Reformulation for Extensional Reasoning

Timothy L. Hinrichs and Michael R. Genesereth

Stanford University
{thinrich,genesereth}@cs.stanford.edu

Abstract. Relational databases have had great industrial success in computer science. The power of the paradigm is made clear both by its widespread adoption and by theoretical analysis. Today, automated theorem provers are not able to take advantage of database query engines and therefore do not routinely leverage that source of power. Extensional Reasoning (ER) is an approach to automated theorem proving where the machine automatically translates a logical entailment query into a database, a set of view definitions, and a database query such that the entailment query can be answered by answering the database query. The techniques developed for ER to date are applicable only when the logical theory is axiomatically complete. This paper discusses techniques for reformulating an incomplete theory into a complete theory so that Extensional Reasoning techniques can be applied.

1 Introduction

Relational databases are one of the most successful applications in computer science, yet today's theorem provers fail to capitalize on that success. Extensional Reasoning (ER)[1] is an approach to automated theorem proving where the system automatically transforms a logical entailment query into a database (a set of extensionally defined tables), a set of view definitions (a set of intensionally defined tables), and a database query. Extensional Reasoning has been shown to have powerful properties in the case of axiomatically complete theories, e.g. orders of magnitude performance improvement and more predictable run-time behavior than traditional techniques. This paper examines techniques for Extensional Reasoning that reformulate incomplete theories into complete theories to achieve those same benefits.

Because theory-completion in the context of Extensional Reasoning is performed solely for the sake of efficiency, there are constraints on how the theory can be completed. To preserve soundness and completeness, a theory must be completed so that any entailment query about the original theory can be transformed into an entailment query about the completed theory where the answers to the queries are the same. Handling incomplete theories in Extensional Reasoning therefore requires developing two transformation algorithms: one for the query and one for the theory.

For example, suppose we have the query $\forall x.(p(x) \Rightarrow q(x))$ and the following set of sentences about p and q.

$$\forall x.(x = a \vee x = b)$$
$$a \neq b$$
$$\neg p(a)$$
$$\neg q(a)$$
$$q(b)$$

This premise set is incomplete because neither $p(b)$ nor $\neg p(b)$ is entailed; however, the universal query $\forall x.(p(x) \Rightarrow q(x))$ is entailed. For Extensional Reasoning the premise set is converted into a complete theory that consists of two definitions: one for all the possible (consistent) values of p and one for all the possible values of q. The query is then rewritten in terms of those definitions.

$$\forall x.(poss_{p(x)}(x) \Rightarrow poss_{q(x)}(x))$$

$poss_{p(x)}$ is defined so that $poss_{p(x)}(t)$ is true for all those t where $p(t)$ is consistent with the premise set; likewise for $poss_{q(x)}$. Since every ground atom is either consistent with the premises or not, the definitions for $poss_{p(x)}$ and $poss_{q(x)}$ are complete.

$$poss_{p(x)}(x) \Leftrightarrow (x = b)$$
$$poss_{q(x)}(x) \Leftrightarrow (x = b)$$

The *poss* version of the query is entailed by the definitions for $poss_{p(x)}$ and $poss_{q(x)}$ if and only if the original entailment query holds. By converting an incomplete theory to a complete theory about *poss* and rewriting the query in terms of *poss*, Extensional Reasoning can be employed to answer entailment queries about incomplete theories. This approach uses what is possible to infer what is necessarily true.

Similar techniques for theory completion (or perhaps more accurately theory minimization) have been studied to great extent in the Nonmonotonic Reasoning literature, e.g. the Closed-World Assumption[2], Predicate Completion[3], Circumscription[4]. Unlike Extensional Reasoning, these settings assume it is acceptable (and in fact desirable) for the reformulation to change the logical consequences of the theory. In contrast, ER completion is consequence-preserving in the sense described above, thus making the Nonmonotonic work insufficient for the purpose of ER.

While we hope Extensional Reasoning will eventually be applied to a wide variety of logics, for the time being we have elected to focus on theories in a decidable logic, placing the issue of efficiency front and center. The particular logic we are studying is a fragment of first-order logic that is a perennial problem in the theorem proving community: it includes the domain closure axiom, which guarantees decidability while allowing arbitrary quantification. This logic, to which the example above belongs, allows us to avoid issues of undecidability at this early stage in the development of Extensional Reasoning, while at the same time giving us the opportunity to make progress on an important class of problems. It should be noted that this logic can be translated into propositional

logic, which allows entailment queries to be answered with a SAT solver; however, the naive translation is exponential.

Orthogonal to the choice of logic, Extensional Reasoning can be applied in a variety of settings that differ in the way efficiency is measured. Our work thus far measures efficiency the same way the theorem proving community does. Once the machine has been given a premise set and a query the clock starts; the clock stops once the machine returns an answer. We do not amortize reformulation costs over multiple queries, and we do not assume the premises or query have ever been seen before or will ever be seen again.

This paper presents theory-completion techniques that run in linear or quadratic time and allow Extensional Reasoning to be performed on theories whose incomplete portion is ground. The intent is to use fast reformulation algorithms that put most of the burden for answering the entailment query on industrial-strength database algorithms, thereby leveraging the many man-hours spent developing them. Our technical contributions can be found in the sections on query rewriting (4) and theory completion (5). The remaining sections discuss background (2), examples (3), and future work (6).

2 Background

In our investigations of Extensional Reasoning thus far, the logic we have considered is function-free first-order logic with unique names axioms (UNA) and a domain closure axiom (DCA). The UNA state that every pair of distinct object constants in the vocabulary is unequal. The DCA states that every object in the universe must be one of the object constants in the vocabulary. Together, the UNA and DCA ensure that the only models that satisfy a given set of sentences are the Herbrand models of those sentences, and because the language is function-free, every such model has a finite universe. We call this logic Finite Herbrand Logic (FHL). It is noteworthy that entailment in FHL is decidable.

