Efficient Propositional Constraint Propagation

Mukesh Dalal
Rutgers University
Computer Science Department
New Brunswick, New Jersey 08903
dalal@cs.rutgers.edu

Abstract

We present an efficient method for inferring facts from a propositional knowledge base, which is not required to be in conjunctive normal form. This logically-incomplete method, called propositional fact propagation, is more powerful and efficient than some forms of boolean constraint propagation. Hence, it can be used for tractable deductive reasoning in many AI applications, including various truth maintenance systems. We also use propositional fact propagation to define a weak logical entailment relation that is more powerful and efficient than some others presented in the literature. Among other applications, this new entailment relation can be used for efficiently answering queries posed to a knowledge base, and for modeling beliefs held by a resource-limited agent.

Introduction

Given a particular knowledge base, it is often important to determine all those facts that can be logically inferred. Since the general problem is intractable,[1] many AI systems forsake completeness and infer only those facts that can be efficiently obtained. For instance, most implementations of logical truth maintenance systems [Reiter and de Kleer, 1987] use a weak reasoning method known as boolean constraint propagation [McAllester, 1980]. However, for boolean constraint propagation to be tractable, the knowledge base must first be converted into conjunctive normal form — a transformation which may increase its size exponentially [de Kleer, 1990].

In this paper, we propose a new method for inferring facts from a propositional knowledge base. This method, called *propositional fact propagation* (PFP), does not require the knowledge base (KB) to be in conjunctive normal form (CNF), and is more powerful and efficient than boolean constraint propagation (BCP). For any KB, PFP obtains at least all the facts, and sometimes many more, that can be obtained by BCP on the CNF transformation of the KB. For any KB of size n, PFP takes at most $O(nk)$ time, where k is the maximum nesting of logical connectives in the KB. PFP is incremental, i.e., formulas can be added to the KB without recomputing the earlier results.

Intuitively, PFP infers facts from the simpler subformulas of the KB, and then uses them to infer facts from the more complex formulas that contain these sub-formulas. Whenever possible, it simplifies the subformulas by propagating the facts inferred from them. For example, if the KB consists of a single formula

$$Q \vee (P \wedge (\neg P \vee Q))$$

then PFP simplifies it and infers Q. Note that the CNF transformation of this KB produces

$$\{Q \vee P,\ Q \vee \neg P\}$$

from which BCP cannot obtain the fact Q.[2] By combining many such steps, PFP can perform some subtle reasoning. For example, consider another KB consisting of a single formula

$$\begin{array}{l}((V \vee W) \wedge ((S \wedge R) \vee S)) \ \vee \\ (P \wedge ((Q \wedge \neg P \wedge R) \vee (S \wedge T))) \ \vee \\ (S \wedge (X \vee W))\end{array} \qquad (1)$$

We show later that PFP simplifies this KB to obtain a much simpler KB

$$\{S,\ (P \wedge T) \vee V \vee W \vee X\}$$

Notice that the CNF transformation of the original KB is significantly larger.

The general problem of determining whether a given KB logically entails a given formula is intractable. By using PFP, we define a weak logical entailment relation, $\mid\sim$, that is tractable. Intuitively, for any KB

[1] For this paper, a problem is intractable if the corresponding decision problem is NP-Hard or CoNP-Hard [Garey and Johnson, 1979]. Also, a knowledge base is any propositional theory, and a fact is any atomic formula or its complement.

[2] For now, it suffices to know that BCP is a variant of unit resolution [Chang and Lee, 1973] for any KB in CNF. A precise definition is given later.

Γ and formula ψ, $\Gamma \mid\!\sim \psi$ iff PFP can determine that $\Gamma \cup \{\neg\psi\}$ is inconsistent. This entailment relation is more powerful and efficient than some others presented in the literature (c.f. the logic of explicit belief [Levesque, 1984b; Frisch, 1987]). We show that $\mid\!\sim$ can be used for efficiently answering queries posed to a KB, and for modeling beliefs held by a resource-limited agent.

