A Reasoning System for a First-Order Logic of Limited Belief

Christoph Schwering
School of Computer Science and Engineering
The University of New South Wales
Sydney NSW 2052, Australia
c.schwering@unsw.edu.au

Abstract

Logics of limited belief aim at enabling computationally feasible reasoning in highly expressive representation languages. These languages are often dialects of first-order logic with a weaker form of logical entailment that keeps reasoning decidable or even tractable. While a number of such logics have been proposed in the past, they tend to remain for theoretical analysis only and their practical relevance is very limited. In this paper, we aim to go beyond the theory. Building on earlier work by Liu, Lakemeyer, and Levesque, we develop a logic of limited belief that is highly expressive while remaining decidable in the first-order and tractable in the propositional case and exhibits some characteristics that make it attractive for an implementation. We introduce a reasoning system that employs this logic as representation language and present experimental results that showcase the benefit of limited belief.

1 Introduction

Dealing with incomplete knowledge is one of the longstanding aims of research in Knowledge Representation and Reasoning. Incompleteness often demands highly expressive languages to be represented accurately. For instance, the statement

"I don't know who Sally's father is, but I know he's rich"

involves an individual (Sally's father) and both knowns (he's rich) and unknowns (his identity) about him. From the representational point of view, first-order and modal logics are excellent tools to formalise such statements. However, reasoning in classical first-order logic very quickly gets undecidable: an existential quantifier and two unary functions with equality can be enough to make validity an undecidable problem [Börger *et al.*, 1997].

One way to get around undecidability of first-order reasoning is through models of *limited belief*.[1] Inspired by natural agents, the idea behind limited belief is to give up the property of *logical omniscience* [Hintikka, 1975]. This separates limited belief from other approaches to decidable reasoning like the classical prefix-vocabulary classes [Börger *et al.*, 1997] or description logics [Baader, 2003], but is similar to approaches of approximate reasoning like [D'Agostino, 2015]. While a number of models of limited belief have been proposed in the past [Konolige, 1986; Vardi, 1986; Fagin and Halpern, 1987; Levesque, 1984b; Patel-Schneider, 1990; Lakemeyer, 1994; Delgrande, 1995], these approaches can be criticised for either being too fine-grained or overly weakening the entailment relation and thus even ruling out the most basic cases of modus ponens.

A more recent proposal for limited belief is due to Liu, Lakemeyer, and Levesque [2004]. Their logic is equipped with a perspicuous semantics based on clause subsumption, unit propagation, and case splitting,[2] and keeps the computational complexity under control by stratifying beliefs in *levels*: level 0 comprises only the explicit beliefs; every following level draws additional inferences by doing another case split. Every query specifies at which belief level it shall be evaluated, and thus controls how much effort should be spent on proving it. The rationale behind this technique of limiting belief by case splits is the hypothesis that in many practical reasoning tasks few case splits – perhaps no more than one or two – suffice.

Let us consider a brief example to illustrate the idea. Suppose we have the following knowledge base (KB):

$$(\text{fatherOf}(\text{Sally}) = \text{Frank} \lor \text{fatherOf}(\text{Sally}) = \text{Fred}) \land$$
$$\forall x(\text{fatherOf}(\text{Sally}) \neq x \lor \text{Rich}(x)).$$

At level 0 the agent believes these clauses, but does not draw any meaningful inferences from them yet. For instance, $\text{Rich}(\text{Frank}) \lor \text{Rich}(\text{Fred})$, while entailed in classical logic, is not believed at level 0. It can however be inferred by splitting the cases for Sally's potential fathers:

- if $\text{fatherOf}(\text{Sally}) = \text{Frank}$, then we obtain $\text{Rich}(\text{Frank})$ by unit propagation with the second clause from the KB;
- if $\text{fatherOf}(\text{Sally}) = \text{Fred}$, then analogously $\text{Rich}(\text{Fred})$;
- if $\text{fatherOf}(\text{Sally})$ is anyone else, then unit propagation with the first clause in the KB yields the empty clause.

Either of the three cases subsumes $\text{Rich}(\text{Frank}) \lor \text{Rich}(\text{Fred})$. Hence, $\text{Rich}(\text{Frank}) \lor \text{Rich}(\text{Fred})$ is believed at level 1.

[1]We use the terms knowledge and belief interchangeably.

[2]A clause is a disjunction of literals. Subsumption refers to inferring a clause c_2 from a clause c_1 when every literal in c_1 subsumes a literal in c_2. Unit propagation of a clause c with a literal ℓ means to remove all literals from c that are complementary to ℓ. A case split means to branch on all possible values a literal or term can take.

Several variants of Liu, Lakemeyer, and Levesque's original theory have been developed [Lakemeyer and Levesque, 2013; 2014; 2016; Klassen *et al.*, 2015; Schwering and Lakemeyer, 2016]; they include concepts like introspection, actions, functions, or conditional beliefs. Despite this progress, the framework has remained a purely theoretical one without practical applications.

In this paper, we want to bring their approach to limited belief to practice. We make two contributions to this end:

- Firstly, we devise a logic of limited belief that unifies some concepts from the earlier proposals and adds several features to make it more attractive for practical use. The result is a sorted first-order logic with functions and equality and two introspective belief operators, one for knowledge and one for what is considered possible. While closely related to the earlier proposals, many technical details in this new language have been changed with practical considerations in mind.
- Secondly, we present a reasoning system that uses this logic as representation language. Our evaluation is twofold. Besides modelling toy domains to showcase the language's expressivity, we have also tested the system's performance with two popular puzzle games, Sudoku and Minesweeper. The results confirm the hypothesis that small belief levels often suffice to achieve good results.

The paper is organised as follows. In Section 2 we introduce an (omniscient) logic of knowledge. Based on this logic, we then develop a logic of limited belief in Section 3. In Section 4 we sketch a decision procedure for limited belief, discuss an implementation of this system, and present experiment results. Proofs of the results presented in this paper can be found in [Schwering, 2017].

2 A Logic of Knowledge

The logic we present in this section is a variant of Levesque's logic of only-knowing [Levesque, 1990] and will serve as a reference for the logic of limited belief in the next section. We refer to the logic from this section as $\mathcal{L}$.

