Polarity Guided Tractable Reasoning

Zbigniew Stachniak

Department of Computer Science
York University
Toronto, Ontario, M3J 1P3 Canada
zbigniew@cs.yorku.ca

Abstract

Non-clausal refinement of Boolean Constraint Propagation inference procedure for classical logic, called P-BCP, is introduced within a new knowledge representational formalism of polarized formulas. P-BCP is a sound, incomplete, and linear-time inference procedure. It is shown that P-BCP can be adopted for tractable reasoning in a number of non-classical logics (including some modal and finitely-valued logics).

Introduction

The use of clausal logic for the representation of knowledge and tractable reasoning is extensive but not without its problems. Knowledge represented in clausal logic frequently results in large, difficult to handle sets of formulas. Extensions of efficient clause-based inference procedures to non-clausal theories frequently result in either intractable procedures (cf. de Kleer, 1990) or in reasoning procedures with a diminished inference power (e.g., when structure preserving CNF transformation algorithms are employed, cf. Empirical Results in (Roy-Chowdhury and Dalal, 1997)).

To find a compromise between the efficiency of reasoning and the expressive power of a knowledge representational framework one can settle for a fully non-clausal and sound reasoning system but either with significantly restricted inference power or of higher computationally complexity. Or, one can aim at the extension of the class of clauses to a class of free-form formulas that enjoy computational properties of clauses while, due to their free-form nature, they eliminate the need for a syntactic transformation into some normal form. The former approach is taken, for instance, in (Roy-Chowdhury and Dalal, 1997). The authors' Restricted Fact Propagation (RFP) is a fully non-clausal, sound, incomplete, and quadratic time inference procedure. Inferentially, RFP is equivalent to CNF-BCP, i.e., to Boolean Constraint Propagation (cf. McAllester, 1990) augmented with the standard CNF transformation of an input set into clauses. The research reported in this paper follows the latter direction. We propose a new knowledge representational formalism of polarized formulas and refine Boolean Constraint Propagation procedure to work with theories of polarized formulas. The notion of a polarized formula is a free-form counterpart of a clause based on the notion of polarity. This notion does not rely on a specific syntactic form (polarized formulas are free-form formulas) but, instead, on polarity values of atoms that occur in such formulas. Representing knowledge using polarized formulas rather than clauses may result in an exponentially smaller theory. The proposed refinement of Boolean Constraint Propagation, named P-BCP, is a sound, incomplete, and linear time inference procedure in the domain of polarized formulas. It is inferentially equivalent to CNF-BCP, i.e., both procedures infer the same literals from the same input set of polarized formulas.

The proposed framework of polarized formulas has other unique features. It allows a universal clause-like representation of knowledge across the class of logical systems. Indeed, the definition of a polarized formula is based on the notion of polarity – the classical notion that is readily available to a variety of logical systems. As a result, P-BCP can be adopted for tractable non-classical reasoning in a number of logical systems.

This paper is organized as follows. The next section, 'Logical Preliminaries', briefly reviews the basic logical notions and notation adopted in this paper. The following two sections present the definitions of the class of polarized formulas for classical logic and of P-BCP. These sections contain the main theoretical results concerning P-BCP; the linear-time refinement of P-BCP is discussed in subsection 'Time Complexity of P-BCP'. The extension of the polarized-formula and P-BCP framework to non-classical logics is presented in section 'P-BCP for Non-Classical Logics'. We show how P-BCP algorithm can be adopted to some intensional and finitely-valued logics (e.g. the modal logic $S4$ and finitely-valued logics of Łukasiewicz). Only propositional logics are discussed.

Logical Preliminaries

Logical systems considered in this paper are pairs $\mathcal{P} = \langle L, \vdash \rangle$, where L is the (propositional) language and $\vdash$ is the inference operation of $\mathcal{P}$. We identify L with the set

of all well-formed formulas which are constructed, in the usual way, from a countably infinite set of propositional variables and a finite set of logical connectives. We assume that among the connectives of $\mathcal{P}$ there are two logical constants: 1, which is a tautology of $\mathcal{P}$, and 0, which is a contradictory formula of $\mathcal{P}$. The inference relation $\vdash$ is a binary relation between sets of formulas and formulas of $\mathcal{P}$. It satisfies *inclusion* ($X \vdash \alpha$, for every $\alpha \in X$), *monotonicity* ($X \vdash \alpha$ and $X \subseteq Y$ implies $Y \vdash \alpha$), and *closure* ($Th(X) \vdash \alpha$ implies $X \vdash \alpha$, where $Th(X) = \{\beta \in L : X \vdash \beta\}$).

If $X \subseteq L$, then $Var(X)$ denotes the set of all variables that occur in formulas of X. We shall write '$\alpha(p_0, \ldots, p_k)$', or '$\alpha(p_i), i \leq k$', to indicate that $p_0, \ldots, p_k$ are among the variables of α. By '$\alpha(p/\beta)$' we denote the result of simultaneous replacement of every occurrence of p in α by a formula $\beta \in L$ (the notation '$\alpha(p_0/\beta_0, \ldots, p_k/\beta_k)$' is self-explanatory). A *tautology* is a formula α such that $\emptyset \vdash \alpha$. α is said to be a *contradictory formula* if for every $\beta \in L$, $\{\alpha\} \vdash \beta$. We call a set $X \subseteq L$ *consistent* if for some $\beta \in L, X \nvdash \beta$. Finally, X is *inconsistent* if it is not consistent.

When semantics for $\mathcal{P}$ is provided, an *interpretation* of L is a mapping from L into a specified set of truth values TV that includes at least the truth-value 1 (true) and 0 (false). For reasons of simplicity, we identify these truth-values with the constants 0 and 1.

Polarized Formulas in Classical Logic

We begin with an example of a reasoning task that will lead us to the definition of a polarized formula – one of the central notions in this paper.

