Efficient Reasoning With Consistent Proper Epistemic Knowledge Bases

Christian Muise, Tim Miller, Paolo Felli, Adrian R. Pearce, Liz Sonenberg
Department of Computing and Information Systems, University of Melbourne
{christian.muise,tmiller,paolo.felli,adrianrp,l.sonenberg}@unimelb.edu.au

ABSTRACT

Reasoning about the nested beliefs or knowledge of other agents is essential for many collaborative and competitive tasks. However, reasoning with nested belief (for example through epistemic logics) is computationally expensive. Proper Epistemic Knowledge Bases (PEKBs) address this by enforcing syntactic restrictions on the knowledge base. By compiling a PEKB and query formula into a specific normal form, entailment can be checked in polynomial time, which is sound and complete for the epistemic logic K_n. The downside is that the complexity of compiling into the normal form is exponential in time and space. In this work, we extend PEKBs to handle belief in the logic of KD_n. We show that this simplifies the complexity of the required reasoning, and importantly, achieves polynomial entailment checking without first having to compile the PEKB into a normal form. Also, we present an alternative approach that calculates the closure of a PEKB, which is exponential in the maximum depth of nested belief, but for which entailment checking is constant on average.

Categories and Subject Descriptors

I.2.4 [**Knowledge Representation Formalisms and Methods**]: Modal logic; I.2.11 [**Distributed Artificial Intelligence**]: Intelligent Agents

General Terms

Algorithms, Theory

Keywords

proper epistemic knowledge bases, reasoning about belief, multi-agent systems

1. INTRODUCTION

Reasoning about the nested beliefs or knowledge of various agents is essential for many collaborative and competitive tasks. Many modal logics for modelling epistemic and doxastic properties have been presented in related literature. Hintikka's seminal work [8] proposed a logic for knowledge at the individual level, while Fagin et al. [4] were some of the first to look at multi-agent logics, including formal models of concepts such as common knowledge and belief. These formalisms are based on the semantic notion of possible worlds, as described in Section 2. These approaches offer neat semantics based on Kripke structures, but unfortunately they suffer from several limitations: notably *logical omniscience* [8] and high complexity [4, 14]. We are particularly interested in complex tasks such as planning that require reasoning with a low complexity. As such, rich epistemic modal logics are not feasible. One alternative is to use syntactic approaches to knowledge bases. The idea of syntactic knowledge bases is not recent. Eberle [3] was one of the first to contrast these semantic approaches to consider syntactic belief and knowledge, while Konolige [9] built on such work for logics that are expressive enough to support nested beliefs. These approaches are expressive enough to model many domains, while also offering some attractive computational properties.

More recently, Lakemeyer and Lespérance (abbreviated as LL from now on) proposed a syntactic restriction on an agent's knowledge base that focuses on collections of nested modal literals [11], specifically *Restricted Modal Literals* (RMLs) that are modal literals that do not contain conjunction, disjunction, $\top$, or $\bot$, and where negation appears only in front of propositional variables. A *Proper Epistemic Knowledge Base* (PEKB) is defined as a set of RMLs consistent under the logic K_n.

LL build on Bienvenu's work on Prime Implicate Normal Form (PINF) [1] and describe how to compile a PEKB into PINF to perform queries efficiently on the knowledge base. The query mechanism that they describe is sound and complete for certain restricted classes of query, and can be made in polynomial time with respect to the size of the knowledge base. The downside is that compiling into PINF takes exponential time, and results in a PINF formula that is exponentially larger than the initial knowledge base. If one is willing to pay the upfront cost of compiling into PINF, then querying can be done efficiently. LL also extend the work to the logic $\mathrm{K45}_n$ (i.e. introducing positive and negative introspection).

We extend the work of LL to consider the logics KD_n and $\mathrm{KD45}_n$, which imply that every agent's belief represented in a PEKB is consistent; i.e. no agent can derive $\bot$ from their beliefs. Moving from K_n to KD_n simplifies reasoning about PEKBs. We present an approach for checking entailment in polynomial time, as is the case with K_n, which additionally does not require pre-processing such as the expensive compilation into PINF.

Moving to KD_n has the advantage that the knowledge base itself is *logically separable*, which means informally that anything entailed by a PEKB is entailed by a single RML in that PEKB. Consider this example PEKB from LL [11, pg. 1]:

$$\{\ \Box_1\Diamond_1 p,\ \ \Diamond_1\Box_1 p,\ \ \Box_1\Box_1\neg p\ \}$$

This PEKB is not satisfiable. Under K_n, to derive the inconsistency requires reasoning about the first two RMLs together to derive that they are inconsistent with the third. However, under KD_n, the third is inconsistent with *either* of the first two RMLs individually. In

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

this paper, we show that this separability property holds for any PEKB under KD_n.

In addition, we propose an alternative approach to reasoning about KD_n PEKBs that first computes the deductive closure of the PEKB, and then uses simple set membership to check entailment. The resulting closure is exponential in the maximum depth of nested belief in the original PEKB, but by using simple techniques such as hashing, the average time complexity for querying if an RML is entailed is constant while the worst-case time complexity is linear in the size of the compiled PEKB. This approach allows classes of epistemic reasoning problems to be encoded as propositional problems, and then fed into propositional reasoning tools. For example, a planning problem involving epistemic modalities could be compiled into a classical planning problem and fed into an existing planner as an off-the-shelf blackbox [12]. Also, we present a simple extension to the logic $KD45_n$, in which $KD45_n$ problems are compiled into KD_n problems in a manner similar to how LL compile $K45_n$ problems into K_n problems.

The contributions of our work are three-fold: (1) we extend the work of LL to handle KD_n instead of K_n; (2) we demonstrate how to use a simplified entailment mechanism for a PEKB in KD_n both with and without a prior compilation step that closes the PEKB deductively; and (3) we evaluate the various query types to assess both the size of the compiled knowledge base and the time to answer queries.

In the following section we provide the necessary background notation for the paper, as well as describe LL's approach for compiling a PEKB to PINF and querying the resulting knowledge base. In Section 3 we extend LL's approach to consider the KD_n logic and discuss options for querying a KD_n knowledge base. We empirically demonstrate the difference between the query mechanisms in Section 4, and conclude with a summary and discussion of future work in Section 5.

2. EPISTEMIC AND DOXASTIC LOGIC

Our larger research goals are to derive ways for reasoning about knowledge and action in the presence of others. For this, we want an agent to be able to represent what others believe (or know) about their world, including what they believe about what we and other agents believe: i.e., *nested belief*.

Epistemic and doxastic[1] logics are useful for this. In particular, we are interested in the modal logic $KD45_n$, which we present here briefly (for a more complete treatment, see Fagin et al. [5]).

2.1 Epistemic and Doxastic Modal Logics

Let $\mathcal{P}$ and Ag respectively be finite sets of propositions and agents. The set of well-formed formulae, $\mathcal{L}$, for epistemic logic is obtained from the following grammar:

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid \Box_i\phi$$

in which $p \in \mathcal{P}$ and $i \in Ag$. Informally, the *modal operator* $\Box_i\phi$ means that agent i believes ϕ. Note that the grammar permits statements of the form $\Box_i\Box_j\Box_k p$, meaning that agent i believes that agent j believes that agent k believes that p is true. Such nestings can be arbitrarily long, and we use $depth(\phi)$ to refer to the maximum number of such nestings in the formula ϕ.

