Artificial Intelligence 57 (1992) 1–42
Elsevier

How to reason defeasibly

John L. Pollock
Department of Philosophy, University of Arizona, Tucson, AZ 85721, USA

Received January 1991
Revised February 1992

Abstract

Pollock, J.L., How to reason defeasibly, Artificial Intelligence 57 (1992) 1–42.

This paper describes the construction of a general-purpose defeasible reasoner that is complete for first-order logic and provably adequate for the argument-based conception of defeasible reasoning that I have developed elsewhere. Because the set of warranted conclusions for a defeasible reasoner will not generally be recursively enumerable, a defeasible reasoner based upon a rich logic like the predicate calculus cannot function like a traditional theorem prover and simply enumerate the warranted conclusions. An alternative criterion of adequacy called *i.d.e.-adequacy* is formulated. This criterion takes seriously the idea that defeasible reasoning may involve indefinitely many cycles of retracting and reinstating conclusions. It is shown how to construct a reasoner that, subject to certain realistic assumptions, is provably i.d.e.-adequate. The most recent version of OSCAR implements this system, and examples are given of OSCAR's operation.

1. Introduction

The aim of the OSCAR project is the construction of a general theory of rationality and its implementation in a computer system, thus producing an artificial rational agent named "OSCAR". This is of central relevance to AI. At least for most purposes, a general constraint on AI systems should be that they draw conclusions and make decisions that we, as human beings, regard as rational. As long as their reasoning remains simple, we can evaluate the rationality of AI systems just by using our intuitions about rationality and without appeal to a general theory of rationality. But as the reasoning of AI systems gets more complicated and the objectives of the systems more ambitious, the reasoning outstrips our uninformed intuitions. Eventually, complex AI systems are going to have to be developed against the background of a theory of rationality.

Correspondence to: J.L. Pollock, Department of Philosophy, University of Arizona, Tucson, AZ 85721, USA. Telephone: (602) 621-3129. E-mail: pollock@ccit.arizona.edu.

This paper addresses an aspect of rationality that is of central importance to much current work in AI—defeasible (nonmonotonic) reasoning. Most reasoning is *defeasible*, in the sense that it can lead not only to the adoption of new conclusions but also to the retraction of previously adopted conclusions. The structure of defeasible reasoning has been a topic of interest in both philosophy and AI, although until recently researchers in the two fields were largely oblivious of each other.[1] The purpose of this paper is not to construct a new theory of defeasible reasoning, but to ask a new question about an old theory—the theory that can probably be regarded as the dominant model in philosophy for the last fifteen years. The question is, how can an automated reasoner be constructed so that it can, in some reasonable sense, be regarded as an implementation of the theory?

Although the general form of the philosophical theory of defeasible reasoning has remained constant since the early 1970s, it has recently undergone considerable refinement in detail, motivated largely by its encounter with AI. These refinements are recounted in [27, 28, 31, 32]. Rather than repeat all of the details and arguments here, I will just give enough of a sketch of the theory to enable the reader to understand its structure, and refer the reader to those earlier publications for the details and for the defense of the theory.

1.1. *Reasons and arguments*

Defeasible reasoning is, *a fortiori*, reasoning. Reasoning proceeds by constructing arguments, where *reasons* provide the atomic links in arguments. *Conclusive reasons* logically entail their conclusions. Defeasibility arises from the fact that not all reasons are conclusive. Those that are not are *prima facie reasons*. Prima facie reasons create a presumption in favor of their conclusion, but it is defeasible. To illustrate, we often judge the colors of objects on the basis of what color they look to us, taking apparent color to provide a prima facie reason for a judgment about actual color. Thus, for example, $\ulcorner x$ looks red to me$\urcorner$ is a prima facie reason for me to believe $\ulcorner x$ is red$\urcorner$. Similarly, $\ulcorner$Most A's are B's, and this is an $A\urcorner$ is a prima facie reason for $\ulcorner$This is a $B\urcorner$. I will represent a reason as an ordered pair $\langle \Gamma, p\rangle$, where Γ is the set of premises of the reason and p is the conclusion. The simplest kind of defeater for a prima facie reason $\langle \Gamma, p\rangle$ is a reason for denying the conclusion. Let us define "$\neg$" as follows: if for some θ, $\varphi = \ulcorner\sim\theta\urcorner$, let $\neg\varphi = \theta$, and let $\neg\varphi = \ulcorner\sim\varphi\urcorner$ otherwise. Then we have:

Definition 1.1. If $\langle \Gamma, p\rangle$ is a prima facie reason, $\langle \Lambda, q\rangle$ is a *rebutting defeater* for $\langle \Gamma, p\rangle$ iff $\langle \Lambda, q\rangle$ is a reason and $q = \ulcorner\neg p\urcorner$.

[1] The work on defeasible reasoning in philosophy stems mainly from the publications of Roderick Chisholm and myself. See [2–4, 21–24, 26, 27, 29]. See also Kyburg's [12, 13].

For instance, if something looks red to me, that gives me a prima facie reason for thinking it is red; but if Jones, whom I regard as reliable, insists that it is not red, that gives me a rebutting defeater. There are also defeaters that are not rebutting defeaters. They attack a prima facie reason without attacking its conclusion. They accomplish this by instead attacking the connection between the premises and the conclusion. They do this by giving us a reason for denying that the premises wouldn't be true unless the conclusion were true. Suppose again that I judge an object to be red on the basis of its looking red to me. If I subsequently learn that the object is illuminated by red lights and such illumination can make things look red when they are not, it is unreasonable for me to maintain my belief that it is red. In other words, this is a defeater. But it is not a reason for denying that the object is red, so it is not a rebutting defeater. It is instead a reason for denying that the object would not look red unless it were red. Symbolizing ⌜it is false that P wouldn't be true unless Q were true⌝ as $\ulcorner P \otimes Q \urcorner$, and letting $\Pi\Gamma$ be the conjunction of the members of a set Γ, we can define:

Definition 1.2. If $\langle \Gamma, p \rangle$ is a prima facie reason, $\langle \Lambda, q \rangle$ is an *undercutting defeater* for $\langle \Gamma, p \rangle$ iff $\langle \Lambda, q \rangle$ is a reason and $q = \ulcorner(\Pi\Gamma \otimes p)\urcorner$.

