Principles and Examples of Plausible Reasoning and Propositional Plausible Logic

David Billington
D.BILLINGTON@GRIFFITH.EDU.AU

School of Information and Communication Technology, Nathan campus,
Griffith University, Brisbane, Queensland 4111, Australia.

Abstract

Plausible reasoning concerns situations whose inherent lack of precision is not quantified; that is, there are no degrees or levels of precision, and hence no use of numbers like probabilities. A hopefully comprehensive set of principles that clarifies what it means for a formal logic to do plausible reasoning is presented. A new propositional logic, called Propositional Plausible Logic (PPL), is defined and applied to some important examples. PPL is the only non-numeric non-monotonic logic we know of that satisfies all the principles and correctly reasons with all the examples. Some important results about PPL are proved.

1. Introduction

We are interested in reasoning about situations that (a) have imprecisely defined parts, and (b) this lack of precision is not quantified. That is, there are no degrees or layers or levels of precision, and in particular there are no numbers like probabilities, that would quantify the lack of precision. These situations are often indicated by the ordinary, rather than technical, use of words such as ‘mostly’, ‘usually’, ‘typically’, ‘normally’, ‘probably’, ‘likely’, ‘plausible’, ‘believable’, and ‘reasonable’. Although these words are not synonymous, they share a common property, which may be expressed by using either frequency of occurrence or weight of evidence. In frequency terms the property is that something is true more often than not; in evidence terms the property is that the evidence for something outweighs the evidence against it. An example is ‘Mammals usually are non-venomous’.

We shall call these situations plausible-reasoning situations because we shall call the reasoning used in such situations plausible reasoning.

This article has two aims. The first is to introduce principles that give a much clearer understanding of what it means for a formal logic to do plausible reasoning; that is the kind of reasoning indicated above. The hope is that this set of principles is comprehensive. Whether it is or not, this seems to be the first such set of principles even though plausible reasoning has been used for at least 2500 years (Walton, Tindale, & Gordon, 2014). However on page 114 of (Walton et al., 2014) there is a list of 11 characteristics of plausible reasoning, rather than characteristics of formal logics that do plausible reasoning.

The second aim is to define a propositional logic, called Propositional Plausible Logic (PPL), that satisfies all these principles of plausible reasoning. This shows that all the principles together are consistent; that is, there is no principle whose negation is implied by all the other principles. A pruned version of PPL is presented in (Billington, 2015).

In this paper we shall be considering only propositional situations; that is, situations that can be fully represented by a propositionally adequate language. That is, by a language
which has an adequate set of propositional connectives. The most common adequate sets of connectives contain negation and at least one of conjunction, or disjunction, or material implication. The connectives we shall use are negation $\neg$, conjunction $\land$, and disjunction $\lor$. A propositionally adequate logic is a logic based on a propositionally adequate language.

Moreover we shall consider only those plausible-reasoning situations that can be specified by a plausible-structure $S = (\text{Fact}(S), \text{Plaus}(S))$ where $\text{Fact}(S)$ is a set of propositional formulas representing the factual part of $S$, and $\text{Plaus}(S)$ is a set representing the plausible part of $S$. The elements of $\text{Plaus}(S)$ can have a variety of forms; for example: defaults are used in Reiter’s Default Logic (Reiter, 1980), defeasible rules are used in ASPIC (Caminada & Amgoud, 2007) and ASPIC$^+$ (Modgil & Prakken, 2013), and defeasible and warning rules are used in Defeasible Logic (Billington, 2008). The plausible-structure syntax is very general, while being specific enough to permit the definition of concepts needed later.

This article is organised into the following sections. The next section defines some ideas and notation from classical propositional logic that are needed in Sections 3 and 5. Section 3 presents the principles of plausible reasoning. Section 4 contains a survey of various non-numeric non-monotonic logics. The definition of PPL is in Section 5. In Section 6 we apply PPL to some examples. In Section 7 we state and discuss some important properties of PPL, and show that PPL satisfies all the principles in Section 3. Section 8 is the conclusion. All the proofs are in the appendices.

2. A Classical Propositional Logic using Resolution

Formulas in classical propositional logics are usually defined using sequences, for example $(a \lor (b \lor a))$. Sequences make unwanted distinctions which are often best removed, for example neither order nor repetitions are needed. So the above example is more clearly written as $\lor\{a,b\}$. We shall define classical propositional formulas that are based on sets rather than sequences. Such set-based formulas simplify the definition of resolution. The classical notions of truth value, valuation, satisfaction, semantic consequence $|=\phantom{0}$, tautology, contradiction, equivalence of formulas, and resolution are as usual.

Let us start by agreeing on some notation. As usual ‘iff’ abbreviates ‘if and only if’. $X$ 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 strict subset of $Y$. The empty set is denoted by $\{\}$, and the set of all integers by $\mathbb{Z}$. The cardinality of a set $S$ is denoted by $|S|$. If $m$ and $n$ are integers then we define the integer interval $[m..n]$ by $[m..n] = \{i \in \mathbb{Z} : m \leq i \leq n\}$.

Our alphabet is the union of the following pairwise disjoint sets of symbols: a non-empty countable set, $\text{Atm}$, of (propositional) atoms; the set $\{\neg, \land, \lor\}$ of connectives with $\neg$, $\land$, $\lor$ denoting negation, conjunction, and disjunction respectively; and the set of punctuation marks consisting of the comma and both braces.

We now define a formula.

**Definition 2.1.**

1) If $a$ is an atom then $a$ is a *formula*.
2) If $f$ is a formula then $\neg f$ is a *formula*.
3) If $F$ is a finite set of formulas then $\land F$ is a *formula* and $\lor F$ is a *formula*.
4) Every formula can be built by a finite number of applications of (1), (2), and (3). The set of all formulas is denoted by $\text{Fml}$.
It is convenient to write $\land F$ and $\lor F$ even though the set $F$ of formulas may be infinite. The next three definitions define some special formulas, the set of all literals in a clause or dual-clause, and the complement of a literal.

**Definition 2.2.**
1) The set, $\text{Lit}$, of all literals is defined by $\text{Lit} = \text{Atm} \cup \{\neg a : a \in \text{Atm}\}$.
2) A clause is either a literal or the disjunction, $\lor L$, of a finite set, $L$, of literals.
3) $\lor \{\} \text{ is the empty clause or falsum.}$
4) A dual-clause is either a literal or the conjunction, $\land L$, of a finite set, $L$, of literals.
5) $\land \{\} \text{ is the empty dual-clause or verum.}$
6) A formula is contingent iff it is not a tautology and it is not a contradiction.

**Definition 2.3.**
1) If $l$ is a literal then $\text{Lit}(l) = \{l\}$.
2) If $L$ is a finite set of literals then $\text{Lit}(\lor L) = L = \text{Lit}(\land L)$.

**Definition 2.4.** Let $a$ be any atom and $L$ be any set of literals.
1) The complement of $a$, $\neg a$, is defined by $\neg a = \neg a$.
2) The complement of $\neg a$, $\neg \neg a$, is defined by $\neg \neg a = a$.
3) The complement of $L$, $\neg L$, is defined by $\neg L = \{\neg l : l \in L\}$.

Let $C$ be any set of clauses. We want to remove from $C$ all the clauses that, when removed, will not change the truth value of $\land C$. By Lemma A.1(1) (See Appendix A) this means removing all tautologies and all clauses that have a strict subclause in $C$. We also want to simplify $C$ by replacing any clause $\lor \{l\}$ in $C$ by $l$. The result will be the core of $C$.

Dually, let $D$ be any set of dual-clauses. We want to remove from $D$ all the dual-clauses that, when removed, will not change the truth value of $\lor D$. By Lemma A.1(2) (See Appendix A) this means removing all contradictions and all dual-clauses that have a strict sub-dual-clause in $D$. We also want to simplify $D$ by replacing any dual-clause $\land \{l\}$ in $D$ by $l$. The result will be the core of $D$.

The following definition does both of these.

**Definition 2.5.** Let $G$ be either a set of clauses or a set of dual-clauses.
1) The set of elements of $G$ that are contingent or empty, $\text{Ctge}(G)$, is defined by $\text{Ctge}(G) = \{g \in G : g \text{ is contingent or empty}\}$.
2) The set of minimal elements of $G$, $\text{Min}(G)$, is defined by $\text{Min}(G) = \{g \in G : \text{ if } g' \in G \text{ then } \text{Lit}(g') \not\subseteq \text{Lit}(g)\}$.
3) The simplification of the formula $f$, $\text{smp}(f)$, is defined as follows.
   - If $f \in \{\land \{g\}, \lor \{g\}\}$, where $g$ is a formula, then $\text{smp}(f) = \text{smp}(g)$; else $\text{smp}(f) = f$.
4) The simplification of $G$, $\text{Smp}(G)$, is defined by $\text{Smp}(G) = \{\text{smp}(g) : g \in G\}$.
5) The core of $G$, $\text{Cor}(G)$, is defined by $\text{Cor}(G) = \text{Smp}(\text{Min}(\text{Ctge}(G)))$.

Let $C$ be any set of clauses. The set of all clauses derivable by resolution from clauses in $C$ is denoted by $\text{Res}(C)$. Also we usually abbreviate $\text{Cor}(\text{Res}(C))$ to $\text{CorRes}(C)$, and $\text{Smp}(\text{Res}(C))$ to $\text{SmpRes}(C)$.

We shall need to convert a formula $f$ into a set $\text{Claus}(f)$ of clauses, such that $\land \text{Claus}(f)$ is equivalent to $f$. Unfortunately there are many such sets of clauses, so we shall follow Sections 2 and 3 of Chapter I of (Nerode & Shore, 1997) to define which set we mean.
We shall denote the true truth value by \( \text{T} \) and the false truth value by \( \text{F} \). Let \( \text{Val} \) denote the set of all valuations, and \( \text{Atm}(f) \) denote the set of atoms in the formula \( f \).

**Definition 2.6.** If \( A \) is a set of atoms then define \( \text{Val}(A) \), the set of valuations which are false outside \( A \), by \( \text{Val}(A) = \{ v \in \text{Val} : \text{for all } a \in \text{Atm}(\neg A), v(a) = \text{F} \} \).

So if \( |A| = n \) then \( |\text{Val}(A)| = 2^n \).

**Definition 2.7.** Let \( f \) be any formula, \( F \) any set of formulas, and \( v \) any valuation.
1) Define \( L(f, v) = \{ a : a \in \text{Atm}(f) \text{ and } v(a) = \text{T} \} \cup \{ \neg a : a \in \text{Atm}(f) \text{ and } v(a) = \text{F} \} \).
2) Define \( \text{Claus}(f) = \{ \forall \neg L(f, v) : v \in \text{Val}(\text{Atm}(f)) \text{ and } v(f) = \text{F} \} \).
3) Define \( \text{Claus}(F) = \bigcup \{ \text{Claus}(f) : f \in F \} \).

If a set \( F \) of formulas is unsatisfiable then classical propositional logic can prove any formula from \( F \); that is, classical propositional logic is explosive. Explosive logics are not ideal because, for unsatisfiable sets of formulas, the idea of ‘proof’ becomes worthless. For example, let \( F_1 = \{ a, \neg a, b \} \). Then from \( F_1 \) an explosive logic can prove \( \neg b \), which does not seem sensible. Logics that are not explosive are called paraconsistent logics. But we want more than mere paraconsistency. In the example above, either \( a \) or \( \neg a \) seems to be a mistake, so it would be reasonable that from \( F_1 \) we can conclude only \( b \), and what follows from \( b \).

Let \( F \) be an unsatisfiable set of formulas. There are several ways of getting a satisfiable subset of \( F \) (see the literature on Belief Revision and Paraconsistent Logics). One way is to take the intersection of all the maximal satisfiable subsets of \( F \), (this is the full meet contraction of \( F \) by a contradiction). An equivalent way is to remove all the minimal unsatisfiable subsets of \( F \). For example, let \( F_2 = \{ a, \neg a, \forall \{ a, b \} \} \). Then the (only) minimal unsatisfiable subset of \( F_2 \) is \( \{ a, \neg a \} \). Removing this from \( F_2 \) leaves \( \{ \forall \{ a, b \} \} \). Alternatively the intersection of all the maximal satisfiable subsets of \( F_2 \) is also \( \{ \forall \{ a, b \} \} \).

However, another way to get a satisfiable subset of \( F \) is to remove all the formulas of \( F \) that are ‘contaminated’ by potential errors, as follows. First we convert \( F \) to the set of clauses \( \text{Claus}(F) \). Then \( F \) is unsatisfiable iff \( \text{Claus}(F) \) is unsatisfiable iff \( \text{Res}(\text{Claus}(F)) \) contains a literal, say \( l \), and its complement, \( \neg l \). At least one of \( l \) and \( \neg l \) is an error. Certainly both \( l \) and \( \neg l \) are potential errors. Potential errors contaminate any clause containing them, making the clause unreliable. We then remove all the contaminated clauses from \( \text{Claus}(F) \) to get the result \( \text{Sat}(\text{Claus}(F)) \).

Applying this to \( F_2 \) we see that \( \text{Claus}(F_2) = \{ \forall \{ a \}, \forall \{ \neg a \}, \forall \{ a, b \} \} \) and the set, \( \text{Err}(\text{Claus}(F_2)) \), of potential error literals of \( \text{Claus}(F_2) \) is \( \text{Err}(\text{Claus}(F_2)) = \{ a, \neg a \} \). Hence every clause in \( \text{Claus}(F_2) \) is contaminated by a potential error, and so \( \text{Sat}(\text{Claus}(F_2)) = \{ \} \).

The formal definition of the functions \( \text{Err}(.) \) and \( \text{Sat}(.) \) follows.

**Definition 2.8.** Let \( C \) be any set of clauses.
\[
\text{Err}(C) = \{ l \in \text{Lit} : \{ l, \neg l \} \subseteq \text{SmpRes}(C) \}.
\]
\[
\text{Sat}(C) = \{ c \in C : c \not\subseteq \forall \{ \} \text{ and } \text{Lit}(c) \cap \text{Err}(C) = \{ \} \}.
\]

Of course if \( C \) is satisfiable then \( \text{Sat}(C) = C \). If \( C \) is any set of clauses then \( \text{Sat}(C) \) is satisfiable, and so \( \text{Sat}(\text{Sat}(C)) = \text{Sat}(C) \).

Let \( C \) be a set of clauses. It can be shown that every literal in every clause in every minimal unsatisfiable subset of \( C \) is a potential error literal and so is in \( \text{Err}(C) \). But \( \text{Sat}(C) \)
Principles and Examples of Plausible Reasoning and Propositional Plausible Logic

removes all the clauses that contain any potential error literal, not just the clauses that are composed entirely of potential error literals. Hence the clauses in Sat(C) may be regarded as at least as reliable as the clauses in the intersection of all the maximal satisfiable subsets of C, as some of these clauses may contain potential error literals, as happens with \( F_2 \).

We can now define our paraconsistent propositional logic by mimicking the standard definition of proof by resolution, which we shall also define.

**Definition 2.9.** Let \( F \) be any set of formulas and \( f \) be any formula. As usual, define \( F \) **proves** \( f \) (by resolution), denoted by \( F \vdash f \), as follows.

\[
F \vdash f \text{ iff } \bigvee \{ \} \in \text{Res}(\text{Claus}(\{\neg f\} \cup F)).
\]

Define \( F \) **judiciously proves** \( f \), denoted by \( F \models f \), as follows.

\[
F \models f \text{ iff } \bigvee \{ \} \in \text{Res}(\text{Claus}(\neg f) \cup \text{Sat}(\text{Claus}(F))).
\]

Explosiveness is a symptom of the fact that, from a set \( F \) of formulas, classical logic proves formulas that, arguably, do not follow from \( F \). For example, if \( a \) and \( b \) are different atoms then from the contradiction \( \bigwedge \{ a, \neg a \} \) we can prove \( b \). Because \( b \) has nothing to do with \( a \) or \( \neg a \), our intuition is that \( b \) does not follow from \( \bigwedge \{ a, \neg a \} \). Although ‘follows from’ is an intuitive concept rather than a formal one, we shall attempt to formally define the concept. To refine our intuition let us consider tautologies.

Let \( F \) be a set of formulas, \( f \) and \( g \) be formulas, \( L \) and \( M \) be finite sets of literals, and \( \top \) be a tautology.

A) We could argue that tautologies stand on their own, they do not depend on any other formula. So if \( \top \notin F \) then \( \top \) does not follow from \( F \).

B) We would like ‘follows from’ to be syntax independent. That is, if \( f \) follows from \( F \) and \( f \) is equivalent to \( g \) then \( g \) follows from \( F \).

C) If \( f \in F \) then it seems reasonable that \( f \) follows from \( F \).

D) If \( \forall L \) follows from \( F \) and \( L \subseteq M \) then it seems reasonable that \( \forall M \) follows from \( F \).

E) If \( a \) and \( b \) are different atoms then it seems reasonable that \( \forall \{ a, \neg a \} \) does not follow from \( \forall \{ b, \neg b \} \). By (C), \( \forall \{ b, \neg b \} \) follows from \( \forall \{ b, \neg b \} \). But this would make ‘follows from’ syntax dependent, contrary to (B).

So tautologies present difficulties for any definition of ‘follows from’. However, in Section 3 we do not want to force all tautologies to be provable, so we shall declare that tautologies do not follow from any set of formulas. Thus we arrive at the following definition.

**Definition 2.10.** Let \( F \) be any set of formulas. Define \( \text{Taut} \) to be the set of all tautologies. Define the set of formulas that follow from \( F \), \( \text{From}(F) \), by

\[
\text{From}(F) = \{ f \in \text{Fml} : F \models f \} - \text{Taut}.
\]

Since \( \text{Sat}(\text{Claus}(F)) \subseteq \text{Claus}(F) \) we have \( \text{From}(F) \subseteq \{ f \in \text{Fml} : F \models f \} \subseteq \{ f \in \text{Fml} : F \vdash f \} \).

3. Principles of Plausible Reasoning

Lists of postulates, properties, or principles that concern special types of reasoning are useful for at least the following reasons.

1) They help **characterise** the intended special type of reasoning.
2) They provide a means of evaluating existing reasoning systems to see how well they perform the intended special type of reasoning.
3) They provide guidelines for creating new reasoning systems for the intended special type of reasoning.
4) They explicitly show a difference between the intended special type of reasoning and an existing form of reasoning.

Notable examples of such lists are the following. The AGM postulates for belief change (Alchourrón, Gärdenfors, & Makinson, 1985; Gärdenfors, 1988), various properties of non-monotonic consequence relations (Makinson, 1988; Kraus, Lehmann, & Magidor, 1990), and the postulates that a rule-based argumentation system should satisfy (Caminada & Amgoud, 2007).

We shall state the principles of this section by referring to the logic or proof algorithm directly; rather than by referring to consequence relations. A consequence relation, say $\vdash$, relates a set $F$ of formulas to a formula $f$; where $F \vdash f$ means that $f$ is a consequence of $F$. Consequence relations are appropriate if the reasoning situations under consideration can be characterised by a set of formulas. But the plausible-reasoning situations we consider are specified by a plausible-structure $S = (\text{Fact}(S), \text{Plaus}(S))$ where the elements of $\text{Plaus}(S)$ may be very different from the formulas in $\text{Fact}(S)$. For these situations consequence relations are much less appropriate. For example consider two fundamental properties that consequence relations may have; namely cut and cautious monotonicity, which together are equivalent to cumulativity, also called lemma addition. If $F$ and $G$ are sets of formulas then let $F \vdash G$ mean for all $g$ in $G$, $F \vdash g$. Then cumulativity is the following property. If $F \vdash G$ then for all formulas $h$, $F \vdash h$ iff $F \cup G \vdash h$. A straightforward translation of $F \vdash f$ into our situation is $S \vdash f$, where $S$ is a plausible-structure. But then it is really hard to know what $F \cup G$ might mean. Essentially we are trying to add proved formulas to $S$. But the only set of formulas in $S$ is $\text{Fact}(S)$. So we could try letting $F \cup G$ be $(\text{Fact}(S) \cup G, \text{Plaus}(S))$. But this is only sensible when the formulas in $G$ have been proved using only $\text{Fact}(S)$. When the formulas in $G$ have been proved using $\text{Plaus}(S)$ then it is no longer sensible to treat the formulas in $G$ as facts; because they are not facts, they are only plausible conclusions.

Some of the principles of plausible reasoning are regarded as necessary and so use the word ‘must’; the other principles are regarded as desirable and so use the word ‘should’.

As well as the principles of plausible reasoning, we shall present 4 plausible-reasoning examples that are signposts to some of the principles. Our first signpost example is a version of the lottery paradox (Kyburg, 1961) in which there are exactly 3 tickets.

**Example 3.1** (The 3-sided Die example). Consider a fair 3-sided die whose sides are labelled 1, 2, and 3. (Such a die could be a cube with sides labelled 1, 1, 2, 2, 3, 3.) For each $i$ in $\{1, 2, 3\}$, $s_i$ means that the (a) side labelled $i$ was up after a roll of the die. Then we have the following.

1) Exactly one element of $\{s_1, s_2, s_3\}$ is true.
2) Each element of $\{s_1, s_2, s_3\}$ is probably false.
3) The disjunction of any 2 elements of $\{s_1, s_2, s_3\}$ is probably true.

This example illustrates some important properties of plausible reasoning that will be considered in several of the following subsections.
The following notation will be convenient. Let $\text{Thm}(\mathcal{L}, \alpha, S)$ denote the set of all formulas derivable from the plausible-structure $S$ by using the proof algorithm $\alpha$ of the logic $\mathcal{L}$. If $F$ is a set of propositional formulas then $\text{Thm}(F)$ denotes all the formulas derivable from $F$ by (the proof algorithm of) any classical propositional logic. This simpler notation is unambiguous because $\text{Thm}(F)$ is independent of the logic (for example Hilbert systems, natural deduction, or resolution systems) and its proof algorithm.

### 3.1 Representation

Plausible-reasoning situations may contain facts as well as plausible information; for instance statement (1) of Example 3.1 is a factual statement, unlike the other two statements which are plausible. Hence the first part of our first principle of plausible reasoning.

Although the inherent lack of precision of plausible-reasoning situations is not quantified, a logic could represent this lack of precision with undue accuracy, for instance by using probabilities. Forbidding this is too restrictive, as the logic may deduce a conclusion using the probabilities but then present that conclusion without using probabilities. All we need is that the conclusions are not unduly precise. In particular if a formula is proved by using plausible information then it should not be regard as a fact. Hence the second part of our first principle of plausible reasoning.

**Principle 3.1** (The Representation Principle).

1. A logic for plausible reasoning must be able to represent, and distinguish between, factual and plausible statements.
2. The formulas proved by a logic for plausible reasoning must not be more precise than the information used to derive them.

We note that when a situation is precisely described, perhaps using probabilities, a logic for plausible reasoning should be able to reason with the corresponding imprecisely defined situation; Example 3.1 is such a situation.

We infer from Principle 3.1(1) that we should be able to distinguish between conclusions that are factual and those that are merely plausible. One way of making this distinction is to have a **factual proof algorithm** that only uses facts and deduces only facts, and also a **plausible proof algorithm** that may use plausible statements and facts and deduces formulas that are only plausible. Of course if a plausible proof algorithm deduces only facts when given just facts then it can be regarded as both a factual and a plausible proof algorithm. The need for multiple proof algorithms is discussed further in Subsection 3.8.

### 3.2 Evidence and Non-Monotonicity

Let us now see if we can establish some general guidelines concerning the provability of a given formula $f$. A plausible-reasoning situation will have evidence for and against $f$. So it seems reasonable to determine whether $f$ is provable or not by just comparing these two sets of evidence, and declaring $f$ provable iff the preponderance of evidence is for $f$.

A consequence of the evidence criterion needs the following definitions. If $S_1$ and $S_2$ are plausible-structures then $S_1 \subseteq S_2$ means $\text{Fact}(S_1) \subseteq \text{Fact}(S_2)$ and $\text{Plaus}(S_1) \subseteq \text{Plaus}(S_2)$. A proof algorithm $\alpha$ of a logic $\mathcal{L}$ is said to be **monotonic** iff for any two plausible-structures, $S_1$ and $S_2$, if $S_1 \subseteq S_2$ then $\text{Thm}(\mathcal{L}, \alpha, S_1) \subseteq \text{Thm}(\mathcal{L}, \alpha, S_2)$. For example, the proof algorithm
of a classical propositional logic is monotonic. A proof algorithm is non-monotonic iff it is not monotonic. A plausible proof algorithm is non-monotonic because the addition of evidence against a previously provable formula can cause it to be unprovable, as shown in our second signpost example.

Example 3.2 (The Non-Monotonicity example). Consider the following two statements. The first is plausible and the second is factual.

1) $a$ is probably true.
2) $\neg a$ is (definitely) true.

From (1) the conclusion is ‘$a$ is plausible’. From (1) and (2), ‘$a$ is plausible’ cannot be deduced, but ‘$\neg a$ is true’ can be.

The discussion above justifies our next principle.

Principle 3.2.

3.2.1) The Evidence Principle.
A plausible proof algorithm can prove a formula $f$ iff all the evidence for $f$ sufficiently outweighs all the evidence against $f$.

3.2.2) The Non-Monotonicity Principle.
A plausible proof algorithm must be non-monotonic.

Exactly what constitutes evidence for or against $f$ can only be determined when the particular logic for plausible reasoning is known. Also ‘sufficiently outweighs’ depends on the intuition that is being modelled, as well as the particular logic.

A proof algorithm that fails the Evidence Principle seems to be seriously flawed. So the Evidence Principle may be a principle that any sensible proof algorithm should satisfy.

3.3 Conjunction

We shall say a proof algorithm $\alpha$ of a logic $\mathcal{L}$ is conjunctive iff for any plausible-structure, $S$, and any two formulas $f$ and $g$, if $\{f, g\} \subseteq \text{Thm}(\mathcal{L}, \alpha, S)$ then $\wedge\{f, g\} \in \text{Thm}(\mathcal{L}, \alpha, S)$. For example, the proof algorithm of any classical propositional logic is conjunctive. A proof algorithm is non-conjunctive iff it is not conjunctive.

Conjunctions of plausible formulas behave very differently from conjunctions of formulas that are certain. In Example 3.1, $\Lambda\{\neg s_1, \neg s_2\}$ is equivalent to $s_3$. So although $\neg s_1$ is plausible and $\neg s_2$ is plausible, $\Lambda\{\neg s_1, \neg s_2\}$ is not plausible. Clearly plausible proof algorithms are not conjunctive.

Although the conjunction of two plausible formulas is not necessarily plausible, the conjunction of two facts is a fact. So what about the conjunction of a fact and a plausible formula? Clearly it cannot be a fact, but is it always plausible? Intuitively, a fact $f$ is always true, and a plausible formula $g$ is true more often that not. So it seems reasonable that their conjunction be true whenever $g$ is true, and hence it is reasonable that the conjunction is plausible. After we account for explosiveness and the problem of tautologies we get the following definition. We shall say a proof algorithm $\alpha$ of a logic $\mathcal{L}$ is plausibly conjunctive iff for any plausible-structure, $S$, and any two formulas $f$ and $g$, if $f \in \text{From} (\text{Fact}(S))$ and $g \in \text{Thm}(\mathcal{L}, \alpha, S)$ then $\Lambda\{f, g\} \in \text{Thm}(\mathcal{L}, \alpha, S)$. For example, the proof algorithm of any classical propositional logic is plausibly conjunctive.
**Principle 3.3** (Conjunction).

3.3.1) **The Non-Conjunction Principle.**
A plausible proof algorithm must not be conjunctive.

3.3.2) **The Plausible Conjunction Principle.**
A plausible proof algorithm should be plausibly conjunctive.

The Non-Conjunction Principle is supported by the fact that the ‘And’ rule of (Kraus et al., 1990), (If $a \sim x$ and $a \sim y$ then $a \sim \land \{x,y\}$), is not probabilistically sound, see (Makinson & Hawthorne, 2014)(Section 2.1) where they call the ‘And’ rule the ‘Right∧+’ rule. Also Definition 2.4 of (Hawthorne & Makinson, 2007) defines an ‘And’ rule that is probabilistically sound and has a similar intuition to our Plausible Conjunction Principle.

**3.4 Disjunction**

We shall say a proof algorithm $\alpha$ of a logic $\mathcal{L}$ is **disjunctive** iff for any plausible-structure, $\mathcal{S}$, and any two formulas $f$ and $g$, if $\bigvee\{f,g\} \in \text{Thm}(\mathcal{L},\alpha,\mathcal{S})$ then either $f \in \text{Thm}(\mathcal{L},\alpha,\mathcal{S})$ or $g \in \text{Thm}(\mathcal{L},\alpha,\mathcal{S})$. A proof algorithm is **non-disjunctive** iff it is not disjunctive. The proof algorithm of any classical propositional logic is non-disjunctive. The 3-sided Die example (Example 3.1) shows that, although $s_1$ and $s_2$ are both unlikely their disjunction $\bigvee\{s_1,s_2\}$ is likely. Hence our next principle is necessary.

**Principle 3.4** (The Non-Disjunction Principle).
A plausible proof algorithm must not be disjunctive.

**3.5 Supraclassicality**

Consider a plausible-structure $\mathcal{S}$. Let $\alpha$ be a plausible proof algorithm of the logic $\mathcal{L}$. Then it is tempting to suggest that $\text{Thm}(\text{Fact}(\mathcal{S})) \subseteq \text{Thm}(\mathcal{L},\alpha,\mathcal{S})$. This is called **supraclassicality**, and could be phrased as ‘what is true is usually true’.

As we saw in Section 2 after Definition 2.7, classical propositional logic is explosive and proves all tautologies. But we do not want to force logics for plausible reasoning to be explosive or to prove all tautologies.

**Definition 3.1.** A proof algorithm $\alpha$ of a logic $\mathcal{L}$ has the **plausible supraclassicality property** and is said to be **plausibly supraclassical** iff for any plausible-structure, $\mathcal{S}$, $\text{From}(\text{Fact}(\mathcal{S})) \subseteq \text{Thm}(\mathcal{L},\alpha,\mathcal{S})$.

**Principle 3.5** (The Plausible Supraclassicality Principle).
Factual and plausible proof algorithms should be plausibly supraclassical.

Since $\text{From}(\text{Fact}(\mathcal{S})) \subseteq \text{Thm}(\text{Fact}(\mathcal{S}))$, if $\alpha$ is supraclassical (that is, $\text{Thm}(\text{Fact}(\mathcal{S})) \subseteq \text{Thm}(\mathcal{L},\alpha,\mathcal{S})$) then it is plausibly supraclassical.

**3.6 Right Weakening**

Right Weakening can be thought of as closure under classical inference. More precisely, a proof algorithm $\alpha$ has the **right weakening property** iff for any plausible-structure, $\mathcal{S}$, and any formula $f$, if $f \in \text{Thm}(\mathcal{L},\alpha,\mathcal{S})$ and $f \models g$ then $g \in \text{Thm}(\mathcal{L},\alpha,\mathcal{S})$. By replacing $g$ with any tautology, we see that a consequence of the right weakening property is $\text{Taut} \subseteq$
\(\text{Thm}(\mathcal{L}, \alpha, S)\). But we do not want to force logics for plausible reasoning to prove all tautologies. We say a proof algorithm \(\alpha\) has the \textbf{weak right weakening property} iff for any plausible-structure, \(S\), and any formula \(f\), if \(f \in \text{Thm}(\mathcal{L}, \alpha, S)\) then \(\text{From}([f]) \subseteq \text{Thm}(\mathcal{L}, \alpha, S)\).

However, suppose that whenever the facts of the plausible-structure \(S\) and a formula \(f\) are true then the formula \(g\) is also true; in symbols \(\text{Fact}(S) \cup \{f\} \models g\). Then in the situation defined by \(S\), \(g\) is true at least as often as \(f\). So if \(f\) is usually true then \(g\) should also be usually true. We say a proof algorithm \(\alpha\) has the \textbf{strong right weakening property} iff for any plausible-structure, \(S\), and any formula \(f\), if \(f \in \text{Thm}(\mathcal{L}, \alpha, S)\) and \(\text{Fact}(S) \cup \{f\} \models g\) then \(g \in \text{Thm}(\mathcal{L}, \alpha, S)\).