The rest of the paper is organized as follows. After some preliminary definitions, we define PFP and illustrate it by some examples. We then present a tractable algorithm to apply PFP on any given KB. Next, we define the weak entailment relation $\mid\!\sim$, and discuss some of its applications. Given the limited space here, we relegate all proofs and detailed motivations and discussions to [Dalal, 1992].

Preliminaries

We denote propositional symbols by upper case letters $P, Q, \ldots$; and literals ($P, \neg P$, etc.) by lower case letters $p, q, \ldots$. The complement operator $\sim$ on literals is defined as follows: if $p = Q$, then $\sim p = \neg Q$; if $p = \neg Q$, then $\sim p = Q$. There are two special literals, $\mathbf{t}$ and $\mathbf{f}$, that denote *true* and *false* respectively, and are complements of each other. For a set, A, of literals, $\sim A$ denotes the set containing the complement of each literal in A.

We consider only those formulas that are in negation normal form (NNF), i.e., finitely built from the literals using only two connectives, $\wedge$ (conjunction) and $\vee$ (disjunction).[3] A formula that's just a literal is called a *fact*. A formula without any conjunctions is called a *clause*. A theory (or a KB) is any set of formulas. A theory is called *clausal* (or in CNF) if all its formulas are clauses. We denote formulas by lower case Greek letters $\psi, \sigma, \ldots$ and theories by upper case Greek letters $\Gamma, \Delta, \ldots$. For any finite theory Γ, we use $\wedge\Gamma$ and $\vee\Gamma$ to denote the conjunction and disjunction respectively, of all the formulas in Γ. The semantic notions of satisfiability, entailment ($\models$) and equivalence ($\equiv$) are defined as usual [Mendelson, 1964].

For any theory Γ, we use lits(Γ) to denote the set of all literals that occur in Γ; facts(Γ) to denote the set of all facts in Γ; and nf(Γ) to denote the set of all non-facts in Γ, i.e., the set (Γ – facts(Γ)). As usual, a singleton set $\{\psi\}$ is often abbreviated as ψ.

Propositional Fact Propagation

Propositional fact propagation is a method for inferring some facts that are logically entailed by a propositional theory. Intuitively, PFP first infers facts from the simplest sub-formulas in the theory, and then uses them to infer facts from the more complex formulas that contain these sub-formulas. Whenever possible, it simplifies the sub-formulas by propagating the facts inferred from them.

Consider the formula, $\psi = Q \vee (P \wedge (\neg P \vee Q))$. Since fact P can be inferred from the sub-formula $(P \wedge (\neg P \vee Q))$, it can be simplified to obtain $(P \wedge Q)$. Since fact Q can now be inferred from both the disjuncts of the simplified ψ, it can also be inferred from ψ. Further simplifying ψ leads to a logically equivalent formula Q. The intuitions used in this example are formalized below.

Facts can be inferred from a formula in two ways. A fact p can be *conjunctively inferred* from a formula ψ iff there is a formula σ such that ψ is logically equivalent to $p \wedge \sigma$. Similarly, a fact p can be *disjunctively inferred* from a formula ψ iff there is a formula σ such that ψ is logically equivalent to $p \vee \sigma$. If follows that a fact p can be conjunctively (or disjunctively) inferred from a formula ψ iff $\psi \models p$ ($p \models \psi$). Thus, in either case, the general problem of determining the set of facts that can be inferred from a particular formula is CoNP-Complete. However, a subset of these facts can be tractably determined using the following recursive definitions of C and D:

1. for any literal p, $C(p) = D(p) = \{p\}$;
2. for any conjunctive formula, $\psi = \sigma \wedge \mu$, $C(\psi) = C(\sigma) \cup C(\mu)$, and $D(\psi) = D(\sigma) \cap D(\mu)$;
3. for any disjunctive formula, $\psi = \sigma \vee \mu$, $C(\psi) = C(\sigma) \cap C(\mu)$, and $D(\psi) = D(\sigma) \cup D(\mu)$;
4. for any theory Γ, $C(\Gamma) = \cup_{\psi \in \Gamma} C(\psi)$, and $D(\Gamma) = \cap_{\psi \in \Gamma} D(\psi)$.

