# Constructive Plausible Logic Is Relatively Consistent

David Billington and Andrew Rock

School of Computing and Information Technology
Nathan Campus, Griffith University
Brisbane, Queensland 4111, Australia
Telephone: +61 (0)7 3875 {5017,5016}
Facsimile: +61 (0)7 3875 5051
{d.billington,a.rock}@griffith.edu.au
http://www.griffith.edu.au

**Abstract.** An implemented, efficient, propositional non-monotonic logic, called Constructive Plausible Logic, is defined and explained. Several important properties enjoyed by this logic are stated. The most important property, relative consistency, means that whenever the set of axioms is consistent so is the set of all formulas proved using defeasible information. Hence the non-monotonic deduction mechanism is trustworthy. This is the first Plausible Logic which has been proved to be relatively consistent. Constructive disjunction is characterised by the property that a disjunction can be proved if and only if at least one of its disjuncts can be proved. Constructive Plausible Logic uses constructive disjunction. Moreover the ambiguity propagating proof algorithm is simpler than the one in Billington and Rock [4].

**K**eywords: Nonmonotonicity; Common-sense reasoning; Logic; Knowledge representation.

## 1 Introduction

In the late 1980s Nute [6] introduced a non-monotonic reasoning formalism called Defeasible Logic. It was a propositional logic which dealt with uncertain and incomplete information, as well as factual information. However the uncertainty and incompleteness was not represented by numbers. Moreover the reasoning was based on principles rather than on the manipulation of numbers. So probabilities and certainty factors have no place in Defeasible Logic. Defeasible Logic has many desirable properties, but perhaps the most important of these is a deterministic polynomial deduction procedure which enables a straightforward implementation of this logic. However, Defeasible Logic can neither represent nor prove disjunctions. In the late 1990s Billington [2] introduced Plausible Logic which was based on Defeasible Logic but which could represent and prove clauses. Both Defeasible Logic and Plausible Logic have a syntax which distinguishes between formulas proved from just the facts or axioms, and those proved using defeasible information.

T.D. Gedeon and L.C.C. Fung (Eds.): AI 2003, LNAI 2903, pp. 954–965, 2003.

Relative consistency means that whenever the set of axioms is consistent so is the set of all formulas proved using defeasible information. Relative consistency is important because it shows that the non-monotonic deduction mechanism does not create inconsistencies. The incorporation of disjunction makes proving relative consistency very difficult. Indeed relative consistency has not been proved for the Plausible Logic introduced in Billington and Rock [4].

Constructive disjunction is characterised by the property that $\bigvee L$ can be proved if and only if at least one element of $L$ can be proved. With classical disjunction it is possible to prove $\bigvee L$ and not be able to prove any element of $L$. Experience with translating business rules and regulations into Plausible Logic indicates that often only constructive disjunction is needed. For example, the eligibility criteria for becoming a member of the IEEE is a disjunctive list, which means that if at least one criterion in the list is satisfied then the candidate is eligible. In general constructive disjunction is exactly what is needed for any disjunctive list of eligibility criteria. The fact that constructive disjunction is computationally simpler than classical disjunction means that constructive disjunction may be just the right compromise between computational efficiency and expressive power. Certainly Constructive Plausible Logic is much more expressive than Defeasible Logic.

The purpose of this paper is to define, and give some intuitions about, a Plausible Logic which is relatively consistent. All attempts at proving relative consistency for many different Plausible Logics with a non-constructive disjunction have failed. As this paper shows, when the simpler constructive disjunction is used then relative consistency can be proved.

A separate issue is that of ambiguity. An atom $a$ is *ambiguous* if and only if neither $a$ nor its negation, $\neg a$, can be proved. Suppose there is evidence for $b$. If $a$ is ambiguous and $a$ is evidence for $\neg b$ then what should be concluded about $b$? A logic is ambiguity blocking if it can conclude $b$; and it is ambiguity propagating if $b$ is ambiguous, because the ambiguity of $a$ has been propagated to $b$.

The Plausible Logic presented in this paper has a simplified ambiguity propagating proof algorithm compared to the Plausible Logic in Billington and Rock [4]. The ambiguity propagating proof algorithm of Billington and Rock [4] required an auxiliary proof algorithm. No such auxiliary algorithm is needed in this paper.

Section 2 of this paper defines and explains Constructive Plausible Logic. Section 3 presents the main results. Unfortunately space limitations allow the inclusion of only the proof of the main theorem on relative consistency. However all the proofs are in Billington [3]. Section 4 describes our implementation of Constructive Plausible Logic, and contains an example of a reasoning problem and its automated proof. Section 5 is the conclusion.

