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 tractable in the first-order case provided that the knowledge base and the query both use a bounded number of variables.

1 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 and 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 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 $(\mathbf{B}KB \supset \mathbf{B}\phi)$, where $\mathbf{B}$ is a belief operator. With the goal of specifying a reasoning service for first-order KBs with disjunctive information, 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. 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.

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 graph-based approach to tractability via the concept of bounded treewidth. For example, in general, the constraint satisfaction problem (CSP) is NP-complete. However, researchers have shown that CSP with bounded-treewidth constraint graphs is in PTIME. Kolaitis and Vardi [2000] give a logical characterization for bounded treewidth via bounded-variable first-order logic, and hence show that

this well-known result about CSP can be explained in terms of 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.

2 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 an edge in a graph, for any k, the property "there is a path of length k from a" (here a is a constant) is definable by an FO^2-formula, as shown by the following. We define $\phi_i(u)$ by induction:

$\phi_0(u) = true$;
$\phi_{2i+1}(u) = \exists x[R(u,x) \wedge \phi_{2i}(x)]$, for $i \geq 0$;
$\phi_{2i+2}(u) = \exists y[R(u,y) \wedge \phi_{2i+1}(y)]$, for $i \geq 0$.

Then for any k, $\phi_k(a) \in$ FO^2, and states that there is a path of length k from a. 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. 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 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. Kolaitis and Vardi are able to confirm this property by showing that bounded-treewidth CSP reduces to database query evaluation for FO^j.

3 The Subjective Logic $\mathcal{SL}$

We begin with the motivation for $\mathcal{SL}$. 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 *et al.* [2004] propose reasoning based on $\mathcal{SL}$, 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 $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, $\boldsymbol{B}_2$, ..., where $\boldsymbol{B}_k$ essentially supports case analysis of depth bounded by k.

3.1 The syntax

The language $\mathcal{L}$ is a standard first-order logic with equality. The language $\mathcal{SL}$ is a first-order logic with equality whose atomic formulas are belief atoms of the form $\boldsymbol{B}_k\phi$ where ϕ is a formula of $\mathcal{L}$ and $\boldsymbol{B}_k$ is a modal operator for any $k \geq 0$. $\boldsymbol{B}_k\phi$ is read as "ϕ 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 are terms.[1] The *literals* are atoms or their negations. We use ρ to range over literals, and we use $\overline{\rho}$ to denote the complement of ρ.

The language $\mathcal{L}$ is the least set of expressions such that

1. if ρ is an atom, then $\rho \in \mathcal{L}$;
2. if t and t' are terms, then $(t = t') \in \mathcal{L}$;
3. if $\phi, \psi \in \mathcal{L}$ and x is a variable, then $\neg\phi$, $(\phi \vee \psi)$, and $\exists x\phi \in \mathcal{L}$.

Clauses, which play an important role in the semantic definition of $\mathcal{SL}$, 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 $(c \vee c')$ is a clause.

A clause is identified with the set of literals it contains. For convenience, a unit clause is identified with the corresponding literal. Only non-empty clauses appear in $\mathcal{L}$. However, the empty clause, denoted by $\Box$, is used in $\mathcal{SL}$.

[1] 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 $\mathcal{SL}$ is the least set of expressions such that

1. if $\phi \in \mathcal{L}$ or ϕ is $\Box$, and $k \geq 0$, then $\boldsymbol{B}_k\phi \in \mathcal{SL}$, and is called a *belief atom* of level k;
2. if t and t' are terms of $\mathcal{L}$, then $(t = t') \in \mathcal{SL}$;
3. if $\alpha, \beta \in \mathcal{SL}$ and x is a variable, then $\neg\alpha$, $(\alpha \vee \beta)$, and $\exists x\alpha \in \mathcal{SL}$.

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

3.2 Belief reductions

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

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

1. $(\boldsymbol{B}_k c)\downarrow = \boldsymbol{B}_k c$, where c is a clause;
2. $(\boldsymbol{B}_k(t = t'))\downarrow = (t = t')$;
3. $(\boldsymbol{B}_k\neg(t = t'))\downarrow = \neg(t = t')$;
4. $(\boldsymbol{B}_k\neg\neg\phi)\downarrow = \boldsymbol{B}_k\phi$;
5. $(\boldsymbol{B}_k(\phi \vee \psi))\downarrow = (\boldsymbol{B}_k\phi \vee \boldsymbol{B}_k\psi)$, where ϕ or ψ is not a clause;
6. $(\boldsymbol{B}_k\neg(\phi \vee \psi))\downarrow = (\boldsymbol{B}_k\neg\phi \wedge \boldsymbol{B}_k\neg\psi)$;
7. $(\boldsymbol{B}_k\exists x\phi)\downarrow = \exists x\boldsymbol{B}_k\phi$;
8. $(\boldsymbol{B}_k\neg\exists x\phi)\downarrow = \forall x\boldsymbol{B}_k\neg\phi$.