Combining the ideas in the preceding two paragraphs produces the following definition and corresponding principle. A proof algorithm \(\alpha\) of a logic \(\mathcal{L}\) has the \textbf{plausible right weakening property} iff for any plausible-structure, \(S\), and any formula \(f\), if \(f \in \text{Thm}(\mathcal{L}, \alpha, S)\) then \(\text{From}(\text{Fact}(S) \cup \{f\}) \subseteq \text{Thm}(\mathcal{L}, \alpha, S)\).

**Principle 3.6** (The Plausible Right Weakening Principle).

A plausible proof algorithm should have the plausible right weakening property.

We note that strong right weakening implies all the other right weakening properties, and weak right weakening is implied by all the other right weakening properties.

### 3.7 Consistency

Of the 11 characteristics of plausible reasoning given on page 114 of (Walton et al., 2014), characteristic 8 is ‘stability’; which seems to mean (bottom of page 97 of (Walton et al., 2014)) that plausible statements are consistent. However, as we shall show, where consistency is concerned the number of plausible statements is important.

We say a proof algorithm \(\alpha\) of a logic \(\mathcal{L}\) is \textbf{n-consistent} iff for any plausible-structure, \(S\), and any set of formulas, \(F\), if \(\text{Fact}(S)\) is satisfiable, and \(F \subseteq \text{Thm}(\mathcal{L}, \alpha, S)\), and \(|F| \leq n\) then \(F\) is satisfiable. Also a proof algorithm \(\alpha\) of a logic \(\mathcal{L}\) is \textbf{strongly n-consistent} iff for any plausible-structure, \(S\), and any set of formulas, \(F\), if \(\text{Fact}(S)\) is satisfiable, and \(F \subseteq \text{Thm}(\mathcal{L}, \alpha, S)\), and \(|F| \leq n\) then \(\text{Fact}(S) \cup F\) is satisfiable.

So if a proof algorithm is strongly \(n\)-consistent then it is \(n\)-consistent. If \(\text{Fact}(S)\) is satisfiable then \(\text{Thm}(\text{Fact}(S))\) is satisfiable; else \(\text{Thm}(\text{Fact}(S))\) contains all formulas.

Contradictions are not plausible, so plausible proof algorithms must be 1-consistent. Hence Principle 3.7.1 below.

Suppose \(S\) is a plausible-structure such that \(\text{Fact}(S)\) is satisfiable. If \(f \in \text{Thm}(\mathcal{L}, \alpha, S)\) then in the situation defined by \(S\), \(f\) is more likely to be true than not. Hence we should expect \(\text{Fact}(S) \cup \{f\}\) to be satisfiable. That is, strong 1-consistency should hold.

Now consider strong 2-consistency. So suppose \(f\) and \(g\) are formulas such that \(\{f, g\} \subseteq \text{Thm}(\mathcal{L}, \alpha, S)\). By strong 1-consistency, both \(\text{Fact}(S) \cup \{f\}\) and \(\text{Fact}(S) \cup \{g\}\) should be satisfiable. If \(\text{Fact}(S) \cup \{f, g\}\) is unsatisfiable then \(\text{Fact}(S) \cup \{g\} \models \overline{f}\). If \(f\) and \(g\) are contingent then the strong right weakening property is reasonable, and so we should expect that \(\overline{f} \in \text{Thm}(\mathcal{L}, \alpha, S)\). Thus we have \(\{f, \overline{f}\} \subseteq \text{Thm}(\mathcal{L}, \alpha, S)\). But, a reasonable property of ‘likely’ is that for any formula \(f\), at most one of \(f\) and \(\overline{f}\) is likely. Therefore we should not have \(\{f, \overline{f}\} \subseteq \text{Thm}(\mathcal{L}, \alpha, S)\). This unsatisfactory situation can be avoided if
Fact(\mathcal{S}) \cup \{f, g\} is satisfiable. So plausible proof algorithms should be strongly 2-consistent. Hence Principle 3.7.2 below.

Consider the 3-sided Die example (Example 3.1) and let $U = \{\neg s_1, \neg s_2, v\{s_1, s_2\}\}$. For each $x$ in $U$, $x$ is likely; and $\neg x$ is not likely. But $U$ is (classically) unsatisfiable. The set $U$ shows the necessity of Principle 3.7.3 below.

**Principle 3.7 (Consistency).**

3.7.1) **The 1-Consistency Principle.**
A plausible proof algorithm must be 1-consistent.

3.7.2) **The Strong 2-Consistency Principle.**
A plausible proof algorithm should be strongly 2-consistent.

3.7.3) **The Non-3-Consistency Principle.**
A plausible proof algorithm that can prove disjunctions must not be 3-consistent.

3.8 Multiple Intuitions: Ambiguity

With the possible exception of tautologies, classical propositional logic captures our intuition about what follows from a satisfiable set of facts. But there are different well-informed intuitions about what follows from a plausible-reasoning situation. For example, as early as 1987 (Section 4.1 of (Touretzky, Horty, & Thomson, 1987)) it was recognised that a plausible-reasoning situation could elicit different sensible conclusions, depending on whether ambiguity was blocked or propagated. The essence of Figure 3 in (Touretzky et al., 1987) is our third signpost example.

**Example 3.3 (The Ambiguity Puzzle).**
1) There is evidence that $a$ is likely.
2) There is evidence that $\neg a$ is likely.
3) There is evidence that $b$ is likely.
4) If $a$ then $\neg b$ is likely.

What can be concluded about $b$? The evidence for $b$ is (3). The evidence against $b$ comes from (1) and (4). If we knew that $a$ was definitely true then the evidence for $b$ and against $b$ would be equal. Ignoring (2), $a$ is only likely by (1), so the evidence against $b$ is weaker than the evidence for $b$. But (2) means that $a$ is even less likely, and so the evidence against $b$ has been further weakened. Thus $b$ is more likely than $\neg b$. Hence many people think that it is reasonable to be able to conclude $b$. Such reasoning might be called ‘best bet’ or ‘most likely’ or ‘balance of probabilities’ reasoning.

A formula $f$ is said to be **ambiguous** iff there is evidence for $f$ and there is evidence against $f$ and neither $f$ nor $\neg f$ can be proved. Since (1) and (2) give equal evidence for and against $a$, $a$ is ambiguous.

If the evidence against $b$ has been weakened sufficiently to allow $b$ to be concluded, then $b$ is not ambiguous. So the ambiguity of $a$ has been blocked from propagating to $b$. An algorithm that can prove $b$ (but not $\neg b$) is said to be **ambiguity blocking**. This level of reasoning is appropriate if the benefit of being right outweighs the penalty for being wrong.

If the evidence against $b$ has not been weakened sufficiently to allow $b$ to be concluded, then $b$ is ambiguous. So the ambiguity of $a$ has been propagated to $b$. An algorithm that
cannot prove \( b \) (or \( \neg b \)) is said to be ambiguity propagating. This more cautious level of reasoning is appropriate if the penalty for being wrong outweighs the benefit of being right.

It is well-known that the Anglo-American legal system uses a hierarchy of proof levels, two of which are the ‘balance of probabilities’ or ‘preponderance of the evidence’ (used in civil cases) which is ambiguity blocking, and ‘beyond reasonable doubt’ (used in criminal cases) which is ambiguity propagating. So there is a need for a proof algorithm that blocks ambiguity and one that propagates ambiguity.

To avoid confusion, one should know which algorithm is used; unless it is irrelevant to the point being made. This, and our observation at the beginning of this section that a logic for plausible reasoning should have a factual proof algorithm, leads to our next principle.

**Principle 3.8 (The Many Proof Algorithms Principle).**

A logic for plausible reasoning should have at least

1) a factual proof algorithm,
2) an ambiguity blocking plausible proof algorithm, and
3) an ambiguity propagating plausible proof algorithm.

Also, the proof algorithm used to prove a formula should be explicit or irrelevant.

Clearly the algorithms in (2) and (3) must be different. But, as indicated after Principle 3.1, the factual algorithm could be the same as a plausible algorithm.

### 3.9 Decisiveness

For a formula, \( f \), a proof algorithm, \( \alpha \), will satisfy exactly one of the following conditions.

i) \( \alpha \) does not terminate.

ii) \( \alpha \) terminates in a state indicating that \( f \) is proved.

iii) \( \alpha \) terminates in a state indicating that \( f \) is not provable.

iv) \( \alpha \) terminates in some other state.

A proof algorithm \( \alpha \) is said to be decisive iff for every formula \( f \), \( \alpha \) terminates in either a state indicating that \( f \) is proved, or a state indicating that \( f \) is not provable.

Our next principle is clearly desirable.

**Principle 3.9 (The Decisiveness Principle).**

Factual and plausible proof algorithms should be decisive.

### 3.10 Truth Values

Let us change our focus from deduction to the more semantic notion of assigning truth values to statements. For classical propositional logic there are exactly two truth values: \( T \) for true and \( F \) for false. If \( v \) is a valuation (that is a function from the set of formulas to the set of truth values) and \( f \) and \( g \) are formulas then

1) Either \( v(f) = T \) or \( v(\neg f) = T \) but not both, (the Excluded Middle property) and

2) \( v(\land \{f, g\}) = T \) if and only if \( v(f) = T \) and \( v(g) = T \), and

3) \( v(\lor \{f, g\}) = T \) if and only if \( v(f) = T \) or \( v(g) = T \).

The 3-sided Die example (Example 3.1) shows that the closest plausible reasoning can get to (2) and (3) is (4) and (5) below.

4) If \( v(\land \{f, g\}) = T \) then \( v(f) = T = v(g) \).

5) If \( v(f) = T \) or \( v(g) = T \) then \( v(\lor \{f, g\}) = T \).
Moreover consider our fourth signpost example.

**Example 3.4** (The 6-sided Die example). Consider a fair (cubic) 6-sided die whose sides are labelled 1, 2, 3, 4, 5, and 6. For each \( i \) in \( \{1, 2, 3, 4, 5, 6\} \), \( s_i \) means that the side labelled \( i \) was up after a roll of the die. Then we have the following.

1) Exactly one element of \( \{s_1, s_2, s_3, s_4, s_5, s_6\} \) is true.

2) Each element of \( \{s_1, s_2, s_3, s_4, s_5, s_6\} \) is probably false.

3) The disjunction of any 2 elements of \( \{s_1, s_2, s_3, s_4, s_5, s_6\} \) is probably false.

4) The disjunction of any 3 of elements of \( \{s_1, s_2, s_3, s_4, s_5, s_6\} \) is not probably true and not probably false.

5) The disjunction of any 4 elements of \( \{s_1, s_2, s_3, s_4, s_5, s_6\} \) is probably true.

6) The disjunction of any 5 elements of \( \{s_1, s_2, s_3, s_4, s_5, s_6\} \) is probably true.

Intuitively some formulas concerning Example 3.4 have different truth values; for example \( V\{s_1, s_2, s_3, s_4, s_5, s_6\} \) is definitely true, \( \neg V\{s_1, s_2, s_3, s_4, s_5, s_6\} \) is definitely false, \( s_1 \) is probably true, \( s_1 \) is probably false, and \( V\{s_1, s_2, s_3\} \) is as likely to be true as false. So plausible reasoning appears to need at least 3 truth values:

one indicating that a formula is more likely to be true than false,

one indicating that a formula is as likely to be true as false, and

one indicating that a formula is more likely to be false than true.

Hence our last principle of plausible reasoning.

**Principle 3.10** (The Included Middle Principle).

A logic for plausible reasoning should have at least 3 truth values.

But what happens if we insist on there being exactly two truth values? Suppose a logic \( \mathcal{L} \) for plausible reasoning has exactly 2 truth values, \( T \) and \( F \). Also suppose that for any formulas \( f \), \( g \), and \( h \), the following truth conditions hold:

TC1) If \( f \) is more likely to be true than false

\[ f \text{ and } \neg f \text{ have different truth values.} \]

TC2) If \( f \) is as likely to be true as false

\[ f \text{ and } \neg f \text{ have the same truth value.} \]

TC3) The truth value of \( V\{f, g, h\} \) is \( T \) iff the truth value of at least one of \( f \), \( g \), or \( h \) is \( T \).

TC4) \( \neg V\{f, g, h\} \) and \( \land\{\neg f, \neg g, \neg h\} \) have the same truth value.

TC5) The truth value of \( \land\{f, g, h\} \) is \( T \) iff the truth value of each one of \( f \), \( g \), and \( h \) is \( T \).

Now apply \( \mathcal{L} \) to Example 3.4. By TC1, for each \( i \) in \( \{1, 2, 3\} \), \( s_i \) and \( \neg s_i \) have different truth values. By TC2, \( V\{s_1, s_2, s_3\} \) and \( \neg V\{s_1, s_2, s_3\} \) have the same truth value, which by TC4 is the same as \( \land\{-s_1, -s_2, -s_3\} \).

If the truth value of \( V\{s_1, s_2, s_3\} \) is \( T \) then by TC3, for some \( i \), the truth value of \( s_i \) is \( T \); hence the truth value of \( -s_i \) is \( F \) and so by TC5 the truth value of \( \land\{-s_1, -s_2, -s_3\} \) is \( F \). On the other hand if the truth value of \( V\{s_1, s_2, s_3\} \) is \( F \) then by TC3, for each \( i \), the truth value of \( s_i \) is \( F \); hence the truth value of each \( -s_i \) is \( T \) and so by TC5 the truth value of \( \land\{-s_1, -s_2, -s_3\} \) is \( T \).

So in both cases \( V\{s_1, s_2, s_3\} \) and \( \land\{-s_1, -s_2, -s_3\} \) have different truth values which contradicts what we had before.

The conditions TC1, TC2, TC3, TC4, and TC5 are so closely related to the meaning of ‘true’, ‘false’, ‘conjunction’, ‘disjunction’, and ‘negation’, that it is hard to reject any of them. Therefore it seems that having only two truth values is an over-simplification.
3.11 Correctness

A logic that satisfies all the previous principles could nonetheless have a fatal flaw. It could give an unsatisfactory answer to a particular example. Some examples may well have no set of answers that are generally agreed upon. But some examples do have a set of answers that are generally agreed upon. We might call these answers the correct answers. So it is tempting to state a principle of correctness similar to “When correct answers exist, a logic must give all the correct answers, and no incorrect answers.”.

The problem with such a principle is that it is impossible to show that any logic satisfies it. The most that can be done is to produce an counter-example that shows a logic fails the principle, or demonstrate that for a chosen set of examples the logic gets the correct answers. But there might exist a counter-example that shows the logic fails the principle of correctness.

Thus we shall refrain from trying to formally state a Correctness Principle.

4. Some Non-Monotonic Logics

We shall consider the relationship between some (non-numeric) non-monotonic logics and the principles and examples of Section 3.

There are three well-known non-monotonic logics, namely Default Logic, Circumscription, and Autoepistemic Logic; see (Antoniou, 1997) for an introduction. Answer Set Programming (ASP) (Baral, 2003) is a well-known Knowledge Representation system.

Each of the proof algorithms of these four well-known systems is conjunctive and so fails the Non-Conjunction Principle (Principle 3.3.1). Also for each of these four proof algorithms, the set of all provable formulas is either satisfiable or contains all formulas. So all four proof algorithms fail the Non-3-Consistency Principle (Principle 3.7.3). Hence none of these logics reasons correctly about Example 3.1. Finally all four of these proof algorithms are ambiguity propagating but not ambiguity blocking. So each of these logics fails the Many Proof Algorithms Principle (Principle 3.8). Hence when ambiguity blocking is required — for instance in civil cases — these logics do not get the right answers.

Logics that deal with only literals are incapable of the reasoning required by Example 3.1. Logics in this category include inheritance networks (Horty, Thomason, & Toretzky, 1990), the DeLP system of (Garcia & Simari, 2004), the ASPIC system mentioned in (Caminada & Amgoud, 2007), the logic in (Prakken & Sartor, 1997), Ordered logic (Geerts, Vermeir, & Nute, 1994), and most Defeasible Logics (Billington, 2008).

Propositional Plausible Logic (PPL), which is defined in the next section, is a member of the family of Defeasible Logics. The only Defeasible Logics that deal with conjunction and disjunction, besides PPL, are the logic in (Billington & Rock, 2001), let’s call it DL1, and the logic in (Billington, 2008), let’s call it DL8. But the plausible proof algorithms of both DL1 and DL8 are conjunctive and so do not satisfy the Non-Conjunction Principle (Principle 3.3.1). Also the Decisiveness Principle (Principle 3.9) fails for the plausible proof algorithms that define the Defeasible Logics in: (Billington, 1993), (Billington & Rock, 2001), (Maier & Nute, 2006), (Billington, 2008), and (Billington, 2011). Since all Defeasible Logics apart from PPL are closely related to a Defeasible Logic in these five citations, all Defeasible Logics apart from PPL fail the Decisiveness Principle. So PPL is the only Defeasible Logic that satisfies all the principles in Section 3. Also PPL is more
expressive than previous Defeasible Logics because the non-strict rules in PPL use formulas whereas previous Defeasible Logics only used literals and clauses. This is significant because a finite set of clauses is very different to the conjunction of those clauses, see the 3-sided Die example (Example 3.1).

Argumentation systems, (Dung, 1995), are well-known non-monotonic reasoning systems that can use rules, for example ASPIC (Caminada & Amgoud, 2007) and ASPIC$^+$ (Modgil & Prakken, 2013). Let $E \in \{\text{admissible, complete, preferred, grounded, ideal, semi-stable, stable}\}$. Then the semantics of ASPIC$^+$ defined by intersecting all $E$-extensions is ambiguity propagating and so fails the Many Proof Algorithms Principle (Principle 3.8(2)).

An early argumentation system is given in (Simari & Loui, 1992) and it also is ambiguity propagating and so fails the Many Proof Algorithms Principle (Principle 3.8(2)). It also has other problems mentioned in (Geerts, Laenens, & Vermier, 1998).

Three postulates that a rule-based argumentation system should satisfy are given in (Caminada & Amgoud, 2007). Postulate 1 is closure under strict rules; that is Modus Ponens for strict rules (Theorem 7.3(3)). It is a kind of right weakening property (Subsection 3.6). Postulate 2 requires the set of all proved literals to be consistent. If only literals can be proved, as in (Caminada & Amgoud, 2007), then this is implied by the Strong 2-Consistency Principle (Principle 2). Postulates 1 and 2 jointly imply Postulate 3.

It is not surprising that Conditional Logics (Nute & Cross, 2001; Arlo-Costa & Egré, 2016) have been used to analyse non-monotonic reasoning. Let $\Rightarrow$ denote a weak conditional; so that for formulas $f$ and $g$, $f \Rightarrow g$ means ‘if $f$ then $g$’ where ‘...’ could be ‘normally’, ‘typically’, ‘probably’, or any other similar word or phrase. A set of such weak conditionals is called a ‘conditional knowledge base’. The following two rules are particularly important for differentiating our plausible reasoning from other kinds of reasoning.

**And:** If $f \Rightarrow g$ and $f \Rightarrow h$ then $f \Rightarrow \land \{g, h\}$.

**Or:** If $f \Rightarrow h$ and $g \Rightarrow h$ then $\lor \{f, g\} \Rightarrow h$.

The And-rule is also called the CC-rule, and the Or-rule is also called the CA-rule.

Let $Ax3 = \land \{\lor \{s_1, s_2, s_3\}, \land \{s_1, s_2\}, \land \{s_1, s_3\}, \land \{s_2, s_3\}\}$ be the formula that characterises a fair 3-sided die, Example 3.1(1). As noted in Subsection 3.3, we have $Ax3 \Rightarrow \neg s_1$ and $Ax3 \Rightarrow \neg s_2$ but not $Ax3 \Rightarrow \land \{\neg s_1, \neg s_2\}$. So reasoning systems that satisfy the And-rule do not do plausible reasoning.

The formula that characterises a fair 6-sided die, Example 3.4(1), is the conjunction of the following 16 formulas: $\lor \{s_1, s_2, s_3, s_4, s_5, s_6\}$, $\land \{s_i, s_j\}$, where $1 \leq i < j \leq 6$; which will be denoted by $Ax6$. Let $f$ be $\land \{Ax6, \neg s_1, \neg s_2, \neg s_3\}$, let $g$ be $\land \{Ax6, \neg s_1, \neg s_2, \neg s_4\}$, and let $h$ be $\lor \{s_5, s_6\}$. Then we have $f \Rightarrow h$ and $g \Rightarrow h$ but not $\lor \{f, g\} \Rightarrow h$ because $\lor \{f, g\}$ is a tautology and $h$ is not the usual result of rolling a fair 6-sided die. So reasoning systems that satisfy the Or-rule do not do plausible reasoning.

In (Delgrande, 2007) it is observed that the following reasoning systems satisfy both the And-rule and the Or-rule and hence do not do our plausible reasoning: systems based on intuitions from probability theory such as (Adams, 1975) and (Pearl, 1988), and from qualitative possibilistic logic (Dubois, Lang, & Prade, 1994), those based on C4 (Lamarre, 1991), CT4 (Boutilier, 1994a), and S (Burgess, 1981).

Geffner and Pearl (Geffner & Pearl, 1992) define a logic called ‘conditional entailment’. The second paragraph on page 235 of (Geffner & Pearl, 1992) contains the following sentence. “In the propositional case, the only difference between conditional entailment and
prioritized circumscription is the source of the priorities: while prioritized circumscription relies on the user, conditional entailment extracts the priorities from the knowledge base itself." As noted near the beginning of this section, circumscription fails the Non-Conjunction Principle (Principle 3.3.1), the Non-3-Consistency Principle (Principle 3.7.3), and the Many Proof Algorithms Principle (Principle 3.8). Hence conditional entailment also fails these principles.

The consequence function of (Makinson, 1988) and the cumulative conditional knowledge bases of (Kraus et al., 1990) satisfy both the And-rule and the Or-rule. Preferential conditional knowledge bases (Kraus et al., 1990) are cumulative. Rational conditional knowledge bases (Lehmann & Magidor, 1992) are preferential. Hence both the preferential and rational closure of a conditional knowledge base satisfies both the And-rule and the Or-rule; and so does not do the plausible reasoning we are trying to characterise.

As noted in (Delgrande, 2007) the following systems are 'essentially the same as' rational closure and hence do not do our plausible reasoning: System Z (Pearl, 1990), systems based on conditional logic (Crocco & Lamarre, 1992), on modal logic (Boutilier, 1994b), on possibilistic logic (Benferhat, Dubois, & Prade, 1992), and on conditional objects (Dubois & Prade, 1991).

The conditional logic C of (Delgrande, 2007) does not satisfy the Plausible Right Weakening Principle (Principle 3.6). Also C and the extensions of C considered in (Delgrande, 2007) have only one proof algorithm and so fail the Many Proof Algorithms Principle (Principle 3.8).

Apart from the problems mentioned in Section 5 of (Goldszmidt & Pearl, 1991), System Z (Pearl, 1990) and System Z+ (Goldszmidt & Pearl, 1991) are ambiguity propagating but not ambiguity blocking. Hence they fail the Many Proof Algorithms Principle (Principle 3.8). Moreover, although they can represent the 3-sided Die example (Example 3.1), they cannot prove anything about the example because the set of rules is not 'consistent' as defined in (Pearl, 1990; Goldszmidt & Pearl, 1991).

The logic implemented by THEORIST (Poole, 1988) and the Preferred Subtheories logic in (Brewka, 1989) both generate consistent extensions and so fail the Non-3-Consistency Principle (Principle 3.7.3).

Every logic reviewed above fails at least one of the following principles: the Non-Conjunction Principle (Principle 3.3.1), the Non-3-Consistency Principle (Principle 3.7.3), the Many Proof Algorithms Principle (Principle 3.8), and the correctness principle as instanced by the 3-sided Die example (Example 3.1). So these principles seem to be central to the difference between the plausible reasoning characterised in Section 3 and other kinds of non-numeric non-monotonic reasoning. As far as we know, Propositional Plausible Logic is the only non-numeric non-monotonic propositionally adequate logic that satisfies all the principles in Section 3.

5. Propositional Plausible Logic (PPL)

The purpose of this section is to define a propositional logic, called Propositional Plausible Logic (PPL), that satisfies all the principles in Section 3. The plausible-structure used in PPL is defined in Subsection 5.1. The proof algorithms are defined in Subsection 5.2.
The notions of ‘proof’ and ‘truth’ are developed in Subsection 5.3 and Subsection 5.4, respectively.

As well as the notation introduced in Section 2, we shall use the following notation concerning sequences. The empty sequence is denoted by (). Let $S$ be a sequence. If $S$ is finite then $S + e$ denotes the sequence formed by just adding $e$ onto the right end of $S$. Define $e \in S$ to mean $e$ is an element of $S$, and $e \notin S$ to mean $e$ is not an element of $S$.

5.1 Plausible Descriptions

Propositional Plausible Logic (PPL) reasons about plausible-reasoning situations that may contain facts, like definitions and membership of categories. These facts are represented by formulas that are converted into clauses called axioms and these axioms are then converted into strict rules. The plausible information is represented by defeasible rules, warning rules, and a priority relation, $\gg$, on rules.

Intuitively the various kinds of rules have the following meanings. The strict rule $A \rightarrow c$ means if every formula in $A$ is true then $c$ is true. So strict rules are like material implication except that $A$ is a finite set of formulas rather than a single formula. (We have already seen that $A$ and $\{A\}$ behave differently.) For example, ‘nautiluses are cephalopods’ could be written as $\{n\} \rightarrow c$, and ‘cephalopods are molluscs’ could be written as $\{c\} \rightarrow m$.

Roughly, the defeasible rule $A \Rightarrow c$ means if every formula in $A$ is true then $c$ is usually true. For example, ‘molluscs usually have shells’ could be written as $\{m\} \Rightarrow s$, and ‘cephalopods usually have no shells’ could be written as $\{c\} \Rightarrow \neg s$.

The warning rule $A \sim c$ roughly means if every formula in $A$ is true then $c$ might be true. So $A \sim \neg c$ warns against concluding usually $c$, but does not support usually $\neg c$. For example, ‘objects that look red in red light might not be red’ could be written as $\{\text{looks-red-in-red-light}\} \sim \neg r$. Warning rules can be used to prevent unwanted chaining. For example, suppose we have ‘if $a$ then usually $b’$ ($\{a\} \Rightarrow b$) and ‘if $b$ then usually $c’$ ($\{b\} \Rightarrow c$). Then it may be too risky to conclude ‘usually $c’ from $a$. Without introducing evidence for $\neg c$, the conclusion of ‘usually $c’ from $a$ can be prevented by the warning rule $\{a\} \sim \neg c$. An instance of this example can be created by letting $a$ be $x \in \{1, 2, 3, 4\}$, $b$ be $x \in \{2, 3, 4\}$, and $c$ be $x \in \{3, 4, 5\}$. Warning rules have also been called ‘defeaters’ and ‘interfering rules’. The formal definition of a rule and its associated terms follows.

Definition 5.1. A rule, $r$, is any triple $(A(r), \text{arrow}(r), c(r))$ such that $A(r)$, called the set of antecedents of $r$, is a finite (possibly empty) set of formulas; $\text{arrow}(r) \in \{\rightarrow, \Rightarrow, \sim\}$; and $c(r)$, called the consequent of $r$, depends on $\text{arrow}(r)$.

If $\text{arrow}(r)$ is the strict arrow, $\rightarrow$, then $c(r)$ is either a formula or the conjunction of a countable set of formulas, and $r$ is written $A(r) \rightarrow c(r)$ and called a strict rule.

If $\text{arrow}(r)$ is the defeasible arrow, $\Rightarrow$, then $c(r)$ is a formula, and $r$ is written $A(r) \Rightarrow c(r)$ and called a defeasible rule.

If $\text{arrow}(r)$ is the warning arrow, $\sim$, then $c(r)$ is a formula, and $r$ is written $A(r) \sim c(r)$ and called a warning rule.

A priority relation, $\gg$, on rules is used to indicate the more relevant of two rules. For instance, the specific rule ‘cephalopods usually have no shells’, $\{(c) \Rightarrow \neg s\}$, is more relevant than the general rule ‘molluscs usually have shells’, $\{(m) \Rightarrow s\}$, when reasoning about the
external appearance of cephalopods. Hence \( \{c\} \Rightarrow \neg s > \{m\} \Rightarrow s \). More generally, some common policies for defining \( > \) are the following. Prefer specific rules over general rules; prefer authoritative rules, (for instance national laws override state laws); prefer recent rules (because they are more up-to-date); and prefer more reliable rules. If \( r \) and \( s \) are rules and \( r > s \) then we often say \( r \) is superior to \( s \) and \( s \) is inferior to \( r \).

Although the priority relation does not have to be transitive, it does have to be acyclic.

**Definition 5.2.** Let \( R \) be any set of rules. A binary relation, \( > \), on \( R \) is cyclic iff there exists a finite sequence, \( (r_1, r_2, ..., r_n) \) where \( n \geq 1 \), of elements of \( R \) such that \( r_1 > r_2 > ... > r_n > r_1 \); that is, \( r_n > r_1 \) and for all \( i \) in \( [1 .. n-1] \), \( r_i > r_{i+1} \). A binary relation, \( > \), is acyclic iff it is not cyclic.

Let us now consider the conversion of the facts of a plausible-reasoning situation represented by a set \( F \) of formulas into strict rules. First we form \( Claus(F) \) which is the set of clauses formed from \( F \). Next we generate the set of axioms, \( Ax \) by defining \( Ax = CorRes(Sat(Claus(F))) \). Finally we convert a contingent clause with \( n \) literals into \( 2^n - 1 \) strict rules. The conversion is done by the function \( Rul(\cdot) \) in the usual way as shown by the following example. \( Rul(\forall \{a, b, c\}) = \{ \{\} \rightarrow \forall \{a, b, c\} \), \( \{\land \{\neg b, \neg c\}\} \rightarrow a \), \( \{\land \{\neg a, \neg c\}\} \rightarrow b \), \( \{\land \{\neg a, \neg b\}\} \rightarrow c \), \( \{\neg a\} \rightarrow \forall \{b, c\} \), \( \{\neg b\} \rightarrow \forall \{c, a\} \), \( \{\neg c\} \rightarrow \forall \{a, b\} \).

The full definition of \( Rul(\cdot) \) and \( Rul(\cdot, \cdot) \), as well as some useful notation, is given in the next definition.

**Definition 5.3.** Let \( R \) be a set of rules, \( F \) be a finite set of formulas, and \( C \) be a set of contingent clauses.

1) \( R_s \) is the set of strict rules in \( R \). That is, \( R_s = \{ r \in R : r \) is a strict rule\}.
2) \( R_d \) is the set of defeasible rules in \( R \). That is, \( R_d = \{ r \in R : r \) is a defeasible rule\}.
3) \( c(R) \) is the set of consequents of the rules in \( R \). That is, \( c(R) = \{ c(r) : r \in R \} \).
4) If \( c \in C \) then \( Rul(c) = \{ \{\} \rightarrow c \} \cup \{ \{\land \sim (L - K)\} \rightarrow \forall K : c = \forall L \text{ and } \{\} \subset K \subset L \} \).
5) \( Rul(C) = \{ Rul(c) : c \in C \} \).
6) \( Rul(C, F) \) is the set of rules in \( Rul(C) \) whose set of antecedents is \( F \). That is, \( Rul(C, F) = \{ r \in Rul(C) : A(r) = F \} \).

We note that the set of antecedents of any strict rule formed by \( Rul(\cdot) \) has at most one element.

Although \( Rul(Ax) \) gives us the strict rules that characterise the set \( F \) of facts we started with, we can reduce the number of these strict rules by ‘anding’ all those that have the same antecedent. For example, the ‘anding’ of \( \{a\} \rightarrow c_1 \), \( \{a\} \rightarrow c_2 \), and \( \{a\} \rightarrow c_3 \) is \( \{a\} \rightarrow \land \{c_1, c_2, c_3\} \). We now have the set of strict rules that we want. This set is formally defined by PD2 below. The formal structure used for describing plausible-reasoning situations is called a plausible description and is defined below.