## 2 Constructive Plausible Logic

We begin by establishing our basic notation and terminology. We often abbreviate "if and only if" by "iff". $X$ is a subset of $Y$ is denoted by $X \subseteq Y$; the

notation $X \subset Y$ means $X \subseteq Y$ and $X \neq Y$, and denotes that $X$ is a *proper subset* of $Y$. The empty set is denoted by $\{\}$, and the set of all integers by $\mathbb{Z}$. If $m$ and $n$ are integers then we define $[m..n] = \{i \in \mathbb{Z} : m \leq i \leq n\}$. The cardinality of a set $S$ is denoted by $|S|$. The length of a sequence $P$ is denoted by $|P|$. Let $P = (P(1), P(2), \ldots, P(|P|))$ be a finite sequence. If $i \in [1..|P|]$ then $P[1..i] = (P(1), \ldots, P(i))$, and if $i = 0$ then $P[1..i] = ()$, the empty sequence. The notation $x \in P$ means that there exists $j$ in $[1..|P|]$ such that $x = P(j)$. And $x \notin P$ means not$(x \in P)$. The concatenation of a sequence $P$ onto the beginning of the sequence $Q$ is denoted by $P\&Q$. Let $S$ be any set. It is sometimes convenient to abbreviate "for all $x$ in $S$" by "$\forall x \in S$". Also "there exists an $x$ in $S$ such that" is sometimes abbreviated to "$\exists x \in S$ such that", and sometimes to just "$\exists x \in S$". $\mathcal{P}(L)$ is the powerset of $L$. When convenient we abbreviate "for all $I$ in $\mathcal{P}(S)$" by "$\forall I \subseteq S$". Also "there exists an $I$ in $\mathcal{P}(S)$ such that" is sometimes abbreviated to "$\exists I \subseteq S$ such that", and sometimes to just "$\exists I \subseteq S$".

Our *alphabet* is the union of the following four pairwise disjoint sets of symbols: a non-empty finite set, $Atm$, of (propositional) atoms; the set $\{\neg, \bigwedge, \bigvee, \rightarrow, \Rightarrow, \rightharpoonup\}$ of connectives; the set $\{+, -, \delta, \gamma, \pi\}$ of proof symbols; and the set of punctuation marks consisting of the comma and both braces. By a *literal* we mean any atom, $a$, or its negation, $\neg a$. A clause, $\bigvee L$, is the disjunction of a finite set, $L$, of literals. $\bigvee\{\}$ is the *empty clause* or *falsum* and is thought of as always being false. If $l$ is a literal then we regard $\bigvee\{l\}$ as another notation for $l$ and so each literal is a clause. A clause $\bigvee L$ is a tautology iff both an atom and its negation are in $L$. A *contingent* clause is a clause which is not empty and not a tautology. A *dual-clause*, $\bigwedge L$, is the conjunction of a finite set, $L$, of literals. $\bigwedge\{\}$ is the *empty dual-clause* or *verum* and is thought of as always being true. If $l$ is a literal then we regard $\bigwedge\{l\}$ as another notation for $l$ and so each literal is a dual-clause. Thus $\bigwedge\{l\} = l = \bigvee\{l\}$. Neither the verum nor the falsum are literals. The verum is not a clause, and the falsum is not a dual-clause. A *cnf-formula*, $\bigwedge C$, is the conjunction of a finite set, $C$, of clauses. A *dnf-formula*, $\bigvee D$, is the disjunction of a finite set, $D$, of dual-clauses. If $c$ is a clause then we regard $\bigwedge\{c\}$ as another notation for $c$. If $d$ is a dual-clause then we regard $\bigvee\{d\}$ as another notation for $d$. Thus both clauses and dual-clauses are both cnf-formulas and dnf-formulas. By a *formula* we mean any cnf-formula or any dnf-formula. The set of all literals is denoted by *Lit*; the set of all clauses is denoted by *Cls*; the set of all dual-clauses is denoted by *DCls*; the set of all cnf-formulas is denoted by *CnfFrm*; the set of all dnf-formulas is denoted by *DnfFrm*; and the set of all formulas is denoted by *Frm*. *Frm* is finite.

We define the *complement*, $\sim f$, of a formula $f$ and the *complement*, $\sim F$, of a set of formulas $F$ as follows. If $f$ is an atom then $\sim f$ is $\neg f$; and $\sim\neg f$ is $f$. If $L$ is a set of literals then $\sim L = \{\sim l : l \in L\}$. If $\bigvee L$ is a clause then $\sim\bigvee L = \bigwedge\sim L$. If $\bigwedge L$ is a dual-clause then $\sim\bigwedge L = \bigvee\sim L$. So the complement of a clause is a dual-clause, and the complement of a dual-clause is a clause. In particular the falsum and the verum are complements of each other. If $E$ is a set of clauses or a set of dual-clauses then $\sim E = \{\sim e : e \in E\}$. If $\bigwedge C$ is a cnf-formula then $\sim\bigwedge C = \bigvee\sim C$. If $\bigvee D$ is a dnf-formula then $\sim\bigvee D = \bigwedge\sim D$. So the complement

of a cnf-formula is a dnf-formula, and the complement of a dnf-formula is a cnf-formula. If $F$ is a set of formulas then $\sim F = \{\sim f : f \in F\}$. Both *Lit* and *Frm* are closed under complementation.

