Tractable Reasoning in First-Order Knowledge Bases with Disjunctive Information

Yongmei Liu and Hector J. Levesque
Department of Computer Science
University of Toronto
Toronto, ON, Canada M5S 3G4
{yliu, hector}@cs.toronto.edu

Abstract
This work proposes a new methodology for establishing the tractability of a reasoning service that deals with expressive first-order knowledge bases. It consists of defining a logic that is weaker than classical logic and that has two properties: first, the entailment problem can be reduced to the model checking problem for a small number of characteristic models; and second, the model checking problem itself is tractable for formulas with a bounded number of variables. We show this methodology in action for the reasoning service previously proposed by Liu, Lakemeyer and Levesque for dealing with disjunctive information. They show that their reasoning is tractable in the propositional case and decidable in the first-order case. Here we apply the methodology and prove that the reasoning is also tractable in the first-order case if the knowledge base and the query both use a bounded number of variables.

Introduction
In the area of Knowledge Representation and Reasoning, there is a well-known tradeoff between the expressiveness of the representation language and the computational tractability of the associated reasoning task. On the one hand, it is well accepted that a general-purpose representation language needs at least the expressiveness of first-order logic; on the other, the logical entailment problem of first-order logic is undecidable. Over the past decades, two main techniques have been proposed to deal with this computational intractability problem. The first is language restriction, as represented, for example, by description logics; the idea is to restrict the expressiveness of the representation language, and especially the types of incomplete knowledge that can be represented. The second is limited reasoning, as represented, for example, by the work on tautological entailment (Levesque 1984; Frisch 1987; Schaerf & Cadoli 1995; Patel-Schneider 1990; Lakemeyer 1994); the idea is to weaken the entailment relation in some way by introducing non-traditional semantics. An emerging direction of research is to combine these two approaches so as to obtain tractable reasoning for representation languages that are not overly restricted and for entailment relations that are not overly weak.

This line of research is initiated by Levesque (1998). He proposes a generalization of a database called a proper KB, which allows a limited form of incomplete knowledge. Since the deduction problem for proper KBs remains undecidable, he defines a reasoning procedure $V$ that is logically sound and sometimes complete. Liu and Levesque (2003) show that despite the incomplete knowledge, $V$ can be implemented efficiently using database techniques.

However, the expressiveness of proper KBs is still quite limited: knowledge may be incomplete, but no disjunctive information is allowed. So Lakemeyer and Levesque (2002) propose an extension to a proper KB called a proper$^+$ KB, which allows simple forms of disjunctive information. They also propose a reasoning procedure $X$ for proper$^+$ KBs that is logically sound and agrees with $V$ on proper KBs. However, the general logical and computational properties of this new reasoning scheme are left unanalyzed.

A popular way of specifying a limited reasoning service is through a logic of belief. Instead of proposing a new entailment relation, the idea is to model reasoning as belief implication, that is, validity of formulas of the form $(KB B_\phi)$, where $B$ is a belief operator. With the goal of specifying a reasoning service for first-order KBs with disjunctive information in the form of proper$^+$ KBs, Liu, Lakemeyer and Levesque (2004) propose a logic of limited belief called the subjective logic $\mathcal{SL}$. Reasoning based on $\mathcal{SL}$ is logically sound and sometimes complete. Given disjunctive information, it performs unit propagation, but only does case analysis in a limited way. Moreover, they show that $\mathcal{SL}$-based reasoning with proper$^+$ KBs is tractable in the propositional case and decidable in the first-order case. The main idea behind these results is that $\mathcal{SL}$-based reasoning with proper$^+$ KBs reduces to a certain model checking problem, which can be implemented by a slight variant of the procedure $X$.

In this paper, we continue this line of research and show that $\mathcal{SL}$-based reasoning with proper$^+$ KBs is not only decidable but also tractable in the first-order case if both the KB and the query use a bounded number of variables. This result is inspired by the tractability of the model checking problem for bounded-variable first-order logic. In this paper, we show that the model checking problem of $\mathcal{SL}$ is tractable for formulas with a bounded number of variables. Combining this result with the aforementioned result that reduces reasoning to model checking, we obtain the main result of this paper.
Bounded-Variable First-Order Logic

The main result of this paper is inspired by the tractability of database query evaluation for first-order formulas with a bounded number of variables. In this section, we briefly review this result, and discuss its significance.