The semantics are given using *Kripke structures* [5]. Each Kripke structure is a tuple $M = (\mathcal{W}, \pi, R_1, \ldots R_n)$, in which W is the set of all worlds considered in a model, $\pi \in \mathcal{W} \rightarrow 2^{\mathcal{P}}$ is a function that maps each world to the set of propositions that hold in that world, and each $R_i \subseteq \mathcal{W} \times \mathcal{W}$ (for each $i \in Ag$) is a belief accessibility relation. Each modelled agent has an accessibility relation, and this relation captures the agent's uncertainty about the world such that for the actual world, w, the set $R_i(w)$ is the set of worlds that agent i considers possible: $R_i(w) = \{w' \mid R_i(w, w')\}$.

Given these definitions, the satisfaction of a formula ϕ in a Kripke structure M and a world w is denoted as $M, w \vDash \phi$, and it is defined inductively over the structure of ϕ:

$$\begin{array}{lll} M, w \vDash p & \text{iff} & p \in \pi(w) \\ M, w \vDash \varphi \wedge \psi & \text{iff} & M, w \vDash \varphi \text{ and } M, w \vDash \psi \\ M, w \vDash \neg\varphi & \text{iff} & M, w \nvDash \varphi \\ M, w \vDash \Box_i\varphi & \text{iff} & \text{for all } v \in R_i(w),\ M, v \vDash \varphi \end{array}$$

We define entailment as: $\phi \vDash \psi$ if and only if for every model M and world w such that $M, w \vDash \phi$, we have $M, w \vDash \psi$.

Additional operators for $\vee$, $\supset$, and $\equiv$ can be derived in the usual way, as can $\top$ (true) and $\bot$ (false). Further, we use a second modal operator, $\Diamond_i\phi$, which indicates that i believes that ϕ is *possibly* true. This is defined as $\Diamond_i\phi \equiv \neg\Box_i\neg\phi$; that is, agent i believes ϕ is possibly true if and only if it does not believe that $\neg\phi$ is true.

As discussed by Fagin et al. [5], placing certain constraints on Kripke structures leads to specific properties of knowledge or belief, which can be represented as axioms of the logic. We are particularly interested in $KD45_n$ systems, where n specifies that there are multiple agents in the environment, and $KD45_n$ corresponds to the axioms named K, D, 4, and 5. The axiom K holds for any standard modal logic, while the axioms D, 4, and 5 hold if the Kripke structures are *serial*, *transitive*, and *Euclidean* respectively:

K	$\Box_i(\phi \supset \psi)$	$\supset$	$(\Box_i\phi \supset \Box_i\psi)$	(Distribution)
D	$\Box_i\phi$	$\supset$	$\neg\Box_i\neg\phi$	(Consistency)
4	$\Box_i\phi$	$\supset$	$\Box_i\Box_i\phi$	(Positive introspection)
5	$\neg\Box_i\phi$	$\supset$	$\Box_i\neg\Box_i\phi$	(Negative introspection)

Such a logic is able to represent expressive statements about the world and the belief of those agents within it.

EXAMPLE 1. *Suppose there are two agents, 1 and 2, who are coworkers. Agent 1 is aware that agent 2 has applied for a promotion. Agent 1 sees an envelope from the human resources department containing the outcome, but has no information about whether agent 2 has opened the envelope. However, agent 1 believes that if agent 2 has opened the envelope, then agent 2 will have formed a belief about whether she has gained her promotion. Agent 1 also believes that if agent 2 has not opened the letter, she will not have formed a belief on this. Assuming that 'opened' and 'promoted' are propositions representing that agent 2 has opened the letter and has been promoted respectively, we can represent the above using modal logic as follows:*

$$\Box_1(opened \supset (\Box_2 promoted \vee \Box_2\neg promoted)) \wedge$$
$$\Box_1(\neg opened \supset \neg(\Box_2 promoted \vee \Box_2\neg promoted))$$

Ladner [10] showed that the satisfiability problem in $KD45_n$ (for a single agent) is NP-complete, while Halpern and Moses [7] demonstrated that for multiple agents, it becomes PSPACE-complete. Later, Halpern showed that if the depth of nesting in modal formulas is bound, the problem is NP-complete [6].

To overcome the complexity of modal logics, Bienvenu [1, 2] presents an algorithm for re-writing both a K_n modal knowledge base and query into a specific normal form called *prime implicate normal form* (PINF), from which entailment of the query from the knowledge base can be checked in polynomial time.

[1]For simplicity, we will follow convention and use "epistemic" to refer to both knowledge and belief throughout the paper.

DEFINITION 1 (MODAL LITERALS, TERMS, AND CLAUSES). *Bienvenu [1] defined the notions of* modal literal *l, a* term *t*[2]*, and* clause *c, which offer desirable properties for compiling and querying knowledge bases:*

$$l ::= \top \mid \bot \mid p \mid \neg p \mid \Box_i \phi \mid \Diamond_i \phi$$
$$t ::= l \mid t \wedge t$$
$$c ::= l \mid c \vee c$$

in which ϕ is any formula in $\mathcal{L}$ where negation appears only in front of propositional variables; that is, ϕ is in negation normal form *(NNF).*

DEFINITION 2 (PRIME IMPLICATE). *A clause c is a* prime implicate *of a formula ϕ if and only if: (1) $\phi \vDash c$; and (2) if $\phi \vDash d$ and $d \vDash c$, for any clause d, then $c \vDash d$ (i.e., c is minimal).*

DEFINITION 3 (PRIME IMPLICATE NORMAL FORM). *As defined by Bienvenu [1], a formula ϕ is in* prime implicate normal form *(PINF) if and only if one of the following holds:*

1. *ϕ is $\top$ or $\bot$; or*
2. *$\phi \nvDash \bot$ and $\top \nvDash \phi$ and $\phi = c_1 \wedge \ldots \wedge c_p$ where:*
   - *(a) $c_i \nvDash c_j$ for $i \neq j$ — that is, there are no redundant conjunctions;*
   - *(b) any prime implicate of ϕ is equivalent to some c_i;*
   - *(c) every c_i is a prime implicate of ϕ such that the following hold:*
     - *i. if d is a disjunct of c_i, then c_i is not logically equivalent to $c_i \setminus \{d\}$ — that is, there are no redundant disjuncts in c_i;*
     - *ii. there is at most one disjunct in c_i of the form $\Box_i \psi$ for every agent i;*
     - *iii. for every disjunct in c_i of the form $\Box_i \psi$ or $\Diamond_i \psi$, ψ is in PINF; and*
     - *iv. for disjuncts $\Box_i \psi'$ and $\Diamond_i \psi''$ in c_i, then $\psi' \vDash \psi''$.*

Essentially, a formula in PINF contains exactly the amount of knowledge it requires to check entailment using a structural subsumption algorithm, and no more. For example, in the logic K_n, the formula $\Box_1 \Diamond_1 p \wedge \Diamond_1 \Box_1 p$ is not in PINF because $\Diamond_1 \Diamond_1 p$ is a prime implicate of this formula, but is not a conjunction in the formula, violating item 2(b). However, in the logic KD_n, the formula is in PINF, because $\Diamond_1 \Diamond_1 p$ is a 'non-prime' implicate: it is derivable from either $\Box_1 \Diamond_1 p$ or from $\Diamond_1 \Box_p$ using axiom D.