It is of interest to inquire about the logical properties of "⊗", but that will not be relevant to the present paper. For now it can be taken as an undefined primitive. It is my conviction that rebutting defeaters and undercutting defeaters constitute the only kinds of defeaters necessary for describing the structure of defeasible reasoning. The basis for this claim is the success I have had over the years in using them as tools in epistemological analysis (see [24, 26, 29]).

Reasoning starts with premises that are input to the reasoner. (In human beings, they are provided by perception.) The input premises comprise the set *input*. (I assume throughout that *input* is finite.) The reasoner then makes inferences (some conclusive, some defeasible) from those premises using reason schemata. Reasons are combined in various patterns to form arguments. The simplest arguments are *linear* arguments. These can be viewed as finite sequences of propositions each of which is either a member of *input* or inferable from previous members of the sequence in accordance with some reason schema. But it is important to realize that not all arguments are linear. We engage in various kinds of *suppositional reasoning*. In suppositional reasoning we "suppose" something that we have not inferred from *input*, draw conclusions from the supposition, and then "discharge" the supposition to obtain a related conclusion that no longer depends upon the supposition. The simplest example of such suppositional reasoning is *conditionalization*. When using conditionalization to obtain a conditional $(p \supset q)$, we suppose the antecedent p, somehow infer the consequent q from it, and then discharge the supposition to infer $(p \supset q)$ independently of the supposition. Similarly, in *reductio ad absurdum* reasoning, to obtain $\neg p$ we may suppose p, somehow infer $\neg p$ on the basis of the supposition, and then discharge the supposition and conclude $\neg p$ independently of the supposition. Another variety of suppositional reasoning is dilemma (reasoning by cases).

In suppositional reasoning, we cannot think of arguments as finite sequences of propositions, because each line of an argument may depend upon suppositions. We can instead think of lines of arguments as ordered triples $\langle X, p, \beta \rangle$, where X is the set of propositions comprising what is supposed on that line, p is the proposition obtained on that line, and β describes the *basis* for the line, β will be taken to be an ordered pair $\langle \lambda, R \rangle$, where R is the rule of inference used to obtain the line and λ is the set of line numbers of the lines from which the present line is inferred by using R. X is the *supposition set* of the line. An argument σ *supports* the proposition p *relative to* the supposition X iff one of its lines has the form $\langle X, p, \beta \rangle$. σ *supports* p iff σ supports p relative to the empty supposition. The *conclusion* of an argument is its last line. For further details about the structure of arguments, see [31, 32].

Arguments defeat other arguments by supporting rebutting or undercutting defeaters for some of their defeasible steps. If we make the simplifying assumption that all reasons are of equal strength, this is described as follows:

Definition 1.3. An argument σ *rebuts* an argument η iff:

(1) some line of η has the form $\langle Y, q, \langle \alpha, \mathit{reason} \rangle\rangle$, where the propositions supported on the lines in α constitute a prima facie reason for q; and
(2) some line of σ has the form $\langle X, \neg q, \beta \rangle$, where $X \subseteq Y$.

Definition 1.4. An argument σ *undercuts* an argument η iff:

(1) some line of η has the form $\langle Y, q, \langle \alpha, \mathit{reason} \rangle\rangle$, where the propositions $p_1, \ldots, p_k$ supported on the lines in α constitute a prima facie reason for q; and
(2) some line of σ has the form $\langle X, ((p_1 \& \cdots \& p_k) \otimes q), \beta \rangle$, where $X \subseteq Y$.

If we do not assume that all reasons are of equal strength, then this account must be complicated somewhat. A proposal is made in [31] for how to do this, and it has been implemented in OSCAR, but to keep the current presentation as simple as possible, I will ignore variations in reason-strength in this paper. We can then define:

An argument σ *defeats* an argument η iff σ either rebuts or undercuts η.

1.2. Justified beliefs and warranted propositions

Theories of reasoning are basically procedural theories. They are concerned with what a reasoner should do next when it finds itself in any particular epistemological situation. Correct reasoning can involve numerous false starts, wherein a belief is adopted, retracted, reinstated, retracted again, and so forth. At each stage of reasoning, if the reasoning is correct then a belief held on the basis of that reasoning is *justified*, even if subsequent reasoning will mandate its retraction. *Epistemic justification*, in this sense, is a procedural notion consisting of the correct rules for belief updating having been followed by the system up to the present time in connection with the belief being evaluated.

We can think of a reasoner as a belief-updater, i.e., an effectively computable set function *update* that operates repeatedly on sets of beliefs to generate new sets of beliefs. The reasoner starts with the set *input*, and each cycle of the reasoner constitutes the application of *update* to the previous set of beliefs. Let us define the function $\mathcal{J}$ recursively by stipulating

$$\mathcal{J}(0) = \mathit{input}\,,$$

$$\mathcal{J}(n+1) = \mathit{update}(\mathcal{J}(n))\,.$$

Then $\mathcal{J}(i)$ is the set of beliefs justified at the ith stage of reasoning.

In contrast to justification, *warrant* is what the system of reasoning is ultimately striving for. A proposition is warranted in a particular epistemic situation iff (if and only if), starting from that epistemic situation, an ideal reasoner unconstrained by time or resource limitations would ultimately be led to believe the proposition. Warranted propositions are those that would be justified "in the long run" if the system were able to do all possible relevant reasoning. A proposition can be justified without being warranted, because although the system has done everything correctly up to the present time and that has led to the adoption of the belief, there may be further reasoning waiting to be done that will mandate the retraction of the belief. Similarly, a proposition can be warranted without being justified, because although reasoning up to the present time may have failed to reveal adequate reasons for adopting the proposition, further reasoning may provide such reasons. Analogously, reasoning up to the present may mandate the adoption of defeaters which, upon further reasoning, will be retracted. So justification and warrant are two importantly different notions, although they are closely related.