Besides the existence of UNA and a DCA, the definitions for FHL are the same as those for function-free first-order logic. Terms and sentences are defined as usual. We say a sentence is *closed* whenever it has no free variables, and an *open* sentence has at least one free variable. The definition for a model is the usual one, but because all the models of FHL are finite Herbrand models, it is often convenient to treat a model as a set of ground atoms. When we do that, satisfaction is defined as follows.

Definition 1 (FHL Satisfaction). *The definition for the satisfaction of closed sentences, where the model M is represented as a set of ground atoms, is as follows.*

$\models_M s = t$ *if and only if* s *and* t *are syntactically identical.*
$\models_M p(t_1, \ldots, t_n)$ *if and only if* $p(t_1, \ldots, t_n) \in M$
$\models_M \neg\psi$ *if and only if* $\not\models_M \psi$
$\models_M \psi_1 \wedge \psi_2$ *if and only if* $\models_M \psi_1$ *and* $\models_M \psi_2$
$\models_M \forall x.\psi(x)$ *if and only if* $\models_M \psi(a)$ *for every object constant* a.

An open sentence $\phi(x_1, \ldots, x_n)$ with free variables $x_1, \ldots, x_n$ is satisfied by M if and only if $\forall x_1 \ldots x_n.\phi(x_1, \ldots, x_n)$ is satisfied by M according to the above definition.

A set of sentences in FHL constitutes an incomplete theory whenever there is more than one Herbrand model that satisfies it. A set of sentences in FHL constitutes a complete theory whenever there is at most one Herbrand model that satisfies it. Logical entailment is defined as usual: $\Delta \models \phi$ in FHL if and only if every model that satisfies Δ also satisfies ϕ.

In this paper, we demonstrate how to transform an entailment query about an incomplete theory into an entailment query about a complete theory, with the eventual goal of transforming the complete theory into a database system. Consequently, the complete theories we will be targeting have a specific form that makes conversion to a database system straightforward. Every set of sentences constituting a complete theory in this paper will be a set of biconditional definitions such that no predicate is ever (transitively) defined in terms of itself, i.e. a set of nonrecursive biconditional definitions.

Definition 2 (Biconditional Definition). *A biconditional definition is a sentence of the form $r(\overline{x}) \Leftrightarrow \phi(\overline{x})$, where r is a predicate, $\overline{x}$ is a nonrepeating sequence of variables, and $\phi(\overline{x})$ is a sentence with free variables $\overline{x}$. We refer to $r(\overline{x})$ as the head and $\phi(\overline{x})$ as the body.*

Definition 3 (Nonrecursive biconditional definitions). *A set of biconditional definitions Δ is nonrecursive if and only if the graph $\langle V, E \rangle$ is acyclic.*

- *V : the set of predicates in Δ*
- *E : $\langle p, q \rangle$ is a directed edge if and only if there is a biconditional in Δ with p in the head and q in the body.*

Theorem 1 (Biconditional Completeness [1]). *Suppose Δ is a finite set of nonrecursive, biconditional definitions in FHL, where there is exactly one definition for each predicate in Δ and no definition for $=$. Δ is complete.*

3 Example

In the case of complete theories, Extensional Reasoning can produce substantial speedups when compared to traditional techniques. Consider the following set of nonrecursive biconditional definitions that state which of the squares in Fig. 1 are located to the west of which other squares.

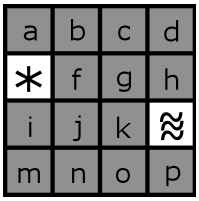

Fig. 1. A Wumpus World snapshot with a shine to the west and a stench to the east

$$westof(x,y) \Leftrightarrow (duewest(x,y) \vee \exists z.(samecolumn(x,z) \wedge duewest(z,y)))$$
$$samecolumn(x,y) \Leftrightarrow (duenorth(x,y) \vee duenorth(y,x))$$
$$duewest(x,y) \Leftrightarrow \begin{pmatrix} onewest(x,y) \vee \\ \exists z.(onewest(x,z) \wedge onewest(z,y)) \vee \\ \exists zw.(onewest(x,z) \wedge onewest(z,w) \wedge onewest(w,y)) \end{pmatrix}$$
$$duenorth(x,y) \Leftrightarrow \begin{pmatrix} onenorth(x,y) \vee \\ \exists z.(onenorth(x,z) \wedge onenorth(z,y)) \vee \\ \exists zw.(onenorth(x,z) \wedge onenorth(z,w) \wedge onenorth(w,y)) \end{pmatrix}$$
$$onewest(x,y) \Leftrightarrow \begin{pmatrix} (x=a \wedge y=b) \vee (x=b \wedge y=c) \vee (x=c \wedge y=d) \vee \\ (x=e \wedge y=f) \vee (x=f \wedge y=g) \vee \ldots \end{pmatrix}$$
$$onenorth(x,y) \Leftrightarrow \begin{pmatrix} (x=a \wedge y=e) \vee (x=e \wedge y=i) \vee (x=i \wedge y=m) \vee \\ (x=b \wedge y=f) \vee (x=f \wedge y=j) \vee \ldots \end{pmatrix}$$

Translating this set of biconditionals into a database system requires linear time. We compared the time required to answer the query $westof(a,p)$ using the axioms as written above and Prover9[1], Otter's successor, with the database version and a homegrown implementation of top-down $datalog^{\neg}$ [5]. Prover9 found a proof in about 300 seconds, and the $datalog^{\neg}$ implementation found an answer in 0.017 seconds (including translation), a speedup factor of 10^4. We also compared times for the query $westof(a,m)$, which is not entailed. Prover9 ran for over 30 minutes without finishing, and the $datalog^{\neg}$ implementation finished in 0.022 seconds, a speedup factor of at least 10^5.