Bienvenu shows that K_n knowledge bases in PINF have desirable properties, such as polynomial time entailment querying [1, 2]. However, the PINF itself is double-exponential in the length of the original knowledge base.

2.2 Proper Epistemic Knowledge Bases

LL [11] investigate the computational complexity of restricted class of K_n epistemic knowledge bases by restricting the syntax for the logic used to store and query knowledge. These knowledge bases, called *proper epistemic knowledge bases* (PEKBs), consist of a set of *restricted modal literals* (RMLs), ϕ satisfying:

$$\phi ::= p \mid \neg p \mid \Box_i \phi \mid \Diamond_i \phi$$

Note that RMLs are in *negation normal form* (NNF), i.e. negation appears only in front of propositional variables. Any formula comprising an arbitrary string of modal operators terminated with a literal can be converted into NNF through the following equivalences:

$$\neg \Box_i \phi \equiv \Diamond_i \neg \phi \qquad \neg \Diamond_i \phi \equiv \Box_i \neg \phi \qquad \neg \neg p \equiv p$$

We use $Lit(\phi)$ to refer to the literal at the end of the RML ϕ:

$$Lit(\phi) = \begin{cases} Lit(\psi) & \text{if } \phi = \Box_i \psi \text{ or } \phi = \Diamond_i \psi \\ \phi & \text{otherwise} \end{cases}$$

This logic is not as expressive as the logic presented in Section 2.1. For example, the formula in Example 1 cannot be expressed because it contains disjunctions. However, it is still expressive enough to be useful in many scenarios, as argued by LL [11], who present an example of a recommender system for making movie recommendations based on others with similar tastes [13].

Querying whether a formula follows from a set of RMLs is not a simple set membership query. First, the PEKB may be inconsistent, in which case any query should return true for any query formula. Second, a query can be a disjunction, conjunction, $\top$, $\bot$, or negation of a modal term, none of which can be in the PEKB. Third, queries may be deducible from a combination of formula. For example, the PEKB $\{\Box_i \Diamond_i p, \Diamond_i \Box_i p\}$ is consistent, and the RML $\Diamond_i \Diamond_i p$ follows from the conjunction of the two formula in the set, but not from either one individually. In terms of Kripke semantics, this is because neither of the RMLs ensures that there is a path of length two from the current world to a world in which p holds (because $\Box_i p$ holds if there are no accessible worlds from the current world). However, together they imply that there is at least one such path.

To overcome these problems, LL follow the work of Bienvenu [2] by compiling PEKBs into prime implicate normal form and restricting queries for a specific normal form called $\mathcal{NF}$.

Converting PEKBs to PINF

LL first define the following abbreviations, where ϕ is a PEKB:

- $B_i(\phi) = \{\psi \mid \Box_i \psi \in \phi\}$
- $D_i(\phi) = \{\psi \mid \Diamond_i \psi \in \phi\}$
- $Prop(\phi) = \{l \mid l \text{ is a non-modal literal and } l \in \phi\}$

Then, they define the function $PEKB2PINF(\phi)$, which takes a non-empty PEKB, ϕ, and returns a *simple formula* in PINF that is equivalent to the conjunction of RMLs in ϕ. A simple formula is a formula in NNF that does not contain any disjunction.

DEFINITION 4 ($PEKB2PINF$). *Lakemeyer and Lesperence [11] define $PEKB2PINF(\phi)$ as:*

1. *Let $F_i(\phi) = \bigwedge_{\psi \in B_i(\phi)} \psi$*

   Let $\Delta(\phi) = Prop(\phi) \cup$
   $\{\Diamond_i(\psi \wedge F_i(\phi)) \mid \psi \in D_i(\phi) \wedge B_i(\phi) \neq \emptyset\} \cup$
   $\{\Diamond_i(\psi) \mid \psi \in D_i(\phi) \wedge B_i(\phi) = \emptyset\} \cup$
   $\{\Box_i(F_i(\phi)) \mid B_i(\phi) \neq \emptyset\}$
2. *For each $l \in \Delta(\phi)$, if l is of the form $\Diamond_i(\psi)$, replace it by $\Diamond_i(PEKB2PINF(\psi))$ and if l is of the form $\Box_i(\psi)$, replace it by $\Box_i(PEKB2PINF(\psi))$.*
3. *If either $\Diamond_i \bot$ is in $\Delta(\phi)$ or if both p and $\neg p$ is in $\Delta(\phi)$, return $\bot$, otherwise return $\bigwedge_{\psi \in \Delta(\phi)} \psi$.*

[2]Bienvenu calls this a *cubal*, but we follow LL's terminology [11].

Essentially, step 1 of the algorithm combines all formulae that begin with $\Box_i$ into a single operator; e.g. $\Box_i p$ and $\Box_i \Diamond_j q$ are combined into $\Box_i(p \wedge \Diamond_j q)$; and infers new knowledge that $\Diamond_i(\phi \wedge \psi)$ holds if $\Box_i \phi$ holds, and agent i believes something is possible ($\Diamond_i \psi$), because from $\Diamond_i \psi$, we know that there is at least one accessible world on our Kripke frame, therefore, if ϕ holds for all accessible worlds, it must also hold for that single world. From $\Box_i \phi$ alone, we do not know if there are any accessible worlds, so this inference occurs only if $\Diamond_i \psi$ also holds. Step 2 ensures that formulae inside modal operators are in PINF, and step 3 checks for consistency of the PEKB.

LL show that the worst-case execution time of $PEKB2PINF$ is $O(|\phi|^{d+2})$. Steps 1 and 3 compare all pairs of RMLs in the PEKB, which is at worst $O(|\phi|^2)$. The recursive calls in step 2 depend on the depth of the modal operators in the formula, therefore, for a maximum depth of d, the worst case of all steps is $O(|\phi|^{d+2})$.

They also show that the size of a PINF is at most exponential in the depth of the PEKB ϕ, and provide an example where this is the case [11, cf. Theorems 2 and 3].

Query evaluation

LL then define a structural subsumption algorithm, V, for querying whether a given formula, ϕ, in NNF is entailed by a formula, Σ, in PINF. A call to $V[\Sigma, \phi]$ returns 1 for a "yes" and 0 for "don't know", where 0 indicates that ϕ could not be proved from the Σ.

