.

If $p$ did not belong to the clause, by the inductive hypothesis that clause would be satisfied by $f(k)$ models. Since the clause is non-tautologous and non-repeating, $p$ does not occur in the remainder of the clause, and hence for every model satisfying the reduced clause where $p$ is true there is another satisfying model exactly the same except that $p$ is false. All those models where $p$ is true will be counted by the presence of $p$ in the enlarged clause, and so as to avoid double counting, we must remove all those models satisfying $p$ from the $f(k)$ count. Since exactly half of those models satisfy $p$, the number of models satisfying the remainder of the clause where $p$ is false is exactly $\frac{1}{2}f(k)$.

Altogether, the number of satisfying models for a clause is $\frac{1}{2}2^n + \frac{1}{2}f(k)$. This completes the inductive step and the proof by mathematical induction. □

## A New Entailment Relation

The commented out theorem below shows that quasi-classical entailment is equivalent to Algorithm 4 with a resolution (and eventually CNF) preprocessor as long as tautologies are removed.

**Theorem 7.** *Suppose $\Delta$ is a propositional clause set, $\Lambda$ is a set of literals, and $p$ is a proposition. Consider the entailment relations $\models_1$ (quasi-classical) and $\models_2$ (Algorithm 4 with a resolution preprocessor).*

$$\Delta \cup \Lambda \models_1 p \quad \textit{if} \quad p \in Res[\Delta \cup \Lambda]$$
$$\Delta \cup \Lambda \models_2 p \quad \textit{if} \quad Res[\Delta] \cup \Lambda \models_E p$$

*If Res denotes the resolution closure with tautology elimination then*

$$\Delta \cup \Lambda \models_1 p \textit{ iff } \Delta \cup \Lambda \models_2 p.$$

*Proof.* Suppose $p \in \Lambda$ then $p \in Res[\Delta \cup \Lambda]$ and hence $\Delta \cup \Lambda \models_1 p$. Also, $\{p\}$ is a subset of $Res[\Delta] \cup \Lambda$ that is satisfiable and entails $p$; hence, $\Delta \cup \Lambda \models_2 p$. So presume that $p \notin \Lambda$.
($\Leftarrow$)

$\Delta \cup \Lambda \models_2 p$
$\Rightarrow Res[\Delta] \cup \Lambda \models_E p$
$\Rightarrow$ $\exists d \in \Delta$ such that $\{d\} \cup \Lambda \models_E p$
$\Rightarrow$ $\exists \Lambda_0 \subseteq \Lambda$ consistent with $d$ s.t. $\{d\} \cup \Lambda_0 \models p$
$\Rightarrow$ $\exists \Lambda_0 \subseteq \Lambda$ consistent with $d$ s.t. $p \in Res[\{d\} \cup \Lambda_0]$

We also know that

$$\begin{aligned} & Res[\Delta \cup \Lambda] \\ &= Res[Res[\Delta] \cup \Lambda] \\ &= Res[d_1 \cup \Lambda] \cup \cdots \cup Res[d_n \cup \Lambda] \end{aligned}$$

for $\{d_1, \ldots, d_n\} = Res[\Delta]$. Thus $Res[\{d\} \cup \Lambda_0]$ is a subset of $Res[\Delta \cup \Lambda]$, and since $p \in Res[\{d\} \cup \Lambda_0]$ then $p \in Res[\Delta \cup \Lambda]$. Ergo $\Delta \cup \Lambda \models_1 p$.
($\Rightarrow$)

$\Delta \cup \Lambda \models_1 p$
$\Rightarrow p \in Res[\Delta \cup \Lambda]$
$\Rightarrow \exists d \in Res[\Delta]$ such that $p \in Res[\{d\} \cup \Lambda]$
$\Rightarrow \exists$ a minimal $\Lambda_0 \subseteq \Lambda$ such that $p \in Res[\{d\} \cup \Lambda_0]$
$\Rightarrow \{d\} \cup \Lambda_0 \models p$

We claim that $\{d\} \cup \Lambda_0$ must be consistent since $\Lambda_0$ is minimal. With this claim,

$\Rightarrow \{d\} \cup \Lambda_0 \models_E p$
$\Rightarrow Res[\Delta] \cup \Lambda \models_E p$
$\Rightarrow \Delta \cup \Lambda \models_2 p$

To prove the claim, consider a single clause $d$ and a minimal set of literals $\Lambda_0$ whose resolution closure includes $p$. Because $p \notin \Lambda$, $d$ must be of the form $p \vee \Phi$. $\Lambda_0$ does not include complementary literals since it is minimal and $d$ is not a tautology. $\Lambda_0$ does not include $\neg p$ for the same reason. Thus, $p \cup \Lambda_0$ is consistent because $\Lambda$ is consistent and can be extended with $p$ because $\neg p$ is not a member. Thus $\{d\} \cup \Lambda_0$ is satisfied by any model that satisfies $p \cup \Lambda_0$. □

**Proposition 21.** *The theorem above does not hold if $Res$ does not eliminate tautologies. In particular, $\Delta \cup \Lambda \models_2 p$ implies $\Delta \cup \Lambda \models_1 p$, but the converse does not hold.*

*Proof.* In the previous proof, ($\Leftarrow$) does not rely on a lack of tautologies; thus, it still holds with tautologies. Now we show a counterexample to ($\Rightarrow$) from the last proof. Suppose $\Delta$ is a singleton set consisting of the tautology $p \vee q \vee \neg q$. Notice that $Res[\Delta]$ is the same as $\Delta$ if tautology elimination is not used. Consider $\Lambda = \{q, \neg q\}$. $Res[\Delta \cup \Lambda]$ includes $p$, but $Res[\Delta] \cup \Lambda$ does not existentially entail $p$; ergo, $\Delta \cup \Lambda \models_1 p$ but $\Delta \cup \Lambda \not\models_2 p$. □