Consider the following two formulas:

$\alpha_0 = (\neg p_1 \wedge \neg q_1) \vee \ldots \vee (\neg p_n \wedge \neg q_n)$

$\alpha_1 = p_1 \wedge \ldots \wedge p_{n-1}$,

$n > 1$, and the reasoning task of determining all the literals that follow from $\{\alpha_0, \alpha_1\}$ in classical propositional logic (CPL). Clearly, these literals are $p_1, \ldots p_{n-1}, \neg p_n, \neg q_n$. Clausal Boolean Constraint Propagation procedure (BCP) will infer all these literals provided that α_0 and α_1 are transformed into clauses. However, converting α_0 alone into clauses using the standard CNF transformation results in 2^n clauses, much too many for BCP to handle even for moderate values of n. On the other hand, structure preserving CNF transformation of α_0 and α_1 (cf. Tseitin, 1983) results in a set of clauses from which BCP can infer neither $\neg p_n$ nor $\neg q_n$.

The formulas α_0 and α_1 have not been selected just to score a theoretical point. In (Nayak and Williams, 1997) one can find the following small fragment of the propositional theory used for on-board real-time model-based diagnosis and recovery for the DS-1 spacecraft:

C_1: $\neg nc_i \vee \neg a \vee nc_0$, C_4: $\neg rf \vee ia$, C_7: $\neg ok \vee \neg uf$
C_2: $\neg ia \vee nc_0$, C_5: $\neg uf \vee ia$, C_8: $\neg rf \vee \neg uf$
C_3: $\neg ok \vee a$, C_6: $\neg ok \vee \neg rf$, C_9: $\neg a \vee \neg ia$.

Clauses C_4 – C_7 result from

$P_2 : (\neg ok \wedge ia) \vee (\neg rf \wedge \neg uf)$

by a standard CNF transformation. Clearly, P_2 has the same structure as α_0 ($n = 2$).

It turns out that BCP can be refined to handle explicitly such formulas as α_0 or P_2 without resorting to a CNF transformation (and, hence, without a danger of an exponential explosion or loss of inferential power). The key observation is that, from the polarity point of view, formulas like α_0, P_2, and α_1 resemble clauses: every variable in these formulas has a unique polarity value 'positive' (e.g. all the variables in α_1, or ia in P_2) or 'negative' (e.g. all the variables in α_0, or ok, rf, and uf in P_2). We shall call such formulas *polarized*. Then, in its local propagation step, BCP can make the required inferences guided by the polarity values of variables in polarized formulas.

We now describe the knowledge representational framework of polarized formulas in details. We assume that the connectives of CPL are: $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), $\rightarrow$ (implication), and the constants 0 and 1. As usual, a *literal* is either a propositional variable (a positive literal) or the negation of a propositional variable (a negative literal). A propositional *clause* is a disjunction of literals with no literal repeated and containing no complementary literals.

The positive and the negative literals in a clause C are exactly the positive and the negative variables in C under the usual polarity assignment algorithm for CPL which assigns one of the polarity values '+' ('positive'), '−' ('negative'), or 'no polarity' to every variable in every formula of CPL:

> Given a formula α and a variable $p \in Var(\{\alpha\})$ the polarity assignment algorithm first assigns a polarity value to every occurrence of p in α. It assigns '+' to an occurrence of p in α if this occurrence is in the scope of an even number of $\neg$ connectives (for the purpose of assigning polarity values to occurrences of variables we identify $\beta \rightarrow \gamma$ with $\neg\beta \vee \gamma$). Otherwise, the selected occurrence of p in α is negative. Then, p is said to be '+' ('−') in α if every occurrence of p in α is '+' (is '−'). The variable p is of no polarity if it is neither '+' nor '−' in α.

The scope of this polarity assignment algorithm is the set of all the formulas of CPL. We can therefore use the algorithm to identify the class of 'polarized' formulas which, form the polarity point of view, is a natural extension of the class of clauses.

Definition 1: *A formula α is called a polarized formula (a p-formula) if every variable of α is either positive ('+') or negative ('−').*

Example A: The fragment of the DS-1 model discussed above can be represented using fewer formulas (regardless of the intended interpretation of the variables):

$P_1 : ((nc_i^- \wedge a^-) \vee ia^-) \rightarrow nc_0^+$ (replaces C_1 and C_2)

$P_2 : (\neg ok^- \wedge ia^+) \vee (\neg rf^- \wedge \neg uf^-)$ (replaces $C_4 - C_7$)

$C_3 : ok^- \rightarrow a^+,\ C_8 : rf^- \rightarrow \neg uf^-,\ C_9 : a^- \rightarrow \neg ia^-.$

It can be easily verified that all these formulas are polarized. Indeed, the polarity assignment algorithm will assign the polarity values to the variables as indicated by the superscripts '+' and '−'. Henceforth we shall use this subscripting method to indicate the polarity values of variables in p-formulas. ■

Proposition 1: *For every p-formula $\alpha(p, q)$:*

(i) $\{\alpha(p^+/0)\} \vdash \alpha(p^+)$ *and* $\{\alpha(p^+)\} \vdash \alpha(p^+/1)$;

(ii) $\{\alpha(p^-/1)\} \vdash \alpha(p^-)$ *and* $\{\alpha(p^-)\} \vdash \alpha(p^-/0)$;

(iii) *q retains its polarity value in $\alpha(p/0)$ and $\alpha(p/1)$.*

Corollary 2: *Let $\alpha(p_i^+, r_j^-)$, $i \leq k, j \leq m$, be a p-formula, where $Var(\{\alpha\}) = \{p_0, \ldots, p_k, r_0, \ldots, r_m\}$. If $\alpha(p_i^+/0, r_j^-/1)$ is a tautology, then so is α.*

P-BCP Algorithm for Classical Logic

Boolean Constraint Propagation (cf. McAllester 1990), a variant of which we describe below, is an inference procedure that accepts a finite set S of CPL formulas and returns a partial substitution h_S of constants 0 and 1 for some (possibly all or none) of the variables in $Var(S)$. In addition, BCP may declare S inconsistent.

Given S, BCP initializes h_S to 'undefined' for every variable in $Var(S)$ and, then, modifies h_S by repeating the following *local propagation step* (lp) until h_S cannot be further modified or for some formula $\beta \in S$, $h_S(\beta)$ is a contradictory formula:

(lp) a formula $\alpha \in S$ and a variable $p \in Var(S)$ are selected such that $\{h_S(\alpha)\} \vdash l$, where l is either p or $\neg p$; then $h_S(p) := 1$, if $l = p$ and $h_S(p) := 0$, if $l = \neg p$.