The next lemma shows that the sets C and D contain conjunctively and disjunctively inferred facts, respectively.

Lemma 1 *Any fact p in $C(\psi)$ (or $D(\psi)$) can be conjunctively (disjunctively) inferred from the formula ψ.*

Both kinds of inferred facts can be propagated to simplify a formula. The basic propagation step is defined as follows: For any formula ψ, and any satisfiable set of facts A, ψ_A is the formula obtained by simplifying ψ after substituting each literal in A by $\mathbf{t}$ and its complement by $\mathbf{f}$. The simplification rules are:

$$\mathbf{f} \wedge \psi \Rightarrow \mathbf{f} \qquad \mathbf{f} \vee \psi \Rightarrow \psi$$
$$\mathbf{t} \wedge \psi \Rightarrow \psi \qquad \mathbf{t} \vee \psi \Rightarrow \mathbf{t}$$

For example, $((P \vee Q) \wedge \neg P)_{\neg P} = Q$. This example also shows that propagation of facts is likely to simplify a formula. The next lemma shows that propagating facts inferred from a formula does not change its logical content.

Lemma 2 *If a fact p can be conjunctively inferred from a formula ψ, then $\psi \equiv p \wedge \psi_p$. If a fact p can be disjunctively inferred from a formula ψ, then $\psi \equiv p \vee \psi_{\sim p}$.* ■

[3] We do not lose any generality due to this syntactic restriction, since any formula also involving $\neg$ (negation) and $\rightarrow$ (implication) can be transformed efficiently into this syntax, without any significant increase in the size of the formula.

Thus, a formula can be simplified by first inferring facts from it, and then propagating them. This can be repeated until no more new facts can be inferred. This process can be recursively applied to each sub-formula. However, it suffices to propagate only conjunctively inferred facts through conjunctive sub-formulas, and disjunctively inferred facts through disjunctive sub-formulas. Special care is needed when the set of inferred facts is unsatisfiable. This happens when either the set contains **f**, or it contains a pair of complementary facts. A theory Γ is said to be *basic inconsistent*, binc(Γ), iff the set facts(Γ) is unsatisfiable. For any formula or a theory, the function f defined below infers some facts and propagates them:

1. for any literal p, $f(p) = p$;
2. for any conjunctive formula ψ,
$$f(\psi) = \begin{cases} \mathbf{f} & \text{if binc}(C(\psi)) \\ \wedge C(\psi) \wedge \psi_{C(\psi)} & \text{otherwise} \end{cases}$$
3. for any disjunctive formula ψ,
$$f(\psi) = \begin{cases} \mathbf{t} & \text{if binc}(D(\psi)) \\ \vee D(\psi) \vee \psi_{\sim D(\psi)} & \text{otherwise} \end{cases}$$
4. for any theory Γ,
$$f(\Gamma) = \begin{cases} \{\mathbf{f}\} & \text{if binc}(C(\Gamma)) \\ C(\Gamma) \cup \{\psi_{C(\Gamma)} : \psi \in \Gamma\} & \text{otherwise} \end{cases}$$

It follows from Lemma 2 that any formula ψ is logically equivalent to $f(\psi)$. Since a theory can be viewed as a (possibly infinite) conjunctive formula, it also follows that any theory Γ is logically equivalent to $f(\Gamma)$. Now consider the following reduction rules:

$$\psi \Rightarrow f(\psi)$$
$$\Gamma \Rightarrow f(\Gamma)$$

Intuitively, a reduction using any of these rules substitutes a sub-formula (or theory) by a logically-equivalent but possibly simplified formula (theory). For any theory, a terminating sequence of reductions is such that no further non-trivial reduction, i.e., a reduction that changes the theory, is possible. For any theory Γ, PFP(Γ) is defined to be any theory obtained by any terminating sequence of reductions using the above two rules.