Answering the database version of the query can sometimes be much faster because the knowledge captured in the classical logic sentences has been organized into a special form that is amenable to processing. In a database system, every predicate corresponds to exactly one table, and the system differentiates between the explicit (extensional) predicates and the implicit (intensional) predicates. Extensionally defined tables can be reasoned about very efficiently because of smart indexing; intensionally defined tables can be reasoned about efficiently because negation as failure is used.

[1] http://www.cs.unm.edu/~mccune/prover9/

Thus, using the techniques introduced in [1], complete theories can be reasoned about very efficiently using database systems. However, if one were to add even just a small amount of incompleteness into a complete theory by extending the theory with new predicates, those techniques could no longer be applied. This is unfortunate since the speedups in the complete case seem to be large enough to absorb some extra overhead for dealing with incomplete theories.

For example, suppose we add to the theory above the following set of sentences.

$$gold(x) \wedge gold(y) \Rightarrow x = y$$
$$gold(a) \vee gold(f) \vee gold(i)$$
$$wumpus(x) \wedge wumpus(y) \Rightarrow x = y$$
$$wumpus(h) \vee wumpus(k) \vee wumpus(p)$$

Together these axioms describe the snapshot of Wumpus World shown in Fig. 1, where an agent has visited two locations. Recall that the Wumpus World contains a hidden pile of gold, the object that is sought, and a hidden wumpus, the beast to be avoided. The percepts sensed in the western cell indicate the gold is in cell a, f, or i, and the percepts in the eastern cell indicate the wumpus is in cell h, k, or p. Consider then the entailment query, "Is the gold situated to the west of the wumpus?"

$$\forall xy.(gold(x) \wedge wumpus(y) \Rightarrow westof(x, y))$$

Clearly the answer is yes since each of the possible gold locations (a, f, and i) is situated to the west of all the possible wumpus locations (h, k, and p).

To answer this query using Extensional Reasoning techniques, we must construct a complete theory that faithfully represents the original. Our approach is to construct complete definitions for new predicates that represent all the possible values for the incomplete predicates. The entailment query is rewritten so that it is expressed in terms of these new predicates.

Consider just the sentences involving *gold*.

$$gold(x) \wedge gold(y) \Rightarrow x = y$$
$$gold(a) \vee gold(f) \vee gold(i)$$

We construct a predicate, $poss_{gold(x)}(x)$ that represents all the gold tuples that are consistent with the *gold* sentences above. In this case, $gold(a)$, $gold(f)$, and $gold(i)$ are the only consistent *gold* atoms.

$$poss_{gold(x)}(x) \Leftrightarrow (x = a \vee x = f \vee x = i)$$

Similar reasoning concludes that $poss_{wumpus(x)}$ is true of h, k, and p.

$$poss_{wumpus(y)}(y) \Leftrightarrow (y = h \vee y = k \vee y = p)$$

The original entailment query, "Is the gold situated to the west of the wumpus?" is written in terms of $poss_{gold(x)}$ and $poss_{wumpus(y)}$.

$$\forall xy.(poss_{gold(x)}(x) \wedge poss_{wumpus(y)}(y) \Rightarrow westof(x, y))$$

Because this query is about a complete theory (the biconditional definitions for $westof$, and the two biconditionals for $poss_{gold(x)}$ and $poss_{wumpus(x)}$), Extensional Reasoning techniques can be applied to answer the query using a database system. We compared the time required to answer the query using the $westof$, $gold$, and $wumpus$ axioms and Prover9 with the $westof$, $poss_{gold(x)}$, and $poss_{wumpus(y)}$ axioms and our $datalog^{\neg}$ implementation. Prover9 found a proof in about 350 seconds, and the $datalog^{\neg}$ implementation found an answer in 0.050 seconds, a speedup factor of 10^3. For the query, "Is the wumpus situated to the west of the gold?", which is not entailed, Prover9 ran for over 35 minutes without finishing, and the $datalog^{\neg}$ implementation finished in 0.020 seconds, a speedup factor of at least 10^5.

This transformation of an incomplete theory to a complete $poss$ theory can be applied in general settings having nothing to do with the Wumpus World. This paper introduces techniques for performing that transformation, using the Wumpus World to illustrate throughout.

4 Reformulating the Query

Satisfiability has long been used in refutational theorem proving to answer logical entailment queries: $\Delta \models \phi$ if and only if $\Delta \cup \{\neg\phi\}$ is unsatisfiable. Satisfiability is usually leveraged solely at the meta-level by people who are trying to justify the soundness and completeness of proof systems. In contrast, the techniques presented here use satisfiability within the logical theory, resulting in proof systems reasoning about satisfiability directly. The central activities required for this approach are the construction and manipulation of definitions for new predicates, each of which represents the satisfiability of a particular sentence in combination with a background theory. $poss_{\phi}$, a predicate representing the satisfiability of sentence ϕ, is true if and only if ϕ is consistent with the theory Δ. When $\phi(\overline{x})$ has n free variables $\overline{x}$, its $poss$ predicate takes n arguments and is true of $\overline{t}$ exactly when $\phi(\overline{t})$ is consistent with Δ.[2]

For example, suppose we wanted to know whether the following sentence were consistent with the Wumpus World theory illustrated in Sect. 3.

$$\exists xy.(gold(x) \wedge wumpus(y) \wedge \neg westof(x, y))$$

In the Extensional Reasoning approach, we would first construct a definition for the 0-ary predicate

$$poss_{\exists xy.(gold(x) \wedge wumpus(y) \wedge \neg westof(x,y))}$$

then convert that definition into a database system, and finally determine satisfiability by posing the database query $poss_{\exists xy.(gold(x) \wedge wumpus(y) \wedge \neg westof(x,y))}$.