BCP is sound: if $h_S(p) = 1$, then $S \vdash p$; if $h_S(p) = 0$, then $S \vdash \neg p$. The local propagation step of BCP involves general entailment which is coNP-complete. However, restricted to clausal propositional theories BCP runs in linear-time in the total size of the input set (McAllester 1980). In the following subsection we adopt BCP to theories of p-formulas.

P-BCP Algorithm

To establish that $\{h_S(\alpha)\} \vdash l$ holds in the local propagation step (lp) it suffices to demonstrate that h_S cannot be extended to all the variables of α in such a way that for the resulting substitution h_S^*, $h_S^*(\alpha)$ is a tautology while $h_S^*(l)$ is a contradictory formula. If α is a p-formula, then the above property can be easily established. Indeed, suppose that l is a variable p and that $\alpha(p^+, p_i^+, r_j^-)$ is a p-formula such that p, $p_i, i \leq k$, and $r_j, j \leq m$, are all the variables of α for which h_S is undefined. By Proposition 1, if h_S^* is any extension of h_S such that $h_S^*(\alpha)$ is a tautology while $h_S^*(p) = 0$, then $h_S(\alpha(p^+/0, p_i^+/1, r_j^-/0))$ must be a tautology as well. Conversely, if $h_S(\alpha(p^+/0, p_i^+/1, r_1^-/0))$ is a tautology, then extending h_S by making the indicated assignments, results in h_S^* such that $h_S^*(\alpha)$ is a tautology while $h_S^*(p) = 0$. We therefore conclude that for $\{h_S(\alpha)\} \vdash p$ to hold, $h_S(\alpha(p/0, p_i^+/1, r_j^-/0))$ must be a contradictory formula. A similar argument shows that for $\{h_S(\alpha)\} \vdash \neg p$ to hold, $h_S(\alpha(p^-/1, p_i^+/1, r_j^-/0))$ must be a contradictory formula (we assume that now p is '−' in α).

This observation is exactly what is needed to make BCP tractable in the domain of p-formulas. For a p-formula α and a variable p (as above), let

$TV(\alpha, p) =$
$\{v \in \{0,1\} : h_S(\alpha(p/v, p_i^+/1, r_j^-/0))$ *is a tautology*$\}$.

The new constraint propagation algorithm – BCP for p-formulas (P-BCP) – is obtained from BCP by restricting inputs to finite sets of p-formulas and by replacing (lp) with

---

(lp*) a formula $\alpha(p, p_0^+, \ldots, p_k^+, r_0^-, \ldots, r_m^-) \in S$ and a variable p are selected, where $p, p_0 \ldots, p_k, r_0, \ldots, r_m$ are all the variables of α for which h_S is undefined;

if p is '+' in α and $TV(\alpha, p) = \{1\}$, then $h_S(p) := 1$;
if p is '−' in α and $TV(\alpha, p) = \{0\}$, then $h_S(p) := 0$.

Example A (cont): Executing P-BCP on the set $S = \{P_1, P_2, C_3, C_8, C_9\}$ results in h_S being undefined for every variable in $Var(S)$. On the other hand, adding the variable ok to S results in assigning 1 to ok and to a (via C_3). Then P-BCP assigns 0 to ia (using C_9) and to rf and uf (using P_2). h_S is undefined for the remaining variables. To sum up, P-BCP infers $ok, a, \neg ia, \neg rf$, and $\neg uf$ from $S \cup \{ok\}$.

P-BCP executed on $\{\alpha_0, \alpha_1\}$ (see the previous section) infers $p_1, \ldots, p_{n-1}$ from α_1 (in $n-1$ steps) and $\neg p_n, \neg q_n$ from α_0 (in two steps). ■

We begin the proof of the correctness of P-BCP with:

Lemma 3: *Let S be a finite set of p-formulas and let h be an interpretation of L such that $h(S) = \{1\}$. Then for every $v \in Var(S)$, if $h_S(v)$ is defined, then $h(v) = h_S(v)$.*

Proof: Let $v_0, \ldots, v_l, \ldots, v_n$ be the order in which P-BCP assigns constants 0 or 1 to propositional variables. Select $v = v_l$ and assume that for every $j < l$, $h(v_j) = h_S(v_j)$.

Suppose that $\alpha(v, p_i^+, r_j^-), i \leq k, j \leq m$, has been used by P-BCP to define $h_S(v)$. Since $h(\alpha) = 1$, $h_S(\alpha(v/h(v), p_i/h(p_i), r_j/h(r_j)))$ is a tautology. Since every p_i is '+' in α, $i \leq k$, and every r_j is '−' in α, $j \leq m$, by Proposition 1, $h_S(\alpha(v/h(v), p_i/1, r_j/0))$ is a tautology. If v is '+' in α, then $TV(\alpha, v) = \{1\}$ and we must have $h(v) = 1$. If v is '−' in α, then $TV(\alpha, v) = \{0\}$ and we have $h(v) = 0$, as required. ■

Theorem 4 (Soundness of P-BCP): *Let S be a finite set of p-formulas, let $p \in Var(S)$, and suppose that $h_S(p)$ is defined when P-BCP is executed on S. Then:*

(i) *$h_S(p) = 1$ implies $S \vdash p$;*

(ii) *$h_S(p) = 0$ implies $S \vdash \neg p$.*

Proof: Let S and p be as stated. Let $h_S(p) = 1$. To show that $S \vdash p$, select any interpretation h of L such that $h(S) = \{1\}$. By Lemma 3, $h(p) = h_S(p) = 1$, as required by (i). The proof of (ii) is similar. ■

Proposition 5: *If executing P-BCP on a set S of p-formulas results in h_S such that for some $\beta \in S, h_S(\beta)$ is a contradictory formula, then S is inconsistent.*

Proof: Suppose that the execution of P-BCP on a consistent set S results in h_S that is defined for every variable of some $\beta \in S$. Let h be an interpretation of L such that $h(S) = \{1\}$. By Lemma 3, if $p \in Var(\{\beta\})$, then $h(p) = h_S(p)$. Since $h(\beta) = 1$, $h_S(\beta)$ is a tautology. ■

P-BCP is not deductively complete. Indeed, $\{p \rightarrow q, \neg p \rightarrow q\} \vdash q$ but after the execution of P-BCP on this set $h_S(q)$ is undefined.