Theorem 1 *For any theory* Γ, PFP(Γ) *is independent of the particular sequence of reductions. For any finite theory, any sequence of non-trivial reductions is of finite length.* ■

As an example, consider a theory Γ consisting of only the Formula 1 given in the introduction. Let ψ denote the conjunctive sub-formula $P \wedge ((Q \wedge \neg P \wedge R) \vee (S \wedge T))$. Since $C(\psi) = \{P\}$, $f(\psi) = P \wedge S \wedge T$. Let σ denote the disjunctive sub-formula $(S \wedge R) \vee S$. Since $D(\sigma) = \{S\}$, $f(\sigma) = S$. After replacing ψ by $f(\psi)$ and σ by $f(\sigma)$, the simplified theory Γ' is

$$\{((V \vee W) \wedge S) \vee (P \wedge S \wedge T) \vee (S \wedge (X \vee W))\}$$

Since $C(\Gamma') = \{S\}$, the theory can be further simplified to obtain $f(\Gamma')$, which is $\{S,\ (P \wedge T) \vee V \vee W \vee X\}$. Since, no further simplification is possible, PFP(Γ) = $f(\Gamma')$.

The next theorem shows that PFP preserves logical equivalence, and is modular, i.e., it can be applied to parts of a theory before applying to the entire theory.

Theorem 2 *For any two theories* Γ *and* Δ, PFP(Γ) $\equiv$ Γ, *and* PFP($\Gamma \cup \Delta$) = PFP(PFP(Γ) $\cup$ Δ) = PFP(PFP(Γ) $\cup$ PFP(Δ)).

PFP is logically incomplete, i.e., it may not infer all facts that are logically entailed by a theory. For example, although P is logically entailed by the theory $\{P \vee \neg Q,\ P \vee Q\}$, it can't be obtained using PFP. This example also demonstrates that the results obtained using PFP depends not only on the logical content of the theory, but also on its syntactic form.

Compute-PFP: An Algorithm

Directly implementing the definition of PFP is likely to be quite expensive, since it would require many passes over the input KB. In this section, we sketch a tractable algorithm, called **Compute-PFP**, that avoids these multiple passes. A detailed description is given in [Dalal, 1992].

Compute-PFP represents the given theory as a labeled tree, where each internal node is labeled by either $\wedge$ (conjunctive node) or $\vee$ (disjunctive node), and each leaf is labeled by a literal. For instance, the theory $\{P \vee \neg Q,\ S \vee P \vee R\}$ is represented by the tree shown in Figure 1. Whenever possible, the tree is simplified by suitably eliminating leaf nodes labeled by **t** or **f** (using the rules described in the definition of ψ_A) and internal nodes that have only one child, and by merging pairs of parent-child nodes labeled by the same connective.

The main task of **Compute-PFP** is identifying facts that can be inferred from the subtree rooted at any node, and then either propagating them through the subtree, or passing them to the parent node. Since the two kinds of nodes are processed similarly, we consider only the disjunctive nodes here. Figure 2 illustrates the situation when a fact p can be disjunctively inferred from a disjunctive node — each node labeled by p (or $\sim p$) in the subtree T is replaced by a node labeled by **f** (**t**), which is then suitably eliminated. Figure 3 illustrates the only case (after all other simplification has been performed) when a fact p can be conjunctively inferred from a disjunctive node — the fact p is passed to the parent of the node.

Compute-PFP maintains three arrays and a counter with each internal node N: $N.max$ keeps a count of the number of children of N; for any literal p, $N.occur[p]$ is a list of all children S of N such that the subtree rooted at S contains a node labeled by p; $N.count[p]$ keeps a count of the number of nodes labeled by p in the subtree rooted at N; while $N.depth[p]$ is the sum of distances of all these nodes from N. The