[2] In this paper, instead of writing out n variables as $x_1, \ldots, x_n$, we will write $\overline{x}$; likewise terms $t_1, \ldots, t_n$ will be written $\overline{t}$. Sometimes a predicate will take some number of variables that we will want to group together; we will use similar shorthand. For example, $p(x_1, \ldots, x_n, y_1, \ldots, y_m)$ will be written $p(\overline{x}, \overline{y})$.

This particular satisfiability query is relevant to the Wumpus World example from the last section because its negation is the entailment query posed there, "Is the gold situated to the west of the wumpus?" Thus, if the above query is inconsistent with the premises then the entailment query must be true. Because the definition for $poss_\phi$ is always complete, the above query is inconsistent if and only if the following is true.

$$\neg poss_{\exists xy.(gold(x) \wedge wumpus(y) \wedge \neg westof(x,y))}$$

More generally, $\Delta \models \phi$ if and only if $\neg poss_{\neg\phi}$ is true. This conversion of an entailment query into a *poss* query is thus straightforward and can be applied to every query, but the construction of *poss* definitions is more complicated. The remainder of this section demonstrates how to simplify a *poss* query to make constructing its definition easier.

Formally, the algorithm that constructs *poss* definitions, which we call *poss*-CONSTRUCTION, must abide by the following constraints.

Definition 4 (*poss*-construction). *poss*-CONSTRUCTION$[\phi(\overline{x}), \Delta]$ *is an operation on a sentence* $\phi(\overline{x})$ *with free variables* $\overline{x}$ *and a set of sentences* Δ *that produces a typed, complete definition for* $poss_{\phi(\overline{x})}(\overline{x})$ *such that* $poss_{\phi(\overline{x})}(\overline{t})$ *is true if and only* $\phi(\overline{t})$ *is consistent with* Δ. $poss_{\phi(\overline{x})}(\overline{x})$ *is typed so that it only applies to object constants in* Δ.

Note that $poss_\phi$ and $poss_\psi$ are logically equivalent if ϕ and ψ are logically equivalent. This is the property of *poss* discussed earlier that allows us to equate the two predicates shown below.

$$\neg poss_{\neg\forall xy.(gold(x) \wedge wumpus(y) \Rightarrow westof(x,y))}$$
$$\Leftrightarrow \neg poss_{\exists xy.(gold(x) \wedge wumpus(y) \wedge \neg westof(x,y))}$$

As a rule of thumb, the definition for $poss_\phi$ is easier to construct the shorter the sentence ϕ. Fortunately, $poss_\phi$ can sometimes be broken up into predicates $poss_{\phi_1}, \ldots, poss_{\phi_n}$ so that ϕ_i is shorter than ϕ for every i. For example, it turns out that *poss* can be distributed across existential quantifiers: $poss_{\exists x.\phi(x)}$ is equivalent to $\exists x.poss_{\phi(x)}(x)$.

Theorem 2 (*poss* distributes over $\exists$). *In FHL,* $poss_{\exists x.\phi(x,\overline{y})}(\overline{y})$ *is logically equivalent to* $\exists x.poss_{\phi(x,\overline{y})}(x, \overline{y})$.

Proof. ($\Rightarrow$) Suppose $poss_{\exists x.\phi(x,\overline{y})}(\overline{t})$ is true. Then there is some model M that satisfies $\exists x.\phi(x, \overline{t})$, which ensures there is some object constant u such that $\models_M \phi(u, \overline{t})$ since Δ is in FHL. By the definition of *poss*, $poss_{\phi(x,\overline{y})}(u, \overline{t})$ must therefore be true, which implies $\exists x.poss_{\phi(x,\overline{y})}(x, \overline{t})$ must be true.

($\Leftarrow$) Suppose $\exists x.poss_{\phi(x,\overline{y})}(x, \overline{t})$ is true. Then because of FHL semantics and the completeness of *poss* definitions, there is some u such that $poss_{\phi(x,\overline{y})}(u, \overline{t})$ is true. By the definition of *poss*, there must be some model that satisfies $\phi(u, \overline{t})$. That same model certainly satisfies $\exists x.\phi(x, \overline{t})$, and serves as a witness for the fact that $poss_{\exists x.\phi(x,\overline{y})}(\overline{t})$ is true.

Because we have shown equivalence for all instances of the theorem, by Herbrand semantics, we have proven the theorem. □

Continuing the Wumpus World example, we can distribute *poss* across the existential quantifier.

$$\neg poss_{\exists xy.(gold(x) \wedge wumpus(y) \wedge \neg westof(x,y))}$$
$$\Leftrightarrow \neg \exists xy.poss_{gold(x) \wedge wumpus(y) \wedge \neg westof(x,y)}(x, y)$$

Because *poss* distributes across existential quantifiers, it is not surprising that it distributes over disjunction because in FHL the two are interconvertible.

Theorem 3 (*poss* distributes over $\vee$). *In FHL,* $poss_{\phi(\overline{x},\overline{y}) \vee \psi(\overline{x},\overline{z})}(\overline{x}, \overline{y}, \overline{z})$ *is logically equivalent to* $poss_{\phi(\overline{x},\overline{y})}(\overline{x}, \overline{y}) \vee poss_{\psi(\overline{x},\overline{z})}(\overline{x}, \overline{z})$.