2.1 The Language

The language of $\mathcal{L}$ is a sorted first-order dialect with functions, equality, standard names, and three epistemic modalities. With minor modifications, this language will also be the language used for the logic of limited belief $\mathcal{LL}$.

We assume an infinite supply of *sorts*, and for each sort we assume an infinite number of *variables*, *function symbols* of every arity $j \geq 0$, and *standard names* (or *names* for short). Standard names serve as special constants that satisfy the unique name assumption and an infinitary version of domain closure. The set of *terms* (of a sort s) contains all variables (of sort s) and names (of sort s) as well as all functions $f(t_1, \ldots, t_j)$ where f is a j-ary function symbol (of sort s) and every t_i is a variable or a name (not necessarily of sort s). A *literal* is an expression of the form $t_1 = t_2$ or $\neg t_1 = t_2$ where t_1 is a variable, name, or function, and t_2 is a variable or a name. The set of *formulas* is the least set that contains all literals and all expressions $\neg\alpha$, $(\alpha \vee \beta)$, $\exists x \alpha$, $\mathbf{K}\alpha$, $\mathbf{M}\alpha$, and $\mathbf{O}\alpha$ where α and β are formulas and x is a variable.

Intuitively, $\mathbf{K}\alpha$ reads as "α is known," $\mathbf{M}\alpha$ as "α is considered possible," and $\mathbf{O}\alpha$ as "α is *all* that is known." We say a formula is *objective* when it mentions no belief operator, and *subjective* when it mentions no function outside of a belief operator. Only-knowing is particularly useful to capture the meaning of a knowledge base. As we only consider objective knowledge bases in this paper, we restrict ourselves from now on to objective ϕ in $\mathbf{O}\phi$.

For convenience, we use the usual abbreviations $\neq$, $\wedge$, $\forall$, $\supset$, and $\equiv$, and we sometimes omit brackets to ease readability.

Some differences between our language and traditional first-order languages are apparent: our language features no predicates; functions cannot be nested; only the left-hand side of a literal may be a function. These restrictions will prove helpful for the implementation of a reasoning system. We remark that none of these restrictions means a limitation of expressivity: a predicate $P(t_1, \ldots, t_j)$ can be simulated by a literal $p(t_1, \ldots, t_j) = \top$ where $\top$ is some standard name chosen to represent truth, and nested functions and literals with a function on the right-hand side can be flattened by introducing a new variable – these transformations preserve equivalence.

As an example, we formalise the introductory statement about Sally's father, who is rich but unknown to the agent:

$$\mathbf{K}\exists x\big(\text{fatherOf}(\text{Sally}) = x \wedge \text{rich}(x) = \top \wedge \\ \mathbf{M}\,\text{fatherOf}(\text{Sally}) \neq x\big),$$

where Sally is a standard name of the same sort (say, 'human') as x and fatherOf, and $\top$ is a name of the same sort (say, 'Boolean') as rich. Quantifying x into the modal context $\mathbf{M}$ expresses that the father's identity is unknown. The next subsection gives a semantic justification to this interpretation.

2.2 The Semantics

The semantics is based on possible worlds. We call a term $f(t_1, \ldots, t_j)$ *primitive* when all t_i are standard names. Let $\mathcal{N}$ and $\mathcal{T}$ be the sets of all names and primitive terms, respectively. To denote the names or primitive terms that occur in a formula α, we write $\mathcal{N}(\alpha)$ and $\mathcal{T}(\alpha)$, respectively, and analogously for sets of formulas. To include only terms of the same sort as t, we write $\mathcal{N}_t$ and $\mathcal{T}_t$. A *world* $w : \mathcal{T} \to \mathcal{N}$ is a sort-preserving mapping from primitive terms to standard names, that is, $w(t) \in \mathcal{N}_t$ for every primitive term t. A *sentence* is a formula without free variables. We denote by α^x_t the result of substituting t for all free occurrences of the variable x in α.

Truth of a sentence α is defined w.r.t. a world w and a set of possible worlds e as follows:

1. $e, w \models t = n$ iff
   - t and n are identical names if t is a name;
   - $w(t)$ and n are identical names otherwise;
2. $e, w \models \neg\alpha$ iff $e, w \not\models \alpha$;
3. $e, w \models (\alpha \vee \beta)$ iff $e, w \models \alpha$ or $e, w \models \beta$;
4. $e, w \models \exists x \alpha$ iff $e, w \models \alpha^x_n$ for some $n \in \mathcal{N}_x$;
5. $e, w \models \mathbf{K}\alpha$ iff $e, w' \models \alpha$ for all $w' \in e$;
6. $e, w \models \mathbf{M}\alpha$ iff $e, w' \models \alpha$ for some $w' \in e$;
7. $e, w \models \mathbf{O}\phi$ iff $e = \{w' \mid e, w' \models \phi\}$.

Unlike classical first-order logic, this semantics handles quantification by substitution of names. Standard names thus effectively serve as a fixed, countably infinite universe of discourse. See [Levesque, 1984a] for a discussion why this is no effective limitation for our purposes.

$\mathbf{O}\phi$ maximises the set of possible worlds e such that $\mathbf{K}\phi$ still holds. Hence the agent knows ϕ, but considers possible everything else provided it is consistent with ϕ. In other words, everything that is not a consequence of ϕ is unknown, for its negation is consistent with ϕ and hence considered possible. Thus $\mathbf{O}\phi$ captures only ϕ and its consequences are known.

As usual, a sentence α *entails* another sentence β, written $\alpha \models \beta$, when $e, w \models \alpha$ implies $e, w \models \beta$ for all e, w. A sentence α is *valid*, written $\models \alpha$, when $e, w \models \alpha$ for all e, w.