**Definition 5.4.** If \( R \) is a set of rules then \((R, >)\) is a plausible description iff PD1, PD2, PD3, and PD4 all hold.

PD1) There is a set \( F \) of formulas such that \( Ax(R) = CorRes(Sat(Claus(F))) \).

Ax(R) is called the set of axioms of \( R \) and is usually denoted by Ax.

PD2) \( R_s = \{ A \rightarrow smp(\land c(Rul(Ax, A))) : A \in \{ A(r) : r \in Rul(Ax) \} \} \).
PD3) If $Ax \neq \emptyset$ then $\tau$ denotes the strict rule $\emptyset \rightarrow \land Ax$.

PD4) $> \text{ is a priority relation on } R$; that is, $> \subseteq R \times (R - \{\tau\})$ and $>$ is not cyclic.

Suppose $(R, >)$ is a plausible description. Then $Ax$ is empty iff $R_s$ is empty. If $Ax \neq \emptyset$ then $\tau \in R_s$. If $R_s$ is not empty we can extract $Ax$ from the consequent of $\tau$. This shows that $Ax(R)$ is indeed dependent on $R$. Different strict rules in $R$ have different sets of antecedents, and no rule is superior ($>$) to $\tau$.

For PPL the plausible-structure is a plausible description $(R, >)$ and the factual part is $Ax(R)$, which by PD2 is equivalent to the strict rules in $R, R_s$. The plausible part consists of the non-strict rules in $R$ and the priority relation $>$. By Lemma A.2(6) (See Appendix A) and PD1, $Ax$ is satisfiable. So in PPL we extend the meaning of ‘fact’ from just being an element of $Ax$ to a formula that is implied by $Ax$. Explicitly, a formula $f$ is said to be a fact iff $Ax \models f$.

5.2 The Proof Relation and the Proof Algorithms

In this subsection we define what it means for a formula to be proved from a plausible description. We shall do this by defining a proof relation, $\models$, and various proof algorithms. This complex task will be done by giving the overall strategy, and then progressively refining this general plan until all the terms used have been defined.

Any method of demonstrating that $Ax \models f$ will do as an algorithm for proving facts; so there is no need to specify a particular one. Let our top level general plan for proving a formula be the following.

Distinguish between proving facts and proving formulas that are not facts.

Lower case Greek letters will be used to denote the proof algorithms that will eventually be defined. A general proof algorithm will be denoted by $\alpha$ (a for alpha and algorithm). We shall use $\varphi$ (f for phi and fact) to denote our factual proof algorithm. Until a further refinement is needed we shall use the notation $\alpha \models f$ to denote that a formula $f$ is proved by the proof algorithm $\alpha$.

Since facts are always true they are (at least) probably true. So we shall decree that facts are provable by all proof algorithms. Thus we have the following.

All algorithms prove all facts. In symbols, if $Ax \models f$ then $\alpha \models f$. The factual algorithm proves a formula iff it is a fact. In symbols, $\varphi \models f$ iff $Ax \models f$.

Now consider formulas $f$ that are not facts, that is, $Ax \not\models f$. To (plausibly) prove $f$ we need to do two things. First, establish some evidence for $f$. Second, defeat all the evidence against $f$. This will satisfy the requirements of the Evidence Principle, Principle 3.2.1. So our first refinement of the general plan is the following.

Refinement 5.1. Suppose $(R, >)$ is a plausible description, $Ax = Ax(R)$, and $f$ is a formula.

1) If $Ax \models f$ then $\alpha \models f$. Also $\varphi \models f$ iff $Ax \models f$.
2) If $Ax \not\models f$ and $\alpha \neq \varphi$ then $\alpha \models f$ iff (2.1) and (2.2) hold.

2.1) Establish some evidence for $f$.
2.2) Defeat all the evidence against $f$.

In accordance with the intuitive meaning of the three kinds of rules given at the beginning of this subsection, the evidence for $f$ consists of strict or defeasible rules that have a
consequent that implies \( f \). However since the axioms are always true, we can weaken this to requiring that \( Ax \cup \{ c(r) \} \) implies \( f \), provided that \( Ax \cup \{ c(r) \} \) is satisfiable. So if \( R' \subseteq R \) it will be convenient to let

\[
R'[f] = \{ r \in R' : Ax \cup \{ c(r) \} \text{ is satisfiable and } Ax \cup \{ c(r) \} \models f \}.
\]

In the case of Refinement 5.1(2.1) we have \( Ax \not\models f \) and so \( r \) cannot support \( f \). Hence the following notation is convenient.

\[
R_d^s = (R_s \cup R_d) - \{ r \}.
\]

So the evidence for \( f \) is all the rules in \( R \) that support \( f \), that is \( R_d^s[f] \). To establish some evidence for \( f \) we must prove the set of antecedents of a rule supporting \( f \). So we need to find a rule \( r \) in \( R_d^s[f] \) and prove \( A(r) \).

But \( A(r) \) is a set of formulas, not a formula. By proving a finite set \( F \) of formulas we shall mean proving every formula in \( F \). In symbols, \( \alpha \models F \) iff \( \forall f \in F, \alpha \models f \). So if \( F \) is empty we have \( \alpha \models \{ \} \).

Collecting these ideas together gives our next refinement.

**Refinement 5.2.** Suppose \((R, >)\) is a plausible description, \( Ax = Ax(R) \), and \( f \) is a formula.

1) If \( F \) is a finite set of formulas then \( \alpha \models F \) iff \( \forall f \in F, \alpha \models f \).
2) If \( Ax \models f \) then \( \alpha \models \neg f \). Also \( \varphi \models f \) iff \( Ax \models f \).
3) If \( Ax \not\models f \) and \( \alpha \not\models \varphi \) then \( \alpha \models \neg f \) iff \( \exists r \in R_d^s[f] \) such that (3.1) and (3.2) hold.
   3.1) \( \alpha \models A(r) \).
   3.2) Defeat all the evidence against \( f \).

Each rule whose consequent implies \( \neg f \) is evidence against \( f \). The set of such rules is \( R[\neg f] \). In Refinement 5.2(3) we have a rule \( r \) in \( R_d^s[f] \). So any rule in \( R[\neg f] \) that is inferior to \( r \) has already been defeated by \( r \) and hence need not be explicitly considered. This reduces the set of evidence against \( f \) that must be considered to the set of rules in \( R[\neg f] \) that are not inferior to \( r \); in symbols, \( \{ s \in R[\neg f] : s \not< r \} \).

A rule \( s \) in \( \{ s \in R[\neg f] : s \not< r \} \) is defeated either by team defeat or by disabling \( s \). The team of rules for \( f \) is \( R_d^s[f] \). The rule \( s \) is defeated by team defeat iff there is a rule \( t \) in the team of rules for \( f \), \( R_d^s[f] \), such that \( t \) is superior to \( s \), \( t > s \), and the set of antecedents of \( t \), \( A(t) \), is proved \( \alpha \models A(t) \). So if \( R' \subseteq R \) it will be convenient to let \( R'[f; s] \) denote the set of all rules in \( R'[f] \) that are superior to \( s \). In symbols,

\[
R'[f; s] = \{ t \in R'[f] : t > s \}.
\]

Alternatively \( s \) is disabled iff the set of antecedents of \( s \), \( A(s) \), cannot be proved, \( \alpha \not\models A(s) \).

Three notations for useful sets of rules have been introduced, so their formal definition is appropriate before our third refinement.

**Definition 5.5.** Suppose \((R, >)\) is a plausible description, \( Ax = Ax(R) \), \( R' \subseteq R \), \( f \) is a formula, and \( s \in R \).

1) \( R_d^s = (R_s \cup R_d) - \{ r \} \).
2) \( R'[f] = \{ r \in R' : Ax \cup \{ c(r) \} \text{ is satisfiable and } Ax \cup \{ c(r) \} \models f \} \).
3) \( R'[f; s] = \{ t \in R'[f] : t > s \} \).

**Refinement 5.3.** Suppose \((R, >)\) is a plausible description, \( Ax = Ax(R) \), and \( f \) is a formula.

1) If \( F \) is a finite set of formulas then \( \alpha \models F \) iff \( \forall f \in F, \alpha \models f \).
2) If $Ax \models f$ then $\alpha \models f$. Also $\varphi \models f$ iff $Ax \models f$.

3) If $Ax \not\models f$ and $\alpha \neq \varphi$ then $\alpha \models f$ iff $\exists r \in R^s_d[f]$ such that (3.1) and (3.2) hold.

3.1) $\alpha \models A(r)$.

3.2) $\forall s \in \{s \in R[-f] : s \not\prec r\}$ either

3.2.1) $\exists t \in R^s_d[f; s]$ such that $\alpha \models A(t)$; or

3.2.2) $\alpha \not\models A(s)$.

Refinement 5.3 has no undefined terms, but unfortunately it has two failings. There is only one plausible proof algorithm (denoted by $\alpha$), and so the Many Proof Algorithms Principle, Principle 3.8, fails. Also, a proof may get into a loop, and hence the Decisiveness Principle, Principle 3.9, fails.

Before we consider looping, let us invent the other proof algorithms. The $\alpha$ in Refinement 5.3(3.2.2) evaluates evidence against $f$; and this need not be the same $\alpha$ as in (3.1) and (3.2.1) which evaluates evidence for $f$. To avoid confusion let us call the $\alpha$ in (3.2.2), $\alpha'$. Replacing $\alpha$ by $\alpha'$ in (3.2.2) creates the need to decide what $(\alpha')'$ is. Let us simplify $(\alpha')'$ to $\alpha''$. Some obvious choices are: $\alpha'' = \alpha$, or $\alpha'' = \alpha'$, or $\alpha''$ is some other proof algorithm. The third choice postpones and complicates the choice that must eventually be made. Experimentation shows that the second choice has some properties that we would rather avoid. So we let $\alpha'' = \alpha$.

Another change that can be made is to the set $\{s \in R[-f] : s \not\prec r\}$ of rules that a proof algorithm regards as evidence against $f$. Let $Foe(\alpha, f, r)$ denote the set of rules that $\alpha$ regards as the evidence against $f$ that is not inferior to $r$.

These ideas gives us our fourth refinement.

**Refinement 5.4.** Suppose $(R, >)$ is a plausible description, $Ax = Ax(R)$, and $f$ is a formula.

1) If $F$ is a finite set of formulas then $\alpha \models F$ iff $\forall f \in F$, $\alpha \models f$.

2) If $Ax \models f$ then $\alpha \models f$. Also $\varphi \models f$ iff $Ax \models f$.

3) If $Ax \not\models f$ and $\alpha \neq \varphi$ then $\alpha \models f$ iff $\exists r \in R^s_d[f]$ such that (3.1) and (3.2) hold.

3.1) $\alpha \models A(r)$.

3.2) $\forall s \in Foe(\alpha, f, r)$ either

3.2.1) $\exists t \in R^s_d[f; s]$ such that $\alpha \models A(t)$; or

3.2.2) $\alpha' \not\models A(s)$.

Let us create our first non-factual proof algorithm, $\beta$, by changing Refinement 5.4 as little as possible. So let $Foe(\beta, f, r) = \{s \in R[-f] : s \not\prec r\}$. Let $\beta$ be defined by replacing each $\alpha$ in Refinement 5.4 with $\beta$. Of course now $\beta'$ must be defined. First let $Foe(\beta', f, r) = \{s \in R[-f] : s \not\prec r\}$. Then let $\beta''$ be defined by replacing each $\alpha$ in Refinement 5.4 with $\beta'$. (Recall that $\beta'' = \beta$.) Later we shall show that $\beta$ is ambiguity blocking ($b$ for beta and blocking). We are not really concerned with any primed algorithm as they only assist with the definition of their non-primed co-algorithm. But later we shall show that $\beta$ and $\beta'$ prove exactly the same formulas. So why is $\beta'$ needed? Without $\beta'$ it is exceedingly difficult to prove the relationship between $\beta$ and the other algorithms we are about to define.

Our next algorithm, $\pi$, will be shown to be ambiguity propagating ($p$ for pi and propagating). We want to make $\pi$ as strong as possible; that is, $\pi$ proves $f$ if there is no evidence against $f$. This can be done by making its co-algorithm $\pi'$ as weak as possible; that is, $\pi'$
ignores all evidence against \(f\); hence \(\text{Foe}(\pi', f, r) = \{\}\). This is the only change we make to Refinement 5.4. Explicitly, let \(\text{Foe}(\pi, f, r) = \{s \in R[-f] : s \not< r\}\). Let \(\pi\) be defined by replacing each \(\alpha\) in Refinement 5.4 with \(\pi\). Let \(\pi'\) be defined by replacing each \(\alpha\) in Refinement 5.4 with \(\pi'\). (Recall that \(\pi'' = \pi\).)

Our last algorithm, \(\psi\), will also be shown to be ambiguity propagating (\(p\) for \(\psi\) and \(\pi\) propagating). We want to make \(\psi\) weaker than \(\pi\). This can be done by making its co-algorithm \(\psi'\) regard those rules that imply \(\neg f\) and are superior to \(r\) as evidence against \(f\); hence \(\text{Foe}(\psi', f, r) = \{s \in R[-f] : s > r\}\). This is the only change we make to Refinement 5.4. Explicitly, let \(\text{Foe}(\psi, f, r) = \{s \in R[-f] : s \not< r\}\). Let \(\psi\) be defined by replacing each \(\alpha\) in Refinement 5.4 with \(\psi\). Let \(\psi'\) be defined by replacing each \(\alpha\) in Refinement 5.4 with \(\psi'\). (Recall that \(\psi'' = \psi\).)

To emphasise that we are only interested in the non-primed proof algorithms we note that there are examples in which both \(\pi'\) and \(\psi'\) can prove both \(f\) and \(\neg f\). This is fine as both \(\pi'\) and \(\psi'\) only assess the evidence against \(f\), rather than try to defeasibly justify accepting \(f\), as the non-primed algorithms do.

The following two formal definitions collect together for ease reference the above notations concerning algorithms and \(\text{Foe}(., ., .)\).

**Definition 5.6.** Define the set, \(\text{Alg}\), of *names of the proof algorithms* by

\[\text{Alg} = \{\varphi, \pi, \psi, \beta, \beta', \psi', \pi'\}\]. Define \(\varphi' = \varphi\). If \(\alpha \in \{\pi, \psi, \beta\}\) then define \((\alpha')' = \alpha'' = \alpha\). If \(\alpha \in \text{Alg}\) then the *co-algorithm* of \(\alpha\) is \(\alpha'\).

**Definition 5.7.** Suppose \((R, >)\) is a plausible description, \(f\) is a formula, and \(r \in R\).

1) If \(\alpha \in \{\varphi, \psi, \psi', \beta', \pi', \pi\}\) and \(r \neq x\) then \(\text{Foe}(\alpha, f, r) = \{s \in R[-f] : s \not< r\}\).
2) If \(\alpha \in \{\varphi, \pi'\}\) or \(r = x\) then \(\text{Foe}(\alpha, f, r) = \{\}\).
3) \(\text{Foe}(\psi', f, r) = \{s \in R[-f] : s > r\} = R[-f; r]\).

Finally, let us consider looping. To prove \(f\) we use \(\alpha\) and a rule \(r\). While proving \(f\) we may have to prove other formulas. During a proof of one of these other formulas, if we choose to use \(\alpha\) and \(r\) again then we will be in a loop and so this choice should fail. To prevent such a looping choice we need to record that \(\alpha\) and \(r\) have been used previously. We shall call such a record of used algorithms and rules a history. Its formal definition follows.

**Definition 5.8.** Suppose \((R, >)\) is a plausible description and \(\alpha \in \text{Alg}\). Define \(\alpha R = \{\alpha r : r \in R\}\). Then \(H\) is an *\(\alpha\)-history* iff \(H\) is a finite sequence of elements of \(\alpha R \cup \alpha'R\) that has no repeated elements.

Unfortunately using a history complicates Refinement 5.4 because we now no longer have just an algorithm proving a formula, but an algorithm and a history proving a formula. Therefore in (1), (2), and (3), \(\alpha \vdash x\) becomes \((\alpha, H) \vdash x\). In (3.1) \(\alpha\) and \(r\) have now been used so \(H\) must be updated to \(H + \alpha r\) and hence \(\alpha \vdash A(r)\) becomes \((\alpha, H + \alpha r) \vdash A(r)\). Also in (3.2.1) \(\alpha\) and \(t\) have been used so \(H\) must be updated to \(H + \alpha t\) and hence \(\alpha \vdash A(t)\) becomes \((\alpha, H + \alpha t) \vdash A(t)\). Similarly in (3.2.2) \(\alpha'\) and \(s\) have been used and so \(H\) must be updated to \(H + \alpha' s\) and hence \(\alpha' \nvdash A(s)\) becomes \((\alpha', H + \alpha' s) \nvdash A(s)\). Finally to prevent looping we must be sure that \(\alpha r \notin H\) in (3.1), \(\alpha t \notin H\) in (3.2.1), and \(\alpha' s \notin H\) in (3.2.2).

Incorporating these changes into Refinement 5.4 gives our formal definition of the proof algorithms and proof relation \(\vdash\). The letter I is attached to these final inference conditions.
**Definition 5.9.** Suppose $\mathcal{P} = (R, >)$ is a plausible description, $Ax = Ax(R)$, $\alpha \in \text{Alg}$, $H$ is an $\alpha$-history, and $f$ is a formula. The **proof relation** for $\mathcal{P}$, $\vdash$, and the **proof algorithms** are defined by I1 to I3.

I1) If $F$ is a finite set of formulas then $(\alpha, H) \vdash F$ iff $\forall f \in F$, $(\alpha, H) \vdash f$.

I2) If $Ax \models f$ then $(\alpha, H) \vdash f$. Also $(\phi, H) \vdash f$ iff $Ax \models f$.

I3) If $Ax \not\models f$ and $\phi \not\models \psi$ then $(\alpha, H) \not\vdash f$ iff $\exists r \in R_{\phi}[f]$ such that I3.1 and I3.2 hold.

I3.1) $\phi \notin H$ and $(\alpha, H + \phi t) \vdash A(t)$.

I3.2) $\forall s \in Foe(\phi, f, r)$ either

I3.2.1) $\exists t \in R_{\phi}[f; s]$ such that $\phi \notin H$ and $(\alpha, H + \phi t) \vdash A(t)$; or

I3.2.2) $\alpha' s \notin H$ and $(\alpha', H + \alpha' s) \not\vdash A(s)$.

The following notation is useful.

**Definition 5.10.** If $\mathcal{P}$ is a plausible description, $\alpha \in \text{Alg}$, and $x$ is either a formula or a finite set of formulas then define $x$ is **$\alpha$-provable** iff $\alpha \models x$ iff $(\alpha, (x)) \vdash x$, and $\mathcal{P}(\alpha) = \{ f \in \text{Fml} : \alpha \models f \}$ to be the set of all $\alpha$-provable formulas.

**A Semantic Aside**

Subsection 5.2, and its culmination in Definition 5.9, can be given a semantic interpretation. By Theorem 7.3(3), the meaning of the strict rule $A \rightarrow f$ is that for any proof algorithm $\alpha$, if $\alpha \models A$ then $\alpha \models f$. The meaning of the defeasible rule $A \Rightarrow f$ is that for any non-factual proof algorithm $\alpha$, if $\alpha \models A$ and the evidence against $f$ is defeated then $\alpha \not\models f$. Exactly what the evidence against $f$ is and how it is defeated is given by I3.2. By I3.2 the warning rule $s = A \sim \neg f$ can only be used as evidence against $f$; and exactly how $s$ can be defeated is also given by I3.2. Thus Definition 5.9 can be seen as giving a meaning to each of the three kinds of rules.

Similarly Definition 5.9, and the explanations preceding it, can be seen as giving a meaning to each of the proof algorithms. By I2, we see that $\phi \models f$ means $Ax \models f$. Each of the non-factual proof algorithms, $\alpha$, regards $R^x_{\phi}[f]$ as the set of potential evidence for $f$; and how $\alpha$ establishes that there is actual evidence for $f$ is given by I3.1. Given some actual evidence $r$ for $f$, the set that $\alpha$ regards as evidence against $f$ is $Foe(\alpha, f, r)$. Exactly how this evidence can be defeated is given by I3.2.

**5.3 A Proof Theory**

Definition 5.9 is recursive, however it can be iterated to yield a rooted tree — defined in Definition 5.13 — that could be regarded as the structure of a proof in PPL. The nodes of this tree will have special labels called tags which we now define.

**Definition 5.11.** Suppose $(R, >)$ is a plausible description, $\alpha \in \text{Alg}$, $F$ is a finite set of formulas, $H$ is an $\alpha$-history, $f$ is a formula, $r \in R^x_{\phi}[f]$, $s \in R[\neg f]$, and $p$ is a node of a tree. The **tag**, $t(p)$, of $p$ is a triple $t(p) = (\text{Subj}(p), \text{op}(p), \text{pv}(p))$.

The **subject** of $p$, $\text{Subj}(p)$, has one of the following forms: $(\alpha, H, F)$, $-(\alpha', H, F)$, $(\alpha, H, f, r)$, or $(\alpha, H, f, r, s)$.

The **operation** of $p$, $\text{op}(p)$, is either min (for minimum), max (for maximum), or $\neg$. If
The **proof value** of a node, $pv(p)$, is either $+1$ or $-1$.

The arithmetic properties of the proof values are defined below. These are as expected, but note that $\max\{\} = -1$ and $\min\{\} = +1$.

**Definition 5.12.** Suppose $S \subseteq \{+1, -1\}$.

1) $\min S = -1$ iff $-1 \in S$. (3) $\max S = +1$ iff $+1 \in S$. (5) $-1 = +1$.

2) $\min S = +1$ iff $-1 \notin S$. (4) $\max S = -1$ iff $+1 \notin S$. (6) $-1 = -1$.

So $\min$ and $\max$ act like quantifiers when applied to a set of proof values. That is, $\min S = -1$ iff there exists $v$ in $S$ such that $v = -1$; $\max S = +1$ iff there exists $v$ in $S$ such that $v = +1$; $\min S = +1$ iff for all $v$ in $S$, $v = +1$; and $\max S = -1$ iff for all $v$ in $S$, $v = -1$.

**Definition 5.13.** Let $\mathcal{P} = (R, >)$ be a plausible description. Then $T$ is an **evaluation tree** of $\mathcal{P}$ if $T$ is a rooted tree constructed as follows. Each node, $p$, of $T$ has exactly one tag, $t(p)$. For each node $p$ of $T$ there is exactly one number, $\#_p$, in $[1..6]$ such that $p$ satisfies $T\#_p$ and $T.7$

T1) $\text{Subj}(p) = (\alpha, H, F)$, $\alpha \in \text{Alg}$, $H$ is an $\alpha$-history, and $F$ is a finite set of formulas.

\[
S(p) = \{(\alpha, H, f) : f \in F\}. \text{ Then } op(p) = \min, p \text{ has } |S(p)| \text{ children, and each element of } S(p) \text{ is the subject of exactly one child of } p. \text{ If } S(p) = \{\} \text{ then } pv(p) = +1.
\]

T2) $\text{Subj}(p) = (\alpha, H, f)$, $\alpha \in \text{Alg}$, $H$ is an $\alpha$-history, $f$ is a formula, and $Ax \models f$.

Then $p$ has no children and $t(p) = ((\alpha, H, f), \min, +1)$.

T3) $\text{Subj}(p) = (\alpha, H, f)$, $\alpha \in \text{Alg} - \{\varphi\}$, $H$ is an $\alpha$-history, $f$ is a formula, and $Ax \not\models f$.

\[
S(p) = \{(\alpha, H, f, r) : \alpha \notin H \text{ and } r \in R^s_\alpha[f]\}. \text{ Then } op(p) = \max, p \text{ has } |S(p)| \text{ children, and each element of } S(p) \text{ is the subject of exactly one child of } p. \text{ If } S(p) = \{\} \text{ then } pv(p) = -1.
\]

T4) $\text{Subj}(p) = (\alpha, H, f, r)$, $\alpha \in \text{Alg} - \{\varphi\}$, $H$ is an $\alpha$-history, $f$ is a formula, $Ax \not\models f$, $\alpha \notin H$, and $r \in R^s_\alpha[f]$. Define $S(p) = \{(\alpha, H + \alpha r, A(r))\} \cup \{(\alpha, H, f, r, s) : s \in \text{Foc}(\alpha, f, r)\}$. Then $op(p) = \min, p \text{ has } |S(p)| \text{ children, and each element of } S(p) \text{ is the subject of exactly one child of } p$. If $S(p) = \{\} \text{ then } pv(p) = -1$.

T5) $\text{Subj}(p) = (\alpha, H, f, r, s)$, $\alpha \in \text{Alg} - \{\varphi, \varphi'\}$, $H$ is an $\alpha$-history, $f$ is a formula, $Ax \not\models f$, $\alpha \notin H$, $r \in R^s_\alpha[f]$, and $s \in \text{Foc}(\alpha, f, r)$. Define $S(p) = \{(\alpha, H + \alpha t, A(t)) : \alpha \notin H \text{ and } t \in R^n_\alpha[f; s]\} \cup \{-(\alpha', H + \alpha' s, A(s)) : \alpha' s \notin H\}$. Then $op(p) = \max, p \text{ has } |S(p)| \text{ children, and each element of } S(p) \text{ is the subject of exactly one child of } p$. If $S(p) = \{\} \text{ then } pv(p) = -1$.

T6) $\text{Subj}(p) = -(\alpha', H, F)$, $\alpha \in \{\pi, \psi, \beta, \beta'\} \cup \{\psi' : \psi' > \text{ is not empty}\}$, $H$ is an $\alpha$-history, and $F$ is a finite set of formulas. Then $op(p) = -; p \text{ has exactly one child, say } p_1; \text{ and } \text{Subj}(p_1) = (\alpha', H, F)$.

T7) If $op(p) = \min$ then $pv(p) = \min\{pv(c) : c \text{ is a child of } p\}$.

If $op(p) = \max$ then $pv(p) = \max\{pv(c) : c \text{ is a child of } p\}$.

If $op(p) = -$ and $c$ is the child of $p$ then $pv(p) = -pv(c)$.

It is possible for an evaluation tree to be infinite. Although this can be prevented by insisting that the set of rules in a plausible description is finite, it is not necessary.
Definition 5.14. A plausible description \( \mathcal{P} \) is a plausible theory iff every evaluation tree of \( \mathcal{P} \) is finite. A Propositional Plausible Logic consists of a plausible theory and its proof relation.

Both the proof relation \( \models \) and evaluation trees are cumbersome to use for derivations by hand. So we shall define a proof function \( P \), which is easier to use and is a straightforward translation of the proof relation \( \models \) of Definition 5.9 into the function \( P \) such that \( (\alpha, H) \models x \iff P(\alpha, H, x) = +1 \) and \( (\alpha, H) \not\models x \iff P(\alpha, H, x) = -1 \). The auxiliary functions: For (evidence for), and \( Dftd \) (defeated), are used in the definition of \( P \).

Definition 5.15. Suppose \( \mathcal{P} = (R, >) \) is a plausible theory, \( \alpha \in \text{Alg}, H \) is an \( \alpha \)-history, and \( f \) is a formula. The proof function for \( \mathcal{P} \), \( P \), and its auxiliary functions \( \text{For} \) and \( Dftd \) are defined by P1 to P5.

P1) If \( F \) is a finite set of formulas, then \( P(\alpha, H, F) = \min \{ P(\alpha, H, f) : f \in F \} \).

P2) If \( Ax \models f \) then \( P(\alpha, H, f) = +1 \). Also \( P(\varphi, H, f) = +1 \) iff \( Ax \models f \).

P3) If \( Ax \not\models f \) and \( \alpha \neq \varphi \) then
\[
P(\alpha, H, f) = \max \{ \text{For}(\alpha, H, f, r) : \alpha r \notin H \text{ and } r \in R^s_d[f] \}.
\]

P4) If \( Ax \not\models f \) and \( \alpha \neq \varphi \) and \( \alpha r \notin H \) and \( r \in R^s_d[f] \) then
\[
\text{For}(\alpha, H, f, r) = \min \{ P(\alpha, H+\alpha r, A(r)) \cup \{ Dftd(\alpha, H, f, r, s) : s \in \text{Foe}(\alpha, f, r) \} \}.
\]

P5) If \( Ax \not\models f \) and \( \alpha \in \text{Alg} - \{ \varphi, \pi' \} \) and \( \alpha r \notin H \) and \( r \in R^s_d[f] \) and \( s \in \text{Foe}(\alpha, f, r) \) then
\[
Dftd(\alpha, H, f, r, s) = \max \{ P(\alpha, H + \alpha t, A(t)) : \alpha t \notin H \text{ and } t \in R^d_d[f ; s] \} \cup \{-P(\alpha', H + \alpha' s, A(s)) : \alpha' s \notin H \}.
\]

We end this subsection by stating the relationship between proof relations (Definition 5.9), evaluation trees (Definition 5.13), and proof functions (Definition 5.15). But before we can do this we need the following notation.

Definition 5.16. Suppose \( \mathcal{P} = (R, >) \) is a plausible theory, \( \alpha \in \text{Alg}, H \) is an \( \alpha \)-history, and \( x \) is either a formula or finite set of formulas. Let \( T[\alpha, H, x] \) denote the evaluation tree of \( \mathcal{P} \) whose root node has the subject \( (\alpha, H, x) \). Let \( T(\alpha, H, x) \) denote the proof value of the root node of \( T[\alpha, H, x] \).

Theorem 5.17 (Notational Equivalence). Suppose \( \mathcal{P} \) is a plausible theory, \( \alpha \in \text{Alg}, H \) is an \( \alpha \)-history, and \( x \) is either a formula or a finite set of formulas. Then \( P(\alpha, H, x) = +1 \) iff \( (\alpha, H) \models x \) iff \( T(\alpha, H, x) = +1 \).

The idea of ‘logical consequence’ in PPL is defined and most easily understood by considering the proof relation \( \models \) of Definition 5.9. The evaluation trees of Definition 5.13 are mainly used to prove results about PPL. Proof functions (Definition 5.15) make hand evaluations easier. So the equivalences expressed in Theorem 5.17 are essential.

5.4 A Truth Theory

Logics often have a function from the set of all formulas to a set of truth values such that (a) the truth value of a formula is related to its proof value, and (b) the truth value of a formula is related to the truth values of its parts. Subsection 3.10 deals with (b), while this subsection is concerned with (a).
Consider the possibilities that could occur when the proof algorithm $\alpha$ evaluates the evidence for and against the formula $f$. If there is sufficient evidence for both $f$ and $\neg f$ then, as far as $\alpha$ is concerned, $f$ and $\neg f$ are ambiguous, and so both should be assigned the ambiguous truth value $a$. If there is sufficient evidence for $f$ but insufficient evidence for $\neg f$ then, as far as $\alpha$ is concerned, $f$ is usually true and $\neg f$ is usually false, so $f$ should be assigned the usually true truth value $t$ and $\neg f$ should be assigned the usually false truth value $f$. If there is insufficient evidence for both $f$ and $\neg f$ then $\alpha$ does not know enough about $f$ or about $\neg f$, and so both should be assigned the undetermined truth value $u$.

Since the truth value of a formula, $f$, depends on the proof algorithm, $\alpha$, evaluating its evidence, we need a veracity (or truth) function $V$ such that $V(\alpha, f)$ is in the set of plausible truth values $\{a, t, f, u\}$.

**Definition 5.18.** Suppose $P = (R, >)$ is a plausible theory, $\alpha \in \mathrm{Alg}$, and $f$ is any formula. The truth function for $P$, $V$, from $\mathrm{Alg} \times \mathrm{Fml}$ to the set of plausible truth values $\{a, t, f, u\}$ is defined by $V_1$ to $V_4$.