DEFINITION 5 ($V[\Sigma, \phi]$). *LL [11] define $V[\Sigma, \phi]$ inductively on the structure of ϕ. Σ is a simple formula in PINF, and $\phi \in \Sigma$ means that ϕ is one of the conjuncts in Σ.*

1. $V[\Sigma, \phi] = 1$ *if* $\Sigma = \bot$*; otherwise:*
2. $V[\Sigma, \top] = 1$*;*
3. $V[\Sigma, \bot] = 0$*;*
4. $V[\Sigma, p] = \begin{cases} 1 & \text{if } p \in \Sigma \\ 0 & \text{otherwise;} \end{cases}$
5. $V[\Sigma, \phi \vee \psi] = \max(V[\Sigma, \phi], V[\Sigma, \psi])$*;*
6. $V[\Sigma, \phi \wedge \psi] = \min(V[\Sigma, \phi], V[\Sigma, \psi])$*;*
7. $V[\Sigma, \Box_i \phi] = \begin{cases} 1 & \text{if for some } \Box_i \Sigma' \in \Sigma,\ V[\Sigma', \phi] = 1 \\ 0 & \text{otherwise;} \end{cases}$
8. $V[\Sigma, \Diamond_i \phi] = \begin{cases} 1 & \text{if for some } \Diamond_i \Sigma' \in \Sigma,\ V[\Sigma', \phi] = 1 \\ 0 & \text{otherwise;} \end{cases}$

LL note that the running time of a naive implementation of this algorithm is $O(n^2)$, in which n is the size of the knowledge base. However, with some indexing on the agent identities and modal operators, this can be reduced to $O(n \log n)$.

They further show that when the knowledge base is converted to PINF, their approach is sound for queries in NNF and complete for queries in a restricted normal form called $\mathcal{NF}$ that includes all queries PINF and more. To define $\mathcal{NF}$, they first define the concept of *logical separability*.

DEFINITION 6 (LOGIC SEPARABILITY [11]). *The set of formulae P is* logically separable *if and only if for every consistent set of simple formulae P' the following holds:*

$$\text{if} \quad P \cup P' \models \bot \quad \text{then} \quad \exists \phi \in P, \text{ s.t. } P' \cup \{\phi\} \models \bot$$

Intuitively, logical separability of the set of formulae P indicates that there is no interaction between the logical statements in P. That is, we cannot infer anything by combining two or more statements that are not inferable from a single statement. For example, the set $\{\Box_i p, \Box_i(p \supset q)\}$ is not logically separable, because we can infer $\Box_i q$ from the combination of the two, yet $\Diamond_i \neg q$ is consistent with each of the formulae in the set. This concepts plays an important role later in the paper when considering only consistent knowledge bases. Because we define logical separability using entailment, we will assume the logic used is obvious from the context (i.e., K_n here and KD_n in the following section).

DEFINITION 7 ($\mathcal{NF}$ [11]). *$\mathcal{NF}$ is the least set such that:*

1. *$p \in \mathcal{NF}$, for all propositional variables p;*
2. *if $\phi \in \mathcal{NF}$ then $\neg\phi \in \mathcal{NF}$;*
3. *if $\phi \in \mathcal{NF}$ then $\Box_i \phi \in \mathcal{NF}$; and*
4. *if $\Gamma \subseteq \mathcal{NF}$, Γ is finite and logically separable, then $\bigwedge_{\phi \in \Gamma} \phi \in \mathcal{NF}$.*

3. QUERYING PDKBs

LL's approach to PEKBs is an approximation of the logic K_n, and they show how to add positive and negative introspection (axioms 4 and 5) in a straightforward manner. In our work, we are concerned with the logics KD_n and $KD45_n$, in which the axioms of $K(45)_n$ are extended with axiom D: that agents cannot have inconsistent beliefs.

In this section, we present three query evaluation algorithms to check entailment of a KD_n PEKB, which we call a *proper doxastic knowledge base* (PDKB) – we will use the symbol P in lieu of ϕ to distinguish the fact that P is a PDKB. All three algorithms assume that the PDKB is consistent prior to the query, and so would require that the knowledge base be constructed using a sound belief change operation. Assuming a consistent PDKB is not strictly required, but simplifies the exposition. Including axiom D results in properties that simplify the complexity of the algorithms relative to the K_n case. The three algorithms that we propose are:

1. *Entailment as structural subsumption*: This is a simple extension to LL's approach, which takes a PDKB and puts it into a specific normal form, after which LL's algorithm $V[\Sigma, \phi]$ (cf. Definition 5) can be used as a sound entailment algorithm.
2. *Entailment without compilation*: We show a polynomial-time algorithm for querying a PDKB without first compiling into a specific form, such as the expensive conversion to INF. This algorithm is possible due to the consideration of axiom D.
3. *Entailment as closure*: We present an algorithm for compiling the PDKB into its *closure*, which is the set of RMLs that are in the deductive closure of the PDKB. As a result, entailment is simple set lookup. The average case time complexity using a hashed set is constant, while the worst case is linear in the size of the compiled knowledge base (which may be exponential in the maximum depth).

3.1 Entailment as structural subsumption

In this section, we show a simple extension to LL's approach for querying PDKBs to account for axiom D. We present a new algorithm for converting a PDKB into *implicate normal form* (INF). By

this, we mean that the resulting formula obeys the items in Definition 3, except item 2(a): not all implicates in an INF are necessarily *prime*. The difference in the algorithm is that we assume that the PDKB is consistent, and that the D axiom can be used to infer new knowledge. We call this algorithm $PDKB2INF$.

DEFINITION 8 ($PDKB2INF$). *Assume P is a consistent PDKB.*

1. *Let* $F_i(P) = \bigwedge_{\psi \in B_i(P)} \psi$
   Let $\Delta(P) = Prop(P) \cup$
   $\{\Diamond_i(\psi) \mid \psi \in D_i(P) \cup B_i(P)\} \cup$
   $\{\Box_i(F_i(P)) \mid B_i(P) \neq \emptyset\}$
2. *For each $l \in \Delta(P)$, if l is of the form $\Diamond_i(\psi)$, replace it by $\Diamond_i(PDKB2INF(\psi))$ and if l is of the form $\Box_i(\psi)$, replace it by $\Box_i(PDKB2INF(\psi))$.*
3. *Return* $\bigwedge_{\psi \in \Delta(P)} \psi$.

There are two differences between the algorithms $PDKB2INF$ and $PEKB2PINF$: (1) the previous step 3 is simplified: because it is assumed that ϕ is consistent, neither $\bot$ nor $\Diamond_i \bot$ can be derived, so we simply return the conjunction of formula; and (2) step 1 is simplified due to the assumption of axiom D. Due to axiom D, we know that $\Box_i \phi \supset \Diamond_i \phi$, so if an RML $\Box_i \phi$ exists in the PDKB, we do not require another RML $\Diamond_i \psi$ (for some ψ) to be able to infer that $\Diamond_i \phi$.