We omit a deeper analysis except to note that $\mathbf{K}$ is a K45 operator [Fagin *et al.*, 1995] and the following equivalences:

Proposition 1

(i) $\models \mathbf{K}\alpha \equiv \neg\mathbf{M}\neg\alpha$*;*

(ii) $\models \forall x \mathbf{K}\alpha \equiv \mathbf{K}\forall x\alpha$ *and* $\models \mathbf{K}\alpha \wedge \mathbf{K}\beta \equiv \mathbf{K}(\alpha \wedge \beta)$*;*

(iii) $\models \exists x \mathbf{M}\alpha \equiv \mathbf{M}\exists x\alpha$ *and* $\models \mathbf{M}\alpha \vee \mathbf{M}\beta \equiv \mathbf{M}(\alpha \vee \beta)$*;*

(iv) $\models \mathbf{O}\phi \supset \mathbf{K}\phi$*.*

To familiarise ourselves with the logic, we show that the query formalised at the end of Section 2.1 is entailed by

$$\mathbf{O}\big((\text{fatherOf}(\text{Sally}) = \text{Frank} \vee \text{fatherOf}(\text{Sally}) = \text{Fred}) \wedge \\ \forall x(\text{fatherOf}(\text{Sally}) \neq x \vee \text{rich}(x) = \top)\big)$$

where Frank and Fred are names of sort 'human.' Let ϕ denote the sentence within $\mathbf{O}$. By Rule 7, $e = \{w \mid e, w \models \phi\}$ is the only set of worlds that satisfies $\mathbf{O}\phi$, so proving the entailment reduces to model checking for e. By assumption, for every $w \in e$, $w(\text{fatherOf}(\text{Sally})) \in \{\text{Frank}, \text{Fred}\}$. Suppose $w(\text{fatherOf}(\text{Sally})) = \text{Frank}$; the case for Fred is analogous. By assumption, $w(\text{rich}(\text{Frank})) = \top$, so it only remains to be shown that $e, w \models \mathbf{M}\,\text{fatherOf}(\text{Sally}) \neq \text{Frank}$, which holds because there are $w' \in e$ with $w'(\text{fatherOf}(\text{Sally})) = \text{Fred}$.

3 A Logic of Limited Belief

We now introduce the logic $\mathcal{LL}$, the limited counterpart of $\mathcal{L}$.

3.1 The Language

The language of $\mathcal{LL}$ follows the rules from $\mathcal{L}$ with the following modifications: the expressions $\mathbf{K}\alpha$ and $\mathbf{M}\alpha$ are replaced with $\mathbf{K}_k\alpha$ and $\mathbf{M}_k\alpha$ where $k \geq 0$ is a natural number, and the expression $\mathbf{G}\alpha$ is added to the language. We read $\mathbf{K}_k\alpha$ and $\mathbf{M}_k\alpha$ as "α is believed at level k" and "α is considered possible at level k," respectively, and the new expression $\mathbf{G}\alpha$ intuitively means "assuming the knowledge base is consistent, α is true." Guaranteeing consistency is motivated by practical applications where it often may reduce the computational cost of reasoning.

3.2 The Semantics

In $\mathcal{LL}$, sets of clauses take over from sets of possible worlds as the semantic primitive that models belief. Intuitively, these clauses will represent the agent's explicit knowledge, like $\text{fatherOf}(\text{Sally}) = \text{Frank} \vee \text{fatherOf}(\text{Sally}) = \text{Fred}$ and $\forall x(\text{fatherOf}(\text{Sally}) \neq x \vee \text{rich}(x) = \top)$ in our running example. By means of case splitting and unit propagation then further inferences can be drawn from these clauses. Before we can formalise this, we need to introduce some terminology.

We call a literal *ground* when it contains no variables. Recall that a primitive term is one of the form $f(n_1, \ldots, n_j)$ where the n_i are names. Therefore every ground literal is of form $n = n'$ or $n \neq n'$ or $f(n_1, \ldots, n_j) = n$ or $f(n_1, \ldots, n_j) \neq n$ for names n_i, n, n'.

A literal is *valid* when it is of the form $t = t$, or $n \neq n'$ for distinct names n, n', or $t \neq t'$ for terms t, t' of distinct sorts. A literal ℓ_1 *subsumes* a literal ℓ_2 when ℓ_1, ℓ_2 are identical or ℓ_1, ℓ_2 are of the form $t = n_1$ and $t \neq n_2$ for distinct names n_1, n_2. Two literals ℓ_1, ℓ_2 are *complementary* when ℓ_1, ℓ_2 are of the form $t = t'$ and $t \neq t'$ (or vice versa), or ℓ_1, ℓ_2 are of the form $t = n_1$ and $t = n_2$ for distinct names n_1, n_2.

A *clause* is a finite set of literals. A clause with a single literal is a *unit clause*. We abuse notation and identify non-empty clauses $\{\ell_1, \ldots, \ell_j\}$ with formulas $(\ell_1 \vee \ldots \vee \ell_j)$. The above terminology for literals carries over to clauses as follows. A clause is *valid* when it contains a valid literal, or a literal $t = t'$ and its negation $t \neq t'$, or two literals $t \neq n_1$ and $t \neq n_2$ for distinct names n_1, n_2. A clause c_1 *subsumes* a clause c_2 if every literal $\ell_1 \in c_1$ subsumes a literal $\ell_2 \in c_2$. The *unit propagation* of a clause c with a literal ℓ is the clause obtained by removing from c all literals that are complementary to ℓ.

A *setup* is a set of ground clauses. We write $\mathsf{UP}(s)$ to denote the closure of s with all valid literals under unit propagation:

- if $c \in s$, then $c \in \mathsf{UP}(s)$;
- if ℓ is a valid literal, then $\ell \in \mathsf{UP}(s)$;
- if $c, \ell \in \mathsf{UP}(s)$ and c' is the unit propagation of c with ℓ, then $c' \in \mathsf{UP}(s)$.

We write $\mathsf{UP}^+(s)$ to denote the result of adding to $\mathsf{UP}(s)$ all valid clauses and all clauses that are subsumed by some clause in $\mathsf{UP}(s)$. Similarly, $\mathsf{UP}^-(s)$ shall denote the setup obtained by removing from $\mathsf{UP}(s)$ all valid clauses and all clauses subsumed by some other clause in $\mathsf{UP}(s)$.