Proof. ($\Rightarrow$) Suppose $poss_{\phi(\overline{x},\overline{y}) \vee \psi(\overline{x},\overline{z})}(\overline{t_1}, \overline{t_2}, \overline{t_3})$ is true. Then there is some model M that satisfies $\phi(\overline{t_1}, \overline{t_2}) \vee \psi(\overline{t_1}, \overline{t_3})$. By the definition of disjunction, M must therefore satisfy either disjunct or both disjuncts. If the model satisfies only one of the disjuncts, then certainly *poss* for that disjunct is true, ensuring the disjunction of the two *poss* statements is true. If the model satisfies both, then the conjunction of the two *poss* statements is true, which ensures their disjunction is also true. In either case, $poss_{\phi(\overline{x},\overline{y})}(\overline{t_1}, \overline{t_2}) \vee poss_{\psi(\overline{x},\overline{z})}(\overline{t_1}, \overline{t_3})$ is true.

($\Leftarrow$) Suppose $poss_{\phi(\overline{x},\overline{y})}(\overline{t_1}, \overline{t_2}) \vee poss_{\psi(\overline{x},\overline{z})}(\overline{t_1}, \overline{t_3})$ is true. Then either there is a model that satisfies $\phi(\overline{t_1}, \overline{t_2})$ or there is a model that satisfies $\psi(\overline{t_1}, \overline{t_3})$ or there is a model that satisfies both. In the first case, since the model satisfies $\phi(\overline{t_1}, \overline{t_2})$, it must satisfy every disjunction including it as a disjunct. That model must therefore satisfy $\phi(\overline{t_1}, \overline{t_2}) \vee \psi(\overline{t_1}, \overline{t_3})$. Likewise for the second case. The third case guarantees the existence of a model that satisfies the conjunction and therefore the disjunction. Therefore, in all three cases, we know that there is a model that satisfies $\phi(\overline{t_1}, \overline{t_2}) \vee \psi(\overline{t_1}, \overline{t_3})$, which means $poss_{\phi(\overline{x},\overline{y}) \vee \psi(\overline{x},\overline{z})}(\overline{t_1}, \overline{t_2}, \overline{t_3})$ is true.

Because we have shown equivalence for all instances of the theorem, by Herbrand semantics, we have proven the theorem. □

It turns out that *poss* does not always distribute over universal quantifiers, conjunction, or negation. If *poss* did distribute over negation, it would also distribute over universal quantifiers and conjunction; hence, the counterexample of negation is the most important. Consider the propositional sentence p in a theory where both p and $\neg p$ are consistent. In this theory both $poss_{\neg p}$ and $poss_p$ are true. If *poss* could be distributed over $\neg$, then $poss_{\neg p}$ implies $\neg poss_p$, which contradicts the fact that $poss_p$ is true.

Nevertheless, when comparing the Wumpus World *poss* query above with a logically equivalent rewriting of the query from Sect. 3, shown below, we see that *poss* can be distributed over conjunction in certain circumstances.

$$\neg \exists xy.(poss_{gold(x)}(x) \wedge poss_{wumpus(y)}(y) \wedge \neg westof(x, y))$$

Intuitively, the possible locations for the gold are entirely independent of the possible locations for the wumpus. This intuition is materialized syntactically by the fact that the sentences constraining *gold* never mention *wumpus*, and the sentences constraining *wumpus* never mention *gold*. Thus if $gold(t)$ were

satisfied by some model and $wumpus(u)$ were satisfied by some model, there must be some model that satisfies $gold(t) \wedge wumpus(u)$. We say that two such sentences are *independent*.

Definition 5 (Sentence Independence). *Let Δ be a sentence set in FHL and $\phi(\overline{x}, \overline{z})$ and $\psi(\overline{y}, \overline{z})$ be sentences whose vocabularies are subsets of the vocabulary of Δ that share the variables $\overline{z}$. $\phi(\overline{x}, \overline{z})$ and $\psi(\overline{y}, \overline{z})$ are independent with respect to Δ if and only if for every tuple of object constants $\overline{t}$, $\overline{u}$, and $\overline{v}$, if $\phi(\overline{t}, \overline{v})$ is consistent with Δ and $\psi(\overline{u}, \overline{v})$ is consistent with Δ then $\phi(\overline{t}, \overline{v}) \wedge \psi(\overline{u}, \overline{v})$ is consistent with Δ.*

Whenever two sentences are independent, *poss* distributes over their conjunction.

Theorem 4 (*poss* Distributes over Conjunctions of Independent Sentences). *In FHL, suppose $\phi(\overline{x}, \overline{z})$ and $\psi(\overline{y}, \overline{z})$ are independent with respect to Δ. Then $poss_{\phi(\overline{x},\overline{z}) \wedge \psi(\overline{y},\overline{z})}(\overline{x}, \overline{y}, \overline{z})$ is equivalent to $poss_{\phi(\overline{x},\overline{z})}(\overline{x}, \overline{z}) \wedge poss_{\psi(\overline{y},\overline{z})}(\overline{y}, \overline{z})$.*

Proof. ($\Rightarrow$) Suppose $poss_{\phi(\overline{x},\overline{z}) \wedge \psi(\overline{y},\overline{z})}(\overline{t}, \overline{u}, \overline{v})$ is true. Then there is some model that satisfies $\phi(\overline{t}, \overline{v}) \wedge \psi(\overline{u}, \overline{v})$. That same model must then also satisfy $\phi(\overline{t}, \overline{v})$ and $\psi(\overline{u}, \overline{v})$ individually, ensuring $poss_{\phi(\overline{x},\overline{z})}(\overline{t}, \overline{v})$ and $poss_{\psi(\overline{y},\overline{z})}(\overline{u}, \overline{v})$ are both true. Thus, their conjunction is true.