Note that the result is not (necessarily) in PINF, because we infer $\Diamond_i \phi$ from $\Box_i \phi$, but $\Diamond_i \phi$ is not a prime implicate. We opt not to reduce to PINF because this would require a separate post-processing step to remove formula such as $\Diamond_i \phi$. For example, consider the PDKB $\{\Box_i \Box_i p, \Diamond_i \Diamond_i p\}$. Because the second formula follows from the first, it is non-prime so should be removed. However, to do this would require an additional algorithm after step 3 that recursively parses the INF to remove non-prime implicates. It cannot be done top-down in $PDKB2INF$ (e.g. using $\{\Diamond_i(\psi) \mid \psi \in D_i(P) \setminus B_i(P)\}$ in step 2), because at the top level for $\Diamond_i \phi$, ϕ is not in $B_i(P)$, requiring compilation of all subformula into PINF, and then checking for all pairs of formula $\Box_i \phi$ and $\Diamond_i \psi$, whether $\phi \supset \psi$.

Adapting LL's proof (which is itself an adaption of Bienvenu's [2]), it is straightforward to see that this algorithm terminates, and produces a formula in INF that is equivalent to the conjunction of RMLs in P. The worst-case execution time for $PDKB2INF$ is $O(|P|^d)$. Due to the fact that this algorithm does not compare pairs of literals as $PEKB2PINF$ does, the 'non-recursive' case is $O(|P|)$. There will be at most d such calls, where d is the depth of the modal operators, meaning that the overall complexity is $O(|P|^d)$.

Following the compilation done by $PDKB2INF$, our first method to query if a PDKB P entails ϕ is to test the following using LL's query algorithm $V[\Sigma, \phi]$ from Definition 5:

$$V[PDKB2INF(P), \phi] = 1$$

V remains sound for PDKBs, and also complete for queries in $\mathcal{NF}$. This result is interesting, as it shows that adding axiom D into our theory results in a lower time complexity to compile an INF formula than a PINF formula, while the complexity of the query remains $O(|P|^2)$.

EXAMPLE 2. *Returning to our example, consider the potential knowledge base of agent 1 where they become aware of the fact that $\neg opened$ holds. The initial PDKB to represent this would be:*

$$P = \{\Box_1 \neg opened, \Box_1 \Diamond_2 promoted, \Box_1 \Diamond_2 \neg promoted\}$$

The procedure $PDKB2INF(P)$ would then result in the following:

$$\Box_1(\neg opened \wedge \Diamond_2 promoted \wedge \Diamond_2 \neg promoted) \wedge$$
$$\Diamond_1 \neg opened \wedge \Diamond_1 \Diamond_2 promoted \wedge \Diamond_1 \Diamond_2 \neg promoted$$

3.2 Entailment without compilation

In the previous section, the $PDKB2INF$ algorithm computes an INF formula in $O(|P|^d)$ time because it only infers new knowledge from single RMLs using axiom D. This indicates an alternative solution in which we only calculate this new knowledge on demand, and eliminate the compilation phase. We consider such a case in this section by relying on the following property:

THEOREM 1 (LOGICAL SEPARABILITY OF PDKBS). *Every consistent PDKB is logically separable.*

Proof. The logical separability of consistent PDKBs can be seen from the definitions of $V[\Sigma, \phi]$ and $PDKB2INF$.

From Definition 6, it is clear that for a set of formula P, and for all formula ϕ such that $P \models \phi$, if ϕ is derivable from a single formula in P, then P is logically separable. We show that this is the case for PDKBs: when we derive the INF formula for a PDKB and then query this formula, we use only single RMLs to derive any new knowledge.

First, consider $PDKB2INF$. It is clear that the only new knowledge is derived from this algorithm is in step 2, where we apply axiom D. However, axiom D is applied only to single RMLs of the form $\Box_i \phi$. This is unlike the original $PEKB2PINF$, which combines two formula at this step.

This leaves V from Definition 5. However, it is clear that this structural subsumption algorithm derives no new knowledge. ■

This result is significant, and gives rise to many desirable properties that simplify reasoning about PDKBs. For example, we do not need to calculate a normal form such as INF to query a knowledge base, because there are no logical puzzles arising from the combination of two formula in a consistent PDKB. Second, to check entailment of an RML, we need to find only a single RML in the knowledge base that entails the formula, allowing us to consider the parts of P in isolation. More specifically, we can take advantage of the following corollary of Theorem 1:

COROLLARY 1 (SINGLE RML ENTAILMENT). *For a consistent PDKB P and RML l, the following holds:*

$$P \models l \quad \textit{iff} \quad \exists m \in P, m \models l$$

Corollary 1 leads us naturally to consider a querying mechanism that looks at each individual RML in a PDKB in turn, rather than viewing the PDKB as a whole. Operating under the KD_n system (axioms K and D), $m \models l$ if and only if there is a proof from m to l under the KD_n axioms. Because m and l are RMLs, we can rule out general applications of axiom K, because RMLs cannot contain implications. Thus, the only axiom applied to an RML is D ($\Box_i \phi \supset \Diamond_i \phi$), and the only application of K is to apply D (which is an implication) to a single RML. Informally, this means that $m \models l$ if and only if we can get from m to l by repeatedly applying axiom D, and $P \models l$ if and only if there is at least one $m \in P$ such that $m \models l$.

DEFINITION 9 ($V_{RML}[P, \phi]$). *First, we define $V_{RML}[m, l]$, in which m and l are both RMLs, as the function that returns 0 (*false*) if one of the following conditions are met, and returns 1 (*true*) otherwise:*

1. *$Lit(m)$ and $Lit(l)$ differ.*

2. *The sequence of agent identifiers on the modalities in m and l differ.*

3. *There exists an agent i and index $k \leq depth(l)$ such that the k^{th} modal operator in m is $\Diamond_i$ and the k^{th} modal operator in l is $\Box_i$.*

Thus, $V_{RML}[m, l] = 1$ if and only if the sequences of agent identifiers on the nested modalities are the same, the literals at the end of the respective sequences are the same, and at each index of the sequence, the modalities in m are the same or stronger than the modality in l ($\Box_i$ implies $\Diamond_i$). We can now define the query for any clause or term ϕ on a PDKB P as:

1. $V_{RML}[P, l] = \begin{cases} 1 & \text{if for some } m \in P,\ V_{RML}[m, l] = 1 \\ 0 & \textit{otherwise}; \end{cases}$

2. $V_{RML}[P, \phi \vee \psi] = \max(V_{RML}[P, \phi],\ V_{RML}[P, \psi])$;

3. $V_{RML}[P, \phi \wedge \psi] = \min(V_{RML}[P, \phi],\ V_{RML}[P, \psi])$;

When restricted to a literal l, checking entailment involves looking through the PDKB to find an RML m such that m entails l from axiom D. For the non-literal case, a clause (term) is entailed if either (both) of the sub-clauses (sub-terms) is entailed.

THEOREM 2. *Let ϕ be in $\mathcal{NF}$. $V_{RML}[P, \phi] = 1$ iff $P \models \phi$.*

Proof. First, we need to establish the soundness and completeness of $V_{RML}[m, l]$ when m and l are both RMLs: $V_{RML}[m, l] = 1$ iff $m \models l$. The left to right case is straightforward: if the propositional literals match, the sequence of agent identifiers match, and the modal operators are always "box to diamond" from left to right (for example $V_{RML}[\Box_1\Diamond_1 p, \Diamond_1\Diamond_1 p]$), then from axiom D, it must be that $m \models l$.