Truth of a sentence α in $\mathcal{LL}$, written $s_0, s, v \approx\!\!\!| \; \alpha$, is defined w.r.t. two setups s_0, s and a set of unit clauses v. The purpose of having these three parameters is to deal with nested beliefs. For the objective part of the semantics, only s is relevant:

1. $s_0, s, v \models\!\!\approx \ell$ iff $\ell \in \mathsf{UP}^+(s)$ if ℓ is a literal;
2. $s_0, s, v \models\!\!\approx (\alpha \vee \beta)$ iff
   - $(\alpha \vee \beta) \in \mathsf{UP}^+(s)$ if $(\alpha \vee \beta)$ is a clause;
   - $s_0, s, v \models\!\!\approx \alpha$ or $s_0, s, v \models\!\!\approx \beta$ otherwise;
3. $s_0, s, v \models\!\!\approx \neg(\alpha \vee \beta)$ iff $s_0, s, v \models\!\!\approx \neg\alpha$ and $s_0, s, v \models\!\!\approx \neg\beta$;
4. $s_0, s, v \models\!\!\approx \exists x\alpha$ iff $s_0, s, v \models\!\!\approx \alpha^x_n$ for some $n \in \mathcal{N}_x$;
5. $s_0, s, v \models\!\!\approx \neg\exists x\alpha$ iff $s_0, s, v \models\!\!\approx \neg\alpha^x_n$ for every $n \in \mathcal{N}_x$;
6. $s_0, s, v \models\!\!\approx \neg\neg\alpha$ iff $s_0, s, v \models\!\!\approx \alpha$.

Note how negation is handled by rules for $(t_1 \neq t_2)$, $\neg(\alpha \vee \beta)$, $\neg\exists x\alpha$, $\neg\neg\alpha$. A rule $s_0, s, v \models\!\!\approx \neg\alpha$ iff $s_0, s, v \not\models\!\!\approx \alpha$ would be unsound, as $\mathcal{LL}$ is incomplete w.r.t. $\mathcal{L}$ (as we shall see).

We proceed with the semantics of $\mathbf{K}_k\alpha$. The idea is that k case splits can be made first, before α is evaluated. A case split means to select some term (say, fatherOf(Sally)) and branch (conjunctively) on the values it could take (namely all standard names of the right sort, such as Frank and Fred). To preserve soundness of introspection, the effect of case splits must not spread into nested beliefs. This is why we need to carefully manage three parameters s_0, s, v. Intuitively, s_0 is the "original" setup without split literals, and v "stores" the split literals. Once the number of case splits is exhausted, $s_0 \cup v$ takes the place of s, so that the objective subformulas of α are interpreted by $s_0 \cup v$, whereas the subjective subformulas of α are interpreted by s_0 (plus future splits from the nested belief operators). We say a setup s is *obviously inconsistent* when $\mathsf{UP}(s)$ contains the empty clause. In this special case, which corresponds to the empty set of worlds in the possible-worlds semantics, everything is known. The semantics of knowledge formalises this idea as follows:

7. $s_0, s, v \mathrel{|\!\approx} \mathbf{K}_0\alpha$ iff
   - $s_0 \cup v$ is obviously inconsistent, or
   - $s_0, s_0 \cup v, \emptyset \mathrel{|\!\approx} \alpha$;
8. $s_0, s, v \mathrel{|\!\approx} \mathbf{K}_{k+1}\alpha$ iff
   for some $t \in \mathcal{T}$ and every $n \in \mathcal{N}_t$,
   $s_0, s, v \cup \{t = n\} \mathrel{|\!\approx} \mathbf{K}_k\alpha$;
9. $s_0, s, v \mathrel{|\!\approx} \neg\mathbf{K}_k\alpha$ iff $s_0, s, v \mathrel{|\!\not\approx} \mathbf{K}_k\alpha$.

Similarly, the idea behind $\mathbf{M}_k\alpha$ is to fix the value of certain terms in order to show that the setup is consistent with α. Intuitively this means that we want to approximate a possible world, that is, an assignment of terms to names, that satisfies α. Often we want to fix not just a single term, but a series of terms with a common pattern, for instance, $f(n) = n$ for all n. To this end, we say two literals ℓ_1, ℓ_2 are *isomorphic* when there is a bijection $* : \mathcal{N} \to \mathcal{N}$ that swaps standard names in a sort-preserving way so that ℓ_1 and ℓ_2^* are identical, and define $v \uplus_s \ell_1 = v \cup \{\ell_2 \mid \ell_1, \ell_2 \text{ are isomorphic and } \neg\ell_2 \notin \mathsf{UP}^+(s \cup v)\}$. In English: $v \uplus_s \ell_1$ adds to v every literal that is isomorphic to ℓ_1 and not obviously inconsistent with the setup $s \cup v$. Furthermore, we need to take care that after fixing these values the setup is not potentially inconsistent. We say a setup s is *potentially inconsistent* when it is obviously inconsistent or when the set $\{\ell \mid \ell \in c \in \mathsf{UP}^-(s)\}$ of all literals in $\mathsf{UP}^-(s)$ contains two complementary literals, or a literal $t = n$ for $n \notin \mathcal{N}_t$, or all literals $t \neq n$ for $n \in \mathcal{N}_t$ for some primitive term t. Note that this consistency test is intentionally naive, for the complexity of $\mathbf{M}_k\alpha$ shall be bounded by k alone. The semantics of the consistency operator is then:

10. $s_0, s, v \mathrel{|\!\approx} \mathbf{M}_0\alpha$ iff
    - $s_0 \cup v$ is not potentially inconsistent, and
    - $s_0, s_0 \cup v, \emptyset \mathrel{|\!\approx} \alpha$;
11. $s_0, s, v \mathrel{|\!\approx} \mathbf{M}_{k+1}\alpha$ iff
    for some $t \in \mathcal{T}$ and $n \in \mathcal{N}_t$,
    $s_0, s, v \cup \{t = n\} \mathrel{|\!\approx} \mathbf{M}_k\alpha$ or
    $s_0, s, v \uplus_{s_0} (t = n) \mathrel{|\!\approx} \mathbf{M}_k\alpha$;
12. $s_0, s, v \mathrel{|\!\approx} \neg\mathbf{M}_k\alpha$ iff $s_0, s, v \mathrel{|\!\not\approx} \mathbf{M}_k\alpha$.