$V_1) V(\alpha, f) = a$ iff $\alpha \vdash f$ and $\alpha \not\vdash \neg f$.
$V_2) V(\alpha, f) = t$ iff $\alpha \vdash f$ and $\alpha \not\vdash \neg f$.
$V_3) V(\alpha, f) = f$ iff $\alpha \not\vdash f$ and $\alpha \vdash \neg f$.
$V_4) V(\alpha, f) = u$ iff $\alpha \not\vdash f$ and $\alpha \not\vdash \neg f$.

Now that PPL is defined we need to show that it is well-behaved and satisfies all the principles in Section 3. But before we do that it is worthwhile to get a better understanding of the logic by applying it to some examples.

### 6. Examples

We shall show how PPL represents and reasons with the first three signpost examples in Section 3. To save space and effort we shall use some of the theorems in Section 7; this will also illustrate some of the utility of these theorems. In some of the following examples we shall use the following equations denoted by $\dagger$ and $\Box$.

$\dagger) P(\alpha, H, \{f\}) = P(\alpha, H, f)$, by P1.
$\Box) P(\alpha, H, \{\}) = \min\{\} = +1$, by P1.

#### 6.1 The Non-Monotonicity Example

Recall the following from Example 3.2.
1) $a$ is probably true.
2) $\neg a$ is (definitely) true.

We show that from (1) the conclusion is ‘$a$ is plausible’; and from (1) and (2), that ‘$a$ is plausible’ cannot be deduced, but ‘$\neg a$ is true’ can be.

The plausible theory $(R, >)$ which models (1) is defined as follows. The priority relation $>$ is empty, and $R = \{r_a\}$, where $r_a = \{} \Rightarrow a$. So $R_a = \{} = Ax(R) = Ax$, $R[\neg a] = \{}$. Also if $l \in \{a, \neg a\}$ and $s \in R$ then $R[l; s] = \{}$.

**Evaluation 6.1.1.** $\alpha \in \{\pi, \psi, \beta\}$ and $\alpha \vdash a$

1a) $P(\alpha, (\_), a) = \text{For}(\alpha, (\_), a, r_a)$, by P3
2a) $P(\alpha, (\alpha r_a), \{\})$, by P4, and $\text{Foe}(\alpha, a, r_a) = \{}$
3α) = +1, by □

Some evaluations can be parameterised by the proof algorithm. The range of such a parameter is given after the number of the evaluation. If an evaluation proves or disproves something then this is given after the number of the evaluation.

Evaluation 6.1.1 and Theorem 5.17(Notational Equivalence), shows that π, ψ, and β can prove α using only (1).

The plausible theory (R, >) which models the Ambiguity Puzzle (Example 3.3) is defined as follows. The priority relation > is empty, and R = {r_a, r_{na}, r_b, r_{amb}}, where r_a is {} ⇒ a, r_{na} is {} ⇒ ¬ a, and r_{amb} is {} ⇒ a. Since R_s = {r_{na}^s}, Ax(R) = Ax = {¬ a}. So by P2, if α ∈ {φ, π, ψ, β} then P(α, (¬ a) = +1.

Hence using only (1) and (2), − a is certain, and by Theorem 7.4(1)(Consistency), π, ψ, and β cannot prove a.

6.2 The Ambiguity Puzzle

We show that the π and ψ proof algorithms are ambiguity propagating and that the β proof algorithm is ambiguity blocking.

The plausible theory (R, >) which models the Ambiguity Puzzle (Example 3.3) is defined as follows. The priority relation > is empty, and R = {r_a, r_{na}, r_b, r_{amb}}, where r_a is {} ⇒ a, r_{na} is {} ⇒ ¬ a, r_b is {} ⇒ b, and r_{amb} is {} ⇒ ¬ b.

Since R_s = {}, Ax(R) = Ax = {}. So R[a] = {r_a}, R[b] = {r_b}, R[¬ a] = {r_{na}}, and R[¬ b] = {r_{amb}}. If l ∈ {a, − a, b, − b} and s ∈ R then R[l; s] = {}.

Evaluation 6.2.1. α ∈ {π, ψ, β}

1α) P(α, (, b) = For(α, (, b, r_b), by P3
2α) = min{P(α, (αr_b), { })}, Dftd(α, (, b, r_b, r_{amb})), by P4
3α) = Dftd(α, (, b, r_b, r_{amb})), by □
4α) = − P(α, (α’r_{amb}), a), by P5, †
5α) = − For(α’, (α’r_{amb}), a, r_a), by P3

Evaluation 6.2.2. α ∈ {π, ψ} and α /∈ b

5α) P(α, (, b) = − For(α’, (α’r_{amb}), a, r_a), by Evaluation 6.2.1
6α) = − P(α’, (α’r_{amb}, α’r_a), { }), by P4
7α) = −1, by □.

Evaluation 6.2.3. β ∈ b

5β) P(β, (, b) = − For(β’, (β’r_{amb}), a, r_a), by Evaluation 6.2.1
6β) = − min{P(β’, (β’r_{amb}, β’r_a), { })}, Dftd(β’, (β’r_{amb}), a, r_a, r_{na})), by P4
7β) = − Dftd(β’, (β’r_{amb}), a, r_a, r_{na}), by □
8β) = − − P(β’, (β’r_{amb}, β’r_{na}), { }), by P5
9β) = +1, by □.

By Evaluation 6.2.2 and Theorems 5.17(Notational Equivalence) and 7.1(Decisiveness), π and ψ cannot prove b and so they are ambiguity propagating. By Evaluation 6.2.3 and Theorem 5.17(Notational Equivalence), β proves b and so is ambiguity blocking.
6.3 The 3-sided Die Example

Recall the following from Example 3.1.
1) Exactly one element of \( \{s_1, s_2, s_3\} \) is true.
2) Each element of \( \neg s_1, \neg s_2, \neg s_3 \) is usually true.
3) The disjunction of any pair of elements of \( \{s_1, s_2, s_3\} \) is usually true.

From (2) we get \( r_{11} \) to \( r_{13} \) below. From (3) we get \( r_{14} \) to \( r_{16} \) below. From (1) we have \( \lor \{s_1, s_2, s_3\}, \neg \land \{s_1, s_2, s_3\}, \land \neg \{s_2, s_3\} \). Converting these facts to clauses gives: \( Ax = \{\lor \{s_1, s_2, s_3\}, \lor \{\neg s_1, \neg s_2, \neg s_2\}, \lor \{\neg s_1, \neg s_3\}, \lor \{s_2, s_3\}\} \).

The plausible theory \( (R, >) \) which models this situation is defined as follows. The priority relation \( > \) is empty, and \( R = \{r_1, r_2, ..., r_{16}\} \), where \( r_1: \{\} \rightarrow \land Ax \), \( r_2: \{\neg s_1\} \rightarrow \lor \{s_2, s_3\} \), \( r_3: \{\neg s_2\} \rightarrow \lor \{s_1, s_3\} \), \( r_4: \{\neg s_3\} \rightarrow \lor \{s_1, s_2\} \), \( r_11: \{\} \Rightarrow \neg s_1 \), \( r_{14}: \{\} \Rightarrow \lor \{s_1, s_2\} \), \( r_{12}: \{\} \Rightarrow \neg s_2 \), \( r_{15}: \{\} \Rightarrow \lor \{s_1, s_3\} \), \( r_{13}: \{\} \Rightarrow \neg s_3 \), \( r_{16}: \{\} \Rightarrow \lor \{s_2, s_3\} \).

Let \( U = \{\neg s_1, \neg s_2, \lor \{s_1, s_2\}\} \). If \( \alpha \in \{\pi, \psi, \beta\} \) then we show \( \alpha \) proves each element of \( U \), \( \alpha \) cannot prove the negation of each element of \( U \), and \( \alpha \) cannot prove \( \land \{\neg s_1, \neg s_2\} \).

Note \( R_d^{1}[\neg s_1] = \{r_2, r_6, r_7, r_9, r_{10}, r_{11}, r_{16}\} \), and \( R_d^{2}[s_1] = \{r_5, r_8\} = R_d^{3}[\land \{\neg s_2, \neg s_3\}] \).

Evaluation 6.3.1. \( \pi \vdash \neg s_1 \)
1) \( P(\pi, \neg s_1) = \max \{F_0(\pi, \neg s_1, r_i) : i \in \{2, 6, 7, 9, 10, 11, 16\}\} \), by P3
2) \( \{\} \rightarrow \land Ax \), \( \{\}\rightarrow \lor \{s_2, s_3\} \), \( \{\} \rightarrow \lor \{s_1, s_3\} \), \( \{\} \rightarrow \lor \{s_2, s_3\} \)
3) \( D_0(\pi, \neg s_1, r_{11}) = \min \{P(\pi, (\pi r_{11}), (\pi r_{11})), P(\pi, (\pi r_{11}), (\pi r_{11})), P(\pi, (\pi r_{11}), (\pi r_{11}))\} \), by P4
4) \( P(\pi', \pi r_5, \land \neg s_2, s_3) = \max \{P(\pi', \pi r_5, \land \neg s_1, s_3), P(\pi', \pi r_5, \land \neg s_3, s_3)\} \), by P3
5) \( = \max \{P(\pi', \pi r_5, \pi r_7, s_1) \), by P4
6) \( = \max \{\}, \) by P3
7) \( = -1 \).
8) \( P(\pi, \neg s_1, r_{11}) = -P(\pi', (\pi r_{11}), s_1) \), by (7) to (2)
9) \( P(\pi, (\pi r_{11}), s_1, r_5) \), by P3
10) \( P(\pi', (\pi r_{11}), s_1, r_5) \), by P4
11) \( = -P(\pi', (\pi r_{11}), s_1, r_5) \), by P4
12) \( = +1 \).
13) \( = P(\pi, \neg s_1) = +1 \), by (12) to (8), and (1).

Because the 3-sided Die example is symmetric in \( s_1, s_2, \) and \( s_3 \), a very similar evaluation gives \( P(\pi, \neg s_2) = +1 \) and \( P(\pi, \neg s_3) = +1 \). Hence by (12), \( P(\pi, (\neg s_3)) = +1 \).

By Theorem 5.17(Notational Equivalence), \( \pi \vdash \neg s_1, \pi \vdash \neg s_2, \) and \( \pi \vdash \neg s_3 \).

By Theorem 7.3(3)(Modus Ponens for strict rules), using \( r_4 \), we get \( \pi \vdash \lor \{s_1, s_2\} \).

Thus \( \pi \) proves each element in \( U = \{\neg s_1, \neg s_2, \lor \{s_1, s_2\}\} \).

Suppose \( \alpha \in \{\pi, \psi, \beta\} \). Then by Theorem 7.6(The proof algorithm hierarchy), \( \alpha \) proves each element of \( U \). By Theorem 7.4(1)(Consistency), the negation of each element of \( U \) cannot be proved by \( \alpha \). Hence by Theorem 7.3(2)(Right Weakening), \( \alpha \) cannot prove \( \land \{\neg s_1, \neg s_2\} \).
7. Properties of Propositional Plausible Logic (PPL)

We shall show that PPL is well-behaved and satisfies all the principles in Section 3.

**Theorem 7.1** (Decisiveness). Suppose \( P \) is a plausible theory, \( \alpha \in \text{Alg} \), \( H \) is an \( \alpha \)-history, and \( x \) is either a formula or a finite set of formulas.
Then either \( T(\alpha, H, x) = +1 \) or \( T(\alpha, H, x) = -1 \), but not both.

Knowing that every evaluation will terminate is very comforting. So at the end of each evaluation either we will have a proof or we will not. If we do not have a proof then maybe there is a proof but we missed it. Fortunately decisiveness assures us that an evaluation will always terminate, and when it does we will have either a proof or a disproof — that is a demonstration that there is no proof.

**Theorem 7.2** (Plausible Conjunction). Suppose \( (R, >) \) is a plausible description, \( Ax = Ax(R) \), \( \alpha \in \text{Alg} \), \( H \) is an \( \alpha \)-history, and \( f \) and \( g \) are both formulas.
If \( Ax \models f \) and \( (\alpha, H) \models g \) then \( (\alpha, H) \models \land \{f, g\} \).

The Plausible Conjunction theorem shows that each \( \alpha \) satisfies the Plausible Conjunction Principle (Principle 3.3.2).

**Theorem 7.3** (Right Weakening). Suppose \( (R, >) \) is a plausible description, \( Ax = Ax(R) \), \( \alpha \in \text{Alg} \), \( H \) is an \( \alpha \)-history, and \( f \) and \( g \) are both formulas.
1) If \( (\alpha, H) \not| f \) and \( Ax \cup \{f\} \models g \) then \( (\alpha, H) \models g \). [Strong Right Weakening]
2) If \( (\alpha, H) \not| f \) and \( f \models g \) then \( (\alpha, H) \models g \). [Right Weakening]
3) If \( A \to g \in R_s \) and \( (\alpha, H) \models A \) then \( (\alpha, H) \not| g \). [Modus Ponens for strict rules]

Theorem 7.3(1) shows that \( \alpha \) has the strong right weakening property, and hence has all the right weakening properties mentioned in Subsection 3.6.

**Theorem 7.4** (Consistency). Suppose \( (R, >) \) is a plausible theory, \( Ax = Ax(R) \), \( \alpha \in \{\varphi, \pi, \psi, \beta, \beta'\} \), and both \( f \) and \( g \) are any formulas.
1) If \( \alpha \not| f \) and \( \alpha \not| g \) then \( Ax \cup \{f, g\} \) is satisfiable.
2) If \( (\psi, H) \not| f \) then \( (\psi', H) \not| -f \).
3) Suppose that whenever \( s \in R_d[R] \) and \( (\pi', H + \pi's) \models A(s) \) then \( R_d[f; s] = \{\} \).
   If \( (\pi, H) \not| f \) then \( (\pi', H) \not| -f \).

Part 1 of Theorem 7.4 says that PPL is strongly 2-consistent. Theorem 7.4(2) says that if there is sufficient evidence for \( \psi \) to prove \( f \) then the evidence for \( -f \) is too weak for \( \psi' \) to register. Theorem 7.4(3) gives conditions under which a similar statement can be said about \( \pi \) and \( \pi' \). In particular when either \( R_d[-f] \) or \( > \) is empty.

**Theorem 7.5** (Truth Values). Suppose \( (R, >) \) is a plausible theory, \( \alpha \in \text{Alg} \), \( F \) is a finite set of formulas, and \( f \) is a formula.
1) \( V(\alpha, \neg f) = V(\alpha, f) \).
2) \( V(\alpha, f) = t \) iff \( V(\alpha, \neg f) = f \).
3) \( V(\alpha, f) = f \) iff \( V(\alpha, \neg f) = t \).
4) \( V(\alpha, f) = a \) iff \( V(\alpha, \neg f) = a \).
5) \( V(\alpha, f) = u \) iff \( V(\alpha, \neg f) = u \).
6) If $V(\alpha, \wedge F) = t$ then for each $f$ in $F$, $V(\alpha, f) = t$.
7) If $f \in F$ and $V(\alpha, f) = t$ then $V(\alpha, \lor F) = t$.
8) If $\alpha \in \{\varphi, \pi, \psi, \beta, \beta'\}$ then $V(\alpha, f) \in \{t, f, u\}$.
9) If $V(\alpha, f) = a$ then $\alpha \not\models f$. (completeness)
10) If $V(\alpha, f) = t$ then $\alpha \models f$. (soundness)
11) If $\alpha \in \{\varphi, \pi, \psi, \beta, \beta'\}$ and $\alpha \models \neg f$ then $V(\alpha, f) = t$. (soundness)

Parts 1 to 5 of Theorem 7.5 show that negation is truth-functional with desirable properties. In Subsection 3.10 the desired relation between the truth values of a conjunction and its conjuncts is given by statement (4), and the desired relation between the truth values of a disjunction and its disjuncts is given by statement (5). Parts 6 and 7 of Theorem 7.5 show that PPL satisfies these relationships.

The primed algorithms $\beta'$, $\psi'$, and $\pi'$, assess the significance of evidence against a formula. Theorem 7.5(9) shows that the threshold of significance for $\psi'$ and $\pi'$ is so low that they can assess the evidence against both $f$ and $\neg f$ as significant. However Theorem 7.5(8) shows that the other algorithms have a 3-valued truth system. The expected completeness and soundness results are given by parts 10 and 11 of Theorem 7.5.

The final result shows the relationships between the various proof algorithms. Recall Definition 5.10 defines $P(\alpha)$ to be the set of all formulas provable from $P$ using the proof algorithm $\alpha$.

**Theorem 7.6** (The proof algorithm hierarchy). Suppose $P = (R, >)$ is a plausible theory.

1) $P(\varphi) \subseteq P(\pi) \subseteq P(\psi) \subseteq P(\beta) = P(\beta') \subseteq P(\psi') \subseteq P(\pi')$.

2) If $>$ is empty then $P(\varphi) \subseteq P(\pi) = P(\psi) \subseteq P(\beta) = P(\beta') \subseteq P(\psi') = P(\pi')$.

So $\beta'$ proves exactly the same formulas as $\beta$. Also if $>$ is empty then $\pi$ and $\psi$ prove exactly the same formulas, as do $\pi'$ and $\psi'$. The set of formulas proved by $\pi'$ is very similar to the union of all extensions of an extension based logic, like Default Logic.

The hierarchy shown in Theorem 7.6 is consistent with the intuition that ambiguity propagating proof algorithms are more cautious than ambiguity blocking algorithms. A similar hierarchy for a Defeasible Logic, also consistent with this intuition, is given in Section 5 of (Billington, Antoniou, Governatori, & Maher, 2010).

When a logic has several proof algorithms it is important to determine how they relate, because this gives a greater theoretical understanding of the logic. Theorem 7.6 shows that the proof algorithms of PPL are totally ordered according to reliability or level of confidence. Suppose that the plausible-reasoning situation gives no information concerning whether ambiguity should be blocked or propagated. If the proof algorithm hierarchy is not totally ordered then we could have two incomparable algorithms only one of which proved the formula of interest. In such circumstances it is not clear what should be concluded. By Theorem 7.6 no such dilemma can occur in PPL.

We shall now check that PPL satisfies all the principles in Section 3.

PPL has strict and defeasible rules, and so can distinguish between factual and plausible statements. Moreover PPL does not use numbers, like probabilities, that could lead to a proved formula being more precise than the information used to derive it. So the Representation Principle (Principle 3.1) is satisfied.
The correspondence between the general ‘plausible-structure’ notation of Sections 1 and 3 and the particular notation of PPL is as follows. The plausible-structure \( S \) corresponds to the plausible description \( \mathcal{P} = (R, >) \). If we let \( Ax = Ax(R) \), then \( \text{Fact}(S) \) corresponds to \( Ax \), \( \text{Thm}(\text{Fact}(S)) \) corresponds to \( \{ f : Ax \models f \} \), and \( \text{Thm}(\mathcal{L}, \alpha, S) \) corresponds to \( \mathcal{P}(\alpha) \).

For the rest of this section suppose \( \alpha \) is in \( \{ \pi, \psi, \beta \} \).

As explained in the paragraph above Refinement 5.1, the Evidence Principle (Principle 3.2.1) is satisfied. In Subsection 6.1 we showed that \( \alpha \) satisfies the Non-Monotonicity Principle (Principle 3.2.2). Hence \( \alpha \) satisfies Principle 3.2.

In Subsection 6.3 we showed that all three elements of \( U = \{ \neg s_1, \neg s_2, \vee \{ s_1, s_2 \} \} \) were \( \alpha \)-provable; but that the conjunction \( \land \{ \neg s_1, \neg s_2 \} \) was not \( \alpha \)-provable. Thus \( \alpha \) satisfies the Non-Conjunction Principle (Principle 3.3.1). Theorem 7.2 shows that \( \alpha \) satisfies the Plausible Conjunction Principle (Principle 3.3.2). By Theorem 7.4(1), both \( s_1 \) and \( s_2 \) are not \( \alpha \)-provable. Thus \( \alpha \) satisfies the Non-Disjunction Principle (Principle 3.4). Because \( U \) is not satisfiable \( \alpha \) satisfies the Non-3-Consistency Principle (Principle 3.7.3). Theorem 7.4(1) shows that \( \alpha \) satisfies the Strong 2-Consistency Principle (Principle 3.7.2) and so satisfies the 1-Consistency Principle (Principle 3.7.1).

By I2 of Definition 5.9, \( \varphi \) and \( \alpha \) are supraclassical. So by the remark after Principle 3.5, they satisfy the Plausible Supraclassicality Principle (Principle 3.5). Theorem 7.3(1) shows that \( \alpha \) has the Strong Right Weakening property. So by the remark after Principle 3.6, \( \alpha \) satisfies the Plausible Right Weakening Principle (Principle 3.6).

By I2 of Definition 5.9, \( \varphi \) is a factual proof algorithm. Subsection 6.2 shows that \( \pi \) and \( \psi \) are ambiguity propagating proof algorithms, and \( \beta \) is an ambiguity blocking proof algorithm. Also PPL makes the proof algorithm used explicit. Hence PPL satisfies the Many Proof Algorithms Principle (Principle 3.8). Theorems 7.1 and 5.17 show that \( \varphi \) and \( \alpha \) satisfy the Decisiveness Principle (Principle 3.9). The truth-value system given in Subsection 5.4 and Theorem 7.5 shows that \( \alpha \) satisfies the Included Middle Principle (Principle 3.10).

Thus PPL satisfies all the principles in Section 3.

8. Conclusion

We have tried to characterise those propositional logics that do plausible reasoning by suggesting some principles that such logics should satisfy. Four important examples of plausible reasoning are presented, and several principles are derived from these examples.

Propositional Plausible Logic (PPL) has been defined. It satisfies all the principles, and deals with negation, conjunction, and disjunction. PPL has been applied to the first three examples, and several theorems about PPL are proved in the appendices. PPL has been implemented by George Wilson under the direction of Dr. Andrew Rock, who has implemented other Defeasible Logics. As far as we know, PPL is the only non-numeric non-monotonic logic that satisfies all the principles in Section 3 and also correctly reasons with all the examples in Section 3.

Future research could make PPL significantly more useful and powerful by incorporating variables in a similar way to the programming language Prolog.

Acknowledgments
The author thanks Michael J. Maher for comments on an earlier version of this paper. The author also thanks René Hexel for helpful discussions about the contents of Section 3 Principles of Plausible Reasoning.

The Appendices

Appendix A. Proof of Theorems 7.1 and 5.17

Lemma A.1. Let $L$ and $M$ be any two sets of literals.
1) $\forall L \models \forall M$ iff either $L \subseteq M$ or $\forall M$ is a tautology.
2) $\wedge M \models \wedge L$ iff either $L \subseteq M$ or $\wedge M$ is a contradiction.

Proof

Let $L$ and $M$ be any two sets of literals.

(1) If $L \subseteq M$ or $\forall M$ is a tautology then $\forall L \models \forall M$.

Conversely suppose $\forall L \models \forall M$. If $L = \{\}$ then $L \subseteq M$. So suppose there is a literal $l$ such that $l \in L - M$. If $\forall M$ is not a tautology then there is a valuation $v$ such that $v(\forall M) = F$ and $v(l) = T$. But that contradicts $\forall L \models \forall M$, so $\forall M$ is must be a tautology.

(2) If $L \subseteq M$ or $\wedge M$ is a contradiction then $\wedge M \models \wedge L$.

Conversely suppose $\wedge M \models \wedge L$. If $L = \{\}$ then $L \subseteq M$. So suppose there is a literal $l$ such that $l \in L - M$. If $\wedge M$ is not a contradiction then there is a valuation $v$ such that $v(\wedge M) = T$ and $v(l) = F$. But that contradicts $\wedge M \models \wedge L$, so $\wedge M$ is must be a contradiction.

EndProofLemA.1

Lemma A.2. Let $C$ be a set of clauses.
1) $l \in \text{Err}(C)$ iff $\neg l \in \text{Err}(C)$.
2) $\text{Sat}(C) \subseteq C$.
3) $\lor \{\} \notin \text{Sat}(C)$.
4) $\lor \{\} \notin \text{Res}(\text{Sat}(C))$.
5) $C$ is satisfiable iff $\lor \{\} \notin \text{Res}(C)$.
6) $\text{Sat}(C)$ is satisfiable, $\text{Res}(\text{Sat}(C))$ is satisfiable, and $\text{CorRes}(\text{Sat}(C))$ is satisfiable.

Proof

Let $C$ be a set of clauses.

(1, 2, 3) These parts follow immediately from Definition 2.8.

(4) Assume $\lor \{\} \in \text{Res}(\text{Sat}(C))$. Since $\lor \{\} \notin \text{Sat}(C)$, there is a literal $l$ such that $\lor \{l\} \in \text{Res}(\text{Sat}(C))$ and $\lor \{\neg l\} \in \text{Res}(\text{Sat}(C))$. Since $\text{Sat}(C) \subseteq C$, $\text{Res}(\text{Sat}(C)) \subseteq \text{Res}(C)$. So $\lor \{l\} \in \text{Res}(C) \text{ and } \lor \{\neg l\} \in \text{Res}(C)$. Hence $l \in \text{Err}(C)$ and $\neg l \in \text{Err}(C)$. So by Definition 2.8, for all $c$ in $\text{Sat}(C)$, $l \notin \text{Lit}(c)$. But $\lor \{l\} \in \text{Res}(\text{Sat}(C))$, so there exists $c$ in $\text{Sat}(C)$ such that $l \in \text{Lit}(c)$. This contradiction shows that $\lor \{\} \notin \text{Res}(\text{Sat}(C))$.

(5) This is well known from classical propositional logic.

(6) By parts (4) and (5) of this lemma, $\text{Sat}(C)$ is satisfiable. Hence $\text{Res}(\text{Sat}(C))$ is satisfiable and so $\text{CorRes}(\text{Sat}(C))$ is satisfiable.

EndProofLemA.2

We say that a set $L$ of literals is contingent iff $L$ is not empty and if $a$ is any atom then $\{a, \neg a\} \notin L$. 

32
Lemma A.3. Suppose \((R, >)\) is a plausible description, \(Ax = Ax(R), \alpha \in Alg, H\) is an \(\alpha\)-history, and \(f\) is a formula.

1) \(Ax\) is satisfiable.
2) Each axiom in \(Ax\) is either contingent or empty.
3) Each axiom in \(Ax\) is satisfiable and \(Ax\) is satisfiable.
4) If \(R[f] \neq \{\}\) then \(Ax \cup \{f\}\) is satisfiable and \(Ax \not\vdash \neg f\).
5) If \((\alpha, H) \vdash f\) then \(Ax \cup \{f\}\) is satisfiable and \(Ax \not\vdash \neg f\).

Proof

Suppose \((R, >)\) is a plausible description, \(Ax = Ax(R), \alpha \in Alg, H\) is an \(\alpha\)-history, and \(f\) is a formula.

(1) By Definition 5.4(PD1), \(Ax = CorRes(Sat(Ax))\). By Lemma A.2(6), \(CorRes(Sat(Ax))\) is satisfiable. Hence \(Ax\) is satisfiable.

(2) By the definitions, \(Ax = CorRes(Sat(Ax)) = SmpMinCtge(Res(Sat(Ax)))\); so each axiom is either contingent or empty. But \(Ax\) is satisfiable, so each axiom is contingent.

(3) This follows from part (2) and \(Ax = Smp(C)\) where \(C\) is a set of clauses.

(4) Suppose \(R[f] \neq \{\}\). By Definition 5.5(2), there exists \(r \in R\) such that \(Ax \cup \{c(r)\}\) is satisfiable and \(Ax \cup \{c(r)\} \vdash f\). Hence \(Ax \cup \{f\}\) is satisfiable, and so \(Ax \not\vdash \neg f\).

(5) Suppose \((\alpha, H) \vdash f\). By Definition 5.9(I2,I3), either \(Ax \vdash f\) or \(R^\alpha[f] \neq \{\}\). If \(Ax \vdash f\) then by part (1), \(Ax \cup \{f\}\) is satisfiable. If \(R^\alpha[f] \neq \{\}\) then by part (4), \(Ax \cup \{f\}\) is satisfiable. But \(Ax \cup \{f\}\) is satisfiable implies \(Ax \not\vdash \neg f\).

EndProofLemA.3

Lemma A.4. Suppose \((R, >)\) is a plausible description, and \(Ax = Ax(R)\).

1) If \(r \in R_s\) then either \(A(r) = \{\}\); or \(A(r) = \{l\}\), where \(l\) is a literal;
   or \(A(r) = \{\land L\}\), where \(|L| \geq 2\) and \(L\) is contingent.
2) If \(r \in R_s\) then \(Ax \cup A(r) \vdash c(r)\).

Proof

Suppose \((R, >)\) is a plausible description, and \(Ax = Ax(R)\).

(1) By Lemma A.3(2), if \(\forall L \in Ax\) then \(\forall L\) is contingent and so \(L\) is contingent. Hence, if \(\{\}\subset K \subset L\) then \(L - K\) is contingent. So \(\sim(L - K)\) is contingent. Therefore the result holds for all \(r\) in \(Rul(Ax)\). So by Definition 5.4(PD2), the result holds for all \(r\) in \(R_s\).

(2) Take any \(r\) in \(Rul(Ax)\), and suppose \(v\) is a valuation such that \(v(Ax \cup A(r))\) is the true truth value; that is, \(v \models Ax \cup A(r)\). Then either \(r\) is \(\{\}\to c\) where \(c \in Ax\), or \(r\) is \(\{\text{smp}(\land \sim(L - K))\} \to \text{smp}(VK)\), where \(\{\}\subset K \subset L\) and \(\forall L \in Ax\). Now \(v \models c\), so in the first case \(v \models c(r)\). In the second case \(v \models \forall L\), and \(v \models \land \sim(l - K)\). Then for all \(l \in L - K\), \(v \models \sim l\) and so \(v \not\models l\). But \(L = K \cup (L - K)\). So \(v \models \forall(K \cup (L - K))\) and hence \(v \models VK\).

Thus \(v \models c(r)\) in the second case too. So the lemma holds for all \(r\) in \(Rul(Ax)\).

Take any \(r_0\) in \(R_0 - Rul(Ax)\), and suppose \(v \models Ax \cup A(r_0)\). Then \(r_0\) is \(A(r_0) \to \land c(Rul(Ax, A(r_0)))\). For each \(r\) in \(Rul(Ax, A(r_0))\), \(r\) is \(A(r_0) \to c(r)\). By the previous paragraph, \(v \models c(r)\). But this is true for every \(r\) in \(Rul(Ax, A(r_0))\) and so \(v \models \land c(Rul(Ax, A(r_0)))\).

EndProofLemA.4

Lemma A.5. Suppose \((R_0, >)\) is a plausible description, \(Ax\) is its set of axioms, \(R \subseteq R_0\), 
\(f\) and \(g\) are formulas, \(\alpha \in Alg\), and \(\{r, s\} \subseteq R_0\).
1) \( R[f] \subseteq R \).
2) If \( R' \subseteq R \) then \( R'[f] \subseteq R[f] \).
3) If \( f \equiv g \) then \( R[f] = R[g] \).
4) If \( Ax \models f \) then \( R[\land \{f, g\}] = R[g] \) and \( \text{Foe}(\alpha, \land \{f, g\}, r) \subseteq \text{Foe}(\alpha, g, r) \).
5) If \( Ax \cup \{f\} \models g \) then (a) \( Ax \cup \{\neg g\} \models \neg f \), (b) \( R[f] \subseteq R[g] \), (c) \( R[f; s] \subseteq R[g; s] \),
   (d) \( R[\neg g] \subseteq R[\neg f] \), (e) \( \text{Foe}(\alpha, g, r) \subseteq \text{Foe}(\alpha, f, r) \).
6) If \( Ax \cup \{f, g\} \) is unsatisfiable then \( R[f] \subseteq R[\neg g] \) and \( R[g] \subseteq R[\neg f] \).