($\Leftarrow$) Suppose $poss_{\phi(\overline{x},\overline{z})}(\overline{t}, \overline{v}) \wedge poss_{\psi(\overline{y},\overline{z})}(\overline{u}, \overline{v})$ is true. Then there must be some model that satisfies $\phi(\overline{t}, \overline{v})$ and some model that satisfies $\psi(\overline{u}, \overline{v})$. By the independence of $\phi(\overline{x}, \overline{z})$ and $\psi(\overline{y}, \overline{z})$, since $\phi(\overline{t}, \overline{v})$ is consistent with Δ and $\psi(\overline{u}, \overline{v})$ is consistent with Δ, $\phi(\overline{t}, \overline{v}) \wedge \psi(\overline{u}, \overline{v})$ is consistent with Δ. Hence,
$poss_{\phi(\overline{x},\overline{z}) \wedge \psi(\overline{y},\overline{z})}(\overline{t}, \overline{u}, \overline{v})$ is true.

Because we have shown equivalence for all instances of the theorem, by Herbrand semantics, we have proven the theorem. □

In light of the last theorem, independence is an important property. One sufficient condition for determining whether $p(\overline{x}, \overline{z})$ and $q(\overline{y}, \overline{z})$ are independent is syntactic: if the constraints on two predicates never (transitively) mention the other predicate (where predicates with complete definitions can be ignored) then the two predicates are independent. Formally, this condition can be defined by examining the dependency graph of a sentence set.

Definition 6 (Undirected Dependency Graph). *An Undirected Dependency Graph for a set of sentences Δ is a graph $\langle V, E \rangle$.*

- *$u \in V$ if and only if u is a predicate in Δ without a complete definition*
- *The undirected edge (u, v) is in E if and only if there is some sentence in Δ that uses both the predicates u and v.*

Theorem 5 (Syntactic Independence). *Let Δ be a satisfiable set of FHL sentences, and let G be its Undirected Dependency Graph. Suppose that in G each of the predicates used in $\phi(\overline{x}, \overline{z})$ is in a different connected component than every predicate in $\psi(\overline{y}, \overline{z})$. Then $\phi(\overline{x}, \overline{z})$ and $\psi(\overline{y}, \overline{z})$ are independent with respect to Δ.*

Connected components can be computed in time linear in the size of the Undirected Dependency Graph, and this graph is linear in the size of Δ.[3] Given the components, checking syntactic independence of two sentences is linear in their size.

Going back to the Wumpus World example, syntactic independence guarantees that *gold*, *wumpus*, and *westof* are all independent. This pushes the simplification one step further than before.

$$\neg\exists xy.poss_{gold(x)\wedge wumpus(y)\wedge\neg westof(x,y)}(x,y)$$
$$\Leftrightarrow \neg\exists xy.(poss_{gold(x)}(x) \wedge poss_{wumpus(y)}(y) \wedge poss_{\neg westof(x,y)}(x,y))$$

The final step requires one more theorem. *westof* is a predicate with a complete definition. Just as independence affects the properties of *poss*, completeness affects those properties as well, and to an even greater extent. (1) A sentence that is complete is independent of every other sentence. (2) $poss_\phi$ is logically equivalent to ϕ when ϕ is complete.

Theorem 6 (Completeness Guarantees Independence). *Let Δ be a satisfiable set of FHL sentences complete with respect to $\phi(\overline{x})$, i.e. for every tuple of object constants $\bar{t}$, $\Delta \models \phi(\bar{t})$ or $\Delta \models \neg\phi(\bar{t})$. Then $\phi(\overline{x})$ is independent of every other sentence ψ with respect to Δ.*

Proof. Suppose that both $\phi(\bar{t})$ and ψ are individually consistent with Δ. Then there must be a model M that satisfies $\phi(\bar{t})$ and a model N that satisfies ψ. Because Δ entails either $\phi(\bar{t})$ or $\neg\phi(\bar{t})$ and M satisfies $\phi(\bar{t})$, Δ must entail $\phi(\bar{t})$ (for M is a countermodel to $\Delta \models \neg\phi(\bar{t})$). Therefore, every model that satisfies Δ must satisfy $\phi(\bar{t})$, including N. Hence, N must satisfy both $\phi(\bar{t})$ and ψ. Since the argument was made for an arbitrary $\bar{t}$, it holds for all $\bar{t}$, and by Herbrand semantics, $\phi(\overline{x})$ and ψ must therefore be independent wrt Δ. □

Theorem 7 (Completeness and *poss* Redundancy). *Let Δ be a satisfiable set of FHL sentences complete with respect to $\phi(\overline{x})$, i.e. for all tuples $\bar{t}$ of object constants in Δ, Δ entails $\phi(\bar{t})$ or Δ entails $\neg\phi(\bar{t})$. Then*

$$\Delta \cup \{poss\text{-}\textsc{construction}[\phi(\overline{x}), \Delta]\} \models poss_{\phi(\overline{x})}(\overline{x}) \Leftrightarrow \phi(\overline{x})$$

Proof. ($\Rightarrow$) Suppose $poss_{\phi(\overline{x})}(\bar{t})$ is true. Then there is some model M that satisfies $\phi(\bar{t})$. Because Δ entails either $\phi(\bar{t})$ or its negation, Δ must entail $\phi(\bar{t})$ since in the other case M is a countermodel.

($\Leftarrow$) Suppose $\Delta \models \phi(\bar{t})$. Then because Δ is satisfiable, there is some model that satisfies Δ and, because entailment holds, that model satisfies $\phi(\bar{t})$. Thus, $poss_{\phi(\overline{x})}(\bar{t})$ is true.

Because we have shown equivalence for all instances of the theorem, by Herbrand semantics, we have proven the theorem. □

[3] Technically, an Undirected Dependency Graph might require a quadratic number of edges but there is a linear graph sufficient for the purpose of computing connected components.

Applying these two theorems to simplify a *poss* query requires knowing which predicates have complete definitions in the theory. The algorithm investigated in [1] detects whether an entire theory is complete using a variation of the well-known CFG algorithm for checking emptiness. A simple variant of that procedure is an anytime algorithm that attempts to find the predicates in a theory that have complete definitions and when run to completion costs low-order polynomial time in the size of the theory.