To capture that $\mathbf{O}\phi$ means that ϕ is all the agent knows, we need to minimise the setup (modulo unit propagation and subsumption), which corresponds to the maximisation of the set of possible worlds in $\mathcal{L}$. We hence define:

13. $s_0, s, v \mathrel{|\!\approx} \mathbf{O}\phi$ iff
    - $s_0, s_0, \emptyset \mathrel{|\!\approx} \phi$, and
    - $s_0, \hat{s}_0, \emptyset \mathrel{|\!\not\approx} \phi$ for every $\hat{s}_0$ with $\mathsf{UP}^+(\hat{s}_0) \subsetneq \mathsf{UP}^+(s_0)$;
14. $s_0, s, v \mathrel{|\!\approx} \neg\mathbf{O}\phi$ iff $s_0, s, v \mathrel{|\!\not\approx} \mathbf{O}\phi$.

Lastly, we define the semantics of the $\mathbf{G}\alpha$ operator, which represents a guarantee that s is consistent and therefore can reduce the size of s to clauses potentially relevant to α. We denote the *grounding* of α by $\mathsf{gnd}(\alpha) = \{\beta^{x_1 \ldots x_j}_{n_1 \ldots n_j} \mid n_i \in \mathcal{N}_{x_i}\}$ where β is the result of rectifying α and removing all quantifiers, and $x_1, \ldots, x_j$ are the free variables in β. Finally, $s|_T$ is the least set such that if $c \in \mathsf{UP}^-(s)$ and c either mentions a term from T, is empty, or shares a term with another clause in $s|_T$, then $c \in s|_T$. Then $\mathbf{G}\alpha$ works as follows:

15. $s_0, s, v \mathrel{|\!\approx} \mathbf{G}\alpha$ iff $s_0|_{\mathcal{T}(\mathsf{gnd}(\alpha))}, s, v \mathrel{|\!\approx} \alpha$;
16. $s_0, s, v \mathrel{|\!\approx} \neg\mathbf{G}\alpha$ iff $s_0, s, v \mathrel{|\!\approx} \mathbf{G}\neg\alpha$.

This completes the semantics. We write $s_0, s \mathrel{|\!\approx} \alpha$ to abbreviate $s_0, s, \emptyset \mathrel{|\!\approx} \alpha$. Note that for subjective formulas σ, s is irrelevant, so we may just write $s_0 \mathrel{|\!\approx} \sigma$. Analogous to $\models$ in $\mathcal{L}$, we overload $\mathrel{|\!\approx}$ for *entailment* and *validity*.

In this paper we are mostly concerned with reasoning tasks of the form $\mathbf{O}\phi \mathrel{|\!\approx} \sigma$ where σ is a subjective query and ϕ is a knowledge base of a special form called *proper*$^+$: ϕ is of the form $\bigwedge_i \forall x_1 \ldots \forall x_j\, c_i$ for clauses c_i.

Observe that a proper$^+$ KB directly corresponds to the setup $\mathsf{gnd}(\phi) = \bigcup_i \mathsf{gnd}(c_i)$. Also note that while existential quantifiers are disallowed in proper$^+$ KBs, they can be simulated as usual by way of Skolemisation.

Reasoning in proper$^+$ KBs is sound in $\mathcal{LL}$ w.r.t. $\mathcal{L}$, provided that the query does not mention belief modalities $\mathbf{K}_k, \mathbf{M}_k$ in a negated context. While the first-order case is incomplete, a restricted completeness result for the propositional case will be given below. We denote by $\sigma_{\mathcal{L}}$ the result of replacing in σ every $\mathbf{K}_k, \mathbf{M}_k$ with $\mathbf{K}, \mathbf{M}$. The soundness theorem follows:

Theorem 2 *Let ϕ be proper$^+$ and σ be subjective, without $\mathbf{O}, \mathbf{G}$, and without negated $\mathbf{K}_k, \mathbf{M}_k$.*
Then $\mathbf{O}\phi \mathrel{|\!\approx} \sigma$ implies $\mathbf{O}\phi \models \sigma_{\mathcal{L}}$.

Negated beliefs break soundness because of their incompleteness. For example, $\mathbf{K}_k(t = n \vee \neg\neg t \neq n)$ in general only holds for $k \geq 1$. Hence $\neg\mathbf{K}_0(t = n \vee \neg\neg t \neq n)$ comes out true, which is unsound w.r.t. $\mathcal{L}$. As a consequence of this incompleteness, $\mathbf{K}_k\alpha \equiv \neg\mathbf{M}_k\neg\alpha$ is not a theorem in $\mathcal{LL}$.

For *propositional* formulas, that is, formulas without quantifiers, high-enough belief levels are complete:

Theorem 3 *Let ϕ, σ be propositional, ϕ be proper$^+$, σ be subjective and without $\mathbf{O}, \mathbf{G}$. Let σ_k be like σ with every $\mathbf{K}_l, \mathbf{M}_l$ replaced with $\mathbf{K}_k, \mathbf{M}_k$.*
Then $\mathbf{O}\phi \models \sigma_{\mathcal{L}}$ implies that there is a k such that $\mathbf{O}\phi \mathrel{|\!\approx} \sigma_k$.

To conclude this section, let us revisit our running example. The KB is the same as in Section 2.2, and the modalities in the query are now indexed with belief levels:

$$\mathbf{K}_1 \exists x \big(\mathsf{fatherOf}(\mathsf{Sally}) = x \wedge \mathsf{rich}(x) = \top \wedge \\ \mathbf{M}_1 \mathsf{fatherOf}(\mathsf{Sally}) \neq x\big).$$

By Rule 13, $s_0 = \{\mathsf{fatherOf}(\mathsf{Sally}) = \mathsf{Frank} \vee \mathsf{fatherOf}(\mathsf{Sally}) = \mathsf{Fred}, \mathsf{fatherOf}(\mathsf{Sally}) \neq n \vee \mathsf{rich}(n) = \top \mid n \in \mathcal{N}_x\}$ is the unique (modulo UP⁺) setup that satisfies the KB. To prove the query, by applying Rule 8 we can split the term fatherOf(Sally). Consider $s_0 \cup \{\text{fatherOf(Sally)} = \text{Frank}\}$. By unit propagation we obtain $\text{rich(Frank)} = \top$, so we can choose Frank for x in Rule 4, and all that remains to be shown is that $s_0 \mid\!\approx \mathbf{M}_1 \text{fatherOf(Sally)} \neq \text{Frank}$. This is done by assigning $\text{fatherOf(Sally)} = \text{Fred}$ in Rule 11, and as the resulting setup $s_0 \cup \{\text{fatherOf(Sally)} = \text{Fred}\}$ is not potentially inconsistent and subsumes $\text{fatherOf(Sally)} \neq \text{Frank}$, the query holds. Returning to the splitting in Rule 8, the case $s_0 \cup \{\text{fatherOf(Sally)} = \text{Fred}\}$ is analogous, and for all other n, the setups $s_0 \cup \{\text{fatherOf(Sally)} = n\}$ are obviously inconsistent and therefore satisfy the query by Rule 7.