The complexity of query evaluation has been one of the main pursuits of database theory. Vardi (1982) proposes two complexity measures for this: combined complexity and data complexity. Combined complexity is measured in terms of the combined size of the database and the query. Data complexity is measured solely in terms of the size of the database and the size of the query is treated as a constant. Vardi studies a number of logical languages and shows that combined complexity is usually one exponential higher than data complexity. For example, the combined complexity of first-order logic is PSPACE-complete, while its data complexity is in PTIME. Later Vardi (1995) shows that the exponential gap between combined and data complexity can be eliminated by restricting the queries to have a bounded number of variables. In particular, Vardi proves that the combined complexity of bounded-variable first-order logic is in PTIME. The basic idea of the proof is to view subformulas of the queries as subqueries. The evaluation is bottom-up and all intermediate results are bounded-arity relations.

We let FO\(^j\) denote the set of all first-order formulas with at most \(j\) distinct variables. The expressiveness of FO\(^j\) is not as limited as it may initially appear because we can reuse variables. For example, given a binary relation \(R\) standing for the edges in a graph, for any \(k\), the property “there is a path of length \(k\) from \(a\) to \(b\)” (where \(a\) and \(b\) are constants) is definable by a formula \(\phi_k(a, b)\) that uses only two variables \(x\) and \(y\). We define the formula \(\phi_i(u, v)\) for any terms \(u\) and \(v\) by induction:

\[
\begin{align*}
\phi_1(u, v) &= R(u, v); \\
\phi_2(u, v) &= \exists x [R(u, x) \land \phi_{2i-1}(x, v)], \text{ for } i \geq 1; \\
\phi_{2i+1}(u, v) &= \exists y [R(u, y) \land \phi_{2i}(y, v)], \text{ for } i \geq 1.
\end{align*}
\]

Then for any \(k\), \(\phi_k(a, b) \in \text{FO}^2\), and states that there is a path of length \(k\) from \(a\) to \(b\). In contrast, the property “there is a clique of size \(k\)” can only be expressed in FO\(^k\).

It turns out that FO\(^j\) provides a logical characterization for the combinatorial notion of bounded treewidth as follows. It is well-known that many algorithmic problems that are “hard” on arbitrary graphs become “easy” on trees. The concept of bounded treewidth generalizes the concept of tree while maintaining its good computational properties. Intuitively, the treewidth of a graph measures its similarity with a tree. This notion can be extended to the treewidth of a relational structure. Kolaitis and Vardi (2000) show that if a finite structure has treewidth less than \(j\), then a certain canonical formula for the structure is definable by an FO\(^j\)-formula that can be constructed in polynomial time.

Interestingly, their result also explains a well-known tractability result about constraint satisfaction problems (CSP). In general, CSP is NP-complete. But Dechter and Pearl (1989) and Freuder (1990) show that CSP with bounded-treewidth constraint graphs is in PTIME.\(^2\) Kolaitis and Vardi are able to confirm this property by showing that bounded-treewidth CSP reduces to database query evaluation for FO\(^j\).

The Subjective Logic \(SL\)

As observed by Lakemeyer and Levesque (2002), although disjunctions can be used in many ways in a common-sense KB, it has two major applications: (1) to represent rules such as in Horn clauses, where the associated reasoning is unit propagation; and (2) to represent incomplete knowledge about individuals, where the associated reasoning is case analysis. They argue that requiring a reasoning service to automatically deal with (2) with no further guidance is asking too much, since this implies the ability to solve combinatorial puzzles. Motivated by this observation, Liu, Lakemeyer and Levesque (2004) propose reasoning based on the subjective logic \(SL\), which supports full unit propagation for (1), but only a controlled form of case analysis for (2). In particular, they introduce a family of belief operators \(B_0, B_1, B_2, \ldots\), where \(B_k\) essentially supports case analysis of depth bounded by \(k\). To save space, later we will refer to (Liu, Lakemeyer, & Levesque 2004) by (LLL04).

The syntax

The language \(L\) is a standard first-order logic with equality. The language \(SL\) is a first-order logic with equality whose atomic formulas are belief atoms of the form \(B_k\phi\) where \(\phi\) is a formula of \(L\) and \(B_k\) is a modal operator for any \(k \geq 0\). \(B_k\phi\) is read as “\(\phi\) is a belief at level \(k\)”.

More precisely, we have the following inductive definitions. There are countably infinite sets of variables and constant symbols, which make up the terms of the language. The constants behave like standard names, and no other function symbols are allowed. The atoms are expressions of the form \(P(t_1, \ldots, t_m)\) where \(P\) is a predicate symbol (excluding equality) and the \(t_i\)’s are terms. The literals are atoms or their negations. We use \(\rho\) to range over literals, and we use \(\mathcal{P}\) to denote the complement of \(\rho\).