Time Complexity of P-BCP

P-BCP is not a very efficient procedure when (lp*) is implemented using the naive search. A better performance can be achieved when the search for α and p in (lp*) is replaced by the deterministic choice. In this subsection we sketch one of the possible solutions that makes P-BCP a linear-time inference procedure in the total size of an input set.

To simplify the presentation, we assume that p-formulas are in *negation normal form* (NNF), i.e., they are constructed from literals using $\vee$ and $\wedge$ as the only binary connectives. (Every formula can be converted into an equivalent NNF formula in linear time.) Furthermore, since $\vee$ and $\wedge$ are associative, we assume that these connectives can conjoin an arbitrary number n of formulas, $n \geq 2$. In this representation a variable q is '$-$' in a (NNF) p-formula α, if every occurrence of q in α is negated, and is '$+$', if no occurrence of q in α is negated. Hence, similarly to clauses, polarity values of variables in a NNF p-formula are explicitly represented through syntax.

We represent every p-formula $\alpha(p_i^+, r_j^-)$ by its parse tree T_α. As usual, every node N of T_α represents a subformula of α, which we denote by α_N. We assume that leaves of T_α represent literals. Every node N of T_α has a label $\langle c_N, V_N \rangle$. c_N is the main connective of α_N; for leaves, $c_N = \alpha_N$. V_N is a subset of $Var(\{\alpha_N\})$. To explain its role and its construction, suppose that $\alpha(p_i^+, r_j^-)$ is satisfiable. By Proposition 1, $\alpha(p_i^+/1, r_j^-/0)$ is a tautology and so is every $\alpha_N(p_i^+/1, r_j^-/0)$. What we want V_N to contain are all the variables q of α_N such that flipping the truth-value of q (from 1 to 0, if q is '$+$' in α, and from 0 to 1, otherwise) causes α_N to become a contradictory formula under the modified assignment. Hence, if N is a leaf and α_N is either q or $\neg q$, then we let $V_N = \{q\}$. To compute V_N for an internal node N, suppose that N has m children labeled with $\langle c_i, V_i \rangle$, $1 \leq i \leq m$. Then

$$V_N = V_1 \cap \ldots \cap V_m$$

if the main connective of α_N is $\vee$, and

$$V_N = V_1 \cup \ldots \cup V_m$$

if the main connective of α_N is $\wedge$. Indeed, suppose that the main connective of α_N is $\vee$ and that q is a variable such that flipping the truth-value of q causes α_N to become a contradiction. So, every child (disjunct) of α_N is a contradiction under the modified assignment and, hence, q must be in every $V_i, 1 \leq i \leq m$. A similar argument can be used to justify the definition of V_N when the main connective of α_N is $\wedge$.

Example B: The parse trees of the formulas in $S = \{p^+ \wedge (q^+ \vee \neg r^-),\ r^+\}$ are shown in Figure 2.

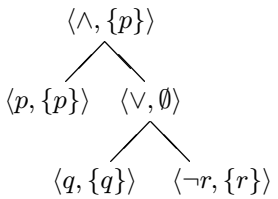


Fig. 2: Parse trees of $p \wedge (q \vee \neg r)$ and r. ■

Let $\langle c, V \rangle$ be the label of the root of a parse tree T_α. We call every $p \in V$ a *terminal variable* of T_α. In example B, the terminal variables are p and r. The main purpose of the construction of parse trees is to determine terminal variables – these are the variables that should be selected in (lp*) to further extend h_S.

Proposition 6: *Let α be a p-formula and let $p \in Var(\{\alpha\})$. The following conditions are equivalent:*

(i) *p is a terminal variable of T_α;*

(ii) *either $TV(\alpha, p) = \{1\}$ and p is '$+$' in α, or $TV(\alpha, p) = \{0\}$ and p is '$-$' in α.*

By Proposition 6, the local propagation step of P-BCP can be simplified to the choice of a terminal variable p of some parse tree T_α. Upon the selection, we define $h_S(p)$ guided by the polarity value of p in α: if p is '$+$', then $h_S(p) := 1$; if p is '$-$', then $h_S(p) := 0$. In other words, (lp*) can be replaced by

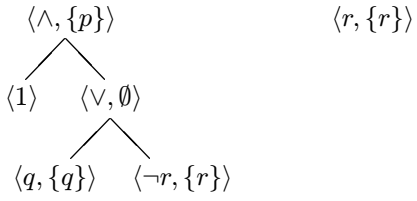
---

a terminal variable p of some T_α is selected; if p is '$+$' in α, then $h_S(p) := 1$, else $h_S(p) := 0$; next p is propagated through all the parse trees.

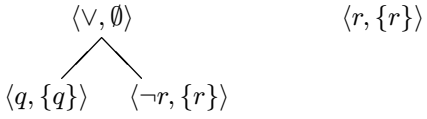
P-BCP repeats this step until h_S cannot be further extended or one of the parse trees has been reduced to the one-node tree labeled with $\langle 0, \emptyset \rangle$.

The propagation of a variable p through a parse tree T_α corresponds, roughly speaking, to the operation of substituting $h_S(p)$ for p in α and, then, simplifying $\alpha(p/h_S(p))$ using Boolean-like reduction rules (such as $(1 \wedge A) \Rightarrow A$, $(A \vee 0) \Rightarrow A$, etc.). We explain this process by continuing Example B.