4 A Reasoning System

We now proceed to describe a decision procedure for reasoning in proper⁺ knowledge bases, and then discuss an implementation as well as experimental results.

4.1 Decidability

Reasoning in proper⁺ knowledge bases is decidable in $\mathcal{LL}$:

Theorem 4 *Let ϕ be proper⁺, σ be subjective and without* $\mathbf{O}$*. Then $\mathbf{O}\phi \mid\!\approx \sigma$ is decidable.*

While space precludes a detailed treatment of the proof and decision procedure, we sketch the idea. For the rest of this section, we use $\mathbf{L}_k$ as a placeholder for $\mathbf{K}_k$ and $\mathbf{M}_k$.

Let us first consider the case where σ is of the form $\mathbf{L}_k\psi$ for objective ψ; we will turn to nested modalities later. As ϕ is proper⁺, $\mathsf{gnd}(\phi)$ gives us the unique (modulo UP⁺) setup that satisfies $\mathbf{O}\phi$, so the reasoning task reduces to model checking of $\mathsf{gnd}(\phi)$. An important characteristic of standard names is that, intuitively, a formula cannot distinguish the names it does not mention. As a consequence, we can limit the grounding $\mathsf{gnd}(\phi)$ and quantification during the model checking to a finite number of names. For every variable x in ϕ or ψ, let p_x be the maximal number of variables occurring in any subformula of ϕ or ψ of the same sort as x. It is then sufficient for the grounding and for quantification of x to consider only the names $\mathcal{N}_x(\phi) \cup \mathcal{N}_x(\psi)$ plus $p_x + 1$ additional new names. Given the finite grounding and quantification, splitting in Rules 8 and 11 can also be confined to finitely many literals.

That way, the rules of the semantics of $\mathcal{LL}$ can be reduced to only deal with finite structures, which immediately yields a decision procedure for $\mathbf{O}\phi \mid\!\approx \mathbf{L}_k\psi$.

In the propositional case, this procedure is tractable:

Theorem 5 *Let ϕ, ψ be propositional, ϕ be proper⁺, ψ be objective. Then $\mathbf{O}\phi \mid\!\approx \mathbf{L}_k\psi$ is decidable in $\mathcal{O}(2^k(|\phi|+|\psi|)^{k+2})$.*

Now we turn to nested beliefs, which are handled using Levesque's representation theorem [Levesque, 1984a]. When ψ mentions a free variable x, the idea is to replace a nested belief $\mathbf{L}_k\psi$ with all instances n for which $\mathbf{L}_k\psi^x_n$ holds. Given a proper⁺ ϕ, a set of primitive terms T, and a formula $\mathbf{L}_k\psi$ for objective ψ, we define $\mathsf{RES}[\phi, T, \mathbf{L}_k\psi]$ as

- if ψ mentions a free variable x:
  $\bigvee_{n \in \mathcal{N}_x(\phi) \cup \mathcal{N}_x(\psi)} (x = n \wedge \mathsf{RES}[\phi, T, \mathbf{L}_k\psi^x_n]) \vee$
  $\big(\bigwedge_{n \in \mathcal{N}_x(\phi) \cup \mathcal{N}_x(\psi)} x \neq n \wedge \mathsf{RES}[\phi, T, \mathbf{L}_k\psi^x_{\hat{n}}]^{\hat{n}}_x\big)$
  where $\hat{n} \in \mathcal{N}_x \setminus (\mathcal{N}_x(\phi) \cup \mathcal{N}_x(\psi))$ is a some new name;
- if ψ mentions no free variables:
  TRUE if $\mathsf{gnd}(\phi)|_T \mid\!\approx \mathbf{L}_k\psi$, and $\neg$TRUE otherwise, where TRUE stands for $\exists x x = x$.

In our running example, $\mathsf{RES}[\phi, \mathcal{T}, \mathbf{M}_1 \text{fatherOf(Sally)} \neq x]$ is $(x = \text{Frank} \wedge \text{TRUE}) \vee (x = \text{Fred} \wedge \text{TRUE}) \vee (x = \text{Sally} \wedge \text{TRUE}) \vee (x \neq \text{Frank} \wedge x \neq \text{Fred} \wedge x \neq \text{Sally} \wedge \text{TRUE})$, which says that everybody is potentially *not* Sally's father.

The RES operator can now be applied recursively to eliminate nested beliefs from the inside to the outside. For proper⁺ ϕ, a set of terms T, and α without $\mathbf{O}$, we define

- $\|t = t'\|_{\phi,T}$ as $t = t'$;
- $\|\neg\alpha\|_{\phi,T}$ as $\neg\|\alpha\|_{\phi,T}$;
- $\|(\alpha \vee \beta)\|_{\phi,T}$ as $(\|\alpha\|_{\phi,T} \vee \|\beta\|_{\phi,T})$;
- $\|\exists x\alpha\|_{\phi,T}$ as $\exists x\|\alpha\|_{\phi,T}$;
- $\|\mathbf{L}_k\alpha\|_{\phi,T}$ as $\mathsf{RES}[\phi, T, \mathbf{L}_k\|\alpha\|_{\phi,T}]$;
- $\|\mathbf{G}\alpha\|_{\phi,T}$ as $\|\alpha\|_{\phi,T \cap \mathcal{T}(\mathsf{gnd}(\alpha))}$.