The information with which constructive plausible logic reasons is either certain or defeasible. All the information is represented by various kinds of rules and a priority relation on those rules. Define $r$ to be a *rule* iff $r = (A(r), arrow(r), c(r))$ where $A(r)$ is a finite set of literals called the *antecedent* of $r$, $arrow(r) \in \{\rightarrow, \Rightarrow, \rightharpoonup\}$, $c(r)$ is a literal called the *consequent* of $r$, $c(r) \notin A(r)$, and $\sim c(r) \notin A(r)$. A rule $r$ which contains the *strict arrow*, $\rightarrow$, is called a *strict rule* and is usually written $A(r) \rightarrow c(r)$. A rule $r$ which contains the *plausible arrow*, $\Rightarrow$, is called a *plausible rule* and is usually written $A(r) \Rightarrow c(r)$. A rule $r$ which contains the *defeater arrow*, $\rightharpoonup$, is called a *defeater rule* and is usually written $A(r) \rightharpoonup c(r)$. The antecedent of a rule can be the empty set. The set of all rules is denoted by *Rul*. *Rul* is finite.

Strict rules, for example $A \rightarrow l$, behave like the material conditional. If all the literals in $A$ are proved then $l$ can be deduced. Plausible rules, for example $A \Rightarrow l$, represent some of the aspects of a situation which are plausible. If all the literals in $A$ are proved then $l$ can be deduced provided that all the evidence against $l$ has been defeated. So we take $A \Rightarrow l$ to mean that, in the absence of evidence against $l$, $A$ is sufficient evidence for concluding $l$. A defeater rule, for example $A \rightharpoonup \sim l$, is evidence against $l$, but it is not evidence for $\sim l$. $A \rightharpoonup \sim l$ can be defeated by defeating $\bigwedge A$. Defeater rules can be used to prevent conclusions which would be too risky. For instance, given the rules $a \Rightarrow b$ and $b \Rightarrow c$ it may be too risky to conclude that things with property $a$ usually have property $c$. In which case we could add the defeater rule $a \rightharpoonup \sim c$. In this case adding $a \Rightarrow \sim c$ would be wrong because having property $a$ is not a reason for having property $\sim c$, indeed it is a weak reason for having property $c$.

Let $R$ be any set of rules. The set of antecedents of $R$ is denoted by $A(R)$; that is $A(R) = \{A(r) : r \in R\}$. The set of consequents of $R$ is denoted by $c(R)$; that is $c(R) = \{c(r) : r \in R\}$. We denote the set of strict rules in $R$ by $R_s$, the set of plausible rules in $R$ by $R_p$, and the set of defeater rules in $R$ by $R_d$. Also we define $R_{pd} = R_p \cup R_d$ and $R_{sp} = R_s \cup R_p$.

Let $l$ be any literal. If $C$ is any set of clauses define $C[l] = \{\bigvee L \in C : l \in L\}$ to be the set of all clauses in $C$ which contain $l$. If $R$ is any set of rules and $L$ is any set of literals then define $R[l] = \{r \in R : l = c(r)\}$ to be the set of all rules in $R$ which end with $l$; and $R[L] = \{r \in R : c(r) \in L\}$ to be the set of all rules in $R$ which have a consequent in $L$.

Any binary relation, $>$, on any set $S$ is *cyclic* iff there exists a sequence, $(r_1, r_2, \ldots, r_n)$ where $n \geq 1$, of elements of $S$ such that $r_1 > r_2 > \ldots > r_n > r_1$. A relation is *acyclic* iff it is not cyclic. If $R$ is a set of rules then $>$ is a *priority relation* on $R$ iff $>$ is an acyclic binary relation on $R$ such that $>$ is a subset of $R_p \times R_{pd}$. We read $r_1 > r_2$ as $r_1$ *beats* $r_2$, or $r_2$ is *beaten* by $r_1$. Notice that strict rules never beat, and are never beaten by, any rule. Also defeater rules never beat any rule. Let $R[l; s] = \{t \in R[l] : t > s\}$ be the set of all rules in $R$ with consequent $l$ that beat $s$.

A *plausible description* of a situation is a 4-tuple $PD = (Ax, R_p, R_d, >)$ such that PD1, PD2, PD3, and PD4 all hold.
(PD1) $Ax$ is a set of contingent clauses.
(PD2) $R_p$ is a set of plausible rules.
(PD3) $R_d$ is a set of defeater rules.
(PD4) $>$ is a priority relation on $R_{pd}$.
The clauses in $Ax$, called *axioms*, characterise the aspects of the situation that are certain. The set of strict rules, $R_s$, is defined by $R_s = \{\sim(L - \{l\}) \to l : l \in L \text{ and } \bigvee L \in Ax\}$. Define $R = R_s \cup R_p \cup R_d$ to be the set of rules *generated* from $PD$. The ordered pair $(R, >)$ is called a *plausible theory*. If $T = (R, >)$ is a plausible theory then $Ax(T) = \{\bigvee(\{c(r)\} \cup \sim A(r)) : r \in R_s\}$ is the set of axioms from which $R_s$ was generated.

Let $S$ be a set of clauses. A clause $C_n$ is *resolution-derivable from* $S$ iff there is a finite sequence of clauses $C_1, \ldots, C_n$ such that for each $i$ in $[1..n]$, either $C_i \in S$ or $C_i$ is the resolvent of two preceding clauses. The sequence $C_1, \ldots, C_n$ is called a *resolution-derivation* of $C_n$ *from* $S$. The set of all clauses which are resolution-derivable from $S$ is denoted by $Res(S)$. Define $Rsn(S) = Res(S) - \{\bigvee\{\}\}$ to be the set of all non-empty clauses in $Res(S)$.