As mentioned earlier, $\mathcal{SL}$ 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 $\mathsf{UP}(s)$ is used to denote the closure of s under unit propagation, that is, the least set s' satisfying:
1. $s \subseteq s'$; and 2. if $\rho \in s'$ and $\{\overline{\rho}\} \cup c \in s'$, then $c \in s'$.
The notation $\mathsf{VP}(s)$ is used to denote the following set:
$\{c \mid c \text{ is a ground clause and there is } c' \in \mathsf{UP}(s) \text{ s.t. } c' \subseteq c\}$.

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

3.3 The semantics

Sentences of $\mathcal{SL}$ 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 $\mathcal{SL}$ 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 $\alpha \in \mathcal{SL}$, $s \models \alpha$ (read "s satisfies α") 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. $s \models \alpha \vee \beta$ 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. $s \models \boldsymbol{B}_k\phi$ iff one of the following holds:
   (a) *subsume*: $k = 0$, ϕ is a clause c, and $c \in \mathsf{VP}(s)$;
   (b) *reduce*: ϕ is not a clause and $s \models (\boldsymbol{B}_k\phi)\downarrow$;
   (c) *split*: $k > 0$ and there is some $c \in s$ such that for all $\rho \in c$, $s \cup \{\rho\} \models \boldsymbol{B}_{k-1}\phi$.

As usual, a sentence $\alpha \in \mathcal{SL}$ 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 $\mathcal{SL}$. The rules for equality and quantification are justified by the fact that the constants are taken to satisfy the unique name assumption and an infinitary version of domain closure. So all the novelty in $\mathcal{SL}$ is due to the interpretation of the $\boldsymbol{B}_k$ operators. Intuitively, the rules propose three different justifications for believing a sentence ϕ at level k:

1. ϕ is a clause, $k = 0$, and after doing unit propagation on our explicit beliefs, we end up with a subclause of ϕ;
2. we already have appropriate beliefs about the subformulas of ϕ, for example, believing both conjuncts 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 ϕ 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 Σ be the set of sentences

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

and let s be the setup defined as the set of instances of Σ:

$$\{G(a), S(b), I(a) \vee I(b), \neg G(a) \vee S(a), \neg G(b) \vee S(b), \ldots\}.$$

Let ϕ be $\exists x(I(x) \wedge S(x))$. We now show that $s \models \boldsymbol{B}_1\phi$.

Clearly, $s \cup \{I(a)\} \models \boldsymbol{B}_0 I(a)$ by subsumption. Also, $s \cup \{I(a)\} \models \boldsymbol{B}_0 S(a)$ by subsumption, since $S(a)$ can be obtained from $G(a)$ and $\neg G(a) \vee S(a)$ by unit propagation. Thus $s \cup \{I(a)\} \models \boldsymbol{B}_0(I(a) \wedge S(a))$ by reduction, and hence $s \cup \{I(a)\} \models \boldsymbol{B}_0\phi$ by reduction. Similarly, $s \cup \{I(b)\} \models \boldsymbol{B}_0\phi$. Thus $s \models \boldsymbol{B}_1\phi$ by splitting on the clause $I(a) \vee I(b)$.

4 $\mathcal{SL}$-based Reasoning with Proper$^+$ KBs

$\mathcal{SL}$ is intended to serve as a 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 $(\boldsymbol{B}_0\Sigma \supset \boldsymbol{B}_k\phi)$, where Σ is a KB, and ϕ is a query. We write $\Sigma \models_k \phi$ if $(\boldsymbol{B}_0\Sigma \supset \boldsymbol{B}_k\phi)$ is valid. The $\mathcal{SL}$-based reasoning problem (for a fixed value k) is as follows: given a KB Σ in $\mathcal{L}$ and a formula ϕ in $\mathcal{L}$, decide whether or not $\Sigma \models_k \phi$. $\mathcal{SL}$-based reasoning is always classically sound: if $\Sigma \models_k \phi$, then Σ classically entails ϕ [Liu *et al.*, 2004]. 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.