Note that $\|\cdot\|$ works from the inside to the outside and always returns an objective formula. In our example, $\|\mathbf{K}_1 \ldots\|_{\phi,\mathcal{T}}$ first determines $\mathsf{RES}[\phi, \mathcal{T}, \mathbf{M}_1 \text{fatherOf(Sally)} \neq x]$, and then $\mathsf{RES}[\phi, \mathcal{T}, \mathbf{K}_1 \exists x(\text{fatherOf(Sally)} = x \wedge \text{rich}(x) = \top \wedge \mathsf{RES}[\phi, \mathcal{T}, \mathbf{M}_1 \text{fatherOf(Sally)} \neq x)]]$ evaluates to TRUE.

Levesque's representation theorem (transferred to $\mathcal{LL}$) states that this reduction is correct:

Theorem 6 *Let ϕ be proper⁺, σ be subjective and without* $\mathbf{O}$*. Then $\mathbf{O}\phi \mid\!\approx \sigma$ iff $\mid\!\approx \|\sigma\|_{\phi,\mathcal{T}}$.*

It follows that propositional reasoning is tractable:

Corollary 7 *Let ϕ, σ be propositional, ϕ be proper⁺, σ be subjective, and $k \geq l$ for every $\mathbf{K}_l, \mathbf{M}_l$ in σ. Then $\mathbf{O}\phi \mid\!\approx \sigma$ is decidable in $\mathcal{O}(2^k(|\phi| + |\sigma|)^{k+3})$.*

4.2 Implementation

The reasoning system LIMBO implements the decision procedure sketched in the previous subsection. LIMBO is written in C++ and available as open source.[3]

Compared to literals in propositional logic, literals in our first-order language are relatively complex objects. As we want to adopt SAT solving technology, care was taken to keep literal objects lightweight and efficient. Firstly, for every term we create only a single full copy and from then on uniquely identify the term with a 30-bit pointer to this full copy. With this compact representation of terms, a literal fits into a 64-bit integer: 60 bits for the left- and right-hand terms, and one bit to encode whether the literal is negated. Two of the remaining bits are used to encode whether the terms are standard names, which allows us to implement the subsumption and complementarity tests for literals (as defined in Section 3.2) using just bitwise operations on the literals' 64-bit representation, without dereferencing the term pointers. An experiment showed this representation to be 24 times faster than a naive encoding.

Subsumption and complementarity tests for literals are mostly used in the context of setups for determining unit propagation and subsumption. To avoid unnecessary blowup, invalid literals in clauses as well as valid clauses in setups are not

[3]Source code: www.github.com/schwering/limbo

	NYT easy	NYT medium	NYT hard	Top 1465
clues	38.0	24.4	24.0	18.0
level 0	42.8	49.5	44.2	45.1
level 1	0.3	6.6	11.2	9.5
level 2	–	0.5	1.8	4.6
level 3	–	–	–	3.1
level 4	–	–	–	0.5
level 5	–	–	–	0.01
time	0.1 s	0.8 s	4.1 s	49.5 m

Table 1: Sudoku experiments over eight puzzles of each category from The New York Times website as well as the first 125 of the "Top 1465" list. The rows show how many cells on average per game were preset or solved at belief level 1, 2, 3, 4, or 5. The last row shows the average time per puzzle.

level		8×8–10	16×16–40	16×30–99	32×64–320
0	win	62.0%	46.0%	1.4%	28.3%
	time	0.01 s	0.06 s	0.24 s	5.08 s
1	win	87.3%	84.9%	37.7%	69.8%
	time	0.01 s	0.08 s	0.43 s	5.46 s
2	win	87.8%	85.0%	39.1%	70.0%
	time	0.02 s	0.10 s	0.64 s	5.60 s
3	win	87.8%	85.0%	39.1%	70.0%
	time	0.07 s	0.25 s	4.94 s	5.90 s

Table 2: Minesweeper experiments over 1000 randomised runs of different configurations, where W×H–M means M mines on a W×H grid. The rows contain results for different maximum belief levels used by the reasoner to figure out whether cells are safe or not. Numbers are the chance of winning and execution time per game.

represented explicitly. To facilitate fast unit propagation and cheap backtracking during splitting, the setup data structure uses the watched-literals scheme [Gomes *et al.*, 2008]. Other SAT technologies like backjumping or clause learning are not used at the current stage. The setup data structure also provides an operation to query the value of a primitive term, which is used to optimise subformulas of the form $\mathbf{K}_k t = x$ in queries.

When the KB is known to be consistent, the query may be wrapped in a $\mathbf{G}$ operator to allow the reasoner to remove irrelevant clauses and thus reduce the branching factor for splitting. With an inconsistent KB, $\mathbf{G}$ is not sound, though.

LIMBO also rewrites formulas, exploiting equivalences that hold in $\mathcal{L}$ but not in $\mathcal{LL}$ like Proposition 1 (ii–iii), and provides syntactic sugar for nested functions and the like.

4.3 Evaluation

Three sample applications were developed to evaluate the reasoning system.[4] For one thing, a textual user interface allows for specification of reasoning problems and has been used to model several small-scale examples, including this paper's running example, to test the system's full expressivity. In this section, however, we focus on the application of limited belief to the games of Sudoku and Minesweeper.

Sudoku is played on a 9×9 grid which is additionally divided into nine 3×3 blocks. The goal is to find a valuation of the cells such that every row, column, and 3×3 block contains every value $[1, 9]$ exactly once. The difficulty depends on how many and which numbers are given as clues from the start.

In Minesweeper the goal is to explore a grid by uncovering all and only those cells that contain no mine. When such a safe cell is uncovered, the player learns how many adjacent cells are safe, but when a mined cell is uncovered, the game is lost. The difficulty depends on the number of mines and grid size.