A characterization of what ought to be believed *given* all possible relevant arguments is a characterization of the set of warranted propositions. Such an account can be given fairly easily in terms of the notion of one argument defeating another. Suppose we have an argument α supporting a conclusion P, and an argument β that defeats α. If these are the only relevant arguments, then P is not warranted. But now suppose we acquire a third argument γ that defeats β. This situation is diagrammed as in Fig. 1. The addition of γ should have the effect of reinstating α, thus making P warranted. We can capture this kind of interplay between arguments by talking about arguments being *in* or *out* at different *levels*. Let us provisionally define:

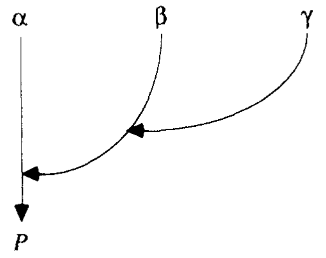


Fig. 1. Interacting arguments.

Definition 1.5.

- All arguments are *in at level* 0.
- An argument is *in at level* $n+1$ iff it is in at level 0 and it is not defeated by any argument in at level n.
- An argument is *ultimately undefeated* iff there is an m such that for every $n \geq m$, the argument is in at level n.

My proposal is then that a proposition is warranted iff it is supported by some ultimately undefeated argument.[2]

1.3. Collective and provisional defeat

Suppose you have two friends, Smith and Jones, that you regard as equally reliable. Smith approaches you in the hall and says, "It is raining outside." Jones then announces, "Don't believe him. It is a fine sunny day." If you have no other evidence regarding the weather, what should you believe? It seems obvious that you should *withhold* belief, believing neither that it is raining nor that it is not. If the state of the weather is important to you then you should seek more evidence rather than deciding at random to hold one or the other of these beliefs. This is an illustration of the phenomenon of *collective defeat.* Consider a simple scenario in which $input = \{p, q\}$, and $\langle\{p\}, r\rangle$ and $\langle\{q\}, \neg r\rangle$ are prima facie reasons of the same strength. Then we can construct two simple arguments:

$$\alpha\colon \quad p {-}{-}{-}{>} r\,,$$

$$\beta\colon \quad q {-}{-}{-}{>} \neg r\,.$$

It follows from our analysis that each argument defeats the other. Accordingly, they are in at level 0, out at level 1, in again at level 2, out again at level 3, and so on. Hence neither is ultimately undefeated, and hence neither r nor $\neg r$ is warranted.

Collective defeat operates in accordance with the following general principle:

[2] I first made this proposal in [25]. It was first published in [26]. A similar proposal was made by Horty, Thomason, and Touretzky in [10].

The Principle of Collective Defeat. If Σ is a set of arguments such that (1) each argument in Σ is defeated by some other argument in Σ and (2) no argument in Σ is defeated by any argument not in Σ, then no argument in Σ is ultimately undefeated.

This is because each argument in Σ will be in at every even level, but then it follows that each will be out at every odd level. We can define:

Definition 1.6.
- An argument σ is *ultimately defeated* iff there is a level n such that σ is out at all higher levels.
- An argument σ is *provisionally defeated* iff there is no level n such that σ is in at all higher levels or out at all higher levels.

Collective defeat is familiar in AI from the discussion of *skeptical* and *credulous* reasoners (see [35]). Roughly, skeptical reasoners withhold belief when they have equally good reasons for and against a conclusion, and credulous reasoners choose a conclusion at random. It has been urged that the choice between skeptical and credulous reasoners is more a matter of taste than a matter of logic, but my own view is that credulous reasoners are just wrong. In the above example, if I announced "I realize that I have no better reason for thinking that it is raining than for thinking that it is not, but I choose to believe that it is raining", no one would regard me as rational. I have argued in [28, p. 129] that this controversy stems from a confusion of theoretical reasoning (reasoning about what to believe) with practical reasoning (reasoning about what actions to perform). In practical reasoning, if one has no basis for choosing between two alternative actions, one should choose at random. The classical illustration of this is the medieval tale of Buridan's ass who starved to death standing midway between two equally succulent bales of hay because he could not decide from which to eat. This marks an important difference between practical reasoning and theoretical reasoning. But regardless of what one thinks about all this, the theory of reasoning being implemented in this paper is skeptical.

Collectively defeated arguments are provisionally defeated, but it turns out that an argument can be provisionally defeated without entering into collective defeat with other arguments. This results from the fact that, although an ultimately defeated argument cannot defeat another argument, provisionally defeated arguments can still provisionally defeat other arguments. To illustrate, suppose α and β defeat one another collectively. In this case, α is in at every even level and out at every odd level. Now suppose α supports a defeater for a third argument γ. This will have the effect that γ is out at every odd level and back in at every even level, so γ is also provisionally defeated, even though it may not in turn defeat the arguments at whose hands it suffers provisional defeat. This observation will turn out to be extremely important in the design of a defeasible reasoner.