Let $S$ be a set of sets. Define the set of *minimal elements* of $S$, $Min(S)$, to be the set of minimal elements of the partially ordered set $(S, \subseteq)$. That is, $Min(S) = \{Y \in S : \text{if } X \subset Y \text{ then } X \notin S\}$.

Let $R$ be the set of rules generated from some plausible description $(Ax, R_p, R_d, >)$. Define $Inc(R) = \{\sim L : \bigvee L \in Rsn(Ax) \cup \{\bigvee\{k, \sim k\} : k \in c(R)\}\}$ to be the set of non-empty sets of literals which are *inconsistent with* $R$. Define $Inc(R, l) = Min(\{I - \{l\} : I \in Inc(R) \text{ and } l \in I\})$. Each member of $Inc(R, l)$ is a minimal set of literals which is *inconsistent with* $l$. Since $Ax$ is finite and $R$ is finite, $Res(Ax)$, $Rsn(Ax)$, $Inc(R)$, and $Inc(R, l)$ are finite.

Cnf-formulas can be proved at three different levels of certainty or confidence. The definite level, indicated by $\delta$, is like classical monotonic proof in that more information cannot defeat a previous proof. If a formula is proved definitely then one should behave as if it is true. Proof at the general [respectively, plausible] level, indicated by $\gamma$ [respectively, $\pi$], is non-monotonic and propagates [respectively, blocks] ambiguity. If a formula is proved at the general or plausible level then one should behave as if it is true, even though it may turn out to be false.

It is sometimes necessary to prove that a cnf-formula is impossible to prove. To signify the six possible cases of proof we use the following *tags*: $+\delta$, $-\delta$, $+\gamma$, $-\gamma$, $+\pi$, $-\pi$. If $\alpha \in \{\delta, \gamma, \pi\}$ then $+\alpha f$ indicates $f$ is proved at the $\alpha$ level, and $-\alpha f$ indicates that we have proved that $+\alpha f$ is impossible prove. A *tagged* formula is a formula preceded by a tag; so all tagged formulas have the form $\pm\alpha f$ where $\pm \in \{+, -\}$, $\alpha \in \{\delta, \gamma, \pi\}$, and $f$ is a formula.

In the following *inference conditions*, $P = (P(1), \ldots, P(|P|))$ is a finite sequence, $(R, >)$ is a plausible theory, $C$ is a non-singleton set of clauses, $L$ is a non-singleton set of literals, $l$ is a literal, and $\alpha \in \{\delta, \gamma, \pi\}$.

$+\bigwedge$) If $P(i+1) = +\alpha\bigwedge C$ then $\forall c \in C$, $+\alpha c \in P[1..i]$.
$-\bigwedge$) If $P(i+1) = -\alpha\bigwedge C$ then $\exists c \in C$, $-\alpha c \in P[1..i]$.
$+\bigvee$) If $P(i+1) = +\alpha\bigvee L$ then $\exists l \in L$, $+\alpha l \in P[1..i]$.
$-\bigvee$) If $P(i+1) = -\alpha\bigvee L$ then $\forall l \in L$, $-\alpha l \in P[1..i]$.
$+\mathcal{L}$) If $P(i+1) = +\alpha l$ then either
.1) $\exists r \in R_s[l], +\alpha\bigwedge A(r) \in P[1..i]$; or
.2) both
.1) $\alpha \in \{\gamma, \pi\}$ and $\exists r \in R_p[l], +\alpha\bigwedge A(r) \in P[1..i]$, and
.2) $\forall J \in Inc(R, l)\ \exists j \in J\ \forall s \in R[j]$ either
.1) $\exists t \in R_p[l; s], +\alpha\bigwedge A(t) \in P[1..i]$; or
.2) $A(s) \neq \{\}$ and $+\alpha\sim\bigwedge A(s) \in P[1..i]$; or
.3) $\alpha = \pi$ and $-\pi\bigwedge A(s) \in P[1..i]$.
$-\mathcal{L}$) If $P(i+1) = -\alpha l$ then
.1) $\forall r \in R_s[l], -\alpha\bigwedge A(r) \in P[1..i]$, and
.2) either
.1) $\alpha = \delta$ or $\forall r \in R_p[l], -\alpha\bigwedge A(r) \in P[1..i]$; or
.2) $\exists J \in Inc(R, l)\ \forall j \in J\ \exists s \in R[j]$ such that
.1) $\forall t \in R_p[l; s], -\alpha\bigwedge A(t) \in P[1..i]$, and
.2) $A(s) = \{\}$ or $-\alpha\sim\bigwedge A(s) \in P[1..i]$, and
.3) $\alpha = \gamma$ or $+\pi\bigwedge A(s) \in P[1..i]$.

Let $T$ be a plausible theory. A formal proof or $T$-*derivation* $P$ is a finite sequence, $P = (P(1), \ldots, P(|P|))$, of tagged cnf-formulas such that for all $i$ in $[0..|P|-1]$ all the inference conditions hold. So () is a $T$-derivation. If the plausible theory $T$ is known or is irrelevant then we often abbreviate $T$-derivation to just derivation. Each element $\pm\alpha f$ of a derivation is called a *line* of the derivation.