The language \(L\) is the least set of expressions such that

1. if \(\rho\) is an atom, then \(\rho \in L\);
2. if \(t\) and \(t’\) are terms, then \((t = t’) \in L\);
3. if \(\phi, \psi \in L\) and \(x\) is a variable, then \(\neg\phi, (\phi \lor \psi), \text{ and } \exists x \phi \in L\).

Clauses, which play an important role in the semantic definition of \(SL\), are inductively defined as follows:

1. a literal is a clause, and is called a unit clause;
2. if \(c\) and \(c’\) are clauses, then \((c \lor c’)\) is a clause.

A clause is identified with the set of literals it contains. Only non-empty clauses appear in \(L\). However, the empty clause, denoted by \(\Box\), is used in \(SL\).\(^3\)

\(^{\text{2}}\)See (Chen 2004) for a recent application of bounded-treewidth.

\(^{\text{3}}\)Unlike the other predicate symbols, equality is taken to have a fixed interpretation (the identity relation) and so behaves more like a logical symbol.
The language $\mathcal{S}L$ is the least set of expressions such that
1. if $\phi \in \mathcal{L}$ or $\phi$ is $\Box$, and $k \geq 0$, then $B_k \phi \in \mathcal{S}L$, and is called a belief atom of level $k$;
2. if $t$ and $t'$ are terms of $\mathcal{L}$, then $(t = t') \in \mathcal{S}L$;
3. if $\alpha, \beta \in \mathcal{S}L$ and $x$ is a variable, then $\neg \alpha$, $(\alpha \lor \beta)$, and $\exists x \alpha \in \mathcal{S}L$.

As usual, $(\alpha \land \beta)$, $(\alpha \implies \beta)$, and $\forall x \alpha$ are used as abbreviations; $\alpha^x_d$ is used to denote $\alpha$ with all free occurrences of $x$ replaced with constant $d$.

**Belief reductions**

Before presenting the semantics of $\mathcal{S}L$, we introduce some preparatory concepts.

When deciding if a sentence $\phi$ is believed, sometimes it is necessary to decide if related subformulas are believed. The notation $(B_k \phi)$ is used to denote this belief reduction. For any $\phi \in \mathcal{L}$, the $\mathcal{S}L$ formula $(B_k \phi)$ is defined as follows:

1. $(B_k c) = B_k c$, where $c$ is a clause;
2. $(B_k (t = t')) = (t = t')$;
3. $(B_k \neg (t = t')) = \neg (t = t')$;
4. $(B_k \neg \neg \phi) = B_k \phi$;
5. $(B_k (\phi \lor \psi)) = (B_k \phi \lor B_k \psi)$, where $\phi$ or $\psi$ is not a clause;
6. $(B_k \neg (\phi \lor \psi)) = (B_k \neg \phi \land B_k \neg \psi)$;
7. $(B_k \exists x \phi) = \exists x B_k \phi$;
8. $(B_k \neg \exists x \phi) = \forall x B_k \neg \phi$.

As mentioned earlier, $\mathcal{S}L$ supports unit propagation, which involves applying unit resolution to clauses until no new clauses are generated. Let $s$ be a set of ground clauses. The notation $UP(s)$ is used to denote the closure of $s$ under unit propagation, that is, the least set $s'$ satisfying:

1. if $c \in s$; and 2. if $\rho \in s'$ and $\{\Box\} \cup c \in s'$, then $c \in s'$.

The notation $VP(s)$ is used to denote the closed set:

Finally, there is a complexity measure $\| \cdot \|$ which maps formulas into natural numbers. It has the following property: for any $\phi$, $\| B_k \phi \| < \| B_{k+1} \phi \|$; and for any $\phi$ that is not a clause, $\| (B_k \phi) \| < \| B_k \phi \|$. We omit its definition here, but mention that this property ensures that the following semantics is well-defined.

**The semantics**

Sentences of $\mathcal{S}L$ are interpreted via a setup, which is a set of non-empty ground clauses, and which specifies what sentences of $\mathcal{L}$ are believed, and consequently what sentences of $\mathcal{S}L$ are true. Intuitively, a setup represents what is explicitly believed as a possibly infinite set of ground clauses. The semantics below then specifies what are the implicit beliefs.