Both games were played by simple agents that use the reasoning system to represent and infer knowledge about the current game state. For Sudoku, we use a function value$(x, y) \in [1, 9]$ and translate the game rules to constraints such as $y_1 = y_2 \vee \text{value}(x, y_1) \neq \text{value}(x, y_2)$. For Minesweeper a Boolean function isMine(x, y) is used, and when a cell (x, y) is uncovered, clauses are added to represent the possible valuations of isMine$(x \pm 1, y \pm 1)$. Both agents use iterative deepening to find their next move: first, they look for a cell (x, y) for which value(x, y) or isMine(x, y) is known at belief level 0; if none exists, they repeat the same for belief level 1; and so on, until a specified maximum level is reached. Once a known cell is found, the corresponding information is added to the knowledge base. In the case of Minesweeper, it is sometimes necessary to guess; we then use a naive strategy that prefers cells that are not next to an uncovered field.

[4]Demos: www.cse.unsw.edu.au/~cschwering/limbo

While both games do not require much expressivity to be modelled, they are nevertheless interesting applications of limited belief because they are known to be computationally hard – Sudoku on N×N grids is NP-complete [Takayuki and Takahiro, 2003], Minesweeper is co-NP-complete [Scott *et al.*, 2011] – yet often easily solved by humans. According to the motivating hypothesis behind limited belief, a small belief level should often suffice to reach human-level performance. Indeed we find this hypothesis confirmed for both games. The results for Sudoku in Table 1 show that most 'easy' instances are solved just by unit propagation, and the number of necessary case splits increases for 'medium' and 'hard.' Significantly more effort is needed to solve games from the "Top 1465" list, a repository of extremely difficult Sudokus. For Minesweeper, Table 2 shows that strong results are achieved at level 1 already, and while belief level 2 increases the chance of winning by 0.5%, there is no improvement at level 3.

The experiments were conducted with custom implementations of both games, compiled with `clang -O3`, and run on an Intel Core i7-4600U CPU at 3.3 GHz.

5 Conclusion

We developed a practical variant of Liu, Lakemeyer, and Levesque's [2004] theory of limited belief and introduced and evaluated LIMBO, a reasoning system that implements our logic. The system features a sorted first-order language of introspective belief with functions and equality; the complexity of reasoning is controlled through the number of case splits. The idea behind limited belief is that often a small number of case splits suffice to bring about useful reasoning results; this hypothesis was confirmed in our experimental evaluation.

A natural next step is to incorporate theories of action, belief change, and multiple agents, which could open up applications in epistemic planning and high-level robot control. Another task is to improve the runtime performance, perhaps using SAT technology like clause learning and backjumping.

References

[Baader, 2003] Franz Baader. *The Description Logic Handbook: Theory, Implementation, and Applications*. 2003.

[Börger *et al.*, 1997] Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. 1997.

[Buffet *et al.*, 2012] Olivier Buffet, Chang-Shing Lee, Woan-Tyng Lin, and Olivier Teytuad. Optimistic heuristics for minesweeper. In *Proc. ICS*, 2012.

[D'Agostino, 2015] Marcello D'Agostino. An informational view of classical logic. *Theor. Comput. Sci.*, 2015.

[Delgrande, 1995] James P. Delgrande. A framework for logics of explicit belief. *Comput. Intell.*, 1995.

[Fagin and Halpern, 1987] Ronald Fagin and Joseph Halpern. Belief, awareness, and limited reasoning. *Artif. Intell.*, 1987.

[Fagin *et al.*, 1995] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about knowledge*. 1995.

[Gomes *et al.*, 2008] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Satisfiability solvers. In *Handbook of Knowledge Representation*. 2008.

[Hintikka, 1975] Jaakko Hintikka. Impossible possible worlds vindicated. *J. Philos. Logic*, 1975.

[Klassen *et al.*, 2015] Toryn Q. Klassen, Sheila A. McIlraith, and Hector J. Levesque. Towards tractable inference for resource-bounded agents. In *Proc. Commonsense*, 2015.

[Konolige, 1986] Kurt Konolige. *A Deduction Model of Belief*. 1986.

[Lakemeyer and Levesque, 2013] Gerhard Lakemeyer and Hector J. Levesque. Decidable reasoning in a logic of limited belief with introspection and unknown individuals. In *Proc. IJCAI*, 2013.

[Lakemeyer and Levesque, 2014] Gerhard Lakemeyer and Hector J. Levesque. Decidable reasoning in a fragment of the epistemic situation calculus. In *Proc. KR*, 2014.

[Lakemeyer and Levesque, 2016] Gerhard Lakemeyer and Hector J. Levesque. Decidable reasoning in a logic of limited belief with function symbols. In *Proc. KR*, 2016.

[Lakemeyer, 1994] Gerhard Lakemeyer. Limited reasoning in first-order knowledge bases. *Artif. Intell.*, 1994.

[Levesque, 1984a] Hector J. Levesque. Foundations of a functional approach to knowledge representation. *Artif. Intell.*, 1984.

[Levesque, 1984b] Hector J. Levesque. A logic of implicit and explicit belief. In *Proc. AAAI*, 1984.

[Levesque, 1990] Hector J. Levesque. All I know: a study in autoepistemic logic. *Artif. Intell.*, 1990.

[Liu *et al.*, 2004] Yongmei Liu, Gerhard Lakemeyer, and Hector J. Levesque. A logic of limited belief for reasoning with disjunctive information. In *Proc. KR*, 2004.

[Patel-Schneider, 1990] Peter F. Patel-Schneider. A decidable first-order logic for knowledge representation. *J. Autom. Reason.*, 1990.

[Schwering and Lakemeyer, 2016] Christoph Schwering and Gerhard Lakemeyer. Decidable reasoning in a first-order logic of limited conditional belief. In *Proc. ECAI*, 2016.

[Schwering, 2017] Christoph Schwering. A reasoning system for a first-order logic of limited belief. *arXiv:1705.01817*.

[Scott *et al.*, 2011] Allan Scott, Ulrike Stege, and Iris Van Rooij. Minesweeper may not be NP-complete but is hard nonetheless. *Math. Intell.*, 2011.

[Takayuki and Takahiro, 2003] Yato Takayuki and Seta Takahiro. Complexity and completeness of finding another solution and its application to puzzles. *IEICE-Tran. Fund. Elec., Comm. & Comp. Sci.*, 2003.

[Vardi, 1986] Moshe Y. Vardi. On epistemic logic and logical omniscience. In *Proc. TARK*, 1986.