1.4. Self-defeating arguments

In [32], I showed that if the only source of defeat among arguments were the rebutting and undercutting relations defined in Section 1.1, purely formal manipulations would produce arguments defeating virtually any defeasible argument, with the result that the whole structure of defeasible reasoning would collapse. For example, suppose P is a prima facie reason for R, Q is an equally good prima facie reason for $\sim R$, S is a prima facie reason for T, and $input = \{P, Q, S\}$. Then we can construct the following three arguments (where defeasible inferences are indicated by dashed arrows):

$$\alpha:\quad P \dashrightarrow R\,,$$

$$\beta:\quad Q \dashrightarrow \sim R\,,$$

$$\sigma:\quad S \dashrightarrow T\,;$$

α and β collectively defeat one another, but σ should be independent of α and β and ultimately undefeated. The difficulty is that we can construct a fourth argument (where deductive inferences are indicated by solid arrows):

$$\eta:\quad \left.\begin{array}{l} P \dashrightarrow R \rightarrow (R \vee \sim T) \longrightarrow \\ Q \dashrightarrow \sim R \longrightarrow \end{array}\right\} \longrightarrow \sim T\,;$$

η uses a standard strategy for deriving an arbitrary conclusion from a contradiction. The problem is now that η rebuts σ. Of course, η itself is defeated by either α or β, or for that matter, by itself (it supports defeaters for its own defeasible steps). But that only results in η being provisionally defeated, and as I pointed out above, a provisionally defeated argument can still provisionally defeat another argument. η is out at every even level, but it is still in at every odd level. Consequently, it still forces σ to be out at every even level, and hence σ is provisionally defeated too. But σ should not be provisionally defeated—it should be ultimately undefeated.

To handle self-defeat correctly, the theory of defeasible reasoning must be made more complicated, and in [32] I made a proposal regarding how this should be done. As a start, the definition of being in at a level is modified so that self-defeating arguments are ruled out at every level. But it turns out that additional complications must be added to the theory to adequately handle collective defeat. The proposal of [32] is incorporated into OSCAR, but to simplify the account, I am going to ignore these complications in the present paper. The justification for this is that they turn out not to affect the overall structure of the automated reasoner. All they do is make it a bit more difficult to compute which arguments defeat which.

1.5. Comparison with AI theories of nonmonotonic reasoning

Although in general outline this theory predates the most familiar theories of nonmonotonic reasoning in AI, it will seem unfamiliar to many researchers in AI because of the past isolation of philosophy and AI from each other. Accordingly, it is useful to compare this theory to theories more familiar in AI. The comparison will be brief, because the point of this paper is not to defend this theory (that has been done elsewhere), but to ask how an automated reasoner could implement it. In spirit, the theory of defeasible reasoning seems close to Reiter's default logic [34], with prima facie reasons and defeaters corresponding to Reiter's defaults. But there are also profound differences between the two theories. First, prima facie reasons are supposed to be logical relationships between concepts. It is a necessary feature of the concept *red* that something's looking red to me gives me a prima facie reason for thinking it is red. (To suppose we have to discover such connections inductively leads to an infinite regress, because we must rely upon perceptual judgments to collect the data for an inductive generalization.) By contrast, Reiter's defaults often represent contingent generalizations. If we know that most birds can fly, then the inference from being a bird to flying may be adopted as a default. In the theory of defeasible reasoning, the latter inference is instead handled in terms of the following prima facie reason schema:

> ⌜Most *A*'s are *B*'s, and this is an *A*⌝ is a prima facie reason for ⌜This is a *B*⌝ .

This is the *statistical syllogism*, and constitutes the central topic of discussion of [29].

A second contrast between the present theory of defeasible reasoning and Reiter's approach is that the latter is semantical (proceeding in terms of an unspecified deductive-consequence relation), whereas the former is argument-theoretic. Argument-based approaches to defeasibility can be found in the work of Loui [16] and also in that of Lin and Shoham [15]. The work of Horty, Thomason, and Touretzky [35] can also be viewed as an argument-based theory of defeasible reasoning, where the arguments are represented as trees. However, these theories are all based upon much simpler conceptions of argument, confining their attention to linear arguments, and they do not investigate the phenomena of collective defeat or self-defeat. None of the AI theories of nonmonotonic reasoning appear to be sensitive to the importance of suppositional reasoning, but suppositional reasoning seems to be essential in any reasoner that is capable of performing deductive and defeasible reasoning simultaneously. In addition, the systems of Horty, Thomason, and Touretzky accommodate only rebutting defeaters.

It is easily proven that if we identify prima facie reasons with defaults, confine our attention to linear arguments, consider only cases in which there is no collective defeat or self-defeat, and identify the deductive-consequence relation with deductive provability using the linear arguments, then the set of warranted conclusions generated by the present theory will be the same as the unique extension generated by Reiter's default logic. In situations in which collective defeat occurs, the two theories yield completely different results, because default logic is credulous and the present theory is skeptical. A skeptical version of default logic can be generated by requiring that default consequences be members of the intersection of all extensions, and I conjecture (but have not proven) that this brings the two theories back into agreement if there is no self-defeat and we consider only linear arguments. Once we allow suppositional reasoning, the two theories diverge again. For instance, if we consider a default theory with the normal default

$$\frac{P : \sim Q}{Q}$$

and the corresponding defeasible theory in which P is a prima facie reason for Q and there are no undercutting defeaters, then from the empty set of premises the present theory of defeasible reasoning will generate the warranted conclusion $(P \supset Q)$, but skeptical default logic will not.

No theories of nonmonotonic reasoning handle self-defeat in the same way as the present theory of defeasible reasoning. For example, suppose we are given P, and P is a prima facie reason for Q, Q is a prima facie reason for R, and R is a prima facie reason for $(P \otimes Q)$. Then the following argument is self-defeating:

$$P ---> Q ---> R ---> (P \otimes Q) .$$

On the theory of defeasible reasoning (as further developed in [32]), Q and R will be warranted, but $(P \otimes Q)$ will not be warranted. If we try to represent this in default logic, we find that the only extension is the logical closure of $\{P, (P \otimes Q)\}$. This has the perverse consequence that although Q and R are unwarranted, $(P \otimes Q)$ is warranted. I take this to be an intuitive counterexample to default logic.