Let $s$ be a setup. For any sentence $\alpha \in \mathcal{S}L$, $s \models \alpha$ (read "$s$ satisfies $\alpha$") is defined inductively on $|\alpha|$ as follows:

1. $s \models (d = d')$ iff $d$ and $d'$ are the same constant;
2. $s \models \neg \alpha$ iff $s \not\models \alpha$;
3. $s \models \alpha \lor \beta$ iff $s \models \alpha$ or $s \models \beta$;
4. $s \models \exists x \alpha$ iff for some constant $d$, $s \models \alpha^x_d$;
5. $s \models B_k \phi$ iff one of the following holds:
   (a) subsume: $k = 0$, $\phi$ is a clause, and $\phi \in VP(s)$;
   (b) reduce: $\phi$ is not a clause and $s \models (B_k \phi)$;
   (c) split: $k > 0$ and there is some $c \in s$ such that for all $\rho \in c$, $s \cup \{\rho\} \models B_{k-1} \phi$.

As usual, a sentence $\alpha \in \mathcal{S}L$ is valid, written $\models \alpha$, if for every setup $s$, we have that $s \models \alpha$.

As can be seen from the rules of interpretation above, negation and disjunction have their usual meaning in $\mathcal{S}L$. The rules for equality and quantification reflect our assumption that there is a bijection between the domain of discourse and the countably infinite set of constants. So all the novelty in $\mathcal{S}L$ is due to the interpretation of the $B_k$ operators. Intuitively, the rules propose three different justifications for believing a sentence $\phi$ at level $k$:

1. $\phi$ is a clause, $k = 0$, and after doing unit propagation on our explicit beliefs, we end up with a subclause of $\phi$;
2. we already have appropriate beliefs about the subformulas of $\phi$, for example, believing both conjunctions of a conjunction, or some instance of an existential;
3. there is a clause in our explicit beliefs that if we split, that is, if we augment our beliefs by a literal in that clause, then in all cases we end up believing $\phi$ at level $k - 1$.

All three of these deal with disjunction but in quite different ways, which we now illustrate with an example.

**Example 1** We assume three predicates: $S(x)$ saying that $x$ is a student, $G(x)$ saying that $x$ is a graduate student, and $I(x)$ saying that $x$ is Irish. We use constant $a$ to stand for Ann and $b$ for Bob. Let $\Sigma$ be the set of sentences

$$
\{ (G(a), S(b), I(a) \lor I(b), \forall x (G(x) \lor S(x)) \},
$$

and let $s$ be the setup defined as the set of instances of $\Sigma$:

$$
\{ (G(a), S(b), I(a) \lor I(b), \forall x (G(x) \lor S(x)) \}.
$$

Let $\phi$ be $\exists x (I(x) \land S(x))$. We now show that $s \models B_1 \phi$.

Clearly, $s \cup \{ I(a) \} \models B_0 \phi$ by subsumption. Also, $s \cup \{ I(a) \} \models B_0 S(a)$ by subsumption, since $S(a)$ can be obtained from $G(a)$ and $\neg G(a) \lor S(a)$. Thus $s \cup \{ I(a) \} \models B_0 \phi$ by reduction, and hence $s \cup \{ I(a) \} \models B_0 \phi$ by reduction. Similarly, $s \cup \{ I(b) \} \models B_0 \phi$. Thus $s \models B_1 \phi$ by splitting on the clause $I(a) \lor I(b)$.

**$\mathcal{S}L$-based Reasoning with Proper KBs**

$\mathcal{S}L$ is intended to serve as the foundation for a semantically coherent and computationally attractive reasoning service. The idea is to model the reasoning service as belief implication, that is, validity of formulas of the form $(B_\Sigma \alpha \supset B_k \phi)$, where $\Sigma$ is a KB, and $\phi$ is a query. We write $\Sigma \models k \phi$ if $(B_\Sigma \phi \supset B_k \phi)$ is valid. The $\mathcal{S}L$-based reasoning problem (for a fixed value $k$) is as follows: given a KB $\Sigma$ in $\mathcal{L}$ and a formula $\phi$ in $\mathcal{L}$, decide whether or not $\Sigma \models k \phi$. $\mathcal{S}L$-based reasoning is always classically sound: if $\Sigma \models k \phi$, then $\phi$ classically entails $\phi$ (LLL04). The converse, logical completeness, does not hold in general. Moreover, in general, $\models k$ is not decidable. To see where $\models k$ becomes decidable, we first define proper KBs.
**Proper** KBs

It is easy to show that if a KB is a simple database, then $\mathcal{S}L$-based reasoning coincides with classical logical entailment and is also decidable. However, a database does not allow any form of incomplete knowledge. Levesque (1998) proposes a generalization of a database called a proper KB, equivalent to a (possibly infinite) consistent set of ground literals. However, Lakemeyer and Levesque (2002) propose an extension to a proper KB called a proper $^+$ KB, equivalent to a possibly infinite set of ground clauses. We now define these notions formally.