First notice that the inference conditions are paired, one positive and one negative. The negative one is just the strong negation of the positive one. That is, the negative condition is obtained by negating the positive condition and then replacing $+X \notin P[1..i]$ by $-X \in P[1..i]$, and $-X \notin P[1..i]$ by $+X \in P[1..i]$. So only the positive inference conditions need explaining. $+\bigwedge$ says that to prove a conjunction at level $\alpha$, all the conjuncts must have been proved at level $\alpha$ previously. $+\bigvee$ says that to prove a disjunction at level $\alpha$, at least one of the disjuncts must have been proved at level $\alpha$ previously. This is just the definition of constructive disjunction.

$+\mathcal{L}$ shows how to prove a literal $l$ at level $\alpha$. $+\mathcal{L}$.1 says that if the conjunction of the antecedent of a strict rule has been previously proved at level $\alpha$, then its consequent can be added to the derivation. $+\mathcal{L}$.1 is just modus ponens for strict rules. $+\mathcal{L}$.2.1 says that $+\mathcal{L}$.1 is the only way to prove a literal at level $\delta$. Hence proof at level $\delta$ only uses strict rules, which were generated from the axioms.

$+\mathcal{L}$.2 gives another way of proving a literal at the defeasible levels. $+\mathcal{L}$.2.1 says there must be some evidence for the literal, and $+\mathcal{L}$.2.2 says that all the evidence against the literal must be defeated. $+\mathcal{L}$.2.1 is similar to $+\mathcal{L}$.1 but with plausible rules replacing strict rules. $+\mathcal{L}$.2.2 says that every set of literals which is inconsistent with $l$ must contain an element $j$ such that every rule $s$ supporting $j$ is defeated. Among other things this ensures that $j$ cannot be

proved at a defeasible level. $+\mathcal{L}.2.2.1$, $+\mathcal{L}.2.2.2$, and $+\mathcal{L}.2.2.3$ give the three ways in which a rule can be defeated. $+\mathcal{L}.2.2.1$ says that $s$ is beaten by an applicable plausible rule, $t$, which supports $l$. $+\mathcal{L}.2.2.2$ says that the antecedent of $s$ is not empty and that the complement of the conjunction of the antecedent of $s$ has previously been proved. $+\mathcal{L}.2.2.3$ gives the $\pi$ level an alternative way to defeat $s$, which is not available at the $\gamma$ level. This is the only difference between these two defeasible levels. $+\mathcal{L}.2.2.3$ says that at the $\pi$ level the conjunction of the antecedent of $s$ has previously been proved to be not provable.

The notation $T \vdash \pm\alpha f$ means that $\pm\alpha f$ is in a $T$-derivation. If $\alpha \in \{\delta, \gamma, \pi\}$ then we define $T(+\alpha) = \{f : T \vdash +\alpha f\}$ and $T(-\alpha) = \{f : T \vdash -\alpha f\}$. A *constructive plausible logic* consists of a plausible theory and the inference conditions.

## 3 Results

This section contains some of the results which have been proved for Constructive Plausible Logic. The proofs of these and other results are in Billington [3].

Since a negative tag means that the corresponding positively tagged formula has been proved to be unprovable, it would be reassuring to know that the same formula cannot have both a positive and a negative tag. This property, called coherence, is proved in the first result.

Intuitively an ambiguity propagating proof procedure should be more reliable than an ambiguity blocking proof procedure. This is because in an ambiguity propagating proof procedure more evidence is required to defeat an attack than in an ambiguity blocking proof procedure. So it would be good to know that every formula that can be proved at the $\delta$ level (using only infallible information) can be proved at the $\gamma$ level (which is ambiguity propagating), and every formula that can be proved at the $\gamma$ level can be proved at the $\pi$ level (which is ambiguity blocking). This hierarchy is established in the first result, as is a corresponding hierarchy for the negative tags.

**Theorem 1 (Coherence and Hierarchy).**
Let $T$ be a plausible theory and suppose $\alpha \in \{\delta, \gamma, \pi\}$.
(1) (Coherence) $T(+\alpha) \cap T(-\alpha) = \{\}$.
(2) (Hierarchy) $T(+\delta) \subseteq T(+\gamma) \subseteq T(+\pi)$, and $T(-\pi) \subseteq T(-\gamma) \subseteq T(-\delta)$.
**End**

Sets in $Inc(R)$ are inconsistent with the rules in $R$. So it would be nice to show that it is not possible to prove every element of a set in $Inc(R)$. Unfortunately it is possible to prove every element of a set in $Inc(R)$, but only if the axioms were inconsistent. This is what lemma 3 says. To help prove lemma 3, we need lemma 2 which shows that if every element of a set in $Inc(R)$ is proved then the fault is confined to just the strict rules (which were derived from the axioms).