In the context of Wumpus World, the results above guarantee that (1) $westof$ is independent of every other sentence and (2) $poss_{\neg westof(x,y)}(x,y)$ is logically equivalent to $\neg westof(x,y)$. Using all the theorems in this section, we see the following sequence of *poss* simplifications of the Wumpus World query.

$\Delta \models \forall xy.(gold(x) \wedge wumpus(y) \Rightarrow westof(x,y))$
$\Leftrightarrow \neg poss_{\neg\forall xy.(gold(x)\wedge wumpus(y)\Rightarrow westof(x,y))}$ (*poss* Semantics)
$\Leftrightarrow \neg poss_{\exists xy.(gold(x)\wedge wumpus(y)\wedge\neg westof(x,y))}$ (Log. Equiv.)
$\Leftrightarrow \neg\exists xy.poss_{gold(x)\wedge wumpus(y)\wedge\neg westof(x,y)}(x,y)$ (Distr. over $\exists$)
$\Leftrightarrow \neg\exists xy.(poss_{gold(x)}(x) \wedge poss_{wumpus(y)}(y) \wedge poss_{\neg westof(x,y)}(x,y))$ (Distr. over $\wedge$)
$\Leftrightarrow \neg\exists xy.(poss_{gold(x)}(x) \wedge poss_{wumpus(y)}(y) \wedge \neg westof(x,y))$ (*poss* Redundancy)
$\Leftrightarrow \forall xy.(poss_{gold(x)}(x) \wedge poss_{wumpus(y)}(y) \Rightarrow westof(x,y))$ (Log. Equiv.)

As demonstrated in this section, any logical entailment query can be written as a *poss* query, and a *poss* query can sometimes be simplified by distributing *poss* over logical connectives, thereby reducing the complexity of constructing *poss* definitions. With the exception of the completeness computation, the simplifications take time linear in the size of the *poss* sentence after spending time linear in the size of the theory to compute which predicates are independent.

5 Completing the Theory

The previous section demonstrated how to transform an entailment query into a *poss* query and enumerated some of the properties of *poss* that allow such a query to be simplified. That simplification is important because it makes constructing definitions for *poss* more efficient. Constructing *poss* definitions is a form of theory-completion because every proper *poss* definition is complete: every sentence is either consistent with a background theory or it is inconsistent with that theory. This section demonstrates how to construct *poss* definitions for ground clauses, where the cost of reformulation is quadratic.

Consider a set of ground clauses that constrain a single predicate p, where we use $[\neg]p(\ldots)$ to indicate that the literal may or may not be negated.

$$\begin{array}{c} [\neg]p(\overline{t_{11}}) \vee \ldots \vee [\neg]p(\overline{t_{1n_1}}) \\ \vdots \\ [\neg]p(\overline{t_{m1}}) \vee \ldots \vee [\neg]p(\overline{t_{mn_m}}) \end{array}$$

Suppose we want to construct a definition for $poss_{p(\overline{x})}(\overline{x})$, i.e. a sentence $\phi(\overline{x})$ with free variables $\overline{x}$ such that $\phi(\overline{t})$ is true if and only if $p(\overline{t})$ is consistent with

the clauses above. First notice that this set of disjunctions is satisfiable if and only if we can select one literal from each disjunction so that a literal and its negation are not both selected. $p(\bar{t})$ is satisfiable and therefore $poss_{p(\overline{x})}(\bar{t})$ is true if and only if there is a satisfying selection of literals that does not include $\neg p(\bar{t})$.

To construct a definition for $poss_{p(x)}(x)$, we take advantage of these observations by writing logical sentences that check whether there is a satisfying selection of literals that does not include $\neg p(\bar{t})$. Because that check requires analyzing the structure of the disjunctions at the metalevel, we reify the structure so that analysis can take place within the logic. For each disjunction $[\neg]p(\overline{t_{i1}}) \vee \ldots \vee [\neg]p(\overline{t_{in_i}})$ we invent a new object constant, conveniently spelled $\pm\overline{t_{i1}} \ldots \pm \overline{t_{in_i}}$ where the $\pm$ means that a $+$ is used if the literal is positive and a $-$ is used if the literal is negative. In addition, we impose an ordering on the disjunctions. The new binary predicate d is true of $\langle k, c\rangle$ if and only if c is the object constant representing the kth disjunction.

$$d(x,y) \Leftrightarrow \begin{pmatrix} (x = 1 \wedge y = \pm\overline{t_{11}} \ldots \pm \overline{t_{1n_1}}) \\ \vee \ldots \vee \\ (x = m \wedge y = \pm\overline{t_{m1}} \ldots \pm \overline{t_{mn_m}}) \end{pmatrix}$$

This reification technique encodes incomplete information within a complete definition. To extract that incomplete information, we use *part*, which also has a complete definition. *part* is a ternary predicate such that $part(\pm\overline{t_{i1}} \ldots \pm\overline{t_{in_i}}, x, y)$ is true whenever y is one of the $\overline{t_{ij}}$ and x is its corresponding sign, i.e. + or -.