We use $\varepsilon$ to range over $\mathcal{E}wff$s, that is, quantifier-free formulas whose only predicate is equality. We use $\forall \phi$ to denote the universal closure of $\phi$. We use $\theta$ to range over substitutions of all variables by constants, and write $\phi \theta$ as the result of applying the substitution $\theta$ to $\phi$.

**Definition 1** Let $\varepsilon$ be an $\mathcal{E}wff$ and $c$ a clause. Then a formula of the form $\forall (c \supset c)$ is called a $\forall$-clause. A KB $\Sigma$ is proper, if it is a finite non-empty set of $\forall$-clauses. Given a proper $^+$ KB $\Sigma$, $\text{gnd}(\Sigma)$ is defined as $\{ e \theta \mid (\forall (c \supset c) \in \Sigma \text{ and } \models (c \theta) \}$. A KB $\Sigma$ is proper if it is proper $^+$ and $\text{gnd}(\Sigma)$ is a consistent set of ground literals.

Note that $\text{gnd}(\Sigma)$ is an $\mathcal{S}L$ setup, as in the example above.

Despite the limitations, proper $^+$ KBs are expressive enough for many real-world applications. To get a feel for this, consider the following example from (LLL04):

**Example 2** Let $\Sigma$ be the following KB with a single predicate $C(p_1, p_2)$ saying that the two persons are compatible:

1. $\forall x \exists y C(x, y) \supset C(y, x)$;
2. $\forall x, C(x, ann) \supset C(x, bob)$;
3. $\neg C(bob, fred)$;
4. $C(carl, eve) \supset C(carl, fred)$;
5. $\forall x. x \neq bob \lor x \neq carl \supset C(dan, x)$;
6. $\neg C(eve, ann) \lor \neg C(eve, fred)$.

Then we have the following:

1. $\Sigma = 0 C(fred, ann)$;
2. $\Sigma = 1 \forall x \exists y C(x, y)$;
3. $\Sigma = \exists x \exists y \exists z [C(x, y) \land C(x, z) \land \neg C(y, z)]$;
4. $\Sigma = \exists x \exists y [x \neq y \land C(x, carl) \land C(y, carl)]$, but $\Sigma = \exists x \exists y [x \neq y \land C(x, carl) \land C(y, carl)]$.

By Theorem 1 below, $\Sigma \models k \phi$ iff $\text{gnd}(\Sigma) \models B_k \phi$. Thus the above can be proved by showing that $\text{gnd}(\Sigma) \models B_k \phi$ (or $\text{gnd}(\Sigma) \not\models B_k \phi$).

**Properties of $\mathcal{S}L$-based reasoning**

As noted above, $\mathcal{S}L$-based reasoning is classically sound but incomplete. However, Liu et al. present the following two results. First, $\mathcal{S}L$-based reasoning is completely for proper KBs and queries in a certain normal form called $\mathcal{N}F$.

Second, $\mathcal{S}L$-based reasoning is “eventually complete” for a propositional proper $^+$ KB $\Sigma$ and a propositional query $\phi$ in $\mathcal{N}F$: if $\Sigma$ classically entails $\phi$, then there is a $k$ such that $\Sigma \models k \phi$. As to the computational property, Liu et al. show that for proper $^+$ KBs, $\mathcal{S}L$-based reasoning reduces to a model checking problem (for a possibly infinite model):

**Theorem 1** (LLL04) Let $\Sigma$ be proper $^+$.

Then $\Sigma \models k \phi$ iff $\text{gnd}(\Sigma) \models B_k \phi$.

Using this theorem, they show that $\mathcal{S}L$-based reasoning with proper $^+$ KBs is tractable in the propositional case and decidable in the first-order case.

**A Tractability Result**

In this section, we show that $\mathcal{S}L$-based reasoning with proper $^+$ KBs is not only decidable but also tractable in the first-order case provided that both the KB and the query use a bounded number of variables.

The main ideas behind this result are as follows. First, by Theorem 1, it suffices to prove that deciding whether $\text{gnd}(\Sigma) \models B_k \phi$ is tractable when both $\Sigma$ and $\phi$ use a bounded number of variables.

Second, as in the case of database query evaluation, instead of answering a Boolean query, we compute the set of substitution $\theta$ such that $\text{gnd}(\Sigma) \models B_k \phi \theta$ where $\phi$ is a formula which may have free variables. Although this set may well be infinite, it has a finite representation, which is what we actually compute. As in the tractability result of (Vardi 1995), we view subformulas of $\phi$ as subqueries, and the fact that $\phi$ uses a bounded number of variables ensures that all intermediate results are bounded-arity database relations.