**Proof**

1) By Definition 5.5(2), \( R[f] \subseteq R \).
2) Suppose \( R' \subseteq R \). Then \( R'[f] = \{ r \in R' : Ax \cup \{c(r)\} \models f \} \subseteq \{ r \in R : Ax \cup \{c(r)\} \models f \} = R[f] \).
3) Suppose \( f \equiv g \). By Definition 5.5(2), \( R[f] = \{ r \in R : Ax \cup \{c(r)\} \models f \} \)
   \( = \{ r \in R : Ax \cup \{c(r)\} \models g \} \).
4) Suppose \( Ax \models f \). By Definition 5.5(2), \( R[g] = \{ r \in R : Ax \cup \{c(r)\} \models g \} \).
5) Suppose \( Ax \cup \{f\} \models g \).
   (a) Every valuation satisfies exactly one of \( f \) or \( \neg f \). Hence \( Ax \cup \{\neg g\} \models \neg f \).
   (b) Take any \( r \in R[f] \). Then \( Ax \cup \{c(r)\} \) is satisfiable and \( Ax \cup \{c(r)\} \models f \). Hence \( Ax \cup \{c(r)\} \models g \) and so \( r \in R[g] \). Thus \( R[f] \subseteq R[g] \).
   (c) Take any \( t \in R[f; s] \). Then \( t \in R[f] \) and \( t > s \). By part (b), \( t \in R[g] \) and so \( t \in R[g; s] \).
   Thus \( R[f; s] \subseteq R[g; s] \).
   (d) This follows from parts (a) and (b).
   (e) This follows from parts (a) and (c).
6) Suppose \( Ax \cup \{f, g\} \) is unsatisfiable. Take any \( r \in R[f] \). Then \( Ax \cup \{c(r)\} \)
   is satisfiable and \( Ax \cup \{c(r)\} \models f \). Hence \( Ax \cup \{c(r)\} \models g \). Therefore \( r \in R[g] \) and so \( R[f] \subseteq R[g] \). By swapping \( f \) and \( g \) we get \( R[g] \subseteq R[f] \).

**EndProofLemA.5**

We need to extend the notation introduced in Definition 5.16.
Definition A.6. Suppose $\mathcal{P} = (R, >)$ is a plausible theory, $\alpha \in \text{Alg}$, $H$ is an $\alpha$-history, $F$ is a finite set of formulas, $f$ is a formula, $r \in R^s_d[f]$, and $s \in R[\neg f]$.

1) Let $T[\alpha, H, F]$ be the evaluation tree of $\mathcal{P}$ whose root has the subject $(\alpha, H, F)$; and let $T(\alpha, H, F)$ be the proof value of the root of $T[\alpha, H, F]$.

2) Let $T[\alpha, H, f]$ be the evaluation tree of $\mathcal{P}$ whose root has the subject $(\alpha, H, f)$; and let $T(\alpha, H, f)$ be the proof value of the root of $T[\alpha, H, f]$.

3) Let $T[\alpha, H, f, r]$ be the evaluation tree of $\mathcal{P}$ whose root has the subject $(\alpha, H, f, r)$; and let $T(\alpha, H, f, r)$ be the proof value of the root of $T[\alpha, H, f, r]$.

4) Let $T[\alpha, H, f, r, s]$ be the evaluation tree of $\mathcal{P}$ whose root has the subject $(\alpha, H, f, r, s)$; and let $T(\alpha, H, f, r, s)$ be the proof value of the root of $T[\alpha, H, f, r, s]$.

5) Let $T[\neg (\alpha, H, F)]$ be the evaluation tree of $\mathcal{P}$ whose root has the subject $-(\alpha, H, F)$; and let $T(- (\alpha, H, F))$ be the proof value of the root of $T[\neg (\alpha, H, F)]$.

Theorem A.7 (Theorem 7.1 Decisiveness). Suppose $\mathcal{P}$ is a plausible theory, $\alpha \in \text{Alg}$, $H$ is an $\alpha$-history, and $x$ is either a formula or a finite set of formulas.

1) $T[\alpha, H, x]$ has finitely many nodes.

2) Either $T(\alpha, H, x) = +1$ or $T(\alpha, H, x) = -1$ but not both.

Proof

(1) follows from Definition 5.14.

(2) follows from part (1) of this lemma and Definition 5.13.

EndProofThmA.7

Theorem A.8 (Theorem 5.17 Notational Equivalence). Suppose $(R, >)$ is a plausible theory, $\alpha \in \text{Alg}$, $H$ is an $\alpha$-history, $F$ is a finite set of formulas, and $f$ is a formula.

1) $P(\alpha, H, F) = +1$ iff $(\alpha, H) \vdash F$ iff $T(\alpha, H, F) = +1$.

2) $P(\alpha, H, f) = +1$ iff $(\alpha, H) \vdash f$ iff $T(\alpha, H, f) = +1$.

3) Suppose $Ax \not\models f$ and $\alpha \not\models \varphi$ and $\alpha r \notin H$ and $r \in R^s_d[f]$ and $f$ is satisfiable.

Then $\text{For}(\alpha, H, f, r) = +1$

iff $T(\alpha, H, f, r) = +1$

iff $(\alpha, H + \alpha r) \vdash A(r)$ and $\forall s \in \text{Foe}(\alpha, f, r), D\text{ftd}(\alpha, H, f, r, s) = +1$.

4) Suppose $Ax \not\models f$ and $\alpha \in \text{Alg} - \{\varphi, \pi^1\}$ and $\alpha r \notin H$ and $r \in R^s_d[f]$ and $f$ is satisfiable and $s \in \text{Foe}(\alpha, f, r)$. Then $D\text{ftd}(\alpha, H, f, r, s) = +1$

iff $T(\alpha, H, f, r, s) = +1$

iff either $\exists t \in R^s_d[f]; s$ such that $\alpha t \notin H$ and $(\alpha, H + \alpha t) \vdash A(t)$;

or $\alpha' s \notin H$ and $(\alpha', H + \alpha' s) \not\vdash A(s)$.

Proof

Let $\mathcal{P} = (R, >)$ be a plausible theory. The proof is by induction on the number of nodes in an evaluation tree of $\mathcal{P}$. Let $p$ be the only node of an evaluation tree of $\mathcal{P}$.

If $p$ satisfies $T1$ then the subject of $p$ is $(\alpha, H, \{\})$ and the proof value of $p$ is $+1$. So $T(\alpha, H, \{\}) = +1$. By P1, $P(\alpha, H, \{\}) = \min\{\} = +1$. By I1, $(\alpha, H) \vdash \{\}$.

If $p$ satisfies $T2$ then the subject of $p$ is $(\alpha, H, f)$ and the proof value of $p$ is $+1$. So $T(\alpha, H, f) = +1$. By P2, $P(\alpha, H, f) = +1$. By I2, $(\alpha, H) \vdash f$.

If $p$ satisfies $T3$ then the subject of $p$ is $(\alpha, H, f)$ and the proof value of $p$ is $-1$. So $T(\alpha, H, f) = -1$. Let $S(p) = \{(\alpha, H, f, r) : \alpha r \notin H$ and $r \in R^s_d[f]\}$. By T3, $S(p)$ is empty. By P3, $P(\alpha, H, f) = \max\{\text{For}(\alpha, H, f, r) : (\alpha, H, f, r) \in S(p)\} = \max\{\} = -1$. Since $S(p)$ is empty, for all $r \in R^s_d[f]$, $\alpha r \notin H$. Hence I3.1 fails and so $(\alpha, H) \not\vdash f$.

35
Since $p$ has no children, $p$ does not satisfy T4 or T6.

If $p$ satisfies T5 then the subject of $p$ is $(\alpha, H, f, r, s)$ and the proof value of $p$ is $-1$. So $T(\alpha, H, f, r, s) = -1$. Let $S(p) = \{(\alpha, H+\alpha t, A(t)) : \alpha t \notin H \text{ and } t \in R^s_d[f; s]\} \cup \{-(\alpha', H+\alpha's, A(s)) : \alpha's \notin H\}$. Also let $S' = \{P(\alpha, H+\alpha t, A(t)) : \alpha t \notin H \text{ and } t \in R^s_d[f; s]\} \cup \{-P(\alpha', H+\alpha's, A(s)) : \alpha's \notin H\}.

By T5, $S(p)$ is empty, and so $S'$ is also empty. By P5, $Dftd(\alpha, H, f, r, s) = \max S' = \max \{\}\ = -1$. Since $S(p) = \{\}$, we have for each $t \in R^s_d[f; s]$, $\alpha t \in H$, and $\alpha's \in H$. Hence the last characterisation of $Dftd(\alpha, H, f, r, s) = +1$ in part (4) is false.

Thus the result holds for all evaluation trees of $P$ that have only the root node.

Take any positive integer $n$. We shall denote the following inductive hypothesis by IndHyp. Suppose the result holds for all evaluation trees of $P$ that have less than $n+1$ nodes. Let $T$ be an evaluation tree of $P$ that has $n+1$ nodes and let $p$ be the root of $T$. Then $p$ has at least one child.

If $p$ satisfies T1 then the subject of $p$ is $(\alpha, H, f)$. Let $S(p) = \{(\alpha, H, f, r) : \alpha r \notin H \text{ and } r \in R^s_d[f]\}$. By T3, IndHyp, P3, and I3, $T(\alpha, H, f) = +1$ if there exists $(\alpha, H, f, r)$ in $S(p)$ such that $T(\alpha, H, f, r) = +1$ if there exists $(\alpha, H, f, r)$ in $S(p)$ such that $\forall s \in Foe(\alpha, f, r)$, $Dftd(\alpha, H, f, r, s) = +1$ if there exists $(\alpha, H, f, r)$ in $S(p)$ such that $(\alpha, H+\alpha r) \vdash A(r)$ and $\forall s \in Foe(\alpha, f, r)$, either $\exists t \in R^s_d[f; s]$ such that $\alpha t \notin H$ and $(\alpha, H+\alpha t) \vdash A(t)$; or $\alpha's \notin H$ and $(\alpha', H+\alpha's) \nvdash A(s)$ if $\exists r \in R^s_d[f]$ such that I3.1 and I3.2 if $(\alpha, H) \vdash f$.

If $p$ satisfies T4 then the subject of $p$ is $(\alpha, H, f, r)$. By T4, IndHyp, and P4, $T(\alpha, H, f, r) = +1$ if $T(\alpha, H+\alpha r, A(r)) = +1$ and $\forall s \in Foe(\alpha, f, r)$, $T(\alpha, H, f, r, s) = +1$ if $(\alpha, H+\alpha r) \vdash A(r)$ and $\forall s \in Foe(\alpha, f, r)$, $Dftd(\alpha, H, f, r, s) = +1$ if $P(\alpha, H+\alpha r, A(r)) = +1$ and $\forall s \in Foe(\alpha, f, r)$, $Dftd(\alpha, H, f, r, s) = +1$ if $For(\alpha, H, f, r) = +1$.

If $p$ satisfies T5 then the subject of $p$ is $(\alpha, H, f, r, s)$. By T5, IndHyp, T6, T7, and P5, $T(\alpha, H, f, r, s) = +1$ if either $\exists t \in R^s_d[f; s]$ such that $\alpha t \notin H$ and $T(\alpha, H+\alpha t, A(t)) = +1$; or $\alpha's \notin H$ and $T(-\alpha', H+\alpha's, A(s)) = +1$ if either $\exists t \in R^s_d[f; s]$ such that $\alpha t \notin H$ and $P(\alpha, H+\alpha t, A(t)) = +1$; or $\alpha's \notin H$ and $T(\alpha', H+\alpha's, A(s)) = -1$ if either $\exists t \in R^s_d[f; s]$ such that $\alpha t \notin H$ and $P(\alpha, H+\alpha t, A(t)) = +1$; or $\alpha's \notin H$ and $P(\alpha', H+\alpha's, A(s)) = -1$.

36
By Definition 5.4, iff \( Dftd(\alpha, H, f, r, s) = +1 \).

If \( p \) satisfies T6 then the subject of \( p \) is \( -(\alpha', H, F) \). By T6, T7, and IndHyp, \( T(-\alpha', H, F)) = +1 \) iff \( T(\alpha', H, F) = -1 \) iff \( P(\alpha', H, F) = -1 \) iff \( (\alpha', H) \not\vdash F \).

Thus the theorem is proved by induction.

**EndProofThmA.8**

**Appendix B. Proof of Theorems 7.2, 7.3, 7.4, and 7.5**

**Theorem B.1** (Theorem 7.2 Plausible Conjunction). Suppose \((R, >)\) is a plausible description, \( Ax = Ax(R) \), \( \alpha \in \text{Alg} \), \( H \) is an \( \alpha \)-history, and \( f \) and \( g \) are both formulas. If \( Ax \models f \) and \( (\alpha, H) \models g \) then \( (\alpha, H) \models \wedge \{f, g\} \).

**Proof**

Suppose \((R, >)\) is a plausible description, \( Ax = Ax(R) \), \( \alpha \in \text{Alg} \), \( H \) is an \( \alpha \)-history, and \( f \) and \( g \) are both formulas. Further suppose that \( Ax \models f \) and \( (\alpha, H) \models g \). We shall use Definition 5.9.

Since \( (\alpha, H) \models g \), by Definition 5.9(I3), there exists \( r \) in \( R_d^\alpha [g] \) such that I3.1 and I3.2 both hold. By Lemma A.5(4), there exists \( r \) in \( R_d^\alpha [\wedge \{f, g\}] \) such that I3.1 and I3.2 both hold. Thus \( (\alpha, H) \models \wedge \{f, g\} \).

**EndProofTheoremB.1**

**Theorem B.2** (Theorem 7.3 Right Weakening). Suppose \((R, >)\) is a plausible description, \( Ax = Ax(R) \), \( \alpha \in \text{Alg} \), \( H \) is an \( \alpha \)-history, and \( f \) and \( g \) are both formulas.

1) If \( (\alpha, H) \models f \) and \( Ax \cup \{f\} \models g \) then \( (\alpha, H) \models g \). [Strong Right Weakening]

2) If \( (\alpha, H) \models f \) and \( f \models g \) then \( (\alpha, H) \models g \). [Right Weakening]

3) If \( A \rightarrow g \in R_s \) and \( (\alpha, H) \models A \) then \( (\alpha, H) \models g \). [Modus Ponens for strict rules]

**Proof**

Suppose \((R, >)\) is a plausible description, \( Ax = Ax(R) \), \( \alpha \in \text{Alg} \), \( H \) is an \( \alpha \)-history, and \( f \) and \( g \) are both formulas. Further suppose that \( (\alpha, H) \models f \). We shall use Definition 5.9.

1) Suppose \( Ax \cup \{f\} \models g \).

If \( Ax \models f \) then \( Ax \models g \) and so by I2, \( (\alpha, H) \models g \). So suppose \( Ax \not\models f \). Then \( \alpha \neq \varphi \).

Since \( (\alpha, H) \models f \), by I3.1 for \( f \), \( \exists r_0 \in R_d^\alpha [f] \) such that \( \alpha r_0 \notin H \) and \( (\alpha, H + \alpha r_0) \models A(r_0) \).

By Lemma A.5(5)(b), \( R_d^\alpha [f] \subseteq R_d^\alpha [g] \). Hence \( r_0 \in R_d^\alpha [g] \). So I3.1 holds for \( g \).

By Lemma A.5(5)(d,e), \( R[-g] \subseteq R[-f] \) and \( \text{Foe}(\alpha, g, r_0) \subseteq \text{Foe}(\alpha, f, r_0) \). By Lemma A.5(5)(b,c), \( R_d^\alpha [f] \subseteq R_d^\alpha [g] \) and \( R_d^\alpha [f; s] \subseteq R_d^\alpha [g; s] \).

Now take any \( s_0 \) in \( \text{Foe}(\alpha, g, r_0) \). Then \( s_0 \in \text{Foe}(\alpha, f, r_0) \). If I3.2.1 holds for \( f \) then \( \exists r_0 \in R_d^\alpha [f; s] \) such that \( \alpha t_0 \notin H \) and \( (\alpha, H + \alpha t_0) \models A(t_0) \). Hence \( t_0 \in R_d^\alpha [g; s_0] \) and so I3.2.1 holds for \( g \). If I3.2.2 holds for \( f \) then \( \alpha s_0 \notin H \) and \( (\alpha', H + \alpha s_0) \not\models A(s_0) \). Hence I3.2.2 holds for \( g \).

Thus I3.2 holds for \( g \) and so \( (\alpha, H) \models g \).

2) Suppose \( f \models g \).

Since \( f \models g \), we have \( Ax \cup \{f\} \models g \). So by part (1), \( (\alpha, H) \models g \).

3) Suppose \( A \rightarrow g \in R_s \) and \( (\alpha, H) \models A \).

By Lemma A.4(1), either \( A = \{\} \), or \( A = \{a\} \) where \( a \) is a formula.

Case 1: \( A = \{\} \).

By Definition 5.4, \( g = \wedge Ax \) and so \( Ax \models g \). By Definition 5.9(I2), \( (\alpha, H) \models g \).
Case 2: \( A = \{a\} \) where \( a \) is formula.
By Definition 5.9(11), \((\alpha, H) \models a\). By Lemma A.4(2), \( Ax \cup \{a\} \models g\).
So by part (1), \((\alpha, H) \models g\).

EndProofThmB.2

Definition B.3. Suppose \( H \) is an \( \alpha \)-history. If \( \alpha = \varphi \) then define \( H(\varphi := \pi') \) to be the sequence formed from \( H \) by just replacing each \( \varphi \) by \( \pi' \).
If \( \alpha \in \{ \pi, \pi', \psi, \beta, \beta' \} \) then define \( H(\alpha := \pi') \) to be the sequence formed from \( H \) by just replacing each \( \alpha \) by \( \pi' \), and each \( \alpha' \) by \( \pi \).

It is clear that \( H(\alpha := \pi') \) is a \( \pi' \)-history.

Lemma B.4. Suppose \( \mathcal{P} = (R, >) \) is a plausible theory, \( \alpha \in \text{Alg} \), \( H \) is an \( \alpha \)-history, and \( x \) is either a formula or a finite set of formulas. If \((\alpha, H) \models x \) then \((\pi', H(\alpha := \pi')) \models x \).
Hence \( \mathcal{P}(\alpha) \subseteq \mathcal{P}(\pi') \).

Proof
Suppose \((R, >) \) is a plausible theory. Let \( Y(n) \) denote the following conditional statement.

“If \( \alpha \in \text{Alg} \), \( H \) is an \( \alpha \)-history, \( f \) is a formula, \( F \) is a finite set of formulas, \( x \in \{ F, f \} \),
\( T(\alpha, H, x) = +1 \), and \( |T(\alpha, H, x)| \leq n \) then \( T(\pi', H(\alpha := \pi'), x) = +1 \)”.

By Theorem A.8(1,2), and Theorem A.7(1), it suffices to prove \( Y(n) \) by induction on \( n \).

Suppose \( n = 1 \). Let the antecedent of \( Y(1) \) hold. Let the only node of \( T(\alpha, H, x) \) be \( p_0 \).
Let the root of \( T[\pi', H(\alpha := \pi'), x] \) be \( q_0 \).
If \( p_0 \) satisfies T1 then \( x = \{\} \) and so \( q_0 \) satisfies T1. So by T1, \( T[\pi', H(\alpha := \pi'), \{\}] \) has only one node and \( T(\pi', H(\alpha := \pi'), \{\}) = +1 \). If \( p_0 \) satisfies T2 then \( x = f \) and \( Ax \models f \).
So \( q_0 \) satisfies T2 and hence \( T(\pi', H(\alpha := \pi'), f) = +1 \). Since \( p_0 \) has no children and the proof value of \( p_0 \) is \( +1 \), \( p_0 \) does not satisfy T3. Since the subject of \( p_0 \) is \((\alpha, H, x)\), \( p_0 \) does not satisfy T4, or T5, or T6. Thus the base case holds.

Take any positive integer \( n \). Suppose \( Y(n) \) is true. We shall prove \( Y(n + 1) \).
Suppose the antecedent of \( Y(n + 1) \) holds and that \( |T(\alpha, H, x)| = n + 1 \). Then \( T(\alpha, H, x) = +1 \). Let \( p_0 \) be the root of \( T(\alpha, H, x) \) and \( q_0 \) be the root of \( T[\pi', H(\alpha := \pi'), x] \).
If \( p_0 \) satisfies T1 then \( x = F \). We see that \( q_0 \) also satisfies T1. So \( t(p_0) = ((\alpha, H, F), \text{min}, +1) \) and \( t(q_0) = ((\pi', H(\alpha := \pi'), F), \text{min}, w_0) \), where \( w_0 = T(\pi', H(\alpha := \pi'), F) \in \{+1, -1\} \). Let \( \{p_f : f \in F\} \) be the set of children of \( p_0 \) and \( \{q_f : f \in F\} \) be the set of children of \( q_0 \). Let \( f \) be any formula in \( F \). Then the subject of \( p_f \) is \((\alpha, H, f)\), and the subject of \( q_f \) is \((\pi', H(\alpha := \pi'), f)\). Also \( |T(\alpha, H, f)| \leq n \) and the proof value of \( p_f \) is \( +1 \) because \( p_0 \) is a min node with proof value \( +1 \). So by \( Y(n) \) the proof value of \( q_f \) is \( +1 \). But this is true for each \( f \), so \( w_0 = +1 \), as required.

Since \( p_0 \) has a child, \( p_0 \) does not satisfy T2.
If \( p_0 \) satisfies T3 then \( x = f \). We see that \( q_0 \) also satisfies T3. So \( t(p_0) = ((\alpha, H, f), \text{max}, +1) \) and \( t(q_0) = ((\pi', H(\alpha := \pi'), f), \text{max}, w_0) \), where \( w_0 = T(\pi', H(\alpha := \pi'), f) \in \{+1, -1\} \).
We shall adopt the following naming conventions. Each non-root node of \( T[\alpha, H, f] \) is denoted by \( p_i(#, y) \) where \( i \) is the level of the node, \( # \) is the number in \([1..6]\) such that the node satisfies \( T# \), and \( y \) is a rule, or a formula, or a set, which distinguishes siblings. The proof value of \( p_i(#, y) \) will be denoted by \( v_i(#, y) \). For non-root nodes in \( T[\pi', H(\alpha := \pi'), f] \) we shall use \( q_i(#, y) \), and its proof value will be denoted by \( w_i(#, y) \).
Let the set of children of $p_0$ be \{ $p_1(4,r) : \alpha \notin H$ and $r \in R^a[f]$ \}, where the tags of these children are: $t(p_1(4,r)) = ((\alpha, H, f, r), \min, v_1(4,r))$. So $+1 = \max \{ v_1(4,r) : \alpha \notin H$ and $r \in R^a[f] \}$.

Let the set of children of $q_0$ be \{ $q_1(4,r) : \pi' \notin H(\alpha := \pi')$ and $r \in R^a[f]$ \}, where the tags of these children are: $t(q_1(4,r)) = ((\pi', H(\alpha := \pi'), f, r), \min, w_1(4,r))$. So $w_0 = \max \{ w_1(4,r) : \pi' \notin H(\alpha := \pi')$ and $r \in R^a[f] \}$.

From above there exists $r_0$ in $R^a[f]$ such that $\alpha r_0 \notin H$ and $v_1(4,r_0) = +1$. By Definition B.3, if $\pi' r_0 \in H(\alpha := \pi')$ then $\alpha r_0 \notin H$. So if $\alpha r_0 \notin H$ then $\pi' r_0 \notin H(\alpha := \pi')$.

Hence $q_1(4,r_0)$ exists. We shall show that $w_1(4,r_0) = +1$ and hence that $w_0 = +1$, as required.

Because $p_1(4,r_0)$ is a min node whose proof value is $+1$, the proof value of every child of $p_1(4,r_0)$ must be $+1$. $p_2(1,r_0)$ is a child of $p_1(4,r_0)$ such that $t(p_2(1,r_0)) = ((\alpha, H + \alpha r_0, A(r_0)), \min, +1)$.

The only child of $q_1(4,r_0)$ is $q_2(1,r_0)$ where $t(q_2(1,r_0)) = ((\pi', H(\alpha := \pi') + \pi' r_0, A(r_0)), \min, w_2(1,r_0))$. So $w_1(4,r_0) = w_2(1,r_0)$.

Since $|T(\alpha, H + \alpha r_0, A(r_0))| \leq n$ and $T(\alpha, H + \alpha r_0, A(r_0)) = +1$, by $Y(n)$ we have $T(\pi', H(\alpha := \pi') + \pi' r_0, A(r_0)) = w_2(1,r_0) = +1$. Hence $w_1(4,r_0) = +1$ and so $w_0 = +1$, as required.

Since the subject of $p_0$ is $(\alpha, H, x)$, $p_0$ does not satisfy T4, or T5, or T6.

Thus $Y(n)$, and hence the lemma, is proved by induction.

**EndProofLemB.4**

**Definition B.5.** Suppose $\mathcal{P}_d = (R, >)$ is a plausible description, $\alpha \in \text{Alg}$, $H$ is an $\alpha$-history, $F$ is a finite set of formulas, $f$ is a formula, $r$ and $s$ are any rules, $T$ is an evaluation tree of $\mathcal{P}_d$, and $p$ is any node of $T$. If $\text{Subj}(p) \in \{(\alpha, H, F), (\alpha, H, f), (\alpha, H, f, r), (\alpha, H, f, r, s), -(\alpha, H, F)\}$ then the history of $p$, $\text{Hist}(p)$, is defined by $\text{Hist}(p) = H$, and the algorithm of $p$, $\text{alg}(p)$, is defined by $\text{alg}(p) = \alpha$.

**Definition B.6.** Suppose $\{ \alpha, \lambda \} \subseteq \text{Alg}$, $H$ is a $\lambda$-history, and $T$ is an evaluation tree of some plausible theory. If $\lambda \notin \{ \alpha, \alpha' \}$ then define $\lambda(\alpha: \alpha') = \lambda$; else define $\alpha(\alpha: \alpha') = \alpha'$ and $\alpha'(\alpha: \alpha') = \alpha$. If $H = (\lambda_1 r_1, ..., \lambda_n r_n)$ then define $H(\alpha: \alpha') = (\lambda_1(\alpha: \alpha') r_1, ..., \lambda_n(\alpha: \alpha') r_n)$. Define $T(\alpha: \alpha')$ to be the tree formed from $T$ by only changing the subject of each node as follows. For each node $p$ of $T$ replace $\text{alg}(p)$ by $\text{alg}(p)(\alpha: \alpha')$, and replace $\text{Hist}(p)$ by $\text{Hist}(p)(\alpha: \alpha')$.

**Definition B.7.** Suppose $\alpha \in \text{Alg}$. $\alpha$ is isomorphic to $\alpha'$, $\alpha \simeq \alpha'$, iff for each plausible theory $\mathcal{P}$, if $T$ is an evaluation tree of $\mathcal{P}$ then $T(\alpha: \alpha')$ is an evaluation tree of $\mathcal{P}$.

It should be clear that if $\alpha \simeq \alpha'$ then for each plausible theory $\mathcal{P}$, $\mathcal{P}(\alpha) = \mathcal{P}(\alpha')$.

**Lemma B.8.** $\beta \simeq \beta'$. Hence for each plausible theory $\mathcal{P}$, $\mathcal{P}(\beta) = \mathcal{P}(\beta')$.

**Proof**

Suppose $\alpha \in \{ \beta, \beta' \}$ and $\mathcal{P} = (R, >)$ is a plausible theory. Let $Y(n)$ denote the following conditional statement. “If $T$ is an evaluation tree of $\mathcal{P}$ and $|T| \leq n$ then $T(\alpha: \alpha')$ is an evaluation tree of $\mathcal{P}$.” By Definition 5.14, it suffices to prove $Y(n)$ by induction on $n$. By Definitions 5.13 and A.6, if $T$ is an evaluation tree of $\mathcal{P}$ then either $T = T[\alpha, H, ...]$ or
\[ T = T[-(\alpha', H, F)]. \] So if \( T(\alpha : \alpha') \) is an evaluation tree of \( \mathcal{P} \) then either \( T(\alpha : \alpha') = T[\alpha', H(\alpha : \alpha'), \ldots] \) or \( T(\alpha : \alpha') = T[-(\alpha, H(\alpha : \alpha'), F)]. \)

Suppose \( n = 1 \) and the antecedent of \( Y(1) \) holds. Let the root of \( T \) be \( p_0 \). Since \( \alpha \in \{\beta, \beta'\} \), without loss of generality we can suppose \( \text{alg}(p_0) = \alpha \). Let the root of \( T(\alpha : \alpha') \) be \( q_0 \). Since \( p_0 \) has no children, \( q_0 \) has no children.

If \( p_0 \) satisfies \( T1 \) then let \( \text{Subj}(p_0) = (\alpha, H, \{\}) \). Hence \( \text{Subj}(q_0) = (\alpha', H(\alpha : \alpha'), \{\}) \) and so \( q_0 \) satisfies \( T1 \). Thus \( T(\alpha : \alpha') \) is an evaluation tree of \( \mathcal{P} \).

If \( p_0 \) satisfies \( T2 \) then let \( \text{Subj}(p_0) = (\alpha, H, f) \). Hence \( \text{Subj}(q_0) = (\alpha', H(\alpha : \alpha'), f) \) and so \( q_0 \) satisfies \( T2 \). Thus \( T(\alpha : \alpha') \) is an evaluation tree of \( \mathcal{P} \).

If \( p_0 \) satisfies \( T3 \) then let \( \text{Subj}(p_0) = (\alpha, H, f, r, s) \). Hence \( \text{Subj}(q_0) = (\alpha', H(\alpha : \alpha'), f, r, s) \). Since \( p_0 \) has no children, \( S(p_0) = \{\} \). So if \( r \in R^\alpha_0[f] \) then \( \alpha r \in H \) iff \( \alpha' r \in H(\alpha : \alpha') \).

Also \( \alpha' s \in H \). Hence \( \alpha s \in H(\alpha : \alpha') \). So \( S(q_0, \alpha') = \{\} \) and so \( q_0 \) satisfies \( T3 \). Thus \( T(\alpha : \alpha') \) is an evaluation tree of \( \mathcal{P} \).

Since \( p_0 \) and \( q_0 \) have no children, \( p_0 \) and \( q_0 \) satisfy neither \( T4 \) nor \( T6 \).

If \( p_0 \) satisfies \( T5 \) then let \( \text{Subj}(p_0) = (\alpha, H, f, r, s) \). Hence \( \text{Subj}(q_0) = (\alpha', H(\alpha : \alpha'), f, r, s) \). Since \( p_0 \) has no children, \( S(p_0) = \{\} \) and so \( S(p_0, \alpha) = \{\} \). Hence if \( t \in R^\alpha_0[f; s] \) then \( \alpha t \in H \) iff \( \alpha' t \in H(\alpha : \alpha') \).

All cases have been considered and so \( Y(1) \) holds.

If \( T \) is any tree and \( p \) is any node of \( T \) for which \( \text{Subj}(p) \) is defined, then define the set \( S(p, T) \) of subjects of the children of \( p \) in \( T \) by \( S(p, T) = \{ \text{Subj}(c) : c \) is a child of \( p \) in \( T \} \).

Take any integer \( n \) such that \( n \geq 1 \). Suppose that \( Y(n) \) is true. We shall prove \( Y(n + 1) \).

Suppose the antecedent of \( Y(n + 1) \) holds and that \( |T| = n + 1 \). Let \( p_0 \) be the root of \( T \). If \( \text{alg}(p_0) \notin \{\alpha, \alpha'\} \) then \( T(\alpha : \alpha') \) is \( T \) and so \( T(\alpha : \alpha') \) is an evaluation tree of \( \mathcal{P} \). So suppose \( \text{alg}(p_0) \in \{\alpha, \alpha'\} \). Let \( q_0 \) be the root of \( T(\alpha : \alpha') \).

If \( p_0 \) satisfies \( T1 \) then let \( \text{Subj}(p_0) = (\alpha, H, F) \). Hence \( \text{Subj}(q_0) = (\alpha', H(\alpha : \alpha'), F) \).

So \( q_0 \) satisfies \( T1 \). Recall that \( S(p_0, T(\alpha, H, F)) = \{\alpha, H, f : \alpha r \in H \} \). So \( S(q_0, T(\alpha, H, F)) = \{\alpha', H(\alpha : \alpha'), f \} \).