**Lemma 2 (Inconsistency Needs Strict Rules).**
Suppose $T = (R, >)$ is a plausible theory, $I \in Inc(R)$, and $\alpha \in \{\delta, \gamma, \pi\}$. Let $P$ be a $T$-derivation such that for all $l \in I$, $+\alpha l \in P$. Then either

1. for all $l \in I$, $+\mathcal{L}.2$ fails, and so $+\mathcal{L}.1$ holds; or
2. $\alpha \in \{\gamma, \pi\}$ and there exists a literal $k$ such that $+\alpha k$ and $+\alpha{\sim}k$ are in a proper prefix of $P$.

**End**

Let $T$ be a plausible theory and $F$ be a set of cnf-formulas. Define $F^{-\wedge} = \{c : \bigwedge C \in F \text{ and } c \in C\}$. Then $F$ is *consistent* [respectively, $T$-*consistent*] iff $\bigvee\{\} \notin Res(F^{-\wedge})$ [respectively, $\bigvee\{\} \notin Res(F^{-\wedge} \cup Ax(T))$]. So $T$-consistent means consistent with the axioms of $T$. $F$ is *inconsistent* iff $F$ is not consistent.

Since $Res(F^{-\wedge}) \subseteq Res(F^{-\wedge} \cup Ax(T))$, if $F$ is $T$-consistent then $F$ is consistent. The converse is not always true. Consider the following counter-example. Let $Ax(T) = \{\bigvee\{a, b\}\}$, and $F = F^{-\wedge} = \{\neg a, \neg b\}$. Then $F$ is consistent but $F$ is not $T$-consistent.

**Lemma 3 (Relative Consistency for Members of $Inc(R)$).**
Suppose $T = (R, >)$ is a plausible theory and $\alpha \in \{\delta, \gamma, \pi\}$. If $I \in Inc(R)$ and there is a $T$-derivation $P$ such that for all $l \in I$, $+\alpha l \in P$ then $Ax(T)$ is inconsistent.
**End**

Consider the following property. If two clauses can be proved then their resolvent (if it exists) can be proved. At least some variation of this "resolution property" seems to be necessary for relative consistency to be proved. Hence a resolution property is not only a useful result by itself, but also highly important. The exact form of the resolution property which Constructive Plausible Logic satisfies is given in lemma 4.

**Lemma 4 (Resolution Property).**
Suppose $T$ is a plausible theory and $\alpha \in \{\delta, \gamma, \pi\}$. Let $L$ and $M$ be two sets of literals and let $l$ be a literal. If $\bigvee(L \cup \{l\}) \in T(+\alpha)$ and $\bigvee(M \cup \{{\sim}l\}) \in T(+\alpha)$ then either $\bigvee(L \cup M) \in T(+\alpha)$ or $\{l, {\sim}l\} \subseteq T(+\alpha)$.
**End**

We would like to show that the set of all proved formulas was consistent. Unfortunately this is not true, because if the axioms are inconsistent then the set of all proved formulas may also be inconsistent. Our final result shows that if the axioms are consistent then the set of all proved formulas is not only consistent but also consistent with the axioms.

**Theorem 5 (Relative Consistency).**
If $T$ is a plausible theory and $\alpha \in \{\delta, \gamma, \pi\}$ then $T(+\alpha)$ is $T$-consistent iff $Ax(T)$ is consistent.

**Proof.**
To prove this theorem we shall need the definition of a subclause and a technical lemma, Lemma TL, concerning only classical propositional resolution.

If $\bigvee L$ and $\bigvee M$ are two clauses then $\bigvee L$ is a *subclause* of $\bigvee M$ iff $L \subseteq M$.

**Lemma TL.**
Let $S$ and $S^*$ be two sets of clauses. Let $L$ and $X$ be two sets of literals.

1. If $S \subseteq S^*$ then $Res(S) \subseteq Res(S^*)$, and $Rsn(S) \subseteq Rsn(S^*)$.
2. If every clause in $S^*$ has a subclause in $S$ then every clause in $Res(S^*)$ has a subclause in $Res(S)$.
3. If $\bigvee X \in Res(S \cup L)$ and $\bigvee X \notin Res(L)$, then there is a finite subset $K$ of $L$ such that $\bigvee(X \cup {\sim}K) \in Res(S)$, and $X \cap {\sim}K = \{\}$.

**EndTL**

Let $Ax = Ax(T)$. Then $Ax^{-\wedge} = Ax$, and $T(+\alpha)^{-\wedge} = T(+\alpha) - \{\bigwedge C : \bigwedge C \in T(+\alpha)$ and $|C| \neq 1\}$. By lemma TL(1), $Res(Ax) \subseteq Res(T(+\alpha)^{-\wedge} \cup Ax)$, so if $T(+\alpha)$ is $T$-consistent then $Ax$ is consistent.