In the following, we use $\mathcal{L}^j$ to denote the set of formulas from $\mathcal{L}$ whose variables are from $X = \{ x_1, \ldots, x_j \}$, where $j \geq 1$. We use $\theta$ to range over substitutions of all variables $x_1, \ldots, x_j$ by constants. We use $D$ to range over finite sets of constants. We use $\theta \in D$ to mean that $\theta$ only takes constants from $D$. We let $\text{gnd}(\Sigma)[D]$ denote the restriction of $\text{gnd}(\Sigma)$ to $D$, that is, the set of clauses from $\text{gnd}(\Sigma)$ that only mention constants from $D$. Let $\Gamma$ be a set of formulas. We use $H(\Gamma)$ to denote the set of constants appearing in $\Gamma$, and we use $H^m_{gnd}(\Gamma)$ to denote the union of the constants appearing in $\Gamma$ and $m$ extra ones.

**Answers to open queries**

**Definition 2** Let $\Sigma \subseteq \mathcal{L}^j$ be proper $^+$, $\phi \in \mathcal{L}^j$, and $k \geq 0$. We define $\text{Ans}(\Sigma, \phi, k)$ as the set $\{ \theta \mid \text{gnd}(\Sigma) \models B_k \phi \theta \}$.

However, $\text{Ans}(\Sigma, \phi, k)$ may well be infinite. Fortunately, we can find a finite representation for it. We let $\text{Ans}(\Sigma, \phi, k)[D]$ denote the restriction of $\text{Ans}(\Sigma, \phi, k)$ to $D$, that is, the set $\{ \theta \in D \mid \text{gnd}(\Sigma) \models B_k \phi \theta \}$.

**Proposition 2** Let $D$ be $H^m_{gnd}(\Sigma \cup \{ \phi \})$ for some $m \geq j$. Then $\text{Ans}(\Sigma, \phi, k)[D]$ is a finite representation for $\text{Ans}(\Sigma, \phi, k)$ in the following sense:
For any substitution θ, θ ∈ Ans(Σ, φ, k) iff θ′ ∈ Ans(Σ, φ, k)|D, where θ′ is like θ except that for all those xi (i = 1, . . . , j) such that θ(xi) /∈ D, θ′ maps them into unique representatives from D − H(Σ ∪ {φ}).

Example 3 Let Σ be the following simple KB:
{∀x.y. C(x, y) ⊃ C(y, x), ∀x.x /∈ ann ⊃ C(x, bob)}. Let φ be C(x, y). Then Ans(Σ, φ, 0) = {(c, bob) | e /∈ ann} ∪ {(bob, c) | e /∈ ann}, which is an infinite set. Now let D = {ann, bob, carl, dan}. Then Ans(Σ, φ, 0, D) = {(bob, bob), (carl, bob), (dan, bob), (bob, carl), (bob, dan)}, and it makes a finite representation for Ans(Σ, φ, 0). First, consider (bob, eve). Since eve /∈ D, we choose carl as its representative; since (bob, carl) ∈ Ans(Σ, φ, 0, D), we know (bob, eve) ∈ Ans(Σ, φ, 0, D). Now consider (eve, fred). Since neither eve nor fred is in D, we choose carl and dan as their representatives; since (carl, dan) /∈ Ans(Σ, φ, 0, D), we know (eve, fred) /∈ Ans(Σ, φ, 0).

The algorithm
We first define two operations to be used in the algorithm. By an ξ-relation over domain D, we mean a set of substitutions θ such that θ ∈ D. We use θ(x/d) to denote the substitution that is the same as θ except that x is assigned d.

Definition 3 Let R be an ξ-relation over domain D, and let x ∈ X. The division of R wrt x, written δξ(x), is the set {θ ∈ D | ∀d ∈ D, θ(x/d) ∈ R}. The projection of R wrt x, written πξ(x), is the set {θ ∈ D | ∃d ∈ D, θ(x/d) ∈ R}. Note that our definition of projection (or division) is somewhat different from that in the database literature: ours is the Cartesian product of theirs and the domain. We use this definition so as to simplify the presentation of the procedure below, where every intermediate relation is an ξ-relation.

Example 4 Let X = {x, y}, D = {a, b, c}, and R = {(a, a), (b, a), (c, a), (a, b)}. Then δξ(x) = {(a, a), (b, a), (a, b)}, and πξ(x) = {(a, a), (b, a), (a, b), (b, a), (c, a), (c, b), (b, b), (a, b), (b, b), (c, b), (b, b)}.