Since \( \alpha \in H \) iff \( \alpha' r \in H(\alpha : \alpha') \), we have \( \alpha r \in H \) iff \( \alpha' r \notin H(\alpha : \alpha') \). So \( S(q_0, T(\alpha, H, F)) = \{\alpha', H(\alpha : \alpha'), f, r : \alpha r \notin H \} \).

Recall that \( S(p_0, T(\alpha, H, F)) = \{\alpha, H, f, r \} \) and \( r \in R^\alpha_0[f] \). Since \( \alpha \in H \) iff \( \alpha' r \in H(\alpha : \alpha') \), we have \( \alpha r \notin H \) iff \( \alpha' r \notin H(\alpha : \alpha') \). So \( q_0 \) satisfies \( T3 \). Recall that \( S(p_0, T(\alpha, H, f, r)) = \{\alpha, H, f, r \} \) and \( r \in R^\alpha_0[f] \). Since \( T(\alpha : \alpha') \) is \( T \) and so \( T(\alpha : \alpha') \) is an evaluation tree of \( \mathcal{P} \). Hence \( T(\alpha : \alpha') = T[\alpha', H(\alpha : \alpha'), f, r] \).

Thus \( T(\alpha : \alpha') = T[\alpha, H, f][\alpha : \alpha'] = T[\alpha', H(\alpha : \alpha'), f] \) which is an evaluation tree of \( \mathcal{P} \).
\[\alpha, A(r)\] is an evaluation tree of \(P\) and \(|T[\alpha, H + \alpha, A(r)]| \leq n\). So by \(Y(n)\), \(T[\alpha, H + \alpha, A(r)](\alpha : \alpha')\) is an evaluation tree of \(P\). Hence \(T[\alpha, H + \alpha, A(r)](\alpha : \alpha') = T[\alpha', H(\alpha : \alpha') + \alpha', A(r)]\). Also for each \((\alpha, H, f, r, s)\) in \(S(p_0, T[\alpha, H, f, r])\), \(T[\alpha, H, f, r, s]\) is an evaluation tree of \(P\) and \(|T[\alpha, H, f, r, s]| \leq n\). So by \(Y(n)\), \(T[\alpha, H, f, r, s](\alpha : \alpha')\) is an evaluation tree of \(P\). Hence \(T[\alpha, H, f, r, s](\alpha : \alpha') = T[\alpha', H(\alpha : \alpha'), f, r, s]\). Thus \(T(\alpha : \alpha') = T[\alpha, H, f, r](\alpha : \alpha') = T[\alpha', H(\alpha : \alpha'), f, r]\) which is an evaluation tree of \(P\).

If \(p_0\) satisfies T5 then let \(Subj(p_0) = (\alpha, H, f, r, s)\). Hence \(Subj(q_0) = (\alpha', H(\alpha : \alpha'), f, r, s)\). Since \(\alpha \in H\) iff \(\alpha' \in H(\alpha : \alpha')\) with \(\alpha' \not\in H\) if \(\alpha' \not\in H(\alpha : \alpha')\). So \(q_0\) satisfies \(T5\). Recall that \(S(p_0, T[\alpha, H, f, r, s]) = \{(\alpha, H + \alpha, A(t)) : \alpha \not\in H\) and \(t \in R^0_\alpha[f; s]\}\) \(\cup \{-\alpha', H + \alpha, A(s)\} : \alpha' \not\in H\}\). Also since \(\alpha' \in H\) if \(\alpha' \in H(\alpha : \alpha')\) we have \(\alpha' \not\in H\) if \(\alpha' \not\in H(\alpha : \alpha')\). So \(S(q_0, T[\alpha, H, f, r, s](\alpha : \alpha')) = \{(\alpha', H(\alpha : \alpha') + \alpha', A(t)) : \alpha' \not\in H\) and \(t \in R^0_\alpha[f; s]\}\) \(\cup \{-\alpha, H(\alpha : \alpha') + \alpha, A(s)\} : \alpha \not\in H(\alpha : \alpha')\}\).

Thus \(T(\alpha : \alpha') = T[\alpha, H, f, r, s](\alpha : \alpha') = T[\alpha', H(\alpha : \alpha'), f, r, s]\) which is an evaluation tree of \(P\).

If \(p_0\) satisfies T6 then let \(Subj(p_0) = -\alpha', H, F\). Hence \(Subj(q_0) = -\alpha, H(\alpha : \alpha'), F\). So \(q_0\) satisfies T6. Recall that \(S(p_0, T[-\alpha', H, F]) = \{(\alpha', H, F)\}\). So \(S(q_0, T[-\alpha', H, F](\alpha : \alpha')) = \{(\alpha', H(\alpha : \alpha'), F)\}\), by T6. But \(T[\alpha', H, F]\) is an evaluation tree of \(P\) and \(|T[\alpha', H, F]| \leq n\). So by \(Y(n)\), \(T[\alpha', H, F](\alpha : \alpha')\) is an evaluation tree of \(P\). Hence \(T[-\alpha', H + \alpha, A(s)\}] = T[-\alpha, H, A(s)\}]\) is an evaluation tree of \(P\). Hence \(T[\alpha', H, F](\alpha : \alpha') = T[-\alpha, H(\alpha : \alpha'), F]\). Thus \(T(\alpha : \alpha') = T[-\alpha, H(\alpha : \alpha'), F]\) which is an evaluation tree of \(P\).

Therefore \(Y(n + 1)\), and hence the lemma, is proved by induction.

EndProof

**Lemma B.9.** Suppose \(P = (R, >)\) is a plausible theory, \(H\) is a \(\psi\)-history, and \(x\) is either a formula or a finite set of formulas. If \((\psi, H) \models x\) then \((\psi', H(\psi : \psi')) \models x\).

Hence \(P(\psi) \subseteq P(\psi')\).

**Proof**

Suppose \((R, >)\) is a plausible theory. Let \(Y(n)\) denote the following conditional statement. “If \(H\) is a \(\psi\)-history, \(F\) is a finite set of formulas, \(f\) is a formula, \(x \in \{F, f\}\), \(T(\psi, H, x) = +1\), and \(|T(\psi, H, x)| \leq n\) then \(T(\psi', H(\psi : \psi'), x) = +1\).”

By Theorem A.8(1,2), and Theorem 7.1(1), it suffices to prove \(Y(n)\) by induction on \(n\).

Suppose \(n = 1\). Let the antecedent of \(Y(1)\) hold. Let the only node of \(T[\psi, H, x]\) be \(p_0\). Let the root of \(T[\psi', H(\psi : \psi'), x]\) be \(q_0\).

If \(p_0\) satisfies T1 then \(x = \{\}\) and so \(q_0\) satisfies T1. So by T1, \(T[\psi', H(\psi : \psi'), \{\}]\) has only one node and \(T[\psi', H(\psi : \psi'), \{\}] = +1\). If \(p_0\) satisfies T2 then \(x = f\) and \(Ax = f\). So \(q_0\) satisfies T2 and hence \(T(\psi', H(\psi : \psi'), f) = +1\). Since \(p_0\) has no children and the
proof value of \( p_0 \) is +1, \( p_0 \) does not satisfy T3. Since the subject of \( p_0 \) is \( (\psi, H, x) \), \( p_0 \) does not satisfy T4, or T5, or T6. Thus the base case holds.

Take any positive integer \( n \). Suppose \( Y(n) \) is true. We shall prove \( Y(n+1) \).

Suppose the antecedent of \( Y(n+1) \) holds and that \( |T[\psi, H, x]| = n+1 \). Then \( T(\psi, H, x) = +1 \). Let \( p_0 \) be the root of \( T[\psi, H, x] \) and \( q_0 \) be the root of \( T[\psi', H(\psi; \psi'), x] \).

If \( p_0 \) satisfies T1 then \( x = F \). We see that \( q_0 \) also satisfies T1. So \( t(p_0) = ((\psi, H, F), \min, +1) \) and \( t(q_0) = ((\psi', H(\psi; \psi'), F), \min, w_0) \), where \( w_0 = T(\psi', H(\psi; \psi'), F) \in \{+1, -1\} \). Let \( \{p_f : f \in F\} \) be the set of children of \( p_0 \) and \( \{q_f : f \in F\} \) be the set of children of \( q_0 \). Let \( f \) be any formula in \( F \). Then the subject of \( p_f \) is \( (\psi, H, f) \), and the subject of \( q_f \) is \( (\psi', H(\psi; \psi'), f) \). Also \( |T[\psi, H, f]| \leq n \) and the proof value of \( p_f \) is +1 because \( p_0 \) is a min node with proof value +1. So by \( Y(n) \) the proof value of \( q_f \) is +1. But this is true for each \( f \), so \( w_0 = +1 \), as required.

Since \( p_0 \) has a child, \( p_0 \) does not satisfy T2.

If \( p_0 \) satisfies T3 then \( x = f \). We see that \( q_0 \) also satisfies T3. So \( t(p_0) = ((\psi, H, f), \max, +1) \) and \( t(q_0) = ((\psi', H(\psi; \psi'), f), \max, w_0) \), where \( w_0 = T(\psi', H(\psi; \psi'), f) \in \{+1, -1\} \).

We shall adopt the following naming conventions. Each non-root node of \( T \) is denoted by \( p_l(\#, y) \) where \( l \) is the level of the node, \( \# \) is the number in [1, 6] such that the node satisfies T\#, and \( y \) is a rule, or a formula, or a set, which distinguishes siblings. The proof value of \( p_l(\#, y) \) will be denoted by \( v_l(\#, y) \). For non-root nodes in \( T[\psi', H(\psi; \psi'), f] \) we shall use \( q_l(\#, y) \), and its proof value will be denoted by \( w_l(\#, y) \).

Let the set of children of \( p_0 \) be \( \{p_1(4, r) : \psi r \notin H \) and \( r \in R_d^s[f]\} \), where the tags of these children are:

\[
t(p_1(4, r)) = ((\psi, H, f, r), \min, v_1(4, r)). \text{ So } +1 = \max\{v_1(4, r) : \psi r \notin H \text{ and } r \in R_d^s[f]\}.
\]

Let the set of children of \( q_0 \) be \( \{q_1(4, r) : \psi' r \notin H(\psi; \psi') \) and \( r \in R_d^s[f]\} \), where the tags of these children are:

\[
t(q_1(4, r)) = ((\psi', H(\psi; \psi'), f, r), \min, w_1(4, r)). \text{ So } w_0 = \max\{w_1(4, r) : \psi' r \notin H(\psi; \psi') \text{ and } r \in R_d^s[f]\}.
\]

From above there exists \( r_0 \) in \( R_d^s[f] \) such that \( \psi r_0 \notin H \) and \( v_1(4, r_0) = +1 \). By Definition B.6, if \( \psi r_0 \notin H(\psi; \psi') \) then \( \psi r_0 \notin H \). So if \( \psi r_0 \notin H \) then \( \psi' r_0 \notin H(\psi; \psi') \). Hence \( q_1(4, r_0) \) exists. We shall show that \( w_1(4, r_0) = +1 \) and hence that \( w_0 = +1 \), as required.

Let the set of children of \( p_1(4, r_0) \) be \( \{p_2(1, r_0) \} \cup \{p_2(5, s) : s \in \text{Foe}(\psi, f, r_0)\} \), where the tags of these children are:

\[
t(p_2(1, r_0)) = ((\psi, H+\psi r_0, A(r_0)), \min, v_2(1, r_0)); \text{ and } t(p_2(5, s)) = ((\psi, H, f, r_0, s), \max, v_2(5, s)).
\]

So \( +1 = v_1(4, r_0) = \min\{v_2(1, r_0) \cup v_2(5, s) : s \in \text{Foe}(\psi, f, r_0)\} \).

Hence \( v_2(1, r_0) = +1 \) and for each \( s \) in \( \text{Foe}(\psi, f, r_0) \), \( v_2(5, s) = +1 \).

Let the set of children of \( q_1(4, r_0) \) be \( \{q_2(1, r_0) \} \cup \{q_2(5, s) : s \in R[-f; r_0]\} \), where the tags of these children are:

\[
t(q_2(1, r_0)) = ((\psi', H(\psi; \psi') + \psi' r_0, A(r_0)), \min, w_2(1, r_0)); \text{ and } t(q_2(5, s)) = ((\psi', H(\psi; \psi'), f, r_0, s), \max, w_2(5, s)).
\]

So \( w_1(4, r_0) = \min\{w_2(1, r_0) \cup w_2(5, s) : s \in R[-f; r_0]\} \).

Since \( v_2(1, r_0) = +1 \), \( T(\psi, H+\psi r_0, A(r_0)) = +1 \) and \( |T[\psi, H+\psi r_0, A(r_0)]| \leq n \). So by \( Y(n) \), \( w_2(1, r_0) = T(\psi', H(\psi; \psi') + \psi' r_0, A(r_0)) = +1 \). Therefore \( w_1(4, r_0) = \min\{w_2(5, s) : s \in R[-f; r_0]\} \). If \( R[-f; r_0] = \{\} \) then \( w_1(4, r_0) = \min\{\} = +1 \) as desired. So suppose \( R[-f; r_0] \neq \{\} \).

Since \( R[-f; r_0] \subseteq \text{Foe}(\psi, f, r_0) \), if \( q_2(5, s) \) exists then \( p_2(5, s) \) exists.
For each node, \( p_2(5, s) \), let the set of children of \( p_2(5, s) \) be \( \{ p_3(1, t) : \psi t \notin H \) and \( t \in R^*_0[f; s] \} \cup \{ p_3(6, s) : \psi s \notin H \) \}, where the tags of these children are: \( t(p_3(1, t)) = ((\psi, H + \psi t, A(t)), \min, v_3(1, t)) \) and \( t(p_3(6, s)) = (-(\psi', H + \psi s, A(s)), -v_3(6, s)) \). So \( +1 = v_2(5, s) = \max\{[v_3(1, t) : \psi t \notin H \) and \( t \in R^*_0[f; s] \} \cup \{ v_3(6, s) : \psi s \notin H \)} \).

For each node, \( q_2(5, s) \), let the set of children of \( q_2(5, s) \) be \( \{ q_3(1, t) : \psi t \notin H(\psi : \psi') \) and \( t \in R^*_0[f; s] \} \cup \{ q_3(6, s) : \psi s \notin H(\psi : \psi') \}, where the tags of these children are: \( t(q_3(1, t)) = ((\psi', H(\psi : \psi') + \psi t, A(t)), \min, w_3(1, t)) \) and \( t(q_3(6, s)) = (-(\psi, H(\psi : \psi') + \psi s, A(s)), -w_3(6, s)) \). So \( w_2(5, s) = \max\{[w_3(1, t) : \psi t \notin H(\psi : \psi') \) and \( t \in R^*_0[f; s] \} \cup \{ w_3(6, s) : \psi s \notin H(\psi : \psi') \} \).

By Definition B.6, \( \psi t \in H \) iff \( \psi t \in H(\psi : \psi') \). So \( \psi t \notin H \) iff \( \psi t \notin H(\psi : \psi') \). Therefore \( p_3(1, t) \) exists iff \( q_3(1, t) \) exists. If there exists \( t_0 \in R^*_0[f; s] \) such that \( v_3(1, t_0) = +1 \) then \( T(\psi, H + \psi t_0, A(t_0)) = +1 \) and \( |T[\psi, H + \psi t_0, A(t_0)]| \leq n \) so by \( Y(n) \), \( w_3(1, t_0) = T(\psi', H(\psi : \psi') + \psi t_0, A(t_0)) = +1 \). Hence \( w_2(5, s) = +1 \).

So suppose no such \( t_0 \in R^*_0[f; s] \) exists. Then \( p_3(6, s) \) exists such that \( \psi s \notin H \) and \( v_3(6, s) = +1 \). By Definition B.6, \( \psi s \in H \) iff \( \psi s \in H(\psi : \psi') \). So \( \psi s \notin H \) iff \( \psi s \notin H(\psi : \psi') \). Hence \( q_3(6, s) \) exists.

Let the child of \( p_3(6, s) \) be \( p_4(1, s) \) where \( t(p_4(1, s)) = ((\psi', H + \psi s, A(s)), \min, v_4(1, s)) \) and \( v_4(6, s) = -v_4(1, s) \). So \( v_4(1, s) = -1 \).

Let the child of \( q_3(6, s) \) be \( q_4(1, s) \) where \( t(q_4(1, s)) = ((\psi, H(\psi : \psi') + \psi s, A(s)), \min, w_4(1, s)) \) and \( w_3(6, s) = -w_4(1, s) \).

Assume \( w_4(1, s) = +1 \). Then \( T(\psi, H(\psi : \psi') + \psi s, A(s)) = +1 \) and \( |T[\psi, H(\psi : \psi') + \psi s, A(s)]| \leq n \) so by \( Y(n) \), \( T(\psi', H(\psi : \psi') + \psi s, A(s)) = +1 \). But \( H(\psi : \psi')(\psi : \psi') = H \). So \( T(\psi', H + \psi s, A(s)) = +1 \). From above \( -1 = v_4(1, s) = T(\psi', H + \psi s, A(s)) = +1 \). This contradiction shows that \( w_4(1, s) = -1 \). Hence \( w_3(6, s) = +1 \) and so \( w_3(6, s) = +1 \).

So in both cases for all \( s \) in \( R[f; r_0] \), \( w_2(5, s) = +1 \) and so \( w_1(4, r_0) = +1 \). Hence \( w_0 = +1 \), as required.

Since the subject of \( p_0 \) is \( (\psi, H, x) \), \( p_0 \) does not satisfy T4, or T5, or T6.

Thus \( Y(n) \), and hence the lemma, is proved by induction.

**EndProof**

**Lemma B.10.** Suppose \( (R, >) \) is a plausible theory, \( Ax = Ax(R) \), and \( f \) is a formula. If \( f \) is satisfiable and \( Ax \not\models f \) then \( R^*_0[f] \) is finite.

**Proof**

Suppose \( (R, >) \) is a plausible theory, \( Ax = Ax(R) \), and \( f \) is a formula. Also suppose \( f \) is satisfiable and \( Ax \not\models f \). Take any \( \alpha \) in \( Alg - \{ \varphi \} \). By Definition 5.14, \( T[\alpha, (\cdot), f] \) is finite, and so its root has only finitely many children. The root of \( T[\alpha, (\cdot), f] \) satisfies T3 of Definition 5.13. Therefore \( R^*_0[f] \) is finite.

**EndProof**

**Theorem B.11** (Theorem 7.4 Consistency). Suppose \( (R, >) \) is a plausible theory, \( Ax = Ax(R) \), \( \alpha \in \{ \varphi, \pi, \psi, \beta, \beta' \} \), and both \( f \) and \( g \) are any formulas.

1) If \( \alpha \vdash f \) and \( \alpha \vdash g \) then \( Ax \cup \{ f, g \} \) is satisfiable.

2) If \( (\psi, H) \vdash f \) then \( (\psi', H) \not\vdash f \).

3) Suppose that whenever \( s \in R^*_0[f] \) and \( (\pi', H + \pi's) \vdash A(s) \) then \( R^*_0[f; s] = \{ \} \).

If \( (\pi, H) \vdash f \) then \( (\pi', H) \not\vdash f \).
Proof

Suppose \((R, >)\) is a plausible theory, \(Ax = Ax(R)\), \(\alpha \in \{\varphi, \pi, \psi, \beta, \beta'\}\), and both \(f\) and \(g\) are any formulas.

(1) Suppose \(\alpha \models f\) and \(\alpha \models g\). So by Theorem A.8(2), \(T(\alpha, f) = +1\) and \(T(\alpha, g) = +1\).

Let \(p_0\) be the root of \(T[\alpha, f]\) and \(q_0\) be the root of \(T[\alpha, g]\). Since the subject of \(p_0\) is \((\alpha, f)\), \(p_0\) does not satisfy T1, or T4, or T5, or T6. Since the subject of \(q_0\) is \((\alpha, g)\), \(q_0\) does not satisfy T1, or T4, or T5, or T6. Therefore \(p_0\) satisfies T2 or T3, and \(q_0\) satisfies T2 or T3. So there are four cases to consider.

Case 1: \(p_0\) satisfies T2 and \(q_0\) satisfies T2.

Then \(Ax \models f\) and \(Ax \models g\). By Lemma A.3(1), \(Ax\) is satisfiable. Therefore \(Ax \cup \{f, g\}\) is satisfiable.

Case 2: \(p_0\) satisfies T2 and \(q_0\) satisfies T3.

Then \(Ax \models f\), \(x \in R[f]\), \(Ax \models g\), and \(\alpha \neq \varphi\). So \(\alpha \in \{\pi, \psi, \beta, \beta'\}\). Assume \(Ax \cup \{f, g\}\) is unsatisfiable. By Lemma A.5(6), \(R[f] \subseteq R[-g]\). So \(x \in R[-g]\). By T3, \(q_0\) has a child, \(q_1\), in \(T[\alpha, g]\) such that \(t(q_1) = ((\alpha, g, r_g), \min, +1)\) and \(r_g \in R^*_d[g]\). By T4, \(q_1\) has a child, \(q_2\), in \(T[\alpha, g]\) such that \(t(q_2) = ((\alpha, g, r_g', r_f), \max, +1)\). By T5, \(q_2\) has a child, \(q_3\), in \(T[\alpha, g]\) such that \(pv(q_3) = +1\). By T5, \(Subj(q_3) \in S(q_2)\). By Definition 5.5(3), \(R^*_d[g; r_f] = \{\}\) and so \(S(q_2) = S(q_2, \alpha)\). However, \(A(x) = \{\}\). So \(t(q_3) = (-(\alpha', (\alpha', r_f), \{\}), -1)\). By T6, \(q_3\) has a child, \(q_4\) in \(T[\alpha, g]\) such that \(Subj(q_4) = (\alpha', (\alpha', r_f), \{\})\). By T1, \(pv(q_4) = +1\). But by T7, \(+1 = pv(q_3) = -pv(q_4) = -1\). This contradiction shows that \(Ax \cup \{f, g\}\) is satisfiable.

Case 3: \(p_0\) satisfies T3 and \(q_0\) satisfies T2.

This case is the same as Case 2 but with \(p\) and \(q\) interchanged and with \(f\) and \(g\) interchanged. So by doing the indicated interchanges the proof for Case 2 becomes a proof for Case 3.

Case 4: \(p_0\) satisfies T3 and \(q_0\) satisfies T3.

Then \(Ax \models f\), \(Ax \models g\), and \(\alpha \neq \varphi\). So \(\alpha \in \{\pi, \psi, \beta, \beta'\}\). By T3, \(p_0\) has a child, \(p_1\), in \(T[\alpha, f]\) such that \(t(p_1) = ((\alpha, f, r_f), \min, +1)\) and \(r_f \in R^*_d[f]\). By T3, \(q_0\) has a child, \(q_1\), in \(T[\alpha, g]\) such that \(t(q_1) = ((\alpha, g, r_g), \min, +1)\) and \(r_g \in R^*_d[g]\).

Assume \(Ax \cup \{f, g\}\) is unsatisfiable. By Lemma A.5(6), \(R^*_d[f] \subseteq R[f] \subseteq R[-g]\) and \(R^*_d[g] \subseteq R[g] \subseteq R[-f]\). So \(r_f \in R[-g]\) and \(r_g \in R[-f]\).

By T4, either \(r_f > r_g\); or \(p_1\) has a child, \(p_2(r_g)\), in \(T[\alpha, f]\) such that \(Subj(p_2(r_g)) = (\alpha, (f, r_f, r_g))\) and \(pv(p_2(r_g)) = +1\). By T5, \(p_2(r_g)\) has a child, \(p_3(r_g)\) in \(T[\alpha, f]\) such that \(pv(p_3(r_g)) = +1\) and \(Subj(p_3(r_g)) \in S(p_2(r_g))\).

Similarly by T4, either \(r_g > r_f\); or \(q_1\) has a child, \(q_2(r_f)\), in \(T[\alpha, g]\) such that \(Subj(q_2(r_f)) = (\alpha, (g, r_g, r_f))\) and \(pv(q_2(r_f)) = +1\). By T5, \(q_2(r_f)\) has a child, \(q_3(r_f)\) in \(T[\alpha, g]\) such that \(pv(q_3(r_f)) = +1\) and \(Subj(q_3(r_f)) \in S(q_2(r_f))\).

Case 4.1: \(Subj(p_3(r_g)) = -(\alpha', (\alpha', r_g), A(r_g))\).
Since \(pv(p_3(r_g)) = +1\), \(T(\alpha', (\alpha', r_g), A(r_g)) = -1\). So by Theorem A.8(1), \((\alpha', (\alpha', r_g)) \not\models A(r_g)\). But \(Subj(q_2(r_g)) = (\alpha, (\alpha', r_g), A(r_g))\) and \(pv(q_2(r_g)) = +1\). So \(T(\alpha, (\alpha', r_g), A(r_g)) = +1\). By Theorem A.8(1), \((\alpha, (\alpha', r_g)) \models A(r_g)\). By Lemmas B.4, B.9, and B.8, \((\alpha', (\alpha', r_g)) \models A(r_g)\). This contradiction shows that Case 4.1 cannot occur. Thus \(Subj(p_3(r_g)) = (\alpha, (\alpha t), A(t))\) where \(t \in R^*_d[f; r_g]\).
Case 4.2: $\text{Subj}(q_3(r_f)) = - (\alpha', (\alpha' r_f), A(r_f))$.

This case is the same as Case 4.1 but with $p$ and $q$ interchanged and with $f$ and $g$ interchanged. So by doing the indicated interchanges the proof that Case 4.1 cannot occur becomes a proof that Case 4.2 cannot occur. Thus $\text{Subj}(q_3(r_f)) = (\alpha, (\alpha t), A(t))$ where $t \in R_d^s[g; r_f]$.

In summary Cases 4.1 and 4.2 have shown that we have

- either $r_f > r_g$ or there is a $t$ in $R_d^s[f; r_g]$; and also
- either $r_g > r_f$ or there is a $t$ in $R_d^s[g; r_f]$.

So there exists $t_f(1)$ in $R_d^s[f; r_g] \subseteq R_d^s[f] \subseteq R[f] \subseteq R[-g]$. Hence $t_f(1) > r_g$ and $t_f(1) \in R[-g]$. So $q_2(r_f)$ can be replaced by $q_2(t_f(1))$, and $q_3(r_f)$ can be replaced by $q_3(t_f(1))$.

Also there exists $t_g(1)$ in $R_d^s[g; r_f] \subseteq R_d^s[g] \subseteq R[g] \subseteq R[-f]$. Hence $t_g(1) > r_f$ and $t_g(1) \in R[-f]$. So $p_2(r_g)$ can be replaced by $p_2(t_g(1))$, and $p_3(r_g)$ can be replaced by $p_3(t_g(1))$.

Similarly, the arguments in Cases 4.1 and 4.2 for these new nodes yield rules $t_f(2)$ and $t_g(2)$ with the following properties: $t_f(2) \in R_d^s[f; t_g(1)] \subseteq R_d^s[f] \subseteq R[f] \subseteq R[-g]$; and $t_g(2) \in R_d^s[g; t_f(1)] \subseteq R_d^s[g] \subseteq R[g] \subseteq R[-f]$. So $t_f(2) > t_g(1)$ and $t_f(2) \in R[-g]$ and $t_g(2) > t_f(1)$ and $t_g(2) \in R[-f]$. Hence $t_f(2) > t_f(1) > r_f$ and $t_g(2) > t_g(1) > r_g$.

This process can be continued indefinitely to yield the following sequences of rules.

- $r_f < t_f(1) < t_f(2) < t_f(3) < t_f(4) < ...$ and $r_g < t_g(1) < t_g(2) < t_g(3) < t_g(4) < ...$

Now each $t_f(i) \in R_d^s[f]$ and each $t_g(i) \in R_d^s[g]$. Since $\alpha \vdash f$ and $\alpha \not\vdash g$, by Lemma A.3(5), both $f$ and $g$ are satisfiable. But $Ax \not\vdash f$ and $Ax \not\vdash g$, so by Lemma B.10, both $R_d^s[f]$ and $R_d^s[g]$ are finite. So for some $i$ and some $j > i$, $t_f(2i) = t_g(2j)$. Hence $> r$ is cyclic, which contradicts the definition of $> r$ as being acyclic. This contradiction shows that $Ax \cup \{f, g\}$ is satisfiable.

(2) If $H$ is a $\psi$-history then $H$ is also a $\psi'$-history. Suppose $(\psi, H) \models f$. We shall use Definition 5.9. Assume $(\psi', H) \not\models f$.

By I3.1 for $\neg f$, $(\psi', H+\psi's_1) \models A(s_1)$. By I3.2 for $f$ either $(\psi', H+\psi's_1) \not\models A(s_1)$, which contradicts (i), or $(\psi', H+\psi's_1) \models A(s_2)$. By I3.2 for $\neg f$ either $(\psi, H+\psi s_2) \models A(r_2)$, which contradicts (*2), or $(\psi, H+\psi s_2) \not\models A(r_2)$. By I3.2 for $f$ either $(\psi', H+\psi's_3) \models A(s_3)$, which contradicts (*3), or $(\psi', H+\psi's_3) \not\models A(s_3)$. So by Lemma B.10, both $R_d^s[f]$ and $R_d^s[g]$ are finite. So there is an even $j$ and an even $k$ such that $j < k$ and $r_j = r_k$. Hence $> r$ is cyclic, contradicting its acyclicity. Thus (2) is proved.

(3) Suppose that (*) whenever $s \in R_d^s[-f]$ and $(\pi', H+\pi's) \models A(s)$ then $R_d^s[f; s] = \emptyset$.

If $H$ is a $\pi$-history then $H$ is also a $\pi'$-history. Suppose $(\pi, H) \models f$. We shall use Definition 5.9. Assume $(\pi', H) \not\models f$.
By I3.1 for \( \neg f \), (** \( \exists s_1 \in R_0^f[\neg f] \) such that \((\pi', H + \pi' s_1) \models A(s_1)\). By I3.2 for \( f \) either \((\pi', H + \pi' s_1) \not\models A(s_1)\), which contradicts (**), or \( \exists r_2 \in R_0^f[s; \pi] \) such that 
\((\pi, H + \pi' r_2) \models A(r_2)\), which contradicts (*). Thus (3) is proved.

EndProofThmB.11

**Theorem B.12** (Theorem 7.5 Truth values). Suppose \((R, >)\) is a plausible theory, \(\alpha \in Alg\), \(F\) is a finite set of formulas, and \(f\) is a formula.

1) \(V(\alpha, \neg f) = V(\alpha, f)\).
2) \(V(\alpha, f) = t\) iff \(V(\alpha, \neg f) = f\).
3) \(V(\alpha, f) = f\) iff \(V(\alpha, \neg f) = t\).
4) \(V(\alpha, f) = a\) iff \(V(\alpha, \neg f) = a\).
5) \(V(\alpha, f) = u\) iff \(V(\alpha, \neg f) = u\).
6) If \(V(\alpha, \Lambda F) = t\) then for each \(f\) in \(F\), \(V(\alpha, f) = t\).
7) If \(f \in F\) and \(V(\alpha, f) = t\) then \(V(\alpha, \Lambda F) = t\).
8) If \(\alpha \in \{\varphi, \pi, \psi, \beta, \beta'\}\) then \(V(\alpha, f) \in \{t, f, u\}\).
9) If \(V(\alpha, f) = a\) then \(\alpha \in \{\psi', \pi'\}\).
10) If \(V(\alpha, f) = t\) then \(\alpha \models f\) (completeness).
11) If \(\alpha \in \{\varphi, \pi, \psi, \beta, \beta'\}\) and \(\alpha \models f\) then \(V(\alpha, f) = t\) (soundness).

**Proof**
Suppose \((R, >)\) is a plausible theory, \(\alpha \in Alg\), \(F\) is a finite set of formulas, and \(f\) is a formula. By Theorem 7.3(2), \(\alpha \models f\) iff \(\alpha \models \neg \neg f\); and so \(\alpha \not\models f\) iff \(\alpha \not\models \neg \neg f\).