2. Criteria of adequacy for a defeasible reasoner

In designing an automated defeasible reasoner, one is faced with the problem of how to evaluate the reasoning that the reasoner performs. We want the reasoning to be "correct", but what is the criterion for correctness? The desideratum is not necessarily to build a reasoner that replicates human reasoning in all respects, because there may be more efficient ways of doing it. However, before we can decide whether a particular procedure is a more efficient way of doing it, we have to determine what the "it" is that we want the reasoner to do. The analysis of warrant constitutes an "argument-based semantics" for defeasible reasoning, but what exactly is the connection between warrant and what we want a reasoner to accomplish? The simplest proposal would be that we want the reasoner to "compute warrant". But if this is understood as requiring that the reasoner implement an effective procedure for deciding warrant, then it is an impossible desideratum. All theorems of logic are automatically warranted because the arguments supporting them are non-defeasible. This includes all theorems of the predicate calculus. If we give the system no non-logical reasons, these are the only warranted propositions. Thus a decision procedure for warrant would give us a decision procedure for the predicate calculus. However, by Church's theorem, the set of theorems of the predicate calculus is not decidable. Thus *no* reasoner can compute warrant in this sense. A weaker proposal would be that we want the reasoner to systematically generate all warranted propositions in some effective way, analogous to the manner in which a complete theorem prover generates all theorems of the predicate calculus. But this desideratum is also provably unsatisfiable, because the set of warranted propositions can fail to be recursively enumerable (henceforth "r.e."). This is because, as has been observed by numerous authors,[3] on any theory of defeasible reasoning, the ultimate correctness of a piece of reasoning (i.e., whether the conclusion of the reasoning will survive an indefinite amount of further reasoning and hence be warranted) will always turn upon something else *being unprovable*. Making this more precise, we have the following theorem:

Theorem 2.1. *There are finite sets of input premises and finite sets of non-logical reasons such that the set of conclusions warranted with respect to them is not r.e.*

Proof. Suppose otherwise. Then for any finite set *input* and finite set of non-logical reasons, there is a way of mechanically generating the list of formulas warranted relative to them. Given any first-order formula P, choose a sentence letter Q not occurring in P, let $input = \{Q\}$, and let the only non-logical reason be the prima facie reason $\langle\{Q\}, P\rangle$. We have a prima facie reason for P, so P is warranted iff there is no ultimately undefeated or provisionally defeated argument defeating this inference. Because Q is logically unrelated to P, and there are no other prima facie reasons, the only possible defeating argument would be a deductive argument for $\sim P$. Thus in this situation, P is warranted iff $\sim P$ is not a theorem of the predicate calculus. Consequently, the mechanical procedure for listing warranted conclusions will list P iff $\sim P$ is not a theorem. This would constitute a recursive enumeration of the non-theorems of the predicate calculus, but that is impossible by Church's theorem. It follows that the set of defeasible consequences of a set of premises

[3] I think that the first were David Israel [11] and Raymond Reiter [34].

may not be r.e., and there can be no effective procedure for generating the set of warranted consequences of an arbitrary set of input premises and non-logical reasons. □

If the desideratum for an automated reasoner is neither that of computing warrant nor recursively enumerating the set of warranted conclusions, what is it? We should take seriously the idea that defeasible reasoning is *defeasible*. That is, a defeasible reasoner may have to adopt a belief, and then retract it in the face of defeaters, and then reinstate the belief because the defeaters are themselves retracted. This cycle may be repeated an indefinite number of times. The most we can require of the reasoner is that its rules for reasoning guarantee that it will systematically modify its belief set so that it comes to approximate the set of warranted propositions more and more closely. We want the set of beliefs to "approach the set of warranted propositions in the limit". In [28, 31], I proposed that we understand this in the following sense:

Proposition 2.2. *The rules for reasoning should be such that*:

(1) *if a proposition p is warranted, then the reasoner will eventually reach a stage where p is adopted and stays adopted*;

(2) *if p is unwarranted, then the reasoner will eventually reach a stage where p is not adopted and stays unadopted.*

So the task of a reasoner is not to compute warrant. It is to generate successive sets of beliefs that approximate warrant more and more closely, in the above sense. We can make this mathematically precise as follows.

Definition 2.3. A set A is *defeasibly enumerable* (henceforth "*d.e.*") iff there is an effectively computable set function σ and a recursive set A_0 such that if we define $A_{i+1} = \sigma(A_i)$ then:

(1) $(\forall x)$ if $x \in A$ then $(\exists n)(\forall m > n)\ x \in A_m$;

(2) $(\forall x)$ if $x \notin A$ then $(\exists n)(\forall m > n)\ x \notin A_m$.

I will say that the pair $\langle A_0, \sigma\rangle$ is a *d.e. approximation* of A, and the sequence A_i of recursive sets is a *defeasible enumeration* of A. An equivalent definition is:

Definition 2.4. A set A is *defeasibly enumerable* iff there is an effectively computable function f such that for each n, $f(n)$ is a recursive set, and

(1) $(\forall x)$ if $x \in A$ then $(\exists n)(\forall m > n)\ x \in f(m)$;

(1) $(\forall x)$ if $x \notin A$ then $(\exists n)(\forall m > n)\ x \notin f(m)$.

Defeasibly enumerable sets are the same as the sets that Gold [9] calls "limiting recursive" and Putnam [33] calls "trial and error". Both authors establish that a set is of this type iff it is Δ_2 in the arithmetic hierarchy.

The intuitive difference between recursively enumerable sets and defeasibly enumerable sets is that recursively enumerable sets can be "systematically approximated from below", while defeasibly enumerable sets that are not recursively enumerable can only be systematically approximated from above and below simultaneously. More precisely, if A is r.e., then there is an effectively computable sequence of sets A_i such that

(1) $(\forall x)$ if $x \in A$ then $(\exists n)(\forall m > n)\ x \in A_m$;
(2) $(\forall x)$ if $x \notin A$ then $(\forall m)\ x \notin A_m$.