We prove the right to left case by contradiction. Define $M_m = \{M \mid M \models m\}$ and $M_l = \{M \mid M \models l\}$. Assume that $m \models l$. Therefore, $M_m \subseteq M_l$. Now, if the right to left case does not hold, then it must be that $V_{RML}[m, l] = 0$, and therefore one of the following must hold: (1) $Lit(m)$ and $Lit(l)$ differ, which cannot be the case because it is straightforward to construct a model M such that $Lit(m) \wedge \neg Lit(l)$ holds in every world, therefore $M_m \not\subseteq M_l$; (2) the agent sequence of identifiers differ, which cannot be the case, because different agents can believe different propositions so it is straightforward to construct a model for any such formula such that $M_m \not\subseteq M_l$; and (3) agent identifiers agree, but at one index m contains $\Diamond_i$ and l contains $\Box_i$, which cannot be the case because it is straightforward to construct a model such that $\Diamond_i l$ holds and $\Box_i l$ does not, and therefore $M_m \not\subseteq M_l$.

From this and Collorary 1, we have that $V_{RML}[P, l]$ iff $P \models l$ where l is an RML.

Next, we need to show cases for terms and clauses, which we do by induction on the size of ϕ. Assume $P \models \phi \wedge \psi$. This holds iff $P \models \phi$ and $P \models \psi$, which by induction, holds iff $V_{RML}[P, \phi] = 1$ and $V_{RML}[P, \psi] = 1$, which holds iff $V_{RML}[P, \phi \wedge \psi] = 1$.

For the clausal case, the left to right case is analagous to the case above for terms. For the right to left case, we follow LL (Theorem 6). Assume $P \models \phi \vee \psi$. From this, we know that $P \cup \{\neg\phi, \neg\psi\}$ is inconsistent. Since $\phi \vee \psi$ is in $\mathcal{NF}$, we know that the set $\{\neg\phi, \neg\psi\}$ is logically separable. From the definition of logical separability, we know that either $P \cup \{\neg\phi\}$ is inconsistent or $P \cup \{\neg\psi\}$ is inconsistent, which implies that $P \models \phi$ or $P \models \psi$. By induction, this implies that $V_{RML}[P, \phi] = 1$ or $V_{RML}[P, \psi] = 1$, which is equivalent to $V_{RML}[P, \phi \vee \psi] = 1$. ■

The worst-case running time of a naïve implementation of the query algorithm $V_{RML}[P, \phi]$ is $O(|P| \cdot d)$. An implementation of $V_{RML}[P, \phi]$ would be required to iterate over all l in P, and for each l it must iterate along the modalities until it reaches a divergence (as in Definition 9) or reaches the end (a successful entailment), which is at most d iterations. This complexity is similar to the $O(|P|^2)$ complexity for the structural subsumption algorithm.

EXAMPLE 3. *Consider the initial PDKB P provided in Example 2 and the query $V_{RML}[P, \Diamond_1 \neg opened]$. It is clear to see from Definition 9 that $V_{RML}[\Box_1 \neg opened, \Diamond_1 \neg opened]$ will hold, and thus so will $V_{RML}[P, \Diamond_1 \neg opened]$. On the other hand, consider $V_{RML}[P, \Box_1\Box_2 \neg promoted]$. The result is false, as every RML fails to entail the query:*

1. *$V_{RML}[\Box_1 \neg opened, \Box_1\Box_2 \neg promoted]$ fails because the sequence of agent identifiers differ (Def. 9, condition 2)*

2. *$V_{RML}[\Box_1\Diamond_2 promoted, \Box_1\Box_2 \neg promoted]$ fails because the literals are different (Def. 9, condition 1)*

3. *$V_{RML}[\Box_1\Diamond_2 \neg promoted, \Box_1\Box_2 \neg promoted]$ fails because the query RML is strictly stronger (Def. 9, condition 3)*

3.3 Entailment as closure

In this section, we present a method for calculating the closure of an entire PDKB, which then allows us to check entailment simply as membership of the closure. This is useful in situations where an underlying reasoning engine cannot handle nested modalities, such as tools for reasoning about propositional knowledge bases. For example, in recent work we defined an automated planning approach for deriving plans in the presence of other agents, and this uses nested modalities to represent the mental states of other agents [12]. This approach encodes the multi-agent epistemic problem description into a classical planning problem. Thus it must remove any fluent that conflicts with new information, and further maintain a state that is closed. Following the intuition behind Corollary 1, we can define the closure of a single RML as follows:

DEFINITION 10 (RML CLOSURE). *Given an RML l, we define $Cl(l)$ to be the smallest set such that $l \in Cl(l)$ and,*

$$\textit{If } \mathcal{X}\Box_i m \in Cl(l), \textit{ then } \mathcal{X}\Diamond_i m \in Cl(l)$$

in which $\mathcal{X}$ is a string of modal operators and m is an RML.

The Cl operator efficiently captures the deductive closure of a single RML, and as we have seen from Corollary 1, this can be a powerful technique when Cl is complete.

THEOREM 3 (RML CLOSURE COMPLETENESS). *For any pair of RMLs l and m, the following holds:*

$$m \models l \Rightarrow l \in Cl(m)$$

Proof. This follows from the correspondence of Definition 10 to the $PDKB2INF$ algorithm – we use the axiom D to replace one or more $\Box$ operators with $\Diamond$ operators. ■

To close an entire PDKB P, we simply close each RML in P:

DEFINITION 11 (PDKB CLOSURE). *The* closure *of a PDKB P, denoted as $Cl(P)$, is the smallest set such that $P \subseteq Cl(P)$ and*

$$l \in Cl(P) \Rightarrow Cl(l) \subseteq Cl(P)$$

With a procedure to deductively close a PDKB, we use the following to query if a formula is entailed by the PDKB P.

DEFINITION 12 ($V_{Cl}[P, \phi]$). *We define the query mechanism for a closed PDKB $Cl(P)$ as:*

1. $V_{Cl}[P, l] = \begin{cases} 1 & \text{if } l \in Cl(P) \\ 0 & \textit{otherwise}; \end{cases}$
2. $V_{Cl}[P, \phi \vee \psi] = \max(V_{Cl}[P, \phi], V_{Cl}[P, \psi])$;
3. $V_{Cl}[P, \phi \wedge \psi] = \min(V_{Cl}[P, \phi], V_{Cl}[P, \psi])$;

To query an RML on PDKB, one needs to compute the entire closure of the PDKB, and check that the RML is a member of the closure set. Conjunction and disjuction are defined as in V_{RML}.