4.1 Proper$^+$ KBs

It is easy to show that if a KB is a simple database, then $\mathcal{SL}$-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. But while a proper KB allows atomic formulas to be unknown, it does not allow any form of disjunctive information. For example, in a proper KB, we cannot express "Ann or Bob is Irish" or "Every graduate student is a student" as in the example above. So 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 e to range over *ewffs*, that is, quantifier-free formulas whose only predicate is equality. We use $\forall\phi$ to denote the universal closure of ϕ. We use θ to range over substitutions of all variables by constants, and write $\phi\theta$ as the result of applying the substitution θ to ϕ.

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

Note that *gnd*(Σ) is an $\mathcal{SL}$ 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 [Liu *et al.*, 2004].

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

1. $\forall x \forall y. C(x, y) \supset C(y, x)$;
2. $\forall x. C(x, \mathit{ann}) \vee C(x, \mathit{bob})$;
3. $\neg C(\mathit{bob}, \mathit{fred})$;
4. $C(\mathit{carl}, \mathit{eve}) \vee C(\mathit{carl}, \mathit{fred})$;
5. $\forall x. x \neq \mathit{bob} \wedge x \neq \mathit{carl} \supset C(\mathit{dan}, x)$;
6. $\neg C(\mathit{eve}, \mathit{ann}) \vee \neg C(\mathit{eve}, \mathit{fred})$.

Then we have the following:

1. $\Sigma \models_0 C(\mathit{fred}, \mathit{ann})$;
2. $\Sigma \models_1 \forall x \exists y C(x, y)$;
3. $\Sigma \models_1 \exists x \exists y \exists z [C(x, y) \wedge C(x, z) \wedge \neg C(y, z)]$;
4. $\Sigma \models_2 \exists x \exists y [x \neq y \wedge C(x, \mathit{carl}) \wedge C(y, \mathit{carl})]$, but $\Sigma \not\models_1 \exists x \exists y [x \neq y \wedge C(x, \mathit{carl}) \wedge C(y, \mathit{carl})]$.

By Theorem 1 below, $\Sigma \models_k \phi$ iff *gnd*$(\Sigma) \models \boldsymbol{B}_k\phi$. Thus the above can be proved by showing that *gnd*$(\Sigma) \models \boldsymbol{B}_k\phi$ (or *gnd*$(\Sigma) \not\models \boldsymbol{B}_k\phi$).

4.2 Properties of $\mathcal{SL}$-based reasoning

As noted above, $\mathcal{SL}$-based reasoning is classically sound but incomplete. However, Liu *et al.* present the following two results. First, $\mathcal{SL}$-based reasoning is classically complete for proper KBs and queries in a certain normal form called $\mathcal{NF}$. Second, $\mathcal{SL}$-based reasoning is "eventually complete" for a propositional proper$^+$ KB Σ and a propositional query ϕ in $\mathcal{NF}$: if Σ classically entails ϕ, then there is a k such that $\Sigma \models_k \phi$. As to the computational property, Liu *et al.* show that for proper$^+$ KBs, $\mathcal{SL}$-based reasoning reduces to a model checking problem (for a possibly infinite model):

Theorem 1 ([Liu *et al.*, 2004]) *Let Σ be proper$^+$. Then $\Sigma \models_k \phi$ iff *gnd*$(\Sigma) \models \boldsymbol{B}_k\phi$.*

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

5 A Tractability Result

In this section, we show that $\mathcal{SL}$-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 *gnd*$(\Sigma) \models \boldsymbol{B}_k\phi$ is tractable when both Σ and ϕ use a bounded number of variables. Although *gnd*(Σ) may well be infinite, as shown in [Liu *et al.*, 2004], it suffices to consider the restriction of *gnd*(Σ) to a finite set of constants, which consists of the constants in either Σ or ϕ, and a few extra ones. Moreover, the fact that Σ uses a bounded number of variables ensures that this restriction has a polynomial size. Second, as in the case of database query evaluation, instead of answering a Boolean query, we compute the set of substitution θ such that *gnd*$(\Sigma) \models \boldsymbol{B}_k\phi\theta$ where ϕ 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 ϕ as subqueries, and the fact that ϕ 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 θ 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 θ only takes constants from D. We let *gnd*$(\Sigma)|D$ denote the restriction of *gnd*(Σ) to D, that is, the set of clauses from *gnd*(Σ) that only mention constants from D. Let Γ be a set of formulas. We use $H(\Gamma)$ to denote the set of constants appearing in Γ, and we use $H^+_m(\Gamma)$ to denote the union of the constants appearing in Γ and m extra ones.