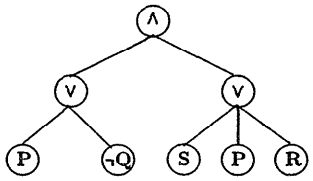

Figure 1: A labeled tree

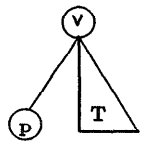

Figure 2: Fact propagation

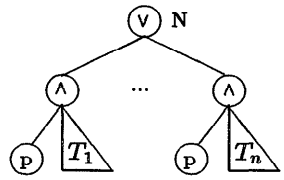

Figure 3: Computing $C(\psi)$

arrays $N.occur$ are used to access all occurrences of a literal in a sub-formula. This is required in propagating a fact p while computing $f(\psi)$ for some sub-formula ψ. All the other information is used for quick detection of whether a fact is in $C(\psi)$ (or $D(\psi)$) for a disjunctive (conjunctive) formula ψ. For instance, in Figure 3 there are exactly $N.max$ ($= n$) occurrences of nodes labeled by p, each at a distance 2 from N, in the subtree rooted at N. Thus, the necessary and sufficient conditions to detect such a case are $N.count[p] = N.max$ and $N.depth[p] = 2 \times N.max$.

Given a theory Γ, let n be its size, m be the number of distinct symbols in it, and $(k-1)$ be the maximum number of alternations of connectives in the formulas of the theory. Initially, the tree has $O(n)$ nodes, each requiring $O(m)$ space to store the various arrays. Thus, the total size of the tree is $O(nm)$. However, only $O(nk)$ space is actively used, and the tree can be built in $O(nk)$ time (using a random-access model of computation). Each simplification step either removes at least one node of the tree, or replaces a literal p by a constant **t** or **f**. Since there are n nodes to begin with, there can be at most $O(n)$ such simplification steps. Each simplification step may require traversing a branch, and can take at most $O(k)$ time. Thus, **Compute-PFP** takes at most $O(nk)$ time and at most $O(nm)$ space in total.

Theorem 3 *For any theory Γ of size n with at most $(k-1)$ alternations of connectives,* **Compute-PFP** *terminates in time $O(nk)$.* ■

Note that k is usually a very small number. For example, $k = 1$ for a theory in either conjunctive or disjunctive normal form. Thus, the worst-case time complexity is quite close to linear. In [Dalal, 1992], we show that it is possible to reduce the space complexity to $O(n\ log(m))$ by increasing the time complexity to $O(nk\ log(m))$.

Tractable Entailment Using PFP

The general problem of determining whether a given theory logically entails a given formula is CoNP-Complete. Various attempts have been made to define weak entailment relations that are tractable (c.f. [Levesque, 1984a]). In this section, we use PFP for defining a new weak but tractable entailment relation that is more powerful and efficient than these previous ones.

Let Γ be any theory and ψ be any formula. It is known that $\Gamma \models \psi$ iff $\Gamma \cup \{\neg\psi\}$ is unsatisfiable. This result can be used to define a weak entailment relation from any sufficient condition for unsatisfiability. In particular, the weak entailment relation $\mid\!\sim$ is defined as follows: $\Gamma \mid\!\sim \psi$ iff $\text{PFP}(\Gamma \cup \{\neg\psi\}) = \{\mathbf{f}\}$.[4] This intuitively means that PFP is able to determine inconsistency of $\Gamma \cup \{\neg\psi\}$. Some properties and examples of $\mid\!\sim$ are listed below:

Soundness: $\mid\!\sim$ is logically sound, i.e., if $\Gamma \mid\!\sim \psi$ then $\Gamma \models \psi$.

Incompleteness: $\mid\!\sim$ is logically incomplete. For example, although $\{\psi\} \models \psi$ for any formula ψ, if $\psi = (a \vee b) \wedge (c \vee d)$ then $\{\psi\} \not\mid\!\sim \psi$. However, PFP is complete for certain restricted classes of theories and formulas. For instance, if Γ contains only Horn clauses and ψ is a conjunctive formula, then $\Gamma \models \psi$ iff $\Gamma \mid\!\sim \psi$.