Given a proper+ KB Σ ⊆ L†, a query φ ∈ L†, a natural number k, and a finite set of constants D, the procedure E returns an ξ-relation over domain D as follows:

1. E(Σ, (t = t′), k, D) = {θ ∈ D | θ is identical to tθ′}. Here t and t′ are variables or constants.
2. E(Σ, (¬(t = t′)), k, D) = {θ ∈ D | tθ is distinct from t′θ′}. This is a subsumption operation, and we will give a detailed procedure for it in the proof of Lemma 8 below.
3. If φ is a clause and k = 0, then E(Σ, φ, k, D) = {θ ∈ D | there is c ∈ UP(gnd(Σ)|D) s.t. c ⊆ φθ}. This is a subsumption operation, and we will give a detailed procedure for it in the proof of Lemma 8 below.
4. If φ is a clause and k > 0, then E(Σ, φ, k, D) = S(Σ, φ, k, D) where we use an abbreviation for

$$
\bigcup_{c \in gnd(\Sigma)|D} \bigcap_{\rho \in c} E(\Sigma \cup \{\rho\}, \phi, k-1, D).
$$

Here S represents a splitting operation, that is, we let c range over clauses in gnd(Σ)|D and take the union of the following: the intersection of E(Σ ∪ {ρ}, φ, k − 1, D) where ρ ranges over literals in c.

5. E(Σ, ¬ψ, k, D) = E(Σ, ψ, k, D).
6. If φ is (ψ ∨ η), but not a clause, then E(Σ, φ, k, D) = S(Σ, φ, k, D) ∪ E(Σ, ψ, k, D) ∪ E(Σ, η, k, D).
7. E(Σ, ¬(ψ ∨ η), k, D) = E(Σ, ¬ψ, k, D) ∩ E(Σ, ¬η, k, D).
8. E(Σ, ∃xψ, k, D) = S(Σ, ∃xψ, k, D) ∪ πx(E(Σ, ψ, k, D)).
9. E(Σ, ¬∃xψ, k, D) = δx(E(Σ, ¬ψ, k, D)).

We now illustrate Cases 1-4 with an example.

**Example 5** Let Σ be the following KB:
{∀x.C(x, ann) ∨ C(x, bob), ¬C(carl, bob), C(dan, ann) ⊃ C(dan, carl), C(dan, bob) ⊃ C(dan, carl)}.

Let D = {ann, bob, carl, dan, eve}. Then
1. E(Σ, (x = y), 0, D) = {(c, c) | c ∈ D}. 2. E(Σ, (¬(x = ann)), 0, D) = {(c, d) | c, d ∈ D, and c /∈ ann}. 3. E(Σ, (C(x, y), 0, D) = {(carl, ann)}. We get (carl, ann) because C(carl, ann) can be obtained from gnd(Σ)|D by unit propagation.
4. E(Σ, (C(x, y), 1, D) = {(carl, ann), (dan, carl)}.

We get (dan, carl) because it appears in both E(Σ ∪ C(dan, ann)), C(x, y), 0, D) and E(Σ ∪ C(dan, bob)), C(x, y), 0, D).

**Correctness proof**
The following theorem states that if D contains all the constants in Σ ∪ {φ} and at least k + 2 extra ones, then the above procedure computes Ans(Σ, φ, k)|D, which is a finite representation for Ans(Σ, φ, k) by Proposition 2.

**Theorem 3** Let Σ ⊆ L† be proper+, φ ∈ L†, and k ≥ 0. Let D be Hmω † (Σ ∪ {φ}) for some m ≥ j(k + 2). Then E(Σ, φ, k, D) = Ans(Σ, φ, k)|D.

The proof is by induction on φ. Cases 1, 2, 5, 7, and 9 use the following properties of beliefs from (LLL04):

\[ B_c e &equiv; c, \text{ where } c \text{ is an ewoff} \]
\[ B_k \neg \phi \equiv B_k \phi \]
\[ B_k (\phi \land \psi) \equiv B_k \phi \land B_k \psi \]
\[ B_k \forall x \phi \equiv \forall x B_k \phi \]

The other cases use the following lemmas, which justify our treatment of subsumption, quantification, and splitting, respectively. We let w(Σ) denote the maximum number of variables in a ∀-clause of Σ.

**Lemma 4** Let D be Hmω † (Σ) for some m ≥ w(Σ). Suppose that c ∈ UP(gnd(Σ)|D). Then c′ ∈ UP(gnd(Σ)|D), where c′ is like c except that constants not in D are replaced with unique representatives from D − H(Σ).