THEOREM 4. *Let ϕ be in $\mathcal{NF}$. Then $V_{Cl}[P, \phi] = 1$ iff $P \models \phi$.*

Proof. The case for l holds directly from Theorem 3. The cases for terms and conjunctions are the same as for V_{RML} (proof of Theorem 2). ■

The simplicity of V_{Cl} comes at the expense of increasing the size of the PDKB to be deductively closed – a single RML l will lead to an additional $(2^k - 1)$ RMLs if l contains k $\Box$ operators. The $\Diamond$ operators add no new RMLs. Further, the compilation time is high. For a PEKB P, each element must be closed, with a complexity of $O(2^d)$, where d is the depth of the formula, leading to a worst-case execution time of $O(|P| \cdot 2^d)$. However, an average case query is constant assuming that a simple hash set implementation is used, while the worst case (all hash values are the same), would require iteration over all RMLs in the PEKB.

EXAMPLE 4. *For the PDKB P introduced in Example 4, the closure would be computed by adding new RMLs entailed by P according to Definition 10. This would result in the following PDKB:*

$$\begin{aligned} Cl(P) = \{\ & \Box_1 \neg opened, \Diamond_1 \neg opened, \\ & \Box_1 \Diamond_2 promoted, \Diamond_1 \Diamond_2 promoted, \\ & \Box_1 \Diamond_2 \neg promoted, \Diamond_1 \Diamond_2 \neg promoted\ \} \end{aligned}$$

3.4 Summary

Table 1 summarises the complexity results of the three approaches, and the approach for K_n knowledge bases proposed by LL [11].

Table 1: Summary of complexity results

Algorithm	**Compilation Time**	**Size**	**Query Time**
Subsump (K_n)	$O(\|P\|^{d+2})$	$O(\|P\| \cdot 2^d)$	$O(\|P\|^2)$
Subsump (KD_n)	$O(\|P\|^d)$	$O(\|P\| \cdot 2^d)$	$O(\|P\|^2)$
No compilation	—	$O(\|P\|)$	$O(\|P\| \cdot d)$
Closure	$O(\|P\| \cdot 2^d)$	$O(\|P\| \cdot 2^d)$	$O(1)$ (avg) $O(\|P\| \cdot 2^d)$

From this table, one can see that the closure approach has the lowest query time complexity, but at a high cost of compilation (size and time). One would expect to make many queries on a PDKB to make this cost valuable; although we note that to insert a new consistent RML, we need only to compute the closure of that RML and not the entire knowledge base.

The approach requiring no compilation is also attractive because the complexity of querying is polynomial, yet there are no setup costs. For the structural subsumption approach, the compilation cost is reduced with the addition of axiom D, while the querying cost remains the same.

3.5 Extending to $KD45_n$

Extending to the $KD45_n$ case — that is, adding positive and negative introspection (axioms 4 and 5 respectively) — is straightforward. We note the following theorems under $KD45_n$:

$$\begin{aligned} \Box_i \Box_i \phi &\equiv \Box_i \phi & \Box_i \Diamond_i \phi &\equiv \Diamond_i \phi \\ \Diamond_i \Box_i \phi &\equiv \Box_i \phi & \Diamond_i \Diamond_i \phi &\equiv \Diamond_i \phi \end{aligned}$$

LL define an *i-objective* formula as a formula that is about the world and agents other than i. For example, $\Box_j(p \wedge \Box_i \neg p)$ is i-objective, but $\Box_j p \wedge \Box_i \neg p$ is not. A formula is *i-reduced* iff for all sub-formulae $\Box_i \phi$ and $\Diamond_i \phi$, ϕ is i-objective.

One can see that any formula $\Box_i \phi$ or $\Diamond_i \phi$ can be i-reduced by applying the equivalences above to strip out consecutive occurrences of modal operators of the same agent. Therefore, one can reduce both a $KD45_n$ PDKB and $KD45_n$ query into a KD_n PDKB and KD_n query respectively, thereby allowing application of the approaches in Section 3.

LL present a similar case of extending K_n to $K45_n$. However, the absence of axiom D means that the formula $\Box_i \Diamond_i \phi$ reduces to $\Box_i \bot \vee \Diamond_i \phi$. While a query containing such a formula can be reduced to a K_n formula in $\mathcal{NF}$, a K_n PEKB containing such a formula cannot be reduced to a K_n PEKB because the resulting RML contains a disjunction.

4. EXPERIMENTAL EVALUATION

In this section, we present an empirical study to assess and compare the three query mechanisms introduced in Section 3. This is a valuable evaluation because, while the theoretical worst-case complexity was analysed in Section 3, the average case is likely to differ significantly. For example, even though the closure approach has a bad theoretical worst case, this worst case is highly unlikely and the approach has the best average case.

Experimental setup. We implemented each approach and tested queries of random RMLs on a range of PDKBs. The implementation for each method was completed in Python, and all experiments were conducted on a Linux Desktop with a 3.4GHz processor.

We measured three key aspects for each of the three query methods: (1) the time taken to compile a knowledge base (i.e. into INF or closed form); (2) the size of the compiled knowledge base; and (3) the time taken to answer a random RML query, which may or may not be entailed by the PDKB. The size of the compiled knowledge base for the first query type is the size of the implicate normal form created by $PDKB2INF$ – every modal operator and proposition counts toward the size. For an unclosed or closed PDKB P, the size of P is measured as $\sum_{\phi \in P} depth(\phi) + 1$.

The complexity analysis presented in Section 1 assumes that no indexing is used on the knowledge bases (except for the closure, for which we assume something similar to a hash set is used). However, our implementations of these concepts all use some form of indexing. We include these in the experiments, first, because we believe anyone using such concepts would indeed use indexing, and second, because indexing will have different effects due to the different knowledge base shapes. For the structural subsumption approach, knowledge bases are implemented as hash sets of formula indexed by the agent identifier. This includes sub-formula; for example, $\Box_i(\Box_j p \wedge \Box_k q)$ will index i at the top level, and the sub-formula will also be represented using a hash set with indices j and k. For the closure approach, knowledge bases are implemented as hash sets of RMLs, with each RML indexed by the hash value of its string format. For the approach with no compilation, knowledge bases are implemented as hash tables with keys indexed by agent prefixes. The values are the RMLs that correspond to that prefix.

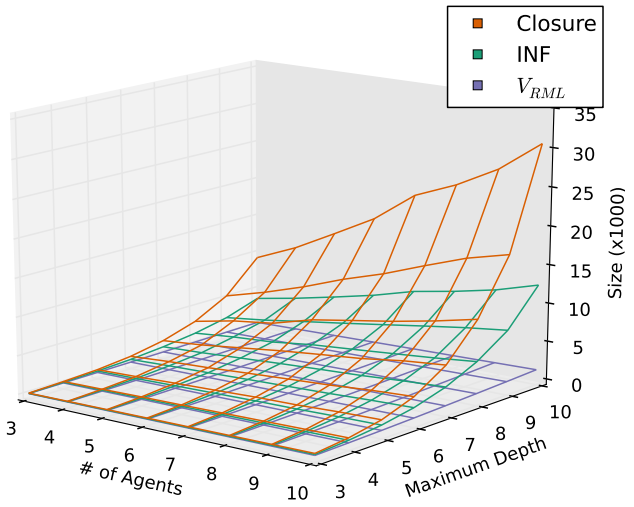

Figure 1: Size of the compiled knowledge bases as a function of the number of agents and maximum depth. Legend indicates the three query approaches from top (largest) to bottom (smallest).