Monotonicity: $\mid\!\sim$ is monotonic, i.e., if $\Gamma \mid\!\sim \psi$ and $\Gamma \subseteq \Gamma'$, then $\Gamma' \mid\!\sim \psi$. It also conforms to some truth-functional properties of the two connectives. For instance, if $\Gamma \mid\!\sim \psi \wedge \sigma$, then $\Gamma \mid\!\sim \psi$, and $\Gamma \mid\!\sim \sigma$. Also, if either $\Gamma \mid\!\sim \psi$ or $\Gamma \mid\!\sim \sigma$, then $\Gamma \mid\!\sim \psi \vee \sigma$. However, the contrapositive of the above two properties, which hold for $\models$, do not hold for $\mid\!\sim$.

Chaining: $\mid\!\sim$ allows some simple chaining, for instance, $\{P,\ \neg P \vee Q\} \mid\!\sim Q$. It also allows slightly more complex chaining, for instance, $\{\neg P \vee Q,\ \neg Q \vee R\} \mid\!\sim \neg P \vee R$.

Syntactic Dependence: $\mid\!\sim$ depends on the syntactic form of the formulas. For instance, although the theories $\{P,\ Q \vee R\}$ and $\{P \vee Q,\ P \vee \neg Q,\ Q \vee R\}$ are logically equivalent, only the former entails P using $\mid\!\sim$. However, $\mid\!\sim$ is independent of some of the finer nuances of syntax — for instance, changing the order of formulas in the theory, or reordering the arguments of a connective.

Tractability: The algorithm **Compute-PFP** can be used to determine $\mid\!\sim$. Given a theory Γ of size n and at most $(k-1)$ alternations of connectives, and

[4] Since PFP requires the input theory to be in NNF, the negation in front of ψ should be first pushed all the way inside to atomic propositions.

a formula ψ of size q and at most $(r-1)$ alternations of connectives, **Compute-PFP** can determine in $O(nk+qr)$ time whether $\Gamma \mid\!\sim \psi$.

PFP can also be used to define another weak entailment $\mid\!\sim'$. Recall that for any theory Γ, PFP(Γ) is a subset of facts that can be logically inferred from Γ. If these facts are sufficient to make a formula ψ true, then ψ can be logically inferred from Γ. Thus, $\mid\!\sim'$ is defined as follows: for any theory Γ and formula ψ, $\Gamma \mid\!\sim' \psi$ iff $\psi_{\text{facts(PFP}(\Gamma))} = \mathbf{t}$.

It follows that $\mid\!\sim'$ is sound, incomplete, monotonic, and depends on the syntactic form of the formulas. Although it can perform some simple chaining, for instance, $\{P, \neg P \vee Q\} \mid\!\sim' Q$, it can't perform the more complex ones, for instance, $\{\neg P \vee Q, \neg Q \vee R\} \not\mid\!\sim' \neg P \vee R$. For any theory Γ and formula ψ, if $\Gamma \mid\!\sim' \psi$, then $\Gamma \mid\!\sim \psi$. Hence, the new relation $\mid\!\sim'$ is strictly weaker than $\mid\!\sim$. If PFP(Γ) is pre-computed and stored suitably such that any fact can be retrieved from it in constant time, then $\mid\!\sim'$ can be determined in only $O(qr)$ time, which is independent of the size of Γ. Thus, $\mid\!\sim'$ can be determined more efficiently than $\mid\!\sim$.

Applications and Related Work

Our problem is closely related to the problem of optimizing a boolean circuit or expression, which has been studied for many years (c.f. [Birkhoff and Bartee, 1970; Chen *et al.*, 1991]). However, the goals there are quite different from ours. For instance, there it is more important to minimize the size (or delay, etc.) of the resulting circuit, or to obtain a canonical form (like CNF) even at the cost of using a more expensive approach. Moreover, the optimizations there make use of don't-cares, specific technology used to realize the circuit, etc. — issues that are not relevant for our purposes.