**Lemma 5** Let φ be a formula with a single free variable x. Let b and d be two constants that do not appear in Σ or φ. Then gnd(Σ) ⊃ B_c φ by splitting on c ∈ gnd(Σ). Then gnd(Σ) ⊃ B_c φ by splitting on some c′ ∈ gnd(Σ)|D.
Complexity analysis

We begin with two lemmas about the complexity of database operations and subsumption operations. We let $j$ denote the number of variables.

**Lemma 7** Each database operation used in procedure $E$ (selection, intersection, union, division and projection) can be done in $O(n^{j})$ time, where $n$ is the size of $D$.

**Proof:** Each relation is of size $O(n^{j})$, and is always kept in sorted form. Each operation can be done in one pass of the input relations, and the result remains sorted. □

**Lemma 8** Let $\phi$ be a clause. Let $D$ be of size $O(n)$, where $n$ is the size of $\Sigma$. Then $E(\Sigma, \phi, 0, D)$ can be computed in $O(n^{j+1})$ time.

**Proof:** The following procedure computes $E(\Sigma, \phi, 0, D)$:
1. Compute $\text{gnd}(\Sigma)|D|$.
2. Perform unit propagation over $\text{gnd}(\Sigma)|D$. Let $U_0$ be the set of minimum clauses of $\text{UP}(\text{gnd}(\Sigma)|D)$.
3. For each $c \in U_0$, check whether it can be unified with a subset of $\phi$; if so, mark those $\theta \in D$ such that $c \subseteq \phi \theta$. Return the set of those marked $\theta$. □

**Theorem 9** Let $\Sigma \subseteq \mathcal{L}^i$ be proper, $\phi \in \mathcal{L}^j$, and $k \geq 0$. Then whether $\Sigma \models_k \phi$ can be decided in $O((ln^{j+1})^{k+1})$ time, where $l$ is the size of $\phi$, and $n$ is the size of $\Sigma$.

**Proof:** To decide if $\Sigma \models_k \phi$, we let $D = H^{j}(k+2)(\Sigma \cup \{\phi\})$, compute $E(\Sigma, \phi, k, D)$, and check whether it is empty.

Let $f(k)$ denote the time complexity of computing $E(\Sigma, \phi, k, D)$. Suppose $k > 0$. For each of the $l$ clauses and logical operators in $\phi$, we perform a database operation and/or a splitting operation. Each database operation can be done in $O(n^{j})$ time by Lemma 7. Each splitting operation considers $O(n^{j+1})$ clauses and takes the union of the corresponding results. Thus $f(k) = O(ln^{j+1}. (f(k-1) + n^{j}))$. Suppose $k = 0$. For each of the $l$ clauses and logical operators in $\phi$, we perform a database operation or a subsumption operation. Each subsumption operation can be done in $O(n^{j+1})$ time by Lemma 8. Thus $f(0) = O(ln^{j+1})$. Solving the recurrence, we have $f(k) = O((ln^{j+1})^{k+1})$. □

This procedure grows exponentially, but only with $j$ (the number of variables) and $k$ (the depth of case splitting).

Obviously, there are two places where we could improve procedure $E$. First, it is possible to reduce the number of clauses we need to consider during a splitting operation. Second, it is also possible to use incremental unit propagation, that is, we perform unit propagation in the very beginning, and then after we add a literal by splitting, we do further unit propagation incurred by this literal.

**Conclusions**

In this paper, we have showed that $\mathcal{SC}$-based reasoning with proper $\gamma$ KBs is tractable in the first-order case when both the KB and the query use a bounded number of variables. The procedure we propose nonetheless scales exponentially with the number of variables and the depth of case analysis. But we expect small values of these parameters to suffice except when the KB encodes a combinatorial puzzle or some other “mathematically interesting” problem. In this sense, $\mathcal{SC}$ provides a computationally viable reasoning service for first-order knowledge bases with disjunctive information.

We believe that the contribution of this paper lies not only in the technical result but also in the methodology. The concept of bounded treewidth has proven valuable for obtaining many tractability results. Kolaitis and Vardi are able to explain this in terms of the tractability of the model checking problem for bounded-variable first-order logic. What we have done here is to show how this idea could be applied to a radically different form of model checking, that is, when the models are the setups of the belief logic $\mathcal{SC}$. In the future, we would like to take this idea even further, and find tractable reasoning services for more expressive representation languages, for example, for knowledge bases that include unknown individuals.

References