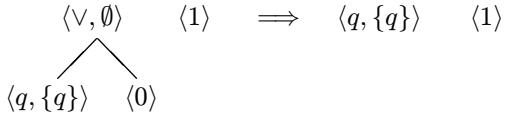
Example B (cont): Suppose that P-BCP selects a terminal variable p. Since p is '+' in $p \wedge (q \vee \neg r)$, the assignment $h_S(p) := 1$ is made. To propagate p through the parse trees in Figure 2, we first relabel all the leaves that are currently labeled with $\langle p, \{p\} \rangle$. The new label is $\langle 1, \emptyset \rangle$, or to keep things simple, it is $\langle 1 \rangle$. Since there is only one such leaf, we get



The resulting tree corresponds to $1 \wedge (q \vee \neg r)$ which clearly can be simplified to $q \vee \neg r$ using the reduction rule $(1 \wedge A) \Rightarrow A$. This simplification step performed on the trees results in the forest



Now, the only terminal variable left is r. Since r is '+' in itself, P-BCP makes the assignment $h_S(r) := 1$ and, then, propagates r through the forest as shown below



using the reduction rules: $\neg 1 \Rightarrow 0$ and $(A \vee 0) \Rightarrow A$. (Note the reduction of the one-node tree $\langle r, \{r\} \rangle$ to $\langle 1 \rangle$.) Now q becomes a terminal variable which forces P-BCP to make the assignment $h_S(q) := 1$. At the end, P-BCP infers p, r, and q from S. ■

Let S be a finite set of p-formulas and let k be the total size of S. The creation of the forest of parse trees for formulas in S can be done in $O(k)$ steps. The total number of the simplification steps performed during the propagation of terminal variables does not exceed k. Hence, this refinement of P-BCP is linear time.

P-BCP and Definite Formulas

A *definite formula* (d-formula) is a p-formula which has at most one positive variable (Stachniak 1998). Informally speaking, d-formulas are free-form counterparts of Horn clauses. The results reported in this subsection are the refinements of Theorem 4 and Proposition 5 in the domain of d-formulas.

Theorem 7: *Suppose that S is a finite consistent set of d-formulas. Execution of P-BCP on S results in h_S such that for every $p \in Var(S)$, $S \vdash p$ iff $h_S(p) = 1$.*

Proof: In the light of Theorem 4, we only need to demonstrate the 'only if' part of the theorem. To this end, let S be as stated, let $S \vdash p$, and let h be an interpretation of L such that $h(S) = \{1\}$. Since $S \vdash p$, S contains at least one non-tautological formula. We prove the theorem by induction on the cardinality of $Var(S)$.

Let $Var(S) = \{p\}$. Note that p is '+' in every non-tautological formula of S. (Indeed, if $\beta(p^-) \in S$, then since $h(p) = 1$, $\beta(p/1)$ is a tautology; by Corollary 2, $\beta(p)$ is a tautology.) Select a non-tautological $\alpha(p^+) \in S$. By Corollary 2, if $\alpha(p/0)$ were a tautology, then so would α. So, $TV(\alpha, p) = \{1\}$ and $h_S(p) = 1$.

Next, suppose that the theorem holds for every consistent set S^* of d-formulas such that $Var(S^*)$ is of cardinality $\leq k$. Suppose that $Var(S)$ is of cardinality $k + 1$. We claim that

(a) $h_S(q)$ is defined for some $q \in Var(S)$.

If (a) were false, then we would be able to construct an interpretation h^* of L such that $h^*(S) = \{1\}$ and $h^*(p) = 0$ which, of course, would contradict $S \vdash p$. Indeed, suppose that (a) is false. For every $\alpha(q^+, r_1^-, \ldots, r_k^-) \in S$, $0 \in TV(\alpha, q)$ and $\alpha(q/0, r_1/0, \ldots, r_k/0)$ is a tautology. For every $\alpha(q^-, r_1^-, \ldots, r_k^-) \in S$, $1 \in TV(\alpha, q)$ and $\alpha(q/1, r_1/0, \ldots, r_k/0)$ is a tautology. Since q is '$-$' in α, by Proposition 1(ii) we conclude that also $\alpha(q/0, r_1/0, \ldots, r_k/0)$ is a tautology. So, if h^* is such that $h^*(q) = 0$, all $q \in Var(S)$, then $h^*(S) = \{1\}$ and $h^*(p) = 0$.

Let q be the first variable for which $h_S(q)$ is defined when executing P-BCP on S. If $p = q$, then by Lemma 3, $h_S(p) = h(p) = 1$. Suppose that $q \neq p$. Let S_0 be obtained from S by replacing q by $h(q)$ in all the formulas of S (and removing tautologies). Note that $h(S_0) = h(S) = \{1\}$ and $S_0 \vdash p$. By inductive hypothesis, $h_{S_0}(p) = 1$ when P-BCP is executed on S_0. When P-BCP is executed on S, $h_S(q)$ gets its value in the first place and, then, the algorithm follows the steps of P-BCP on S_0. Hence, at some point, P-BCP will assign the value to $h_S(p)$. By Lemma 3, $h_S(p) = 1$. ■

Proposition 8: *Let S be a finite set of d-formulas. S is inconsistent iff execution of P-BCP on S results in h_S such that for some $\alpha \in S$, $h_S(\alpha)$ is a contradictory formula.*

Proof: Suppose that S is inconsistent. Let h be the interpretation of L defined as follows: $h(q) = h_S(q)$, if $h_S(q)$ is defined, and $h(q) = 0$, for the remaining variables. Since S is inconsistent, there is $\alpha \in S$ such that $h(\alpha) = 0$. We claim that h_S is defined for every $p \in Var(\{\alpha\})$. Suppose not. Let $p_0, \ldots, p_k$ be all the variables of α for which h_S is undefined. Suppose that p_0 is '+' in α. Since $0 \in TV(\alpha, p_0)$, $\alpha(p_0/0, p_1/0, \ldots, p_k/0)$ is a tautology, or $h(\alpha) = 1$ – a contradiction. If all the variables $p_0, \ldots, p_k$ are negative in α, then $1 \in TV(\alpha, p_0)$ which means that $\alpha(p_0/1, p_1/0, \ldots, p_k/0)$ is a tautology. Since p_0 is '−' in α, by Proposition 1(ii), $\alpha(p_0/0, p_1/0, \ldots, p_k/0)$ is a tautology and, again, $h(\alpha) = 1$, which is impossible.

Since h_S is defined for every variable in $Var(\{\alpha\})$, by the definition of h and the fact that $h(\alpha) = 0$ we conclude that $h_S(\alpha)$ is a contradictory formula.