5.1 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 $Ans(\Sigma, \phi, k)$ as the set $\{\theta \mid$ *gnd*$(\Sigma) \models \boldsymbol{B}_k\phi\theta\}$.

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

Proposition 2 *Let D be $H^+_m(\Sigma \cup \{\phi\})$ for some $m \geq j$. Then $Ans(\Sigma, \phi, k)|D$ is a finite representation for $Ans(\Sigma, \phi, k)$ in the following sense:*

For any substitution θ, $\theta \in Ans(\Sigma, \phi, k)$ iff $\theta' \in Ans(\Sigma, \phi, k)|D$, where θ' is like θ except that for all x_i ($i = 1, \ldots, j$) s.t. $\theta(x_i) \notin D$, θ' maps them into unique representatives from $D - H(\Sigma \cup \{\phi\})$.

Example 3 Let Σ be the following simple KB:

$$\{\forall x \forall y. C(x, y) \supset C(y, x), \ \ \forall x. x \neq \mathit{ann} \supset C(x, \mathit{bob})\}.$$

Let ϕ be $C(x, y)$. Then $Ans(\Sigma, \phi, 0) = \{(c, \mathit{bob}) \mid c \neq \mathit{ann}\} \cup \{(\mathit{bob}, c) \mid c \neq \mathit{ann}\}$, which is an infinite set. Now let $D = \{\mathit{ann}, \mathit{bob}, \mathit{carl}, \mathit{dan}\}$. Then $Ans(\Sigma, \phi, 0, D) = \{(\mathit{bob}, \mathit{bob}), (\mathit{carl}, \mathit{bob}), (\mathit{dan}, \mathit{bob}), (\mathit{bob}, \mathit{carl}), (\mathit{bob}, \mathit{dan})\}$, and it makes a finite representation for $Ans(\Sigma, \phi, 0)$. First, consider $(\mathit{bob}, \mathit{eve})$. Since *eve* $\notin D$, we choose *carl* as its representative; since $(\mathit{bob}, \mathit{carl}) \in Ans(\Sigma, \phi, 0, D)$, we know $(\mathit{bob}, \mathit{eve}) \in Ans(\Sigma, \phi, 0)$. Now consider $(\mathit{eve}, \mathit{fred})$. Since neither *eve* nor *fred* is in D, we choose *carl* and *dan* as their representatives; since $(\mathit{carl}, \mathit{dan}) \notin Ans(\Sigma, \phi, 0, D)$, we know $(\mathit{eve}, \mathit{fred}) \notin Ans(\Sigma, \phi, 0)$.

5.2 The algorithm

We first define two operations to be used in the algorithm. By an X-relation over domain D, we mean a subset of $\{\theta \mid \theta \in D\}$. We use $\theta(x/d)$ to denote the substitution which is the same as θ except that x is assigned d.

Definition 3 Let R be an X-relation over domain D, and let $x \in X$. The *division* of R wrt x, written $\delta_x(R)$, is the set $\{\theta \in D \mid \forall d \in D, \theta(x/d) \in R\}$. The *projection* of R wrt x, written $\pi_x(R)$, is the set $\{\theta \in D \mid \exists d \in D, \theta(x/d) \in R\}$.

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 X-relation.

Example 4 Let $X = \{x, y\}$, $D = \{a, b, c\}$, and $R = \{(a, a), (b, a), (c, a), (a, b)\}$. Then $\delta_x(R) = \{(a, a), (b, a), (c, a)\}$, and $\pi_x(R) = \{(a, a), (b, a), (c, a), (a, b), (b, b), (c, b)\}$.

Given a proper$^+$ KB $\Sigma \subseteq \mathcal{L}^j$, a query $\phi \in \mathcal{L}^j$, a natural number k, and a finite set of constants D, the procedure E returns an X-relation over domain D as follows:

1. $E(\Sigma, (t = t'), k, D) = \{\theta \in D \mid t\theta \text{ is identical to } t'\theta\}$. Here t and t' are variables or constants.
2. $E(\Sigma, \neg(t = t'), k, D) = \{\theta \in D \mid t\theta \text{ is distinct from } t'\theta\}$.
3. If ϕ is a clause and $k = 0$, then $E(\Sigma, \phi, k, D) = \{\theta \in D \mid \text{there is } c \in \mathsf{UP}(\mathit{gnd}(\Sigma)|D) \text{ s.t. } c \subseteq \phi\theta\}$. 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 $E(\Sigma, \phi, k, D) = S(\Sigma, \phi, k, D)$, which we use as an abbreviation for
$$\bigcup_{c \in \mathit{gnd}(\Sigma)|D} \ \bigcap_{\rho \in c} E(\Sigma \cup \{\rho\}, \phi, k - 1, D).$$
Here S represents a splitting operation, that is, we let c range over clauses in $\mathit{gnd}(\Sigma)|D$ and take the union of the following: the intersection of $E(\Sigma \cup \{\rho\}, \phi, k - 1, D)$ where ρ ranges over literals in c.
5. $E(\Sigma, \neg\neg\psi, k, D) = E(\Sigma, \psi, k, D)$.
6. If ϕ is $(\psi \vee \eta)$, but not a clause, then $E(\Sigma, \phi, k, D) = S(\Sigma, \phi, k, D) \cup E(\Sigma, \psi, k, D) \cup E(\Sigma, \eta, k, D)$.
7. $E(\Sigma, \neg(\psi \vee \eta), k, D) = E(\Sigma, \neg\psi, k, D) \cap E(\Sigma, \neg\eta, k, D)$.
8. $E(\Sigma, \exists x\psi, k, D) = S(\Sigma, \exists x\psi, k, D) \cup \pi_x(E(\Sigma, \psi, k, D))$.
9. $E(\Sigma, \neg\exists x\psi, k, D) = \delta_x(E(\Sigma, \neg\psi, k, D))$.

We now illustrate Cases 1-4 with an example.

Example 5 Let Σ be the following KB:

$$\{\forall x. C(x, \mathit{ann}) \vee C(x, \mathit{bob}), \neg C(\mathit{carl}, \mathit{bob}), \ C(\mathit{dan}, \mathit{ann}) \supset C(\mathit{dan}, \mathit{carl}), C(\mathit{dan}, \mathit{bob}) \supset C(\mathit{dan}, \mathit{carl})\}.$$

Let $D = \{\mathit{ann}, \mathit{bob}, \mathit{carl}, \mathit{dan}, \mathit{eve}\}$. Then

1. $E(\Sigma, (x = y), 0, D) = \{(c, c) \mid c \in D\}$.
2. $E(\Sigma, \neg(x = \mathit{ann}), 0, D) = \{(c, d) \mid c, d \in D, \text{ and } c \neq \mathit{ann}\}$.
3. $E(\Sigma, C(x, y), 0, D) = \{(\mathit{carl}, \mathit{ann})\}$. We get $(\mathit{carl}, \mathit{ann})$ because $C(\mathit{carl}, \mathit{ann})$ can be obtained from $\mathit{gnd}(\Sigma)|D$ by unit propagation.
4. $E(\Sigma, C(x, y), 1, D) = \{(\mathit{carl}, \mathit{ann}), (\mathit{dan}, \mathit{carl})\}$. We get $(\mathit{dan}, \mathit{carl})$ because it appears in both $E(\Sigma \cup \{C(\mathit{dan}, \mathit{ann})\}, C(x, y), 0, D)$ and $E(\Sigma \cup \{C(\mathit{dan}, \mathit{bob})\}, C(x, y), 0, D)$.

5.3 Correctness proof

The following theorem states that if D contains all the constants in $\Sigma \cup \{\phi\}$ and at least $j(k + 2)$ extra ones, then the above procedure computes $Ans(\Sigma, \phi, k)|D$, which is a finite representation for $Ans(\Sigma, \phi, k)$ by Proposition 2.

Theorem 3 *Let $\Sigma \subseteq \mathcal{L}^j$ be proper$^+$, $\phi \in \mathcal{L}^j$, and $k \geq 0$. Let D be $H^+_m(\Sigma \cup \{\phi\})$ for some $m \geq j(k + 2)$. Then $E(\Sigma, \phi, k, D) = Ans(\Sigma, \phi, k)|D$.*

The proof is by induction on ϕ. Cases 1, 2, 5, 7, and 9 use the following properties of beliefs from [Liu *et al.*, 2004]:

$$\models \boldsymbol{B}_k e \equiv e, \text{ where } e \text{ is an ewff}$$
$$\models \boldsymbol{B}_k \neg\neg\phi \equiv \boldsymbol{B}_k \phi$$
$$\models \boldsymbol{B}_k(\phi \wedge \psi) \equiv \boldsymbol{B}_k \phi \wedge \boldsymbol{B}_k \psi$$
$$\models \boldsymbol{B}_k \forall x \phi \equiv \forall x \boldsymbol{B}_k \phi$$