1) This follows from Definition 5.18 and the equivalences noted above.
2) \(V(\alpha, f) = t\) iff \(\alpha \models f\) and \(\alpha \not\models \neg f\). \(V(\alpha, \neg f) = f\) iff \(\alpha \not\models f\) and \(\alpha \models \neg \neg f\). So (2) holds.
3) \(V(\alpha, f) = f\) iff \(\alpha \not\models f\) and \(\alpha \models \neg f\). \(V(\alpha, \neg f) = t\) iff \(\alpha \models \neg f\) and \(\alpha \not\models \neg f\). So (3) holds.
4) \(V(\alpha, f) = a\) iff \(\alpha \models f\) and \(\alpha \not\models \neg f\). \(V(\alpha, \neg f) = a\) iff \(\alpha \not\models f\) and \(\alpha \models \neg f\). So (4) holds.
5) \(V(\alpha, f) = u\) iff \(\alpha \not\models f\) and \(\alpha \not\models \neg f\). \(V(\alpha, \neg f) = u\) iff \(\alpha \not\models f\) and \(\alpha \not\models \neg f\). So (5) holds.
6) Suppose \(V(\alpha, \Lambda F) = t\). Then \(\alpha \models \Lambda F\) and \(\alpha \not\models \neg \Lambda F\). By Theorem B.2(2), for each \(f\) in \(F\), \(\alpha \models f\). Take any \(f\) in \(F\) and assume \(\alpha \models \neg f\). By Theorem B.2(2), \(\alpha \models \neg \Lambda F\), where \(\neg F = \{\neg f : f \in F\}\). So by Theorem B.2(2), \(\alpha \models \neg \Lambda F\). This contradiction shows that for each \(f\) in \(F\), \(\alpha \not\models f\). Thus for each \(f\) in \(F\), \(V(\alpha, f) = t\).
7) Suppose \(f \in F\) and \(V(\alpha, f) = t\). Then \(\alpha \models f\) and \(\alpha \not\models \neg f\). By Theorem B.2(2), \(\alpha \models \Lambda F\). Assume \(\alpha \models \neg \Lambda F\). By Theorem B.2(2), \(\alpha \models \Lambda F\) and so \(\alpha \models f\). This contradiction shows that \(\alpha \not\models \Lambda F\). Thus \(V(\alpha, \Lambda F) = t\).
8) Suppose \(\alpha \in \{\varphi, \pi, \psi, \beta, \beta'\}\). Recall \(V(\alpha, f) = a\) iff \(\alpha \models f\) and \(\alpha \not\models \neg f\). So by Theorem 7.4(1), \(V(\alpha, f) \neq a\).
9) This is just the contrapositive of part (8).
10) Recall \(V(\alpha, f) = t\) iff \(\alpha \models f\) and \(\alpha \not\models \neg f\).
11) Suppose \(\alpha \in \{\varphi, \pi, \psi, \beta, \beta'\}\) and \(\alpha \models f\). By Definition 5.18 and \(\alpha \models f\) we have \(V(\alpha, f) \in \{a, t\}\). So by part (8), \(V(\alpha, f) = t\).

EndProofThmB.12
Appendix C. Proof of Theorem 7.6

Lemma C.1. Suppose $\mathcal{P} = (R, >)$ is a plausible theory, $\alpha \in \text{Alg}$, $I$ is a $\varphi$-history, $H$ is an $\alpha$-history, and $x$ is either a formula or a finite set of formulas. Let $(\varphi, I) \models x$ then $(\alpha, H) \models x$. Hence $\mathcal{P}(\varphi) \subseteq \mathcal{P}(\alpha)$.

Proof

Suppose $(R, >)$ is a plausible theory, $Ax = Ax(R)$, $\alpha \in \text{Alg}$, $H$ is an $\alpha$-history, and $x$ is either a formula or a finite set of formulas. Let $(\varphi, I) \models x$. We shall use Definition 5.9.

Case 1: $x$ is a formula.

Let $x = f$. Then $(\varphi, I) \not\models f$. By Definition 5.9(I2), $Ax \models f$ and $(\alpha, H) \models f$.

Case 2: $x$ is a finite set of formulas.

Let $x = F$. Then $(\varphi, I) \not\models F$. By I1, for all $f$ in $F$, $(\varphi, I) \not\models f$. By Case 1, $(\alpha, H) \models f$.

So by I1, $(\alpha, H) \models F$.

EndProof

Lemma C.3. Suppose $\mathcal{P} = (R, >)$ is a plausible theory, $H$ is a $\pi$-history, and $x$ is either a formula or a finite set of formulas. If $(\pi, H) \models x$ then $(\psi, H(\pi := \psi)) \models x$.

Hence $\mathcal{P}(\pi) \subseteq \mathcal{P}(\psi)$.

Proof

Suppose $(R, >)$ is a plausible theory and $Ax$ is its set of axioms. Let $Y(n)$ denote the following conditional statement.

“If $H$ is a $\pi$-history, $x$ is either a formula or a finite set of formulas, $T(\pi, H, x) = +1$, and $|T[\pi, H, x]| \leq n$ then $T(\psi, H(\pi := \psi), x) = +1$.”

By Theorem A.8(1,2), and Theorem A.7(1), it suffices to prove $Y(n)$ by induction on $n$.

Suppose $n = 1$. Let the antecedent of $Y(1)$ hold. Let $p_0$ be the root of $T[\pi, H, x]$ and $q_0$ be the root of $T[\psi, H(\pi := \psi), x]$. Then $p_0$ has no children. If $p_0$ satisfies T1 then $x = \{\} \psi$ and so $q_0$ satisfies T1. So by T1, $T[\psi, H(\pi := \psi), \{\}]$ has only one node and $T(\psi, H(\pi := \psi), \{\}) = +1$. If $p_0$ satisfies T2 then $x = f$ and $Ax \models f$. So $q_0$ satisfies T2 and hence $T(\psi, H(\pi := \psi), f) = +1$. Since $p_0$ has no children and the proof value of $p_0$ is +1, $p_0$ does not satisfy T3. Since the subject of $p_0$ is $(\pi, H, x)$, $p_0$ does not satisfy T4, or T5, or T6. Thus the base case holds.

Take any positive integer $n$. Suppose that $Y(n)$ is true. We shall prove $Y(n+1)$.

Suppose the antecedent of $Y(n+1)$ holds and that $|T[\pi, H, x]| = n+1$. Let $p_0$ be the root of $T[\pi, H, x]$ and $q_0$ be the root of $T[\psi, H(\pi := \psi), x]$.

If $p_0$ satisfies T1 then let $x$ be $F$. We see that $q_0$ also satisfies T1. So $t(p_0) = ((\pi, H, F), \text{min}, +1)$ and $t(q_0) = ((\psi, H(\pi := \psi), F), \text{min}, w_0)$, where $w_0 = T(\psi, H(\pi := \psi), F) \in \{+1, -1\}$. Let $\{p_f : f \in F\}$ be the set of children of $p_0$ and $\{q_f : f \in F\}$ the set of children of $q_0$. Let $f$ be any formula in $F$. Then the subject of $p_f$ is $(\pi, H, f)$, and the subject of $q_f$ is $(\psi, H(\pi := \psi), f)$. Also $|T[\pi, H, f]| \leq n$ and the proof value of $p_f$ is +1 because $p_0$ is a min node with proof value +1. So by $Y(n)$ the proof value of $q_f$ is +1. But this is true for each $f$, so $w_0 = +1$, as required.

Since $p_0$ has a child, $p_0$ does not satisfy T2.
If $p_0$ satisfies T3 then let $x$ be $f$. We see that $q_0$ also satisfies T3. So $t(p_0) = ((\pi, H, f), \max, +1)$ and $t(q_0) = ((\psi, H(\pi := \psi), f), \max, w_0)$, where $w_0 = T(\psi, H(\pi := \psi), f) \in \{+1, -1\}$.

We shall adopt the following naming conventions. Each non-root node of $T[\pi, H, f]$ is denoted by $p_i(\#, y)$ where $i$ is the level of the node, $\#$ is the number in $[1..6]$ such that the node satisfies $T\#$, and $y$ is a rule, or a formula, or a set, which distinguishes siblings. The proof value of $p_i(\#, y)$ will be denoted by $v_i(\#, y)$. For non-root nodes in $T[\psi, H(\pi := \psi), f]$ we shall use $q_i(\#, y)$, and its proof value will be denoted by $w_i(\#, y)$.

Let the set of children of $p_0$ be $\{p_1(4, r) : \pi r \notin H$ and $r \in R^*_0[f]\}$, where the tags of these children are:

$t(p_1(4, r)) = ((\pi, H, f, r), \min, v_1(4, r))$. So $+1 = \max\{v_1(4, r) : \pi r \notin H$ and $r \in R^*_0[f]\}$.

Let the set of children of $q_0$ be $\{q_1(4, r) : \psi r \notin H(\pi := \psi)$ and $r \in R^*_0[f]\}$, where the tags of these children are: $t(q_1(4, r)) = ((\psi, H(\pi := \psi), f, r), \min, w_1(4, r))$. So $w_0 = \max\{w_1(4, r) : \psi r \notin H(\pi := \psi)$ and $r \in R^*_0[f]\}$.

From above there exists $r_0$ in $R^*_0[f]$ such that $\pi r_0 \notin H$ and $v_1(4, r_0) = +1$. So $t(p_1(4, r_0)) = ((\pi, H, f, r_0), \min, +1)$.

Let the set of children of $p_1(4, r_0)$ be $\{p_2(1, r_0)\} \cup \{p_2(5, s) : s \in Foe(\pi, f, r_0)\}$, where the tags of these children are: $t(p_2(1, r_0)) = ((\pi, H, f, r_0), \min, v_2(1, r_0))$; and $t(p_2(5, s)) = ((\pi, H, f, r_0, s), \max, v_2(5, s))$. So $+1 = v_1(4, r_0) = \min\{v_2(1, r_0)\} \cup \{v_2(5, s) : s \in Foe(\pi, f, r_0)\}$. Hence $v_2(1, r_0) = +1$; and for each $s$ in $Foe(\pi, f, r_0)$, $v_2(5, s) = +1$.

Let the set of children of $q_1(4, r_0)$ be $\{q_2(1, r_0)\} \cup \{q_2(5, s) : s \in Foe(\psi, f, r_0)\}$, where the tags of these children are: $t(q_2(1, r_0)) = ((\psi, H(\pi := \psi) + \psi r_0, A(r_0)), \min, w_2(1, r_0))$; and $t(q_2(5, s)) = ((\psi, H(\pi := \psi), f, r_0, s), \max, w_2(5, s))$.

So $w_1(4, r_0) = \min\{w_2(1, r_0)\} \cup \{w_2(5, s) : s \in Foe(\psi, f, r_0)\}$.

Since $|T[\pi, H + \pi r_0, A(r_0)]| < n$ and $T(\pi, H + \pi r_0, A(r_0)) = v_2(1, r_0) = +1$, by $Y(n)$ we have $T(\psi, H(\pi := \psi) + \psi r_0, A(r_0)) = w_2(1, r_0) = +1$. Hence $w_2(1, r_0) = \min\{w_2(5, s) : s \in Foe(\psi, f, r_0)\}$.

For each $s$ in $Foe(\pi, f, r_0)$ let the set of children of $p_2(5, s)$ be $\{p_3(1, t) : \pi t \notin H$ and $t \in R^*_d[f; s]\} \cup \{p_3(6, s) : \pi' t \notin H\}$, where the tags of these children are: $t(p_3(1, t)) = ((\pi, H + \pi t, A(t)), \min, v_3(1, t))$; and $t(p_3(6, s)) = (-(\pi', H + \pi s, A(s)), -v_3(6, s))$. So $+1 = v_2(5, s) = \max\{v_3(1, t) : \pi t \notin H$ and $t \in R^*_d[f; s]\} \cup \{v_3(6, s) : \pi' t \notin H\}$.

For each $s$ in $Foe(\psi, f, r_0)$ let the set of children of $q_2(5, s)$ be $\{q_3(1, t) : \psi t \notin H(\pi := \psi)$ and $t \in R^*_d[f; s]\} \cup \{q_3(6, s) : \psi' t \notin H(\pi := \psi)\}$, where the tags of these children are: $t(q_3(1, t)) = ((\psi, H(\pi := \psi) + \psi t, A(t)), \min, w_3(1, t))$; and $t(q_3(6, s)) = (-(\psi', H(\pi := \psi) + \psi' s, A(s)), -w_3(6, s))$. So $w_2(5, s) = \max\{w_3(1, t) : \psi t \notin H(\pi := \psi)$ and $t \in R^*_d[f; s]\} \cup \{w_3(6, s) : \psi' t \notin H(\pi := \psi)\}$.

Take any $s$ in $Foe(\psi, f, r_0)$. We shall show that $w_2(5, s) = +1$.

Suppose there exists $t_0$ such that $\pi t_0 \notin H$ and $t_0 \in R^*_d[f; s]$ and $+1 = v_3(1, t_0) = T(\pi, H + \pi t_0, A(t_0))$. Then $p_3(1, t_0)$ exists, and $\psi t_0 \notin H(\pi := \psi)$ and so $q_3(1, t_0)$ exists. Since $|T[\pi, H + \pi t_0, A(t_0)]| < n$, then by $Y(n)$ we have $T(\psi, H(\pi := \psi) + \psi t_0, A(t_0)) = w_3(1, t_0) = +1$. Hence $w_2(5, s) = +1$. 

48
So suppose that such a \( t_0 \) does not exist. Then \( v_3(6, s) = +1 \) and so \( p_3(6, s) \) exists and \( \pi' s \notin H \). Hence \( \psi' s \notin H(\pi := \psi) \), and so \( q_3(6, s) \) exists. Let the child of \( p_3(6, s) \) be \( p_4(1, s) \) where \( t(p_4(1, s)) = ((\pi', H + \pi' s, A(s)), \text{min}, v_4(1, s)). \) Then \( +1 = v_3(6, s) = -v_4(1, s) \). So \( v_4(1, s) = -1 \) and hence \( T(\pi', H + \pi' s, A(s)) = -1 \). By Theorem A.8(1), 
\[(\pi', H + \pi' s) \not\models A(s).\]
By Definition B.3 with \( \alpha = \psi' \) and Definition C.2, 
\[H(\pi := \psi')(\psi' := \pi') = H \quad \text{and} \quad (\psi' s)(\psi' := \pi') = (\pi' s).\]

Let the child of \( q_3(6, s) \) be \( q_4(1, s) \) where
\[t(q_4(1, s)) = ((\psi', H(\pi := \psi) + \psi' s, A(s)), \text{min}, w_4(1, s)). \] Then \( w_3(6, s) = -w_4(1, s). \)
Assume \( w_4(1, s) = +1. \) Then \( T(\psi', H(\pi := \psi) + \psi' s, A(s)) = +1 \) and so by
Theorem A.8(1), 
\[(\pi', H(\pi := \psi) + \psi' s) \models A(s). \]
By Lemma B.4 with \( \alpha = \psi' \),
\[H(\pi := \psi)(\psi' := \pi') \models A(s). \]
But from the previous paragraph, this simplifies to \( (\pi', H + \pi' s) \models A(s) \). This contradiction shows that \( w_4(1, s) = -1. \) Hence \( w_3(6, s) = +1. \) Therefore \( w_2(5, s) = +1. \)

Thus for all \( s \) in \( \text{Foe}(\psi, f, r_0) \), \( w_2(5, s) = +1. \) So by (*) \( w_1(4, r_0) = +1 \) and hence \( w_0 = +1 \), as required.

Since the subject of \( p_0 \) is \( (\pi, H, x) \), \( p_0 \) does not satisfy \( T4 \) or \( T5 \) or \( T6 \).

Thus \( Y(n) \), and hence the lemma, is proved by induction.

EndProofLemC.3

Definition C.4. If \( H \) is a \( \psi \)-history then define \( H(\psi := \beta) \) to be the sequence formed from \( H \) by just replacing each \( \psi \) by \( \beta \), and each \( \psi' \) by \( \beta' \).

Lemma C.5. Suppose \( P = (R, >) \) is a plausible theory, \( H \) is a \( \psi \)-history, and \( x \) is either a formula or a finite set of formulas.
1) If \( \models (\psi, H) \models x \) then \( (\beta, H(\psi := \beta)) \models x \).
2) If \( \not\models (\psi', H) \models x \) then \( (\beta', H(\psi := \beta)) \not\models x \).
Hence \( P(\psi) \subseteq P(\beta) \) and \( P(\beta') \subseteq P(\psi') \).

Proof
Suppose \( (R, >) \) is a plausible theory, \( Ax \) is its set of axioms, \( H \) is a \( \psi \)-history, and \( x \) is either a formula or a finite set of formulas. Note that \( H \) is a \( \psi \)-history iff \( H \) is a \( \psi' \)-history. Let \( Y(k) \) and \( Z(k) \) denote the following conditional statements.

\( Y(k) \): If \( H \) is a \( \psi \)-history, \( x \) is either a formula or a finite set of formulas,
\[T(\psi, H, x) = +1, \text{ and } |T[\psi, H, x]| \leq k \text{ then } T(\beta, H(\psi := \beta), x) = +1.\]

\( Z(k) \): If \( H \) is a \( \psi' \)-history, \( x \) is either a formula or a finite set of formulas,
\[T(\psi', H, x) = -1, \text{ and } |T[\psi', H, x]| \leq k \text{ then } T(\beta', H(\psi := \beta), x) = -1.\]

By Theorem A.8(1,2) and Theorem A.7(2) it suffices to prove \( Y(k) \) and \( Z(k) \) by induction on \( k \).

Suppose \( k = 1 \).

Let the antecedent of \( Y(1) \) hold. Let \( p_0 \) be the root of \( T[\psi, H, x] \) and \( q_0 \) be the root of \( T[\beta, H(\psi := \beta), x] \). Then \( p_0 \) has no children. If \( p_0 \) satisfies \( T1 \) then \( x = \emptyset \) and so \( q_0 \) satisfies \( T1 \). So by \( T1, T(\beta, H(\psi := \beta), \{\}) \) has only one node and \( T(\beta, H(\psi := \beta), \{\}) = +1 \). If \( p_0 \) satisfies \( T2 \) then \( x = f \) and \( Ax \models f \). So \( q_0 \) satisfies \( T2 \) and hence \( T(\beta, H(\psi := \beta), f) = +1 \). Since \( p_0 \) has no children and the proof value of \( p_0 \) is \( +1 \), \( p_0 \) does not satisfy \( T3 \). Since the subject of \( p_0 \) is \( (\psi, H, x) \), \( p_0 \) does not satisfy \( T4 \) or \( T5 \) or \( T6 \). Thus \( Y(1) \) holds.
Let the antecedent of $Z(1)$ hold. Let $m_0$ be the root of $T[\psi', H, x]$ and $n_0$ be the root of $T[\beta', H(\psi := \beta), x]$. Then $m_0$ has no children. Since $pv(m_0) = T(\psi', H, x) = -1$, $m_0$ does not satisfy T1 or T2. If $m_0$ satisfies T3 then $x = f$ and for each $r \in R^s_0[f]$, $\psi'r \in H$.

So $n_0$ satisfies T3 and for each $r \in R^s_0[f]$, $\beta'r \in H(\psi := \beta)$. Hence by T3, $n_0$ has no children and so $-1 = pv(n_0) = T(\beta', H(\psi := \beta), f)$. Since the subject of $m_0$ is $(\psi', H, x)$, $m_0$ does not satisfy T4 or T5 or T6. Thus $Z(1)$ holds.

Take any positive integer $k$. Suppose that both $Y(k)$ and $Z(k)$ are true. We shall prove both $Y(k+1)$ and $Z(k+1)$.

Suppose the antecedent of $Y(k+1)$ holds and $|T[\psi, H, x]| = k+1$. We must show $T(\beta, H(\psi := \beta), x) = +1$. Let $p_0$ be the root of $T[\psi, H, x]$ and $q_0$ be the root of $T[\beta, H(\psi := \beta), x]$.

If $p_0$ satisfies T1 then let $x$ be $F$. We see that $q_0$ also satisfies T1. So $t(p_0) = ((\psi, H, f), \min, +1)$ and $t(q_0) = ((\beta, H(\psi := \beta), f), \min, w_0)$, where $w_0 = T(\beta, H(\psi := \beta), F)$ $\in \{+1, -1\}$. Let $\{p_f : f \in F\}$ be the set of children of $p_0$ and $\{q_g : f \in F\}$ be the set of children of $q_0$. Let $f$ be any formula in $F$. Then the subject of $p_f$ is $(\psi, H, f)$, and the subject of $q_g$ is $(\beta, H(\psi := \beta), f)$. Also $|T[\psi, H, f]| \leq k$ and the proof value of $p_f$ is +1 because $p_0$ is a min node with proof value +1. So by $Y(k)$ the proof value of $q_g$ is +1. But this is true for each $f$, so by $T_1$, $w_0 = +1$, as required.

Since $p_0$ has a child, $p_0$ does not satisfy T2.

If $p_0$ satisfies T3 then let $x$ be $f$. We see that $q_0$ also satisfies T3. So $t(p_0) = ((\psi, H, f), \max, +1)$ and $t(q_0) = ((\beta, H(\psi := \beta), f), \max, w_0)$, where $w_0 = T(\beta, H(\psi := \beta), f) \in \{+1, -1\}$.

We shall adopt the following naming conventions. Each non-root node of $T[\psi, H, f]$ is denoted by $p_l(\#, y)$ where $l$ is the level of the node, $\#$ is the number in $[1..6]$ such that the node satisfies $T\#$, and $y$ is a rule, or a formula, or a set, which distinguishes siblings. The proof value of $p_l(\#, y)$ will be denoted by $v_l(\#, y)$. For non-root nodes in $T[\beta, H(\psi := \beta), f]$ we shall use $q_l(\#, y)$, and its proof value will be denoted by $w_l(\#, y)$.

Let the set of children of $p_0$ be $\{p_1(4, r) : \psi r \notin H \text{ and } r \in R^s_0[f]\}$, where the tags of these children are:

$t(p_1(4, r)) = ((\psi, H, f, r), \min, v_1(4, r))$. So $+1 = \max\{v_1(4, r) : \psi r \notin H \text{ and } r \in R^s_0[f]\}$.

Hence there exists $r_0$ in $R^s_0[f]$ such that $\psi r_0 \notin H$ and $v_1(4, r_0) = +1$. So $t(p_1(4, r_0)) = ((\psi, H, f, r_0), \min, +1)$.

Let the set of children of $q_0$ be $\{q_1(4, r) : \beta r \notin H(\psi := \beta) \text{ and } r \in R^s_0[f]\}$, where the tags of these children are: $t(q_1(4, r)) = ((\beta, H(\psi := \beta), f, r), \min, w_1(4, r))$.

So $w_0 = \max\{w_1(4, r) : \beta r \notin H(\psi := \beta) \text{ and } r \in R^s_0[f]\}$.

Let the set of children of $p_1(4, r_0)$ be $\{p_2(1, r_0) \cup p_2(5, s) : s \in Foe(\psi, f, r_0)\}$, where the tags of these children are: $t(p_2(1, r_0)) = ((\psi, H + \psi r_0, A(r_0), \min, v_2(1, r_0))$ and $t(p_2(5, s)) = ((\psi, H, f, r_0, s), \max, v_2(5, s))$. So $+1 = v_1(4, r_0) = \min\{\{v_2(1, r_0) \cup \{v_2(5, s) : s \in Foe(\psi, f, r_0)\}\}$. Hence $v_2(1, r_0) = +1$; and for each $s$ in $Foe(\psi, f, r_0)$, $v_2(5, s) = +1$.

Let the set of children of $q_1(4, r_0)$ be $\{q_2(1, r_0) \cup q_2(5, s) : s \in Foe(\beta, f, r_0)\}$, where the tags of these children are: $t(q_2(1, r_0)) = ((\beta, H(\psi := \beta) + \beta r_0, A(r_0)), \min, w_2(1, r_0))$; and $t(q_2(5, s)) = ((\beta, H(\psi := \beta), f, r_0, s), \max, w_2(5, s))$.

So $w_1(4, r_0) = \min\{\{w_2(1, r_0) \cup \{w_2(5, s) : s \in Foe(\beta, f, r_0)\}\}$. 

Billington
Since \(|T[\psi, H+\psi r_0, A(r_0)]|_{k} \leq k \) and \(T(\psi, H+\psi r_0, A(r_0)) = v_2(1, r_0) = +1\), by \(Y(k)\) we have \(T(\beta, H(\psi := \beta)+\beta r_0, A(r_0)) = w_2(1, r_0) = +1\). Hence
\[^{(*)} w_1(4, r_0) = \min\{w_2(5, s) : s \in \text{Foe}(\beta, f, r_0)\}\].

For each \(s\) in \(\text{Foe}(\psi, f, r_0)\) let the set of children of \(p_2(5, s)\) be \(\{p_3(1, t) : \psi t \notin H \text{ and } t \in R_0^a[f; s]\} \cup \{p_2(6, s) : \psi' s \notin H\}\), where the tags of these children are: \(t(p_3(1, t)) = ((\psi, H+\psi t, A(t)), \min, v_3(1, t))\); and \(t(p_3(6, s)) = (- (\psi', H+\psi' s, A(s)), - , v_3(6, s))\).

So \(+1 = v_3(5, s) = \max\{|v_3(1, t) : \psi t \notin H \text{ and } t \in R^a_0[f; s]\} \cup \{v_3(6, s) : \psi' s \notin H\}|\).

For each \(s\) in \(\text{Foe}(\beta, f, r_0)\) let the set of children of \(q_2(5, s)\) be \(\{q_3(1, t) : \beta t \notin H(\psi := \beta) \text{ and } t \in R^a_0[f; s]\} \cup \{q_2(6, s) : \beta' s \notin H(\psi := \beta)\}\), where the tags of these children are:
\(t(q_3(1, t)) = ((\beta, H(\psi := \beta)+\beta t, A(t)), \min, w_3(1, t))\); and \(t(q_3(6, s)) = (- (\beta', H(\psi := \beta)+\beta' s, A(s)), - , w_3(6, s))\).

So \(w_2(5, s) = \max\{|w_2(1, t) : \beta t \notin H(\psi := \beta) \text{ and } t \in R^a_0[f; s]\} \cup \{w_2(6, s) : \beta' s \notin H(\psi := \beta)\}|\).

Take any \(s\) in \(\text{Foe}(\beta, f, r_0)\). We shall show that \(w_2(5, s) = +1\). Observe that \(\text{Foe}(\psi, f, r_0) = \text{Foe}(\beta, f, r_0)\).

Suppose there exists \(t_0\) such that \(\psi t_0 \notin H\) and \(t_0 \in R_0^a[f; s]\) and \(+1 = v_3(1, t_0) = T(\psi, H+\psi t_0, A(t_0))\). Then \(p_2(1, t_0)\) exists. Since \(\psi t_0 \in H\) iff \(\beta t_0 \in H(\psi := \beta)\), we have \(\beta t_0 \notin H(\psi := \beta)\) and so \(q_2(1, t)\) exists. Since \(|T[\psi, H+\psi t_0, A(t_0)|_k \leq k\), then by \(Y(k)\) we have
\(T(\beta, H(\psi := \beta)+\beta t_0, A(t_0)) = w_2(1, t_0) = +1\). Hence \(w_2(5, s) = +1\).

So suppose that such a \(\beta t_0\) does not exist. Then \(v_3(6, s) = +1\) and so \(p_3(6, s)\) exists and \(\psi' s \notin H\). Since \(\psi' s \in H\) iff \(\beta' s \in H(\psi := \beta)\), we have \(\beta' s \notin H(\psi := \beta)\), and so \(q_3(6, s)\) exists.

Let the child of \(p_3(6, s)\) be \(p_4(1, s)\) where \(t(p_4(1, s)) = ((\psi', H+\psi' s, A(s)), \min, v_4(1, s))\).

Then \(+1 = v_2(5, s) = v_4(1, s)\). So \(v_2(1, s) = -1\) and hence \(T(\psi', H+\psi' s, A(s)) = -1\).

Since \(|T[\psi', H+\psi' s, A(s)]|_k \leq k\), then by \(Z(k)\), \(T(\beta', H(\psi := \beta)+\beta' s, A(s)) = -1\).

Let the child of \(q_3(6, s)\) be \(q_4(1, s)\) where \(t(q_4(1, s)) = ((\beta', H(\psi := \beta)+\beta' s, A(s)), \min, w_4(1, s))\). Then \(w_3(6, s) = -w_4(1, s) = -T(\beta', H(\psi := \beta)+\beta' s, A(s)) = +1\). Therefore \(w_2(5, s) = +1\).

Thus for all \(s\) in \(\text{Foe}(\beta, f, r_0)\), \(w_2(5, s) = +1\). By \(^{(*)}\), \(w_1(4, r_0) = +1\) and hence \(w_0 = +1\), as required.

Since the subject of \(p_0\) is \((\psi, H, x)\), \(p_0\) does not satisfy T4, or T5, or T6.

Thus \(Y(k+1)\) is proved.

To prove \(Z(k+1)\) we suppose the antecedent of \(Z(k+1)\) holds and \(|T[\psi', H, x]|_k = k+1\). We must show \(T(\beta', H(\psi := \beta), x) = -1\). Let \(m_0\) be the root of \(T[\psi', H, x]\) and \(n_0\) be the root of \(T[\psi', H(\psi := \beta), x]\). Then \(m_0\) has a child and \(pv(m_0) = T(\psi', H, x) = -1\).

If \(m_0\) satisfies T1 then let \(x\) be \(F\). We see that \(n_0\) also satisfies T1. So \(t(m_0) = ((\psi', H, F), \min, -1)\) and \(t(n_0) = ((\beta', H(\psi := \beta), F), \min, pv(n_0))\), where \(pv(m_0) = T(\beta', H(\psi := \beta), F) \in \{+1, -1\}\). Let \(\{m_f : f \in F\}\) be the set of children of \(m_0\) and \(\{n_f : f \in F\}\) be the set of children of \(n_0\). Let \(F\) be any formula in \(F\). Then the subject of \(m_f\) is \((\psi', H, f)\) and the subject of \(n_f\) is \((\beta', H(\psi := \beta), f)\). Also \(|T[\psi', H, f]|_k \leq k\). There exists \(f_0 \in F\) such at \(pv(m_{f_0}) = -1\) because \(m_0\) is a min node with proof value \(-1\). So by \(Z(k)\), \(pv(n_{f_0}) = T(\beta', H(\psi := \beta), f_0) = -1\). But \(n_0\) is a min node, so \(pv(n_0) = -1\), as required.

Since \(pv(m_0) = T(\psi', H, x) = -1\), \(m_0\) does not satisfy T2.
If \( m_0 \) satisfies T3 then let \( x = f \). We see that \( n_0 \) also satisfies T3. So 
\[
\begin{align*}
t(m_0) &= ((\psi', H, f), \max, -1) \quad \text{and} \quad t(n_0) = ((\beta', H(\psi := \beta), f), \max, \pv(n_0)),
\end{align*}
\]
where \( \pv(n_0) = T(\beta', H(\psi := \beta), f) \in \{+1, -1\} \).

We shall adopt the following naming conventions. Each non-root node of \( T[\psi', H, f] \) is denoted by \( m_l(#, y) \) where \( l \) is the level of the node, \( # \) is the number in \([1, 6]\) such that the node satisfies T#, and \( y \) is a rule, or a formula, or a set, which distinguishes siblings. For non-root nodes in \( T[\beta', H(\psi := \beta), f] \) we shall use \( n_l(#, y) \).

Let the set of children of \( m_0 \) be \( \{m_1(4, r) : \psi' r \notin H \text{ and } r \in R^a[\beta']\} \), where the tags of these children are: 
\[
\begin{align*}
t(m_1(4, r)) &= (((\psi', H, f, r), \min, \pv(m_1(4, r)))).
\end{align*}
\]
Recall that \( m_0 \) has at least one child. So 
\[
\begin{align*}
-1 &= \max\{\pv(m_1(4, r)) : \psi' r \notin H \text{ and } r \in R^a[\beta']\}. \quad \text{Hence if } \psi' r \notin H \text{ and } r \in R^a[\beta'] \text{ then } \pv(m_1(4, r)) = -1.
\end{align*}
\]

Let the set of children of \( n_0 \) be \( \{n_1(4, r) : \beta' r \notin H(\psi := \beta) \text{ and } r \in R^a[\beta']\} \), where the tags of these children are: 
\[
\begin{align*}
t(n_1(4, r)) &= (((\beta', H(\psi := \beta), f, r), \min, \pv(n_1(4, r)))).
\end{align*}
\]
Let \( \pv(n_0) = \max\{\pv(n_1(4, r)) : \beta' r \notin H(\psi := \beta) \text{ and } r \in R^a[\beta']\} \). If \( n_0 \) does not have a child then \( \pv(n_0) = \max\{\} = -1 \), as required. So suppose that \( n_0 \) has at least one child.