The sets A_i approximate A from below in the sense that they are all subsets of A and they grow monotonically, approaching A in the limit. If A is defeasibly enumerable, however, the sets A_i need not be subsets of A. They may only approach A from above and below simultaneously, in the sense that they may contain elements not contained in A. Every such element must eventually be taken out of the A_i's, but there need not be any point at which they have *all* been removed. The process of defeasible enumeration can be pictured by thinking of A as a spherical region of space and the A_i's as representing successive stages of a reverberating elastic ball whose center coincides with the center of A. As the reverberations dampen out, the outer surface of the ball will come to approximate that of the spherical surface more and more closely, but there will never be a point at which the ball is contained entirely within the spherical surface.

The reverberating sphere metaphor can be used to give a precise mathematical characterization of the difference between A being r.e. (approximation from below) and d.e. (approximation from above and below simultaneously). If A is r.e. then

$$A = \bigcup_{n \in \omega} A_n\ .$$

On the other hand, if A is d.e. then what we have is:

$$A = \bigcap_{n \in \omega} \bigcup_{m \geq n} A_m = \bigcup_{n \in \omega} \bigcap_{m \geq n} A_m\ .$$

My proposal regarding reasoning and warrant is that the set of warranted propositions is defeasibly enumerable, and the rules for reasoning are rules for successively approximating warrant in this way, i.e., they are rules for constructing a d.e. approximation. More accurately, thinking of a reasoner as an effective set function *update* that starts with the set *input* and operates repeatedly on sets of beliefs to generate new sets of beliefs, we have:

Definition 2.5. A reasoner *update* is *d.e.-adequate* iff, for any set *input* of inputs, $\langle$*input*,*update*$\rangle$ is a d.e. approximation of the set of propositions that are warranted given that set of inputs.

Equivalently:

Definition 2.6. If $\mathcal{J}(i)$ is the set of propositions justified after i applications of *update* to *input*, the reasoner is d.e.-adequate iff $\mathcal{J}$ is a defeasible enumeration of the set of warranted propositions.

I propose d.e.-adequacy as the primary criterion of adequacy for a reasoner, and the objective of this paper is to investigate the question of how to construct a reasoner that is d.e.-adequate.[4]

In actual practice, an automated reasoner does not strive to do all possible reasoning. Instead, it is constrained to construct arguments built using only limited resources for argument formation. For instance, it might construct only arguments that can be formulated within a certain system of first-order logic. These arguments will comprise a class $\mathcal{A}$. The preceding adequacy condition must be relativized to $\mathcal{A}$ to make it applicable to such reasoners. Given any class $\mathcal{A}$ of arguments, we can relativize all our argument-based concepts to $\mathcal{A}$ by relativizing all quantification over arguments to the class $\mathcal{A}$. In particular:

Definition 2.7. P is *warranted relative to* $\mathcal{A}$ iff P is supported by some argument in $\mathcal{A}$ that is ultimately undefeated relative to $\mathcal{A}$.

Definition 2.8. A reasoner *update* is *d.e.-adequate relative to* $\mathcal{A}$ iff, for any set *input* of inputs, $\langle$*input*,*update*$\rangle$ is a d.e. approximation to the set of propositions that are warranted relative to $\mathcal{A}$ given that set of inputs.

Our actual objective will be to construct reasoners that are d.e.-adequate relative to particular classes of arguments.

The requirement that a reasoner provide a d.e. approximation to warrant is a *minimal* criterion of adequacy. Other criteria must also be involved in the choice of a reasoner. At the very least we must consider efficiency. But there is a different kind of adequacy condition that must also be met. If a reasoner is d.e.-adequate, there will be cases in which it will never stop reasoning. Any given proposition may be adopted, retracted, and reinstated many times. Every warranted proposition will eventually be adopted without subsequently being retracted, and every unwarranted proposition will eventually become unadopted without subsequently being adopted, but the reasoner may never know that a given proposition has reached this stable state. It can inform us that "so far" a certain conclusion is justified, but it may have to continue

[4] It can be observed that d.e.-adequate reasoners might also be constructed for other theories of defeasible and nonmonotonic reasoning. I have not pursued that, because it is my conviction that insofar as such theories disagree with the present theory they give incorrect accounts of rationality. But, of course, others will disagree with me on this.

forever in a possibly fruitless search for defeating arguments. This, of course, is just the way people work. This highlights a distinction between two concepts of defeasibility. The sense in which correct human reasoning is defeasible is that we regard such reasoning as "innocent until proven guilty". Once a conclusion becomes justified, it is reasonable to accept it provisionally and act upon it. By contrast, AI theories of nonmonotonic reasoning (default logic, circumscription, etc.) have usually focused on a stronger notion of defeasibility according to which a defeasible conclusion is acceptable only if it has been established that it is objectively devoid of faults. The latter amounts to *proving* that the conclusion is warranted. This has made it seem mysterious how nonmonotonic reasoning can possibly function in a finite agent. The solution is to instead adopt the "innocent until proven guilty" construal of defeasibility, and allow a rational agent to act on its defeasible conclusions even though it has not conclusively established that there are no defeaters and even though, in the absence of more pressing tasks, it will continue to search for defeaters.

The reasoning employed by such a rational agent must be *interruptible*,[5] in the sense that if at some point the agent must stop reasoning and act, it is reasonable to act on the conclusions drawn to that point. This is not ensured by d.e.-adequacy. For example, let R_1 be a reasoner that is both interruptible and d.e.-adequate. Let R_2 be just like R_1 except that for the first million steps it draws conclusions purely at random, and then after one million steps it withdraws all those randomly drawn conclusions and begins reasoning as in R_1. Clearly, it would be unreasonable to make use of any of the conclusions drawn by R_2 during its first one million inference steps, so it is not interruptible. On the other hand, R_2 is still d.e.-adequate, because that concerns only its behavior in the limit, and its behavior in the limit is the same as that of R_1.