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

Lemma 4 *Let D be $H^+_m(\Sigma)$ for some $m \geq w(\Sigma)$. Suppose that $c \in \mathsf{UP}(\mathit{gnd}(\Sigma))$. Then $c' \in \mathsf{UP}(\mathit{gnd}(\Sigma)|D)$, where c' is like c except that constants not in D are replaced with unique representatives from $D - H(\Sigma)$.*

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 $\mathit{gnd}(\Sigma) \models \boldsymbol{B}_k \phi^x_b$ iff $\mathit{gnd}(\Sigma) \models \boldsymbol{B}_k \phi^x_d$.*

Lemma 6 *Let D be $H^+_m(\Sigma \cup \{\phi\})$ for some $m \geq w(\Sigma)$. Suppose that $\mathit{gnd}(\Sigma) \models \boldsymbol{B}_k \phi$ by splitting on $c \in \mathit{gnd}(\Sigma)$. Then $\mathit{gnd}(\Sigma) \models \boldsymbol{B}_k \phi$ by splitting on some $c' \in \mathit{gnd}(\Sigma)|D$.*

5.4 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 ϕ be a clause. Let D be of size $O(n)$, where n is the size of Σ. 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 *gnd*$(\Sigma)|D$.
2. Perform unit propagation over *gnd*$(\Sigma)|D$. Let U_0 be the set of minimum clauses of $\mathsf{UP}($*gnd*$(\Sigma)|D)$.
3. For each $c \in U_0$, check whether it can be unified with a subset of ϕ; if so, mark those $\theta \in D$ such that $c \subseteq \phi\theta$. Return the set of those marked θ. ■

Theorem 9 *Let $\Sigma \subseteq \mathcal{L}^j$ 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 ϕ, and n is the size of Σ.*

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 ϕ, 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 $f(k) = O(ln^{j+1} \cdot (f(k-1) + n^j))$. Suppose $k = 0$. For each of the l clauses and logical operators in ϕ, 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 with both 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.

6 Conclusions

In this paper, we have showed that $\mathcal{SL}$-based reasoning with proper$^+$ 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, $\mathcal{SL}$ 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 $\mathcal{SL}$. 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

[Dechter and Pearl, 1989] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38(3):353–366.

[Freuder, 1990] E. C. Freuder. Complexity of K-tree structured constraint satisfaction problems. In *Proc. AAAI-90*, pages 4–9.

[Frisch, 1987] A. M. Frisch. Inference without chaining. In *Proc. IJCAI-87*, pages 515–519, 1987.

[Kolaitis and Vardi, 2000] Ph. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, 61(2):14–23, 2000.

[Lakemeyer and Levesque, 2002] G. Lakemeyer and H. J. Levesque. Evaluation-based reasoning with disjunctive information in first-order knowledge bases. In *Proc. KR-02*, pages 73–81, 2002.

[Lakemeyer, 1994] G. Lakemeyer. Limited reasoning in first-order knowledge bases. *Artificial Intelligence*, 71(2):213–255, 1994.

[Levesque, 1984] H. J. Levesque. A logic of implicit and explicit belief. In *Proc. of AAAI-84*, pages 198–202, 1984.

[Levesque, 1998] H. J. Levesque. A completeness result for reasoning with incomplete first-order knowledge bases. In *Proc. KR-98*, pages 302–332, 1998.

[Liu and Levesque, 2003] Y. Liu and H. J. Levesque. A tractability result for reasoning with incomplete first-order knowledge bases. In *Proc. IJCAI-03*, pages 83–88, 2003.

[Liu *et al.*, 2004] Y. Liu, G. Lakemeyer, and H. J. Levesque. A logic of limited belief for reasoning with disjunctive information. In *Proc. KR-04*, pages 587–597, 2004.

[Patel-Schneider, 1990] P. F. Patel-Schneider. A decidable first-order logic for knowledge representation. *Journal of Automated Reasoning*, 6:361–388, 1990.

[Schaerf and Cadoli, 1995] M. Schaerf and M. Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74:249–310.

[Vardi, 1982] M. Y. Vardi. The complexity of relational query languages (extended abstract). In *Proc. 14th Annual ACM Symposium on Theory of Computing*, pages 137–146, 1982.

[Vardi, 1995] M. Y. Vardi. On the complexity of bounded-variable queries. In *Proc. 14th ACM Symposium on Principles of Database Systems (PODS '95)*, pages 266–276, 1995.