Logical TMSs and BCP

Logical truth maintenance systems [Reiter and de Kleer, 1987; de Kleer, 1986] perform limited deductive reasoning from the set of formulas given to them. Most implementations of a Logical TMS use BCP to infer facts [McAllester, 1990]. Given a theory Γ, BCP monotonically expands it by adding facts as follows. In each step, if any *single* formula in Γ and *all the facts* in Γ taken together *logically entails* any other fact, then the new fact is added to the theory Γ. This step is repeated until no new fact can be so obtained, or the theory Γ becomes basic inconsistent.[5]

Given any clause and any set of facts, it is easy to determine any new facts that can be inferred. Thus, *clausal BCP*, which is BCP restricted to clauses, is tractable. Since the general problem of determining facts that can be inferred from an arbitrary formula is intractable, *formula BCP*, which is the unrestricted BCP, is inherently intractable. de Kleer [1990] suggests two techniques for using clausal BCP for theories containing arbitrary formulas. In one, *CNF-BCP*, the formulas are first converted into CNF; and in the other, *Prime-BCP*, clausal BCP is applied to the prime implicants [Reiter and de Kleer, 1987] of each formula. Since computing prime implicants is itself an intractable problem, Prime-BCP is also inherently intractable.

If no new symbols are added, then conversion to CNF may increase the size of the given theory exponentially. But conversion to CNF can be done in linear time and space by adding new symbols, each representing a sub-formula of the theory [Cook, 1971]. However, with either kind of conversion, CNF-BCP is strictly weaker than PFP, as shown in the next theorem (and the first example in the introduction).

Theorem 4 *For any theory* Γ, *either* PFP(Γ) = {f} *or* facts(BCP(CNF(Γ))) $\subseteq$ facts(PFP(Γ)). ■

The above theorem and the tractability of **Compute-PFP** makes PFP an attractive technique for reasoning in TMS and other AI applications.

Explicit Belief and Tractable Entailment

Responding to the problem of logical omniscience [Hintikka, 1975], Levesque [1984b] introduces a semantic notion of explicit belief, which is the set of beliefs that a resource-limited agent can infer from its implicit beliefs.[6] This notion of explicit belief does not allow any chaining, i.e., an agent who explicitly believes in P and $\neg P \vee Q$ may not believe in Q. The next theorem (and the observation that $\mid\!\sim$ allows some chaining) shows that the entailment $\mid\!\sim$ based on PFP is more powerful than the notion of explicit belief.

Theorem 5 *If belief in* Γ *leads to an explicit belief in* ψ, *then* $\Gamma \mid\!\sim \psi$. ■

The entailment relation $\mid\!\sim$ is also computationally less expensive, since it takes $O(nq)$ time to determine whether a formula of size q can be explicitly believed, given a theory of size n [Levesque, 1984b]. Note also that computing explicit beliefs is known to be tractable only if all the formulas are in CNF, something not required by $\mid\!\sim$. Thus, the weak entailment relation, $\mid\!\sim$, presented in this paper is a significant improvement over the notion of explicit belief.

Efficient Query Answering

Querying is an important function that should be provided by any KB [Levesque, 1984a]. Let Ask(Γ, ψ) be

[5] In the TMS literature (c.f. [de Kleer, 1990]), a distinction is usually made between the facts (called assumptions) and the other formulas (called constraints) in the theory. However, this distinction does not effect the results obtained using BCP.

[6] Later, Frisch [1987] developed a different but equivalent model theory.

the answer returned by the knowledge base Γ to the query ψ. It is traditionally defined as:

$$\text{Ask}(\Gamma, \psi) = \begin{cases} \text{"yes"} & \text{if } \Gamma \models \psi \\ \text{"no"} & \text{if } \Gamma \models \neg\psi \\ \text{"unknown"} & \text{otherwise} \end{cases}$$

Reliance on $\models$ makes *Ask* intractable even for propositional KBs. However, *Ask* can be made tractable by using a different entailment relation. We propose that the weak but tractable entailment relation $\vdash\!\!\sim$ be used instead. Let *Ask'* be the function obtained after replacing $\models$ by $\vdash\!\!\sim$ in the definition of *Ask*.