It is not clear how to construct a formal criterion of adequacy that will ensure interruptibility. It is tempting to at least require that the conditional probability that a conclusion is warranted given that it is drawn at a certain stage of the reasoning is (1) high, and (2) a monotonic increasing function of the number of the stage. But this is still insufficient to insure interruptibility. For example, a reasoner satisfying this condition will continue to satisfy it if we modify it to draw conclusions at random when it is dealing with a certain isolated subject matter.

Because of the difficulty in formulating a mathematically precise characterization of interruptibility, I am going to ignore that condition in this paper, but it is a topic that must eventually be addressed with care. My objective here will be merely to design a reasoner that is d.e.-adequate. I will return to interruptibility briefly in Section 7.

[5] This point and the terminology are due to George Smith.

3. Building a defeasible reasoner

3.1. The monotonic reasoner

In constructing a defeasible reasoner, we need rules governing the adoption of beliefs, the retraction of beliefs in response to the adoption of defeaters, and the reinstatement of beliefs in response to the retraction of defeaters. I propose to begin with the construction of a reasoner that ignores defeaters. This *monotonic reasoner* will be analogous to a deductive reasoner in that it constructs arguments and adopts as beliefs any conclusions supported by any of the arguments it constructs, but it will differ from a deductive reasoner in that it will use prima facie reasons as well as conclusive reasons as links in arguments. For the most part, it need not distinguish between prima facie and conclusive reasons.

3.2. Defeasible reasoning without collective defeat or self-defeat

If we did not have to contend with collective defeat or self-defeat, it would be quite easy to build a defeasible reasoner by modifying the monotonic reasoner. A defeasible reasoner must perform three kinds of operations: belief adoption, retraction, and reinstatement. These proceed as follows:

(1) The reasoner must adopt beliefs in response to constructing arguments, provided no defeaters have already been adopted for any step of the argument. This can be handled just as in the monotonic reasoner, except that when a defeasible inference occurs, there must be a check to ascertain whether a defeater for it has already been adopted as a belief.
(2) The reasoner must keep track of the bases upon which its beliefs are held. When a new belief is adopted that is a defeater for a previous inference step, then the reasoner must retract that inference step and all beliefs inferred from it.
(3) The reasoner must keep track of defeated inferences, and when a defeater is itself retracted (in accordance with (2)), this should reinstate the defeated inference. The reasoner can then either repeat the reasoning that followed from that defeated inference, or the reasoner can be constructed in such a way that it keeps track of that reasoning and reinstates it *en block*.

It is simple to build a reasoner that performs these functions, and in fact such a reasoner is described in [27]. Let us call a reasoner performing these functions a *stage-I* reasoner. One of the salient characteristics of a stage-I reasoner is that once a defeasible inference step is defeated, the reasoner ceases exploring its consequences. The reasoner expends its resources investigating the consequences of a defeated inference only if the inference is reinstated. This seems like the only sensible way to proceed in defeasible reasoning. Why put effort into developing the consequences of an inference you already know to be defeated? But as we will see, this seemingly obvious design feature leads to apparently insuperable difficulties when we begin to worry about collective defeat.

3.3. Collective defeat and provisional defeat

Some special device must be adopted for handling collective defeat. For example, suppose *input* = $\{P, Q\}$, and P is a prima facie reason for R while Q is an equally good prima facie reason for $\sim R$. The stage-I reasoner will fail to discover the collective defeat. It will adopt P and Q, and then infer R. The next move would be to infer $\sim R$, but it has already adopted a defeater for that inference (namely, R), so it will refrain from making that inference. It will thus never adopt a defeater for the inference to R, and hence that belief will remain adopted, whereas it should be provisionally defeated.

A first stab at handling collective rebutting defeat might have the reasoner note that $\sim R$ is a defeater for the inference to R, and so in addition to refraining from inferring $\sim R$, the reasoner will also retract the belief in R. However, the reasoner cannot stop there, because upon retracting R, it will no longer have a defeater for the inference to $\sim R$, and hence that inference would be reinstated according to the rules governing the stage-I reasoner. To block this, the reasoner must maintain a special database recording the essential facts governing any collective defeat that has occurred. By appealing to the database, the reasoner can avoid repeating collectively defeated reasoning unless it is reinstated, and it can also use the database to govern reinstatement. In collective rebutting defeat, a set X of prima facie reasons all of whose premises are adopted as beliefs is shown to deductively entail a contradiction. A convenient way to record such collective rebutting defeat is to have the database be a list of such sets X. If one of the reasons in X is subsequently defeated in some other way, or if its premises are retracted on the basis of defeat elsewhere, that should reinstate the other reasons in X. Let us call a reasoner handling collective rebutting defeat in this way a *stage-II* reasoner. The reasoner of [27] was also a stage-II reasoner. Note that stage-II reasoners are like stage-I reasoners in that it is still true that once a defeasible inference step is defeated, the reasoner ceases exploring its consequences.

In [27], I observed that the reasoner described there did not handle some cases of collective undercutting defeat correctly. At the time, I supposed this could be corrected by adding a special database and rules for processing it analogous to the above treatment of rebutting defeaters. But I have subsequently come to realize that this is incorrect, and furthermore that the stage-II reasoner handles some cases of collective rebutting defeat incorrectly. The problem stems from the seemingly desirable feature that once a defeasible inference step is defeated, the reasoner ceases exploring its consequences.

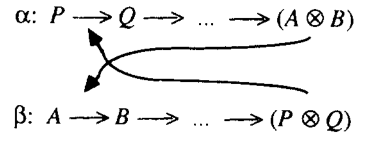


Fig. 2. Collective undercutting defeat.