For the other half of the proof see Proposition 5. ■

P-BCP and CNF-BCP Compared

Several extensions of clausal BCP to non-clausal theories have been proposed in the literature. (McAllester, 1990) and (de Kleer, 1990) briefly review general BCP. (de Kleer, 1990) further explores CNF-BCP (we review this inference procedure below) and Prime Implicate BCP. (Dalal, 1990) and (Roy-Chowdhury and Dalal, 1997) offer two more fact propagation procedures: Fact Propagation and Restricted Fact Propagation (RFP). Of all these procedures, RFP is a reasonable compromise between efficiency and inferential power; it runs in quadratic time in the total size of an input set and is inferentially equivalent to CNF-BCP. P-BCP is a linear time procedure that operates on p-formulas. In this subsection we demonstrate that P-BCP and CNF-BCP are inferentially equivalent in the domain of p-formulas.

CNF-BCP accepts a finite set S of propositional formulas, converts S into a set S_{cnf} of clauses and, then, follows the steps of BCP on S_{cnf}. The conversion into S_{cnf} is done using the standard CNF transformation algorithm (using the laws: $\neg\neg\alpha \equiv \alpha$, $(\alpha \rightarrow \beta) \equiv (\neg\alpha \vee \beta)$, $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$, $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$, and $\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$).

Theorem 9: *Given a finite set S of p-formulas, P-BCP and CNF-BCP infer the same literals from S.*

Proof: Let S be as stated. If $\alpha \in S$, then by α^* we denote the formula obtained from α by applying one of the laws used to convert S into S_{cnf}. Let S^* be obtained from S by replacing one of $\alpha \in S$ with α^*. We note that

(a) α and α^* are logically equivalent and have the same variables; the polarity values of the variables in α^* are the same as in α (so, α^* is a p-formula).

Hence, P-BCP executed on S and on S^* computes the same substitution, i.e., $h_S = h_{S^*}$. Repeating the process of replacement of some formula α with α^* we conclude that

(b) P-BCP infers the same literals from S as from S_{cnf}.

To conclude the proof of the theorem we need

(c) P-BCP and BCP infer the same literals from S_{cnf}.

Let $v_0, \ldots, v_i, \ldots, v_n$ be the order in which P-BCP assigns constants 0 and 1 to propositional variables. Select v_i and assume that for every $j < i$, BCP assigns $h_{S_{cnf}}(v_j)$ to v_j. If $h_{S_{cnf}}(v_i) = 1$, then there is a clause $v_i \vee C \in S$. Since $TV(\alpha, v_i) = \{1\}$, $0 \vee C$ is a contradictory formula which means that either C is the empty clause or every literal in C evaluates to 0 under $h_{S_{cnf}}$. This means that BCP must assign 1 to v_i, as required. In the same way we show that if $h_{S_{cnf}}(v_i) = 0$, then BCP assigns 0 to v_i.

To conclude the proof of (c), one needs to show that if for some $v \in Var(S_{cnf})$, BCP assigns c to v, then $h_{S_{cnf}}(v) = c$. This can be done as in the first part of the proof of (c) and is left to the reader. ■

P-BCP for Non-classical Logics

Knowledge representation frequently resorts to non-classical logics (modal, temporal, many-valued). The Definition 1 of a polarized formula is meaningful for an arbitrary logical system $\mathcal{P}$ provided that some *polarity assignment algorithm* is specified which assigns polarity values '+', '−', or 'no polarity' to variables in formulas of $\mathcal{P}$. A number of such polarity assignment algorithms for classical as well as non-classical logics are discussed in (Manna and Waldinger, 1986) and (Stachniak, 1996). In this section we adopt the P-BCP algorithm to some non-classical logics.

Let $\mathcal{P} = \langle L, \vdash \rangle$ be an arbitrary propositional logic and let $\mathcal{A}$ be a polarity assignment algorithm which assigns the polarity values '+', '−', or 'no polarity' to variables in formulas of $\mathcal{P}$. We assume that

(i1) *the class of p-formulas determined by $\mathcal{A}$ has the properties described in Proposition 1.*

Going through the proof of Theorem 4 one realizes that P-BCP can be proved correct for a logic $\mathcal{P}$ if, roughly speaking, $\mathcal{P}$ can be defined using two-valued semantics (not necessarily truth-functional) in which interpretations are arbitrary mappings of formulas of $\mathcal{P}$ into, say, $\{0, 1\}$. In (Suszko, 1977) the author argues that 'every logic is two-valued'. Suszko shows that for every (propositional) logic $\mathcal{P} = \langle L, \vdash \rangle$ one can find a set $H_\vdash$ of functions from L into $\{0, 1\}$ (*logical interpretations*) such that for every $X \cup \{\alpha\} \subseteq L$,

$X \vdash \alpha$ *iff* for every $h \in H_\vdash$, $h(X) \subseteq \{1\}$ implies $h(\alpha) = 1$.

Indeed, the required set $H_\vdash$ can be define as follows. For every set $X \subseteq L$ consistent in $\mathcal{P}$ define $h_X : L \longrightarrow \{0, 1\}$ in the following way: $h(\alpha) = 1$ if $X \vdash \alpha$, and $h(\alpha) = 0$, if $X \nvdash \alpha$. Now, take $H_\vdash$ to be the set of all such mappings. If $\mathcal{P}$ has the property that every consistent set is included in a maximal consistent set,

then it is sufficient to include in $H_\vdash$ only mappings h_X, where X is maximal consistent.

Some restrictions on $H_\vdash$ and, hence, on the logics that we shall subject to our further analysis are necessary. First, we need the proper interpretation of the constants 0 and 1:

(i2) *for every* $h \in H_\vdash, h(0) = 0$ *and* $h(1) = 1$.

Second, we need some sort of negation to allow the representation of negative information. We therefore assume that the negation connective $\neg$ is available in $\mathcal{P}$ and that it satisfies at least the following condition:

(i3) *for every p-formula* $\alpha(q)$*, if* q *is '+' (is '−') in* α*, then* q *is '−' (is '+') in* $\neg\alpha$*; moreover, for every* $h \in H_\vdash, h(\neg 0) = 1$ *and* $h(\neg 1) = 0$.