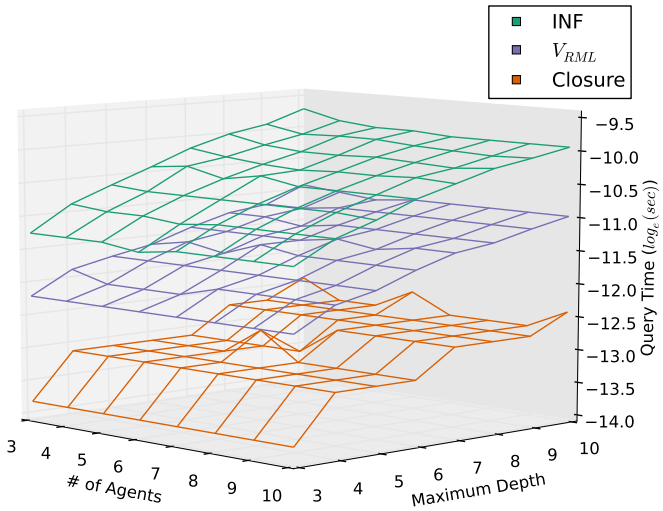

Figure 2: Average query time as a function of the number of agents and maximum depth. Legend indicates the three query approaches from top (slowest) to bottom (fastest).

We generated random PDKBs for a range of parameters that include: (1) the number of agents (ranging from 3 to 10); (2) the maximum depth (3 to 10); and (3) the number of RMLs (20 to 200). The number of RMLs was defined as a function of the maximum depth and number of agents. To generate a PDKB, we inserted random RMLs one after another while continually checking whether or not the resulting PDKB was consistent and skipping RMLs that caused the PDKB to become inconsistent. In total, we generated 100 PDKBs in this manner for every configuration of agents and maximum depth (64 in total), and took the average over the 100 PDKBs generated for each particular parameter setting.

For the queries, we generated 200 random RMLs for each of the 100 PDKBs for a particular parameter setting, half of which were entailed by the PDKB. We measured the query time for each approach, and averaged the result over all queries and PDKBs.

Results. Figure 1 shows a size comparison as a function of the maximum depth and number of agents. We find a strict dominance of $V_{RML} < INF < Closure$ (lower is better). As expected, we also find an exponential increase in the size of both Cl and (to a lesser extent) INF when we scale either the number of agents or the maximum depth. Both follow from the exponential increase in RMLs that would be entailed.

Figure 2 shows the query timing for all three query types. Again, we find a strict dominance between the query types (lower is better): $Closure < V_{RML} < INF$. Note that we have used a log scale for the query time of the three approaches.

Graphs for the compilation time are omitted, however, the shape of the graph is very much similar to Figure 1, except V_{RML} is not present (no compilation required) and the difference between the INF and closure times is not as great.

From these results, one can see that the closure is the fastest, but this is at the expense of a longer compilation time and larger compiled size than the other two. However, one can also see that for logic KD_n, compilation into INF is unnecessary: using an uncompiled knowledge base gives a similar querying time[3] with a smaller size and without an expensive compilation phase.

[3]It is faster in our experiments, but perhaps a similar indexing technique can be used on the INF formulae to reduce this gap.

5. SUMMARY AND DISCUSSION

For an agent reasoning about the nested belief of others, it is crucial to maintain a consistent view of the world. If the agent can believe that a door is both locked and unlocked simultaneously, we cannot expect that agent to reason through an efficient course of action for stepping through the door. In this paper we extended PEKBs, which deal primarily with the logic $K45_n$, to address consistent knowledge bases (i.e., the logic $KD45_n$). We found that adding the axiom D not only provides a natural level of reasoning for an agents mental state, but it also simplifies the computational mechanism required for dealing with proper epistemic knowledge bases. Because of the properties inherent in a PDKB that is presumed to be consistent, we were able to construct two additional query methods for checking entailment; each trading off the storage requirement and response time for answering a query. Through an empirical evaluation of the three query methods, we found that closing a PDKB offered the fastest query time (at the expense of requiring a large amount of space), while keeping the PDKB unclosed resulted in the smallest knowledge base with a query time that was still faster than using implicate normal form.

A key assumption of this work is that the knowledge base is consistent. While it is a straightforward polynomial-time task to check whether or not a PDKB is consistent, in future work we are particularly interested in algorithms for *belief update* and *belief revision*, which maintain a consistent knowledge base. This makes it possible to use knowledge bases as the state representation for a planning process (e.g., building of our current work on encoding PDKBs for automated planning [12]). Additionally, we hope to investigate the feasibility of extending PDKBs to capture limited forms of disjunctive belief to capture concepts such as "know whether", and also to consider how to represent notions of group beliefs.

Acknowledgements This research is funded by Australian Research Council Discovery Grant DP130102825, *Foundations of Human-Agent Collaboration: Situation-Relevant Information Sharing*.

REFERENCES

[1] Meghyn Bienvenu. Prime implicate normal form for ALC concepts. In *AAAI*, pages 412–417, 2008.

[2] Meghyn Bienvenu. Prime implicates and prime implicants: From propositional to modal logic. *Journal of Artificial Intelligence Research*, 36(1):71–128, 2009.

[3] Rolf A Eberle. A logic of believing, knowing, and inferring. *Synthese*, 26(3):356–382, 1974.

[4] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about knowledge*, volume 4. MIT press Cambridge, 1995.

[5] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Y Vardi. *Reasoning about knowledge*, volume 4. MIT press Cambridge, 1995.

[6] Joseph Y Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75(2):361–372, 1995.

[7] Joseph Y Halpern and Yoram Moses. A guide to the modal logics of knowledge and belief: Preliminary draft. In *Proceedings of the 9th international joint conference on Artificial intelligence*, pages 480–490. Morgan Kaufmann Publishers, 1985.

[8] Jaako Hintikka. *Knowledge and belief*, volume 414. Cornell University Press, Ithaca, 1962.

[9] Kurt Konolige. A deductive model of belief. In *IJCAI*, pages 377–381, 1983.

[10] Richard E Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM journal on computing*, 6(3):467–480, 1977.

[11] Gerhard Lakemeyer and Yves Lespérance. Efficient reasoning in multiagent epistemic logics. In *ECAI 2012 - 20th European Conference on Artificial Intelligence.*, pages 498–503, 2012.

[12] Christian Muise, Vaishak Belle, Paolo Felli, Sheila McIlraith, Tim Miller, Adrian Pearce, and Liz Sonenberg. Planning over multi-agent epistemic states: A classical planning approach. In *The 29th AAAI Conference on Artificial Intelligence*, 2015.

[13] Paul Resnick and Hal R. Varian. Recommender systems — introduction to the special section. *Communications of the ACM*, 40(3):56–58, 1997.

[14] Hans van Ditmarsch, Wiebe van der Hoek, and Barteld Pieter Kooi. *Dynamic epistemic logic*, volume 337. Springer, 2007.