Conversely, suppose $T(+\alpha)$ is not $T$-consistent. Then $\bigvee\{\} \in Res(T(+\alpha)^{-\wedge} \cup Ax)$. Let $T = (R, >)$. If there is a literal $l$ such that $\{l, {\sim}l\} \subseteq T(+\alpha)^{-\wedge}$, then $\{l, {\sim}l\} \in Inc(R)$, and so by lemma 3, $Ax$ is inconsistent. So suppose there is no literal $l$ such that $\{l, {\sim}l\} \subseteq T(+\alpha)^{-\wedge}$. By lemma 4, $Rsn(T(+\alpha)^{-\wedge}) = T(+\alpha)^{-\wedge}$. If $\bigvee\{\} \in Res(T(+\alpha)^{-\wedge})$ then there is a literal $k$ such that $\{k, {\sim}k\} \subseteq Rsn(T(+\alpha)^{-\wedge})$, and hence $\{k, {\sim}k\} \subseteq T(+\alpha)^{-\wedge}$. So suppose $\bigvee\{\} \notin Res(T(+\alpha)^{-\wedge})$. Let $L$ be the set of all literals in $T(+\alpha)$. Then $L \subseteq T(+\alpha)^{-\wedge}$, and so by lemma TL(1), $\bigvee\{\} \notin Res(L)$. By $+\bigvee$, every clause in $T(+\alpha)^{-\wedge} \cup Ax$ has a subclause in $L \cup Ax$, and so by lemma TL(2), every clause in $Res(T(+\alpha)^{-\wedge} \cup Ax)$ has a subclause in $Res(L \cup Ax)$. Hence $\bigvee\{\} \in Res(L \cup Ax)$. By lemma TL(3), there is a finite subset $K$ of $L$ such that $\bigvee({\sim}K) \in Res(Ax)$. If ${\sim}K = \{\}$ then $Ax$ is inconsistent. So suppose ${\sim}K$ is not empty. Then $\bigvee({\sim}K) \in Rsn(Ax)$ and so $K \in Inc(R)$. By lemma 3, $Ax$ is inconsistent.
**EndProof5**

## 4 Implementation

Constructive Plausible Logic has been implemented as a tool, CPL, that attempts the proofs of tagged formulas. The tool is written in literate Haskell, and is fully listed and documented in Rock [9]. It is implemented in a manner emphasising correctness and simplicity, similar to that of previous versions of Plausible Logic [4, 8], and Defeasible Logic [5, 7], but does not at present contain as many speed optimisations to cope with hundreds of thousands of rules, and is simpler to use. It could be extended with more usage options and optimisations to match the previous systems. It presently consists of about 1300 logical lines of code, compared to 5000 for the previous implementation of Plausible Logic [8].

To demonstrate the tool, the following reasoning puzzle will be used. We know for certain that Hans is a native speaker of Pennsylvania-Dutch, *nspd*, that native speakers of Pennsylvania-Dutch are native speakers of German, $nspd \rightarrow nsg$ or $\bigvee\{\neg nspd, nsg\}$, and that persons born in Pennsylvania are born in the United States of America, $bp \rightarrow busa$ or $\bigvee\{\neg bp, busa\}$. We also know that *usually* native

```
/* file:    PennDutch.d
   purpose: Generalised competitors. */

% plausible description:
    nspd.
    \/{~nspd, nsg}.
    \/{~bp, busa}.
R1: nsg =>~busa.
R2: nspd => bp.
    R2 > R1.

% the requested proofs:
output{+d busa}.  output{+g busa}.  output{+p busa}.
output{-d busa}.  output{-g busa}.  output{-p busa}.
output{+d~busa}.  output{+g~busa}.  output{+p~busa}.
output{-d~busa}.  output{-g~busa}.  output{-p~busa}.
output{+d bp}.    output{+g bp}.    output{+p bp}.
output{-d bp}.    output{-g bp}.    output{-p bp}.
output{+d~bp}.    output{+g~bp}.    output{+p~bp}.
output{-d~bp}.    output{-g~bp}.    output{-p~bp}.
```

**Fig. 1.** CPL input file containing the Pennsylvania-Dutch plausible description and requests to output specific proofs

speakers of German are not born in the United States, $nsg \Rightarrow \neg busa$, but that *usually* native speakers of Pennsylvania-Dutch are born in Pennsylvania, $nspd \Rightarrow bp$. The latter rule, being more specific, should take priority over the former. These axioms, plausible rules and the priority form a plausible description.

$nspd$
$\bigvee\{\neg nspd, nsg\}$
$\bigvee\{\neg bp, busa\}$
$r_1$: $nsg \Rightarrow \neg busa$
$r_2$: $nspd \Rightarrow bp$
$r_2 > r_1$

We would like to know whether we can conclude that Hans was born in the USA. Figure 1 shows the plausible description coded for input to the CPL tool. $\bigvee$ is approximated by `\/`. `%`, `/*` and `*/` delimit comments as in Prolog. The input includes `output` directives to request proofs of tagged formulas. In these `d`, `g` and `p` stand for $\delta$, $\gamma$ and $\pi$ respectively.