Finally,

(i4) *if for some* $h \in H_\vdash$, $h(\alpha(p_0, \ldots, p_k)) = 1$*, then* $\alpha(p_0/h(p_0), \ldots, p_k/h(p_k))$ *is a tautology.*

Example C: Among the logics that satisfy (i1)–(i4) there is the modal logic $S4$ (cf. Mints, 1992) over the propositional language that contains the classical connectives $\neg, \vee, \wedge, \rightarrow$, and one modal connective $\Box$. For $S4$, logical interpretations in $H_\vdash$ can be defined in terms of maximal consistent sets of formulas (cf. Mints, 1992, Section 4). The class of $S4$ p-formulas is defined using the polarity assignment algorithm for CPL reviewed in section 'Polarized Formulas in Classical Logic'.

The proofs of (i1)–(i3) for $S4$ are straightforward and are omitted for reasons of space limitation. The proof of (i4) requires more work. In what follows, '$\alpha \equiv \beta$' abbreviates '$(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$'.

Lemma A: *Let* $\alpha(p)$ *be a p-formula of* $S4$*. Then:*

(i) $\{p \rightarrow 0\} \vdash_{S_4} \alpha(p) \equiv \alpha(p/0)$;

(ii) $\{1 \rightarrow p\} \vdash_{S_4} \alpha(p) \equiv \alpha(p/1)$.

Proof: By induction on the complexity of α. ■

Lemma B: *Suppose that* α *is a p-formula of* $S4$ *without variables. If for some* $h_X \in H_\vdash$, $h_X(\alpha) = 1$*, then* α *is a tautology.*

Proof: Let α be as stated, let W be any maximal consistent set, and let α^* be obtained from α by removing all occurrences of $\Box$. Using induction on the complexity of α one can show that

(a) $\alpha \in W$ iff $\alpha^* \in W$.

Let $h_X \in H_\vdash$ be such that $h_X(\alpha) = 1$. To show that α is a tautology we must show that for every $h_W \in H_\vdash, h_W(\alpha) = 1$ or, equivalently, that $\alpha \in W$. Select $h_W \in H_\vdash$. Since $h_X(\alpha) = 1$, $\alpha \in X$ and, by (a), $\alpha^* \in X$. Since α^* is a formula of classical logic, by the consistency of X, α^* is a tautology. Since W is maximal consistent, $\alpha^* \in W$. By (a), $\alpha \in W$. ■

Now, we are ready to demonstrate (i4). Let $\alpha(p_0, p_1)$ be a formula of $S4$ (for simplicity, assume that α has only two variables) and let $h_X(\alpha) = 1$, for some $h_X \in H_\vdash$. Suppose that $h_X(p_0) = 1$ and $h_X(p_1) = 0$. Since X is maximal consistent, $p_0 \in X$ and $p_1 \notin X$, which means that $1 \rightarrow p_0, p_1 \rightarrow 0 \in X$. By Lemma A, $\alpha(p_0/1, p_1/0) \in X$. Since $\alpha(p_0/1, p_1/0)$ has no variables, by Lemma B, this formula is a tautology. ■

Soundness of Non-classical P-BCP

Let $\mathcal{P} = \langle L, \vdash \rangle$ be an arbitrary but fixed propositional logic that satisfies (i1)–(i4) with respect to some polarity assignment algorithm. Let $H_\vdash$ be a set of logical interpretations which defines $\mathcal{P}$.

Lemma 10: *Let* S *be a finite set of p-formulas of* $\mathcal{P}$ *and let* $h \in H_\vdash$ *be such that* $h(S) = \{1\}$*. Then for every* $p \in Var(S)$*, if* $h_S(p)$ *is defined when P-BCP is executed on* S*, then* $h_S(p) = h(p)$.

Proof: Let $v_0, \ldots, v_l, \ldots, v_n$ be the order in which P-BCP assigns values 0 or 1 to propositional variables. Select $v = v_l$ and assume that for every variable v_j, $j < l$, $h(v_j) = h_S(v_j)$.

Suppose that $\alpha(v, p_i^+, r_j^-)$, $i \leq k$, $j \leq m$, has been used by P-BCP to define $h_S(v)$ and that h_S is undefined for $v, p_i, i \leq k$, and r_j, $j \leq m$. Since $h(\alpha) = 1$, by (i4), $h_S(\alpha(v/h(v), p_i/h(p_i), r_j/h(r_j)))$ is a tautology. The rest of the proof is as in the proof of Lemma 3, with (i1) replacing Proposition 1. ■

Theorem 11: *Let* S *be a finite set of p-formulas of* $\mathcal{P}$ *and let* $p \in Var(S)$ *be such that* $h_S(p)$ *is defined when P-BCP is executed on* S*. Then*

(i) $h_S(p) = 1$ *implies* $S \vdash p$;

(ii) $h_S(p) = 0$ *implies* $S \vdash \neg p$.

Theorem 11 can be proved in the same way as Theorem 4, using Lemma 10 instead of Lemma 3. Following the proof of Proposition 4 (using (i4) and Lemma 10) we can prove that

Proposition 12: *If executing P-BCP on a set* S *of p-formulas of* $\mathcal{P}$ *results in* h_S *such that for some* $\beta \in S, h_S(\beta)$ *is a contradictory formula, then* S *is inconsistent.*

Example C (cont): Every formula in $S = \{\Box p^+ \wedge \neg q^-,\ \Box q^+ \vee r^+\}$ is a p-formula of $S4$. Executing P-BCP on S results in the assignments $h_S(q) := 0$ and $h_S(p) = 1$ (using $\Box p \wedge \neg q$), and, then, $h_S(r) := 1$ (using $\Box q \vee r$). By Theorem 11, S entails p, r, and $\neg q$ in $S4$. ■

P-BCP and Many-Valued Logics

While the general two-valued semantics discussed above is also available to many-valued logics, these calculi frequently fail (i4) and, hence, the soundness of P-BCP may be lost. Here is an example.