Using d and *part*, we can construct a definition for all consistent selections of literals from the m disjunctions, where one literal is selected per disjunction. The definition of sat_1 is naturally recursive. Suppose we had a selection of literals from the last $m-1$ disjunctions. Then to select a literal from the remaining disjunction requires simply finding a literal that is consistent with the selection so far. The definition for selection from the last $m-i$ disjunctions depends on a definition for selecting from the last $m-(i+1)$ disjunctions. Often such a definition is written recursively, but because at the time this definition is written, the number of disjunctions is known, we can instead unroll the recursion and write m definitions. This unrolling is important because when the definitions are nonrecursive, they comprise a complete theory, which is the goal of this reformulation.

$$sat_1(x_1, \overline{y_1}, \ldots, x_m, \overline{y_m}) \Leftrightarrow \begin{pmatrix} \exists z.(d(1,z) \wedge part(z, x_1, \overline{y_1})) \wedge \\ sat_2(x_2, \overline{y_2}, \ldots, x_m, \overline{y_m}) \wedge \\ (\overline{y_1} = \overline{y_2} \Rightarrow x_1 = x_2) \wedge \\ (\overline{y_1} = \overline{y_3} \Rightarrow x_1 = x_3) \\ \wedge \ldots \wedge \\ (\overline{y_1} = \overline{y_m} \Rightarrow x_1 = x_m) \end{pmatrix}$$

$$\vdots$$

$$sat_m(x_m, \overline{y_m}) \Leftrightarrow \exists z.(d(m,z) \wedge part(z, x_m, \overline{y_m}))$$

Finally we need a definition for $poss_{p(\overline{x})}(\overline{x})$ that is true whenever there is some sat_1 where none of the arguments are $\overline{x}$ with sign $-$.

$$poss_{p(\overline{x})}(\overline{x}) \Leftrightarrow \exists x_1, \overline{y_1}, \ldots, x_m, \overline{y_m}. \begin{pmatrix} sat_1(x_1, \overline{y_1}, \ldots, x_m, \overline{y_m}) \wedge \\ \neg(x_1 = - \wedge \overline{y_1} = \overline{x}) \\ \wedge \ldots \wedge \\ \neg(x_m = - \wedge \overline{y_m} = \overline{x}) \end{pmatrix}$$

A constraint that says p may be true of no more than n tuples can be handled by checking whether a satisfying selection of literals contains more than n positive literals or not.

To illustrate, we use the Wumpus World, which includes ground clauses and size constraints for *gold* and *wumpus*. Consider just the *gold* sentences.

$$gold(a) \vee gold(f) \vee gold(i)$$
$$gold(x) \wedge gold(y) \Rightarrow x = y$$

Because there is a single ground disjunction, the definitions for d and *part* are particularly simple.

$$d(x, y) \Leftrightarrow (x = 1 \wedge y = +a + f + i)$$

$$part(x, y, z) \Leftrightarrow \begin{pmatrix} (x = +a + f + i \wedge y = + \wedge z = a) \vee \\ (x = +a + f + i \wedge y = + \wedge z = f) \vee \\ (x = +a + f + i \wedge y = + \wedge z = i) \end{pmatrix}$$

Because there is one ground clause, there is just one *sat* predicate.

$$sat_1(x_1, y_1) \Leftrightarrow \exists z.(d(1, z) \wedge part(z, x_1, y_1))$$

The definition for $poss_{gold(x)}(x)$ is shown here, and takes into account the fact that there can be only one value for *gold*.

$$poss_{gold(x)}(x) \Leftrightarrow \exists x_1 y_1. \begin{pmatrix} sat_1(x_1, y_1) \wedge \\ \neg(x_1 = - \wedge y_1 = x) \wedge \\ \neg(x_1 = + \wedge y_1 \neq x) \end{pmatrix}$$

In general, the definitions prescribed in this section are fairly efficient to compute. The definitions for d and *part* cost time linear in the size of the clause set. The *sat* definitions altogether cost time quadratic in the number of clauses, and the cost of computing $poss_{p(\overline{x})}(x)$ is linear in the number of clauses.

This *poss* definition construction can be generalized to handle ground clauses that constrain multiple predicates by including information in the new object constants about not only the sign of the tuple but also the predicate it belongs to, and the same complexity analysis holds.

6 Conclusion and Future Work

Extensional Reasoning is a promising new approach to automated theorem proving that leverages one of the most successful applications of logic today: the relational database. The two key insights discussed in this paper are (1) how *poss* can be used to apply Extensional Reasoning to entailment queries about incomplete theories and (2) how *poss* definitions can be constructed efficiently by creating new object constants that represent syntactic features of an axiom set. By reformulating a theory into a complete theory about *poss*, and rewriting the query in terms of *poss*, ER techniques developed for complete theories can sometimes be leveraged to answer entailment queries orders of magnitude more quickly than traditional techniques.

The first and most obvious extension to the work presented here is an expansion of the class of theories that can be completed using *poss*. Ground clauses form a good starting point but are less compelling than we would like. We have experimented with computing *poss* definitions via abduction, but because our abduction algorithms relied on traditional proof techniques, run-times were unpredictable, and for some problems a great deal of time was spent on the reformulation step. It is unclear at this point whether there are better techniques for computing *poss* definitions in general.

The laws for distributing *poss* across logical connectives are relatively complete, but further conditions that identify independent sentences may turn out to be important. Such conditions would enlarge the class of cases for which *poss* distributes over conjunction, thereby breaking the *poss* definition computation up into smaller pieces.

Finally, there is the possibility that we could turn this work upside down. Instead of viewing the task as using databases to speed-up theorem proving, we could also view it as expanding traditional database techniques to allow for disjunctive data and more flexible query languages. (We are currently analyzing how this work relates to the work on incomplete databases, e.g. [5, 6].)

References

1. Hinrichs, T.L., Genesereth, M.R.: Extensional reasoning. In preparation: http://logic.stanford.edu/~thinrich/papers/hinrichs2007extensional.pdf (2007)
2. Reiter, R.: On closed world data bases. Logic and Databases (1978)
3. Lloyd, J.: Foundations of Logic Programming. Springer Verlag (1984)
4. McCarthy, J.: Circumscription - a form of non-monotonic reasoning. Artificial Intelligence (1980)
5. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
6. Zimanyi, E., Pirotte, A.: Imperfect knowledge in databases. In Motro, A., Smets, P., eds.: Uncertainty Management in Information Systems: From Needs to Solutions. Kluwer Academic Publishers (1997) 35–87