If \( \psi' r \notin H \text{ and } r \in R^a[\beta'] \) then let the set of children of \( m_1(4, r) \) be \( \{m_2(1, r) \cup \{m_2(5, s) : s \in R[\beta'; r]\}\} \), where the tags of these children are: 
\[
\begin{align*}
t(m_2(1, r)) &= (((\psi', H + \psi' r, A(r)), \min, \pv(m_2(1, r)))) \quad \text{and}
\end{align*}
\]
\[
\begin{align*}
t(m_2(5, s)) &= (((\psi', H(\psi := \beta), f, r, s), \max, \pv(m_2(5, s)))).
\end{align*}
\]
So \( \pv(n_1(4, r)) = \min\{\pv(m_2(1, r)) \cup \pv(m_2(5, s)) : s \in R[\beta'; r]\} \). Therefore either \( \pv(m_2(1, r)) = -1 \) or there exists \( s_0 \) in \( R[\beta'; r] \) such that \( \pv(m_2(5, s_0)) = -1 \).

If \( \beta' r \notin H(\psi := \beta) \text{ and } r \in R^a[\beta'] \) then let the set of children of \( n_1(4, r) \) be \( \{n_2(1, r) \cup \{n_2(5, s) : s \in \For(\beta', f, r)\} \), where the tags of these children are: 
\[
\begin{align*}
t(n_2(1, r)) &= (((\beta', H(\psi := \beta) + \beta' r, A(r)), \min, \pv(n_2(1, r)))) \quad \text{and}
\end{align*}
\]
\[
\begin{align*}
t(n_2(5, s)) &= (((\beta', H(\psi := \beta), f, r, s), \max, \pv(n_2(5, s)))).
\end{align*}
\]
So \( \pv(n_1(4, r)) = \min\{\pv(n_2(1, r)) \cup \pv(n_2(5, s)) : s \in \For(\beta', f, r)\} \).

We show that for each \( r \) in \( R^a[\beta'] \) such that \( \beta' r \notin H(\psi := \beta) \), \( \pv(n_1(4, r)) = -1 \).

We have \( |T[\psi', H + \psi' r, A(r)]| \leq k \). If \( -1 = \pv(n_2(1, r)) = T[\psi', H + \psi' r, A(r)] \) then by \( Z(k) \), \( \pv(n_2(1, r)) = T(\beta', H(\psi := \beta) + \beta' r, A(r)) = -1 \). Hence \( \pv(n_1(4, r)) = -1 \).

So suppose there exists \( s_0 \) in \( R[\beta'; r] \) such that \( T[\psi', H, f, r, s_0] = \pv(n_2(5, s_0)) = -1 \). Then \( s_0 \in \For(\beta', f, r) \). We shall show that \( T(\beta', H(\psi := \beta), f, r, s_0) = \pv(n_2(5, s_0)) = -1 \), and hence that \( \pv(n_1(4, r)) = -1 \).

Let the set of children of \( m_2(5, s_0) \) be 
\[
\begin{align*}
\{m_3(1, t) : \psi' t \notin H \text{ and } t \in R^a[\beta'; s_0]\} \cup \{m_3(6, s_0) : \psi' s_0 \notin H\},
\end{align*}
\]
where the tags of these children are: 
\[
\begin{align*}
t(m_3(1, t)) &= (((\psi', H + \psi' t, A(t)), \min, \pv(m_3(1, t)))) \quad \text{and}
\end{align*}
\]
\[
\begin{align*}
t(m_3(6, s_0)) &= (((-\psi, H + \psi' s_0, A(s_0)), -, \pv(m_3(6, s_0)))).
\end{align*}
\]
So \( -1 = \pv(m_3(2, s_0)) = \max\{\pv(m_3(1, t)) : \psi' t \notin H \text{ and } t \in R^a[\beta'; s_0]\} \cup \pv(m_3(6, s_0)) \). Hence for all \( t \) in \( R^a[\beta'; s_0] \) such that \( \psi' t \notin H \) we have 
\[
\begin{align*}
-1 &= \pv(m_3(1, t)) = T(\psi', H + \psi' t, A(t)). \quad \text{Also if } \psi s_0 \notin H \text{ then }
\end{align*}
\]
\[
\begin{align*}
-1 &= \pv(m_3(6, s_0)) = T(-\psi, H + \psi' s_0, A(s_0)).
\end{align*}
\]
Let the set of children of \( n_2(5, s_0) \) be 
\[
\begin{align*}
\{n_3(1, t) : \beta' t \notin H(\psi := \beta) \text{ and } t \in R^a[\beta' s_0]\} \cup \{n_3(6, s_0) : \beta s_0 \notin H(\psi := \beta)\},
\end{align*}
\]
where the tags of these children are: 
\[
\begin{align*}
t(n_3(1, t)) &= (((\beta', H(\psi := \beta) + \beta' t, A(t)), \min, \pv(n_3(1, t)))) \quad \text{and}
\end{align*}
\]
\[
\begin{align*}
t(n_3(6, s_0)) &= (((-\beta, H(\psi := \beta) + \beta s_0, A(s_0)), -, \pv(n_3(6, s_0)))).
\end{align*}
\]
So \( \pv(n_2(5, s_0)) = \max\{\pv(n_3(1, t)) : \beta' t \notin H(\psi := \beta) \text{ and } t \in R^a[\beta'; s_0]\} \cup \\max\{\pv(n_3(6, s_0)) : \beta s_0 \notin H(\psi := \beta)\} \).
\{pv(n_3(6, s_0)) : \beta s_0 \notin H(\psi := \beta)\}. If n_2(5, s_0) has no children then

\[pv(n_2(5, s_0)) = \max\{\} = -1, \text{ as required. So suppose that } n_2(5, s_0) \text{ has at least one child.}\]

Case 1: \(n_3(1, t)\) is a child of \(n_2(5, s_0)\).

Then \(\beta t \notin H(\psi := \beta)\) and \(t \in R_{\psi}^n[f; s_0]\). Hence \(\psi t \notin H\). So from above,

\[-1 = pv(m_3(1, t)) = T(\psi', H + \psi t, A(t)). \text{ Also } |T[\psi', H + \psi t, A(t)]| \leq k. \text{ So by } Z(k),\]

\[-1 = T(\beta', H(\psi := \beta) + \beta t, A(t)) = pv(t(n_3(1, t))).\]

Case 2: \(n_3(6, s_0)\) is a child of \(n_2(5, s_0)\).

Then \(\beta s_0 \notin H(\psi := \beta)\). Hence \(\psi s_0 \notin H\). So from above,

\[-1 = pv(m_3(6, s_0)) = T(-\psi, H + \psi s_0, A(s_0))). \text{ Therefore } T(\psi, H + \psi s_0, A(s_0)) = +1. \text{ Also } |T[\psi, H + \psi s_0, A(s_0)]| \leq k. \text{ So by } Y(k),\]

\[-1 = T(\beta, H(\psi := \beta) + \beta s_0, A(s_0)) = - + 1 = -1.\]

These two cases show that \(pv(n_2(5, s_0)) = -1, \text{ as required. Hence } pv(n_1(4, r)) = -1.\) Therefore \(pv(n_0) = -1\). Thus \(Z(k+1)\) is proved.

Therefore \(Y(k)\) and \(Z(k)\) are proved by induction, and so the lemma is proved.

**EndProofLemC.5**

**Definition C.6.** Suppose \(\{\alpha, \gamma, \lambda\} \subseteq Alg, H\) is a \(\alpha\)-history, and \(T\) is an evaluation tree of some plausible theory. If \(\alpha \notin \{\gamma, \gamma', \lambda, \lambda'\}\) then define \(\alpha(\gamma, \lambda) = \alpha\); else define \(\gamma(\gamma, \lambda) = \lambda, \gamma'(\gamma, \lambda) = \lambda', \lambda(\gamma, \lambda) = \gamma, \text{ and } \lambda'(\gamma, \lambda) = \gamma'\).

If \(H = (\alpha_1 r_1, ..., \alpha_n r_n)\) then define \(H(\gamma, \lambda) = (\alpha_1(\gamma, \lambda)r_1, ..., \alpha_n(\gamma, \lambda)r_n)\).

Define \(T(\gamma, \lambda)\) to be the tree formed from \(T\) by only changing the subject of each node as follows. For each node \(p\) of \(T\) replace \(alg(p)\) by \(alg(p)(\gamma : \lambda)\), and replace \(Hist(p)\) by \(Hist(p)(\gamma : \lambda)\).

**Lemma C.7.** Suppose \(P = (R, >)\) is a plausible theory such that \(>\) is empty. If \(T\) is an evaluation tree of \(P\) then \(T(\pi : \psi)\) is an evaluation tree of \(P\).

Hence \(P(\psi) = P(\pi)\) and \(P(\pi') = P(\psi')\).

**Proof**

Let \(P = (R, >)\) be a plausible theory such that \(>\) is empty. Then \(Foe(\psi', f, r) = \{\} = Foe(\pi', f, r)\) and \(Foe(\pi, f, r) = R[\neg f] = Foe(\psi, f, r)\). Let \(Y(n)\) denote the following conditional statement.

“If \(T\) is an evaluation tree of \(P\) and \(|T| \leq n\) then \(T(\pi : \psi)\) is an evaluation tree of \(P\).” By Definition 5.14, it suffices to prove \(Y(n)\) by induction on \(n\).

Suppose \(n = 1\) and the antecedent of \(Y(1)\) holds. Let the root of \(T\) be \(p_0\) and \(alg(p_0) = \alpha\). If \(\alpha \notin \{\pi, \psi, \psi', \pi'\}\) then \(T(\pi : \psi) = T\) and so \(Y(1)\) holds. So suppose \(\alpha \in \{\pi, \psi, \psi', \pi'\}\). Let the root of \(T(\pi : \psi)\) be \(q_0\), and let \(alg(q_0) = \lambda\). So \(\alpha(\pi : \psi) = \lambda\).

Since \(p_0\) has no children, \(q_0\) has no children.

If \(p_0\) satisfies \(T1\) then let \(Subj(p_0) = (\alpha, H, \{\})\). Hence \(Subj(q_0) = (\lambda, H(\pi : \psi), \{\})\) and so \(q_0\) satisfies \(T1\). Thus \(T(\pi : \psi)\) is an evaluation tree of \(P\).

If \(p_0\) satisfies \(T2\) then let \(Subj(p_0) = (\alpha, H, f)\). Hence \(Subj(q_0) = (\lambda, H(\pi : \psi), f)\) and so \(q_0\) satisfies \(T2\). Thus \(T(\pi : \psi)\) is an evaluation tree of \(P\).

If \(p_0\) satisfies \(T3\) then let \(Subj(p_0) = (\alpha, H, f)\). Hence \(Subj(q_0) = (\lambda, H(\pi : \psi), f)\). Since \(p_0\) has no children, \(S(q_0) = \{\}\). So if \(r \in R_{\pi}^2[f]\) then \(\alpha r \in H\). Now \(\alpha r \in H\) if \(\lambda r \in H(\pi : \psi)\). Hence \(S(q_0) = \{\}\) and so \(q_0\) satisfies \(T3\). Thus \(T(\pi : \psi)\) is an evaluation tree of \(P\).

Since \(p_0\) and \(q_0\) have no children, \(p_0\) and \(q_0\) satisfy neither \(T4\) nor \(T6\).

If \(p_0\) satisfies \(T5\) then let \(Subj(p_0) = (\alpha, H, f, r, s)\). Hence \(Subj(q_0) = (\lambda, H(\pi : \psi), f, r, s)\).
Since \( p_0 \) has no children, \( S(p_0) = \{ \} \). Hence if \( t \in R^*_a[f,s] \) then \( \alpha t \in H \). Now \( \alpha t \in H \) iff \( \lambda t \in H(\pi;\psi) \). Also \( \alpha s \in H \). Hence \( \lambda s \in H(\pi;\psi) \). So \( S(q_0) = \{ \} \). Thus \( q_0 \) satisfies T5 and so \( T(\pi;\psi) \) is an evaluation tree of \( P \).

All cases have been considered and so \( Y(1) \) holds.

If \( T \) is any tree and \( p \) is any node of \( T \) for which \( \text{Subj}(p) \) is defined, then define the set \( S(p,T) \) of subjects of the children of \( p \) in \( T \) by \( S(p,T) = \{ \text{Subj}(c) : c \text{ is a child of } p \text{ in } T \} \).

Take any integer \( n \) such that \( n \geq 1 \). Suppose that \( Y(n) \) is true. We shall prove \( Y(n+1) \). Suppose the antecedent of \( Y(n+1) \) holds and that \( |T| = n+1 \). Let the root of \( T \) be \( p_0 \) and \( \text{alg}(p_0) = \alpha \). If \( \alpha \notin \{ \pi,\psi,\psi',\pi' \} \) then \( T(\pi;\psi) = T \) and so \( Y(n+1) \) holds. So suppose \( \alpha \notin \{ \pi,\psi,\psi',\pi' \} \). Let the root of \( T(\pi;\psi) \) be \( q_0 \), and let \( \text{alg}(q_0) = \lambda \). So \( \alpha(\pi;\psi) = \lambda \).

If \( p_0 \) satisfies T1 then let \( \text{Subj}(p_0) = (\alpha,H,F) \). Hence \( \text{Subj}(q_0) = (\lambda,H(\pi;\psi),F) \). So \( q_0 \) satisfies T1. Recall that \( S(p_0,T[\alpha,H,F]) = \{ (\alpha,H,f) : f \in F \} \). So \( S(q_0,T[\alpha,H,F](\pi;\psi)) = \{ (\lambda,H(\pi;\psi),f) : f \in F \} = S(q_0,T[\lambda,H(\pi;\psi),F]) \), by T1. But for each \( (\alpha,H,f) \) in \( S(p_0,T[\alpha,H,F]) \), \( T[\alpha,H,f] \) is an evaluation tree of \( P \) and \( |T[\alpha,H,f]| \leq n \). So by \( Y(n) \), \( T[\alpha,H,f](\pi;\psi) \) is an evaluation tree of \( P \). Hence \( T[\alpha,H,f](\pi;\psi) = T[\lambda,H(\pi;\psi),f] \). Thus \( T(\pi;\psi) = T[\alpha,H,F](\pi;\psi) = T[\lambda,H(\pi;\psi),F] \) which is an evaluation tree of \( P \).

Since \( p_0 \) has a child, \( p_0 \) does not satisfy T2.

If \( p_0 \) satisfies T3 then let \( \text{Subj}(p_0) = (\alpha,H,f) \). Hence \( \text{Subj}(q_0) = (\lambda,H(\pi;\psi),f) \). So \( q_0 \) satisfies T3. Recall that \( S(p_0,T[\alpha,H,f]) = \{ (\alpha,H,f) : \alpha f \notin H \} \). Since \( \alpha f \in H \) iff \( \alpha r \in H(\pi;\psi) \) we have \( \alpha f \notin H \) iff \( \alpha f \notin H(\pi;\psi) \). So \( S(q_0,T[\alpha,H,f](\pi;\psi)) = \{ (\lambda,H(\pi;\psi),f) : f \in F \} = \{ (\alpha,H(\pi;\psi),f) : f \in F \} \), by T3. But for each \( (\alpha,H,f) \) in \( S(p_0,T[\alpha,H,f]) \), \( T[\alpha,H,f](\pi;\psi) \) is an evaluation tree of \( P \) and \( |T[\alpha,H,f]| \leq n \). So by \( Y(n) \), \( T[\alpha,H,f](\pi;\psi) \) is an evaluation tree of \( P \). Hence \( T[\lambda,H(\pi;\psi),f] \) which is an evaluation tree of \( P \).

If \( p_0 \) satisfies T4 then let \( \text{Subj}(p_0) = (\alpha,H,f,r) \). Hence \( \text{Subj}(q_0) = (\lambda,H(\pi;\psi),f,r) \). Since \( \alpha f \in H \) iff \( \alpha r \in H(\pi;\psi) \) we have \( \alpha f \notin H \) iff \( \alpha r \notin H(\pi;\psi) \). So \( q_0 \) satisfies T4. Recall that \( S(p_0,T[\alpha,H,f,r]) = \{ (\alpha,H,f) : f \in F \} \cup \{ (\alpha,H,f,r,s) : s \in \text{Foe}(\alpha,f,r) \} \). Since \( \text{Foe}(\alpha,f,r) = \text{Foe}(\lambda,f,r) \), \( S(q_0,T[\alpha,H,f,r](\pi;\psi)) = \{ (\lambda,H(\pi;\psi),\lambda r) : r \in A(\alpha) \} \cup \{ (\lambda,H(\pi;\psi),f,r,s) : s \in \text{Foe}(\alpha,f,r) \} \), by T4. But \( T[\alpha,H,f,r](\pi;\psi) \) is an evaluation tree of \( P \) and \( |T[\alpha,H,f,r]| \leq n \). So by \( Y(n) \), \( T[\alpha,H,f,r](\pi;\psi) \) is an evaluation tree of \( P \). Hence \( T[\alpha,H,f,r,s](\pi;\psi) = T[\lambda,H(\pi;\psi),f,r,s] \) which is an evaluation tree of \( P \).

If \( p_0 \) satisfies T5 then let \( \text{Subj}(p_0) = (\alpha,H,f,r,s) \). Then \( \alpha \neq \pi' \). Since \( s \in \text{Foe}(\alpha,f,r), \text{Foe}(\alpha,f,r) \neq \{ \} \) and so \( \alpha \neq \psi' \). Therefore \( \alpha \in \{ \pi,\psi \} \) and so \( \lambda \in \{ \pi,\psi \} \). Now \( \text{Subj}(q_0) = (\lambda,H(\pi;\psi),f,r,s) \). Since \( \alpha f \in H \) iff \( \alpha r \in H(\pi;\psi) \) we have \( \alpha f \notin H \) iff \( \alpha r \notin H(\pi;\psi) \). So \( q_0 \) satisfies T5. Recall that \( S(p_0,T[\alpha,H,f,r,s]) = \{ (\alpha,H,f) : f \in F \} \cup \{ (\alpha,H,f,r,s) : s \in \text{Foe}(\alpha,f,r,s) \} \).

BDI
= \{ (\lambda, H(\pi: \psi) + \lambda t, A(t)) : \alpha t \notin H \text{ and } t \in R_0^\pi[f; s] \} \cup \{ - (\lambda', H(\pi: \psi) + \lambda' s, A(s)) : \alpha' s \notin H \} \\
= \{ (\lambda, H(\pi: \psi) + \lambda t, A(t)) : \lambda t \notin H(\pi: \psi) \text{ and } t \in R_0^\pi[f; s] \} \cup \\
\{ - (\lambda', H(\pi: \psi) + \lambda' s, A(s)) : \lambda' s \notin H(\pi: \psi) \} \\
= S(q_0, T[\lambda, H(\pi: \psi), f, r, s]), \text{ by T5.} \\
\]

But for each $\alpha$, $H + \alpha t, A(t))$ in $S(p_0, T[\alpha, H, f, r, s]), T[\alpha, H + \alpha t, A(t)]$ is an evaluation tree of $\mathcal{P}$ and $|T[\alpha, H + \alpha t, A(t)]| \leq n$. So by $Y(n)$, $T[\alpha, H + \alpha t, A(t)](\pi: \psi)$ is an evaluation tree of $\mathcal{P}$. Hence $T[\alpha, H + \alpha t, A(t)](\pi: \psi) = T[\lambda, H(\pi: \psi) + \lambda t, A(t)].$ Also if 

$-(\alpha', H + \alpha' s, A(s)) \in S(p_0, T[\alpha, H, f, r, s]),$ then $T[-(\alpha', H + \alpha' s, A(s))]$ is an evaluation tree of $\mathcal{P}$ and $|T[-(\alpha', H + \alpha' s, A(s))]| \leq n$. So by $Y(n)$, $T[-(\alpha', H + \alpha' s, A(s))]((\pi: \psi)$ is an evaluation tree of $\mathcal{P}$. Hence $T[-(\alpha', H + \alpha' s, A(s))](\pi: \psi) = T[\lambda', H(\pi: \psi) + \lambda' s, A(s)].$

Thus $T(\pi: \psi) = T[\alpha, H, f, r, s](\pi: \psi) = T[\lambda, H(\pi: \psi), f, r, s]$ which is an evaluation tree of $\mathcal{P}$.

If $p_0$ satisfies T6 then let $\text{Subj}(p_0) = -(\alpha', H, F)$. Then $\alpha \in \{ \pi, \psi \}$ and so $\lambda \in \{ \pi, \psi \}$. Hence $\text{Subj}(q_0) = - (\lambda', H(\pi: \psi), F)$. So $q_0$ satisfies T6. Recall that $S(p_0, T[-(\alpha', H, F)]) = \{ (\alpha', H, F) \}$. So $S(q_0, T[\lambda', H(\pi: \psi), F])) = \{ (\lambda', H(\pi: \psi), F) \} = S(q_0, T[\lambda', H(\pi: \psi), F]),$ by T6. But $T[\lambda', H, F]$ is an evaluation tree of $\mathcal{P}$ and $|T[\lambda', H, F]| \leq n$. So by $Y(n)$, $T[\lambda', H, F](\pi: \psi)$ is an evaluation tree of $\mathcal{P}$. Hence 

$T[\lambda', H, F](\pi: \psi) = T[\lambda', H(\pi: \psi), F]$. Thus $T(\pi: \psi) = T[\lambda', H(\pi: \psi), F]$ which is an evaluation tree of $\mathcal{P}$.

Therefore $Y(n + 1)$, and hence the lemma, is proved by induction.

EndProofLemC.7

Theorem C.8 (Theorem 7.6 The proof algorithm hierarchy). Suppose $\mathcal{P} = (R, >)$ is a plausible theory.

1) $\mathcal{P}(\varphi) \subseteq \mathcal{P}(\pi) \subseteq \mathcal{P}(\psi) \subseteq \mathcal{P}(\beta) = \mathcal{P}(\beta') \subseteq \mathcal{P}(\psi') \subseteq \mathcal{P}(\pi').$

2) If $>$ is empty then $\mathcal{P}(\varphi) \subseteq \mathcal{P}(\pi) = \mathcal{P}(\psi) \subseteq \mathcal{P}(\beta) = \mathcal{P}(\beta') \subseteq \mathcal{P}(\psi') = \mathcal{P}(\pi').$

Proof
Suppose $\mathcal{P} = (R, >)$ is a plausible theory. By Lemma C.1, $\mathcal{P}(\varphi) \subseteq \mathcal{P}(\pi)$. By Lemma C.3, $\mathcal{P}(\pi) \subseteq \mathcal{P}(\psi)$. By Lemma C.5(1), $\mathcal{P}(\psi) \subseteq \mathcal{P}(\beta)$. By Lemma B.8, $\mathcal{P}(\beta) = \mathcal{P}(\beta')$. By Lemma C.5(2), $\mathcal{P}(\beta') \subseteq \mathcal{P}(\psi')$. By Lemma B.4, $\mathcal{P}(\psi') \subseteq \mathcal{P}(\pi')$. So part (1) holds.

Part (2) holds by part (1) and Lemma C.7.

EndProofThmC.8

References

Adams, E. W. (1975). The Logic of Conditionals: An Application of Probability to Deductive Logic. D. Reidel Publishing Co., Dordrecht, Holland.

Alchourrón, C. E., Gärdenfors, P., & Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. Journal of Symbolic Logic, 50(2), 510–530.

Antoniou, G. (1997). Nonmonotonic Reasoning. MIT Press.

Arlo-Costa, H., & Egré, P. (2016). The logic of conditionals. In Zalta, E. N. (Ed.), The Stanford Encyclopedia of Philosophy (Winter 2016 edition). Metaphysics Research Lab, Stanford University, https://plato.stanford.edu/archives/win2016/entries/logic-conditionals/.
Baral, C. (2003). Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press.

Benferhat, S., Dubois, D., & Prade, H. (1992). Representing default rules in possibilistic logic. In Nebel, B., Rich, C., & Swartout, W. (Eds.), Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning (KR92), pp. 673–684. Morgan Kaufmann Publishers, Inc.

Billington, D. (2011). A defeasible logic for clauses. In Wang, D., & Reynolds, M. (Eds.), AI 2011: Advances in Artificial Intelligence 24th Australasian Joint Conference Perth, Australia, December 5-8, 2011 Proceedings, Vol. 7106 of Lecture Notes in Artificial Intelligence, pp. 472–480. Springer.

Billington, D., Antoniou, G., Governatori, G., & Maher, M. J. (2010). An inclusion theorem for defeasible logics. ACM Transactions on Computational Logic, 12(1), Article 6.

Billington, D., & Rock, A. (2001). Propositional plausible logic: Introduction and implementation. Studia Logica, 67(2), 243–269.

Billington, D. (1993). Defeasible logic is stable. Journal of Logic and Computation, 3(4), 379–400.

Billington, D. (2008). Propositional clausal defeasible logic. In Holldobler, S., Lutz, C., & Wansing, H. (Eds.), Logics in Artificial Intelligence, Vol. 5293 of Lecture Notes in Artificial Intelligence, pp. 34–47, Dresden, Germany. 11th European Conference on Logics in Artificial Intelligence (JELIA2008), Springer.

Billington, D. (2015). A propositional plausible logic. In Pfahringer, B., & Renz, J. (Eds.), AI 2015: Advances in Artificial Intelligence, Vol. 9457 of Lecture Notes in Artificial Intelligence, pp. 76–82. Springer.

Boutilier, C. (1994a). Conditional logics of normality: A modal approach. Artificial Intelligence, 68(1), 87–154.

Boutilier, C. (1994b). Unifying default reasoning and belief revision in a modal framework. Artificial Intelligence, 68(1), 33–85.

Brewka, G. (1989). Preferred subtheories: An extended logical framework for default reasoning. In Sridharan, N. S. (Ed.), Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI89), Vol. 2, pp. 1043–1048. Morgan Kaufmann Publishers, Inc.

Burgess, J. P. (1981). Quick completeness proofs for some logics of conditionals. Notre Dame Journal of Formal Logic, 22(1).

Caminada, M., & Amgoud, L. (2007). On the evaluation of argumentation formalisms. Artificial Intelligence, 171, 286–310.

Crocco, G., & Lamarre, P. (1992). On the connection between non-monotonic inference systems and conditional logics. In Nebel, B., Rich, C., & Swartout, W. (Eds.), Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning (KR92), pp. 565–571. Morgan Kaufmann Publishers, Inc.

Delgrande, J. P. (2007). On a rule-based interpretation of default conditionals. Annals of Mathematics and Artificial Intelligence, 48, 135–167.
Dubois, D., Lang, J., & Prade, H. (1994). Possibilistic logic. In Gabbay, D. M., Hogger, C., & Robinson, J. (Eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3 Nonmonotonic Reasoning and Uncertain Reasoning, pp. 439–513. Oxford Science Publications.

Dubois, D., & Prade, H. (1991). Conditional objects and non-monotonic reasoning. In Allen, J., Fikes, R., & Sandewall, E. (Eds.), Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR91), pp. 175–185. Morgan Kaufmann Publishers, Inc.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming, and n-person games. Artificial Intelligence, 77(2), 321–357.

Garcia, A. J., & Simari, G. R. (2004). Defeasible logic programming: an argumentative approach. Theory and Practice of Logic Programming, 4(1,2), 95–138.

Gärdenfors, P. (1988). Knowledge in Flux. Modeling the Dynamics of Epistemic States. MIT Press.

Geerts, P., Laenens, E., & Vermier, D. (1998). Defeasible logics. In Gabbay, D. M., & Smets, P. (Eds.), Handbook of Defeasible Reasoning and Uncertainty Management Systems, Vol. 2, pp. 175–210. Kluwer Academic.

Geerts, P., Vermeir, D., & Nute, D. (1994). Ordered logic: defeasible reasoning for multiple agents. Decision Support Systems, 11, 157–190.

Geffner, H., & Pearl, J. (1992). Conditional entailment: bridging two approaches to default reasoning. Artificial Intelligence, 53, 209–244.

Goldszmidt, M., & Pearl, J. (1991). System-z+: A formalism for reasoning with variable-strength defaults. In Proceedings AAAI-91, pp. 399–404.

Hawthorne, J., & Makinson, D. (2007). The quantitative/qualitative watershed for rules of uncertain inference. Studia Logica, 86, 247–297.

Horty, J. F., Thomason, R. H., & Touretzky, D. S. (1990). A skeptical theory of inheritance in nonmonotonic semantic networks. Artificial Intelligence, 42, 311–348.

Kraus, S., Lehmann, D., & Magidor, M. (1990). Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence, 44(1-2), 167–207.

Kyburg, H. E. (1961). Probability and the Logic of Rational Belief. Wesleyan University Press, Middletown, CT.

Lamarre, P. (1991). S4 as the conditional logic of nonmonotonicity. In Allen, J., Fikes, R., & Sandewall, E. (Eds.), Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR91), pp. 357–367. Morgan Kaufmann Publishers, Inc.

Lehmann, D., & Magidor, M. (1992). What does a conditional knowledge base entail?. Artificial Intelligence, 55(1), 1–60.

Maier, F., & Nute, D. (2006). Ambiguity propagating defeasible logic and the well-founded semantics. In 10th European Conference on Logics in Artificial Intelli-
Makinson, D. (1988). General theory of cumulative inference. In Proceedings of the Second International Workshop on Non-Monotonic Reasoning, Vol. 346 of Lecture Notes in Artificial Intelligence, pp. 1–18. Springer-Verlag.

Makinson, D., & Hawthorne, J. (2014). Lossy inference rules and their bounds: a brief review. In Loslow, A., & Buchsbaum, A. (Eds.), The Road to Universal Logic, Vol. 1, pp. 385–408. Springer.

Modgil, S., & Prakken, H. (2013). A general account of argumentation with preferences. Artificial Intelligence, 195, 361–397.

Nerode, A., & Shore, R. A. (1997). Logic for Applications (2nd edition). No. ISBN 0-387-94893-7 in Graduate Texts in Computer Science. Springer.

Nute, D., & Cross, C. B. (2001). Conditional logic. In Gabbay, D. M., & Guenthner, F. (Eds.), Handbook of Philosophical Logic (2nd edition), Vol. 4, pp. 1–98. Kluwer Academic Publishers.

Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers, Inc.

Pearl, J. (1990). System z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning. In Parikh, R. (Ed.), Theoretical Aspects of Reasoning about Knowledge (TARK-III), pp. 121–135. Morgan Kaufmann Publishers, Inc.

Poole, D. (1988). A logical framework for default reasoning. Artificial Intelligence, 36, 27–47.

Prakken, H., & Sartor, G. (1997). Argument-based extended logic programming with defeasible priorities. Journal of Applied Non-Classical Logics, 7(1-2), 25–75.

Reiter, R. (1980). A logic for default reasoning. Artificial Intelligence, 13, 81–132.

Simari, G. R., & Loui, R. P. (1992). A mathematical treatment of defeasible reasoning and its implementation. Artificial Intelligence, 53, 125–157.

Touretzky, D., Horty, J., & Thomason, R. (1987). A clash of intuitions: The current state of nonmonotonic multiple inheritance systems. In McDermott, J. P. (Ed.), Proceedings of the 10th International Joint Conference on Artificial Intelligence, Vol. 1, pp. 476–482. Morgan Kaufmann Publishers, Inc.

Walton, D., Tindale, C., & Gordon, T. (2014). Applying recent argumentation methods to some ancient examples of plausible reasoning. Argumentation, 28, 85–119.