The CPL tool prints traces for all of the requested proofs and a summary table of the results. The trace of the proof of $+\pi busa$ is listed in Figure 2. The trace starts with the tagged formulas to be proved, the goal. Each line of each inference rule used is indicated by printing the label for that line, *e.g.* `+L.1` for line $+\mathcal{L}.1$, and `-/\` for line $-\bigwedge$. As variables scoped by $\forall$ and $\exists$ are instantiated their values are printed. The attempted proofs of subgoals are printed indented.

```
To Prove: +p busa
.  +L.1
.  r = {bp} -> busa
.  To Prove: +p bp
.  .  +L.1
.  .  +L.2.1
.  .  r = R2: {nspd} => bp
.  .  To Prove: +p nspd
.  .  .  +L.1
.  .  .  r = {} -> nspd
.  .  .  To Prove: +p /\{}
.  .  .  .  +/\
.  .  .  Proved: +p /\{}
.  .  Proved: +p nspd
.  .  +L.2.2
.  .  J = [~busa]
.  .  j =~busa
.  .  s = R1: {nsg} =>~busa
.  .  +L.2.2.1
.  .  t = R2: {nspd} => bp
.  .  Proved previously: +p nspd
.  .  J = [~bp]
.  .  j =~bp
.  .  s = {~busa} ->~bp
.  .  +L.2.2.1
.  .  +L.2.2.2
.  .  Loop detected: +p busa
.  .  +L.2.2.3
.  .  To Prove: -p~busa
.  .  .  -L.1
.  .  .  -L.2.1
.  .  .  r = R1: {nsg} =>~busa
.  .  .  To Prove: -p nsg
.  .  .  .  -L.1
.  .  .  .  r = {nspd} -> nsg
.  .  .  .  To Prove: -p nspd
.  .  .  .  .  -L.1
.  .  .  .  .  r = {} -> nspd
.  .  .  .  .  To Prove: -p /\{}
.  .  .  .  .  .  -/\
.  .  .  .  .  Not proved: -p /\{}
.  .  .  .  Not proved: -p nspd
.  .  .  Not proved: -p nsg
.  .  .  -L.2.2
.  .  .  J = [bp]
.  .  .  j = bp
.  .  .  s = R2: {nspd} => bp
.  .  .  -L.2.2.1
.  .  .  -L.2.2.2
.  .  .  To Prove: -p~nspd
.  .  .  .  -L.1
.  .  .  .  r = {~nsg} ->~nspd
.  .  .  .  To Prove: -p~nsg
.  .  .  .  .  -L.1
.  .  .  .  .  -L.2.1
.  .  .  .  Proved: -p~nsg
.  .  .  .  -L.2.1
.  .  .  Proved: -p~nspd
.  .  .  -L.2.2.3
.  .  .  Proved previously: +p nspd
.  .  Proved: -p~busa
.  Proved: +p bp
Proved: +p busa
Goal count = 10
```

**Fig. 2.** CPL trace of the proof $+\pi busa$

Duplicated proofs of subgoals are avoided by maintaining a history of prior results of attempted proofs. This same history is used to detect loops, where a goal generates itself as a subgoal. Where a loop is detected the proof must be achieved by an alternate path if possible. The trace ends with a count of the goals and subgoals, a measure of the effort required for that proof.

## 5 Conclusion

A constructive Plausible Logic has been defined and explained and implemented. This is the first Plausible Logic which has been proved to be relatively consistent, an important property ensuring that the non-monotonic deduction mechanism

is trustworthy. It also has the desirable properties of coherence, hierarchy, and resolution. Moreover its ambiguity propagating proof algorithm is simpler than the one in Billington and Rock [4]. Its implementation has been at least as straightforward and efficient as its predecessors.

In the past many Plausible Logics have been defined with a non-constructive disjunction, and then the proof of relative consistency has been attempted. In every case the attempt has failed. But now we can start with this constructive Plausible Logic and try to generalise the disjunction while maintaining relative consistency. Antoniou and Billington [1] showed that an ambiguity propagating version of Defeasible Logic could be embedded into Default Logic with Priorities. An attempt to embed an ambiguity propagating version of the Plausible Logic of Billington and Rock [4] into Default Logic with Priorities failed because the relative consistency of that Plausible Logic could not be proved. Now that constructive Plausible Logic is relatively consistent it is worthwhile to see if it can be embedded into Default Logic with Priorities.

## References


[1] G. Antoniou and D. Billington. Relating defeasible and default logic. In *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence*, volume 2256 of *Lecture Notes in Artificial Intelligence*, pages 13–24. Springer, 2001. 965

[2] D. Billington. Defeasible deduction with arbitrary propositions. In *Poster Proceedings of the 11th Australian Joint Conference on Artificial Intelligence*, pages 3–14, Griffith University, 1998. 954

[3] David Billington. Constructive Plausible Logic version 1.2 repository. Available from the author, 2003. 955, 960

[4] David Billington and Andrew Rock. Propositional plausible logic: Introduction and implementation. *Studia Logica*, 67:243–269, 2001. 954, 955, 962, 965

[5] Michael J. Maher, Andrew Rock, Grigoris Antoniou, David Billington, and Tristan Miller. Efficient defeasible reasoning systems. *International Journal on Artificial Intelligence Tools*, 10(4):483–501, 2001. 962

[6] D. Nute. Defeasible reasoning. In *Proceedings of the 20th Hawaii International Conference on System Science*, pages 470–477, University of Hawaii, 1987. 954

[7] Andrew Rock. *Deimos*: A query answering Defeasible Logic system. Available from the author, 2000. 962

[8] Andrew Rock. *Phobos*: A query answering Plausible Logic system. Available from the author, 2000. 962

[9] Andrew Rock. Implementation of Constructive Plausible Logic (version 1.2). Available from the author, 2003. 962