Apart from the linear time worst-case complexity of $\vdash\!\!\sim$, there is an additional advantage due to the modularity of PFP. If the KB does not change very often, then PFP(Γ) can be computed once and then used for many different queries. Thus, the cost of PFP(Γ) is amortized over many queries, further reducing the actual time taken to answer each of them.

Conclusions

We presented PFP, which is an efficient method for inferring facts from a propositional knowledge base. Though it is logically-incomplete, it does not require the knowledge base to be in CNF, and is more powerful and efficient than CNF-BCP. We also presented an algorithm, **Compute-PFP**, whose worst-case time complexity is linear in the product of the size of the knowledge base and the maximum number of alternations of connectives. We also used PFP to define two notions of weak but tractable entailment, and analyzed some of their properties.

PFP has some limitations. Firstly, it is quite weak in some cases. For instance, it does not infer the fact P from the theory $\{P \vee Q,\ P \vee \neg Q\}$.[7] Secondly, it lacks a model theory that provides an independent characterization.

It seems that the PFP can be extended to logics with quantifiers and modality. It also seems possible to strengthen propagation while maintaining efficiency and the other properties, by limited use of case analysis. There is also a possibility of getting stronger propagation rules, that are still tractable. Our current research is directed towards realizing these possibilities. We are also in the process of implementing PFP, and testing its performance on large and realistic theories.

PFP appears to be the first tractable fact propagation technique that does not require converting formulas to CNF. We believe that PFP is likely to have significant applications in TMS and other similar AI systems, in modeling the beliefs of resource-limited agents, and in efficiently answering queries posed to a knowledge base.

[7] Note that both BCP and and the notion of explicit belief also suffer from this weakness.

Acknowledgments

I would like to thank Alex Borgida, David Etherington, Hari Hampapuram, Ringo Ling, Kumar Vadaparty, Ke-Thia Yao, Haym Hirsh and Kerstin Voigt for helpful discussions and comments on the earlier drafts of this paper. I also wish to thank AT&T Bell Laboratories for partial support during this research.

References

Birkhoff, G. and Bartee, T.C. 1970. *Modern Applied Algebra.* McGraw-Hill, New York.

Chang, C. and Lee, R.C. 1973. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, London.

Chen, K.-C.; Matsunaga, Y.; Fujita, M.; and Muroga, S. 1991. A resynthesis approach for network optimization. In *Proceedings 28th ACM/IEEE Design Automation Conference.* 458–463.

Cook, S.A. 1971. The complexity of theorem proving procedures. In *Proceedings Third Annual ACM Symposium on the Theory of Computing.* 151–158.

Dalal, M. 1992. Fact propagation functions. In preparation.

de Kleer, J. 1986. An assumption-based truth maintenance system. *Artificial Intelligence* 28:127–162.

de Kleer, J. 1990. Exploiting locality in a TMS. In *Proceedings Eight National Conference on Artificial Intelligence (AAAI-90).* 264–271.

Frisch, A.M. 1987. Inference without chaining. In *Proceedings Tenth International Joint Conference on Artificial Intelligence (IJCAI-87).* 515–519.

Garey, M. and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, W. H., NY.

Hintikka, J. 1975. Impossible possible worlds vindicated. *Journal of Philosophical Logic* 4:475–484.

Levesque, H.J. 1984a. Foundations of a functional approach to knowledge representation. *Artificial Intelligence* 23(2):155–212.

Levesque, H.J. 1984b. A logic of implicit and explicit belief. *Proceedings National Conference on Artificial Intelligence (AAAI-84)* 198–202.

McAllester, D. 1980. An outlook on truth maintenance. Memo 551, MIT AI Lab.

McAllester, D. 1990. Truth maintenance. In *Proceedings Eight National Conference on Artificial Intelligence (AAAI-90).* 1109–1116.

Mendelson, E. 1964. *Introduction to Mathematical Logic.* Van Nostrand, Princeton, N.J.

Reiter, R. and de Kleer, J. 1987. Foundations of assumption-based truth maintenance systems: Preliminary report. In *Proceedings Sixth National Conference on Artificial Intelligence (AAAI-87).* 183–188.