Example D: The 3-valued logic of Łukasiewicz $\mathcal{P}_3 = \langle L, \vdash_3 \rangle$ is semantically defined in terms of three logical values: $0, \frac{1}{2}, 1$, where 1 is the only designated truth-value (cf. Stachniak, 1996). Logical connectives $\neg, \vee, \wedge, \rightarrow$ are interpreted as functions $1-x$, $max(x,y)$, $min(x,y)$, $min(1, 1-x+y)$, respectively, where $x, y \in \{0, \frac{1}{2}, 1\}$. It can be easily verified that $\mathcal{P}_3$ satisfies (i1)–(i3). Unfortunately, $\mathcal{P}_3$ fails (i4). Take $S = \{\neg p \rightarrow p\}$ and let $H_{\vdash_3}$ be any set of interpretations of L into $\{0,1\}$ which defines $\vdash_3$. Since $S \not\vdash_3 p$, there is $h \in H_{\vdash_3}$ such that $h(\neg p \rightarrow p) = 1$ and $h(p) = 0$. If (i4) were true, then $\neg 0 \rightarrow 0$ would be a tautology of $\mathcal{P}_3$. However, $\neg 0 \rightarrow 0$ is a contradictory formula of $\mathcal{P}_3$.

In $\mathcal{P}_3$, p is '+' in $\neg p \rightarrow p$ under the so-called vo-polarity assignment algorithm. The formal definition of this algorithm can be found in (Stachniak, 1996); the class of p-formulas of $\mathcal{P}_3$ determined by vo-polarity assignment algorithm is the same as the class of p-formulas for CPL discussed in the previous sections. Applying P-BCP to S results in the assignment $h_S(p) := 1$ in spite of the fact that $S \not\vdash_3 p$. ■

P-BCP can regain its soundness in the domain of finitely-valued logics if the definition of $TV(\alpha, p)$ in the local propagation step is adopted to a larger set of truth-values. To explain how this can be done, let us assume that a finitely-valued logic $\mathcal{P} = \langle L, \vdash \rangle$ is defined using a finite set of truth-values $TV = \{0, \frac{1}{n}, \ldots, \frac{n-1}{n}, 1\}$ and that 1 is the only designated truth-value. The connectives of $\mathcal{P}$ are semantically defined by truth-tables over TV. An interpretation is a mapping that assigns a truth-value from TV to every propositional variable. Every interpretation can be extended to all the formulas of L in the usual way using the truth-tables; we shall call such an extension an interpretation of L in TV. We adopt the standard semantic definition of $\vdash$: for every $X \cup \{\alpha\} \subseteq L$,

$$X \vdash \alpha \text{ iff for every interpretation } h \text{ of } L \text{ in } TV, \\ h(X) \subseteq \{1\} \text{ implies } h(\alpha) = 1.$$

Let h be an interpretation, let p be a propositional variable and let $v \in TV$. By $h[p/v]$ we denote an interpretation that is defined exactly like h except that $h[p/v](p) = v$ (the notation $h[p_0/v_0, \ldots, p_k/v_k]$ is self-explanatory). In the P-BCP algorithm, h_S is a partial substitution of constants 0 and 1 for propositional variables. To adopt P-BCP to a finitely-valued semantics we shall view h_S as a partial interpretation that assigns truth-values 0 and 1 to propositional variables. In the new algorithm, the set $TV(\alpha(p, p_i^+, r_j^-), p)$ is defined as

$$\{v \in TV : h_S[p/v, p_i/1, r_j/0](\alpha(p, p_i^+, r_j^-)) = 1\}.$$

Let us refer to this variant of P-BCP as MVL-BCP.

Theorem 13: *Let S be a finite set of p-formulas of a finitely-valued logic $\mathcal{P}$ which satisfies (i1)–(i3), and let p be a variable such that $h_S(p)$ is defined when MVL-BCP is executed on S. Then:*

(i) $h_S(p) = 1$ *implies* $S \vdash p$;

(ii) $h_S(p) = 0$ *implies* $S \vdash \neg p$.

The proof of this theorem is analogous to that of Theorem 3; it is based on Lemma 3 that also holds for MVL-BCP.

Example D (cont): Let us reconsider the set $S = \{\neg p \rightarrow p\}$. Executing MVL-BCP on S we get $TV(\neg p \rightarrow p, p) = \{1, \frac{1}{2}\} \neq \{1\}$. Hence, $h_S(p)$ is left undefined. On the other hand, executing P-BCP for CPL on S gives us $TV(\neg p \rightarrow p, p) = \{1\}$ and $h_S(p) = 1$. By Theorems 13 and 4, $S \not\vdash_3 p$ and $S \vdash p$ in CPL, as expected. ■

Acknowledgments

Research supported by a grant from the Natural Science and Engineering Research Council of Canada. Thanks to the anonymous reviewers for their insightful comments, and to Steven Bernstein for his work on the implementation of P-BCP.

References

de Kleer, J. 1990. Exploiting Locality in a TMS. In *Proc. of the Eighth National Conference on Artificial Intelligence*, 264–271.

Manna, Z. and Waldinger, R. 1986. Special Relations in Automated Deduction. *J. ACM* 33, 1–59.

McAllester, D. 1980. An Outlook on Truth Maintenance. Memo 551, MIT AI Laboratory, August.

McAllester, D. 1990. Truth Maintenance. In *Proc. of the Eighth National Conference on Artificial Intelligence*, 1109–1116.

Mints, G.E. 1992. Lewis' Systems and System T (1965–1973). In Minc G.E., *Selected Papers in Proof Theory*, Bibliopolis and North-Holland.

Nayak, P. and Williams, B. 1997. Fast Context Switching in Real-time Propositional Reasoning. In *Proc. of the Fourtinth National Conference on Artificial Intelligence*, 50–56.

Roy-Chowdhury, R. and Dalal, M. 1997. Model-Theoretic Semantics and Tractable Algorithm for CNF-BCP. In *Proc. of the Fourtinth National Conference on Artificial Intelligence*, 227–232.

Stachniak, Z. 1998. Non-Clausal Reasoning with Propositional Definite Theories. In *Proc. of the Int. Conf. on Artificial Intelligence and Symbolic Computation, Lecture Notes in Artificial Intelligence* 1476, 296–307, Springer-Verlag.

Stachniak, Z. 1996. *Resolution Proof Systems: An Algebraic Theory*, Kluwer Academic Publishers.

Suszko, R. 1977. The Frege Axiom and Polish Mathematical Logic in the 1920^s. *Studia Logica* 34, 377–380..

Tseitin, G.S. 1983. On the complexity of derivation in propositional calculus. *Automated Reasoning* (J. Siekmann and G. Wrightson, eds.), vol. 2, 466–483, Springer-Verlag.