The cases in which collective rebutting defeat is handled incorrectly arise from the fact noted in Section 1.3 that collectively defeated arguments can still render other arguments provisionally defeated. However, such provisional defeat will not be discovered by a stage-II reasoner. If it simply retracts beliefs in the face of collective defeat, this will prevent their provisionally defeating other beliefs. For instance, suppose we have an argument α for $\sim(P \mathbin{\&} Q)$, and β for $(P \mathbin{\&} Q)$. α and β are collectively defeated. If the reasoner then constructs an argument γ for $\sim P$, it will not notice the conflict between that and $(P \mathbin{\&} Q)$, because the latter was retracted before the reasoner had a chance to infer P from it. Consequently, the reasoner will not take γ to be provisionally defeated. In order to discover the provisional defeat of γ, it would have to continue reasoning from $(P \mathbin{\&} Q)$ to P, even after β has been provisionally defeated. (I am assuming here that without some special reason for doing so, the reasoner will not automatically infer $\sim(P \mathbin{\&} Q)$ from $\sim P$. I believe that this is true of all existing monotonic reasoners. The alternative is combinatorially explosive.) It appears that the only way to handle this correctly is to flag provisionally defeated beliefs as provisionally defeated, and have the reasoner continue to reason from them instead of simply discarding them as in the stage-II reasoner.

An analogous problem occurs in collective undercutting defeat. Suppose α and β are long arguments each of which undercuts an early defeasible inference in the other. This is diagrammed in Fig. 2. If the reasoner first discovers the defeat of β by α, it will stop reasoning from propositions supported by β and never discover the defeat of α and β. The only way to handle this is to continue reasoning with beliefs even when they are defeated outright (not just provisionally defeated). Again, the reasoner must flag them as defeated, and continue reasoning with them.

3.4. *Flag-based reasoners*

The preceding considerations seem to indicate that the production of arguments by the monotonic reasoner must be relatively insensitive to their defeat status, because the reasoner must continue reasoning from conclusions even when they are defeated. This suggests building a defeasible reasoner out of two relatively autonomous modules. The first is the monotonic reasoner, which systematically makes inferences (generates arguments) without concern for their defeat status, and the second is a module that computes the defeat status of all the arguments produced at each stage of reasoning relative to the set of all arguments so far produced. The reasoner will then be a simple loop:

```
(loop
  (make-an-inference)
  (recompute-defeat-statuses))
```

Each time the reasoner makes an inference, it thereby produces a new argument. Let $\mathcal{A}_i$ be the set of arguments produced after i inferences. It is assumed that the set $\mathcal{A}$ of arguments relative to which we assess warrant is the union of the $\mathcal{A}_i$.

Definition 3.1.

- An argument α is *ultimately undefeated at stage* i iff α is ultimately undefeated relative to $\mathcal{A}_i$.
- *An argument* α is *ultimately defeated at stage* i iff α is ultimately defeated relative to $\mathcal{A}_i$.
- An argument α is *provisionally defeated at stage* i iff α is provisionally defeated relative to $\mathcal{A}_i$.
- A proposition P is *justified at stage* i iff P is supported by some argument ultimately undefeated at stage i.

The function *recompute-defeat-status* determines which of the arguments in $\mathcal{A}_i$ are ultimately defeated, which are ultimately undefeated, and which are provisionally defeated, at stage i. To avoid confusion, the reader should bear in mind that the levels of arguments employed in the recursive definition of warrant are unrelated to the stages of inference employed here. I will refer to the latter as *stages of the monotonic reasoner*. Let $\mathcal{J}(i)$ be the set of all propositions justified at stage i of the monotonic reasoner. These will be taken to be the interim conclusions (the current beliefs) of the reasoner. The hope is that the sequence of justification sets $\mathcal{J}(i)$ will constitute a defeasible enumeration of the set of warranted conclusions. I will call a reasoner of this sort a *flag-based reasoner*. Whether this approach will actually work remains to be seen, but the principal result of this paper will be that flag-based reasoners can be made to work.

Flag-based reasoners mark a serious divergence from stage-II reasoners. Flagging beliefs as provisionally defeated or defeated outright, and then continuing to reason with them, seems outrageously expensive. But there appears to be no alternative. It is worth at least noting that human reasoners are not totally insensitive to the consequences of defeated beliefs, even if they do not usually expend a large amount of effort in developing those consequences. In Section 6, I will mention possibilities for minimizing this cost. But first, in Sections 4 and 5, I will address theoretical issues concerning the possibility of constructing a d.e.-adequate flag-based reasoner.

4. The defeat graph

The main result of this section will be a theorem to the effect that, given certain reasonable assumptions, it is possible to construct a d.e.-adequate flag-based reasoner for a class of arguments $\mathscr{A}$. Such a reasoner consists of a monotonic reasoner and an algorithm for computing defeat status. Initially, the only assumption that will be made about the monotonic reasoner is that it systematically generates every argument in $\mathscr{A}$. An algorithm will be constructed for computing defeat status, and then we will consider what additional constraints must be imposed on the monotonic reasoner in order to ensure that the resulting flag-based reasoner is d.e.-adequate.

The algorithm for computing defeat status will be based rather directly upon the argument-based semantics. This will facilitate understanding how it works, but it would be impractical to try to implement it directly because of the inordinate demands it would make on memory. To surmount this difficulty, a second algorithm will be described in Section 5. The second algorithm is derived from the first by adopting a different representation of arguments, and it can be implemented in a more practical way. However, a d.e.-adequate flag-based reasoner based on either algorithm can be seen to require an unreasonable assumption about the monotonic reasoner. In order to relax this assumption, we must modify the criterion of d.e.-adequacy to make it interest-relative. Section 6 will address the task of constructing a truly practical defeasible reasoner based upon an interest-driven monotonic reasoner.

The first algorithm is based upon the *defeat graph*, which represents defeat information in a graphical form.

Definition 4.1. The *defeat graph* for a set arg of arguments is a graph whose nodes are the members of arg and which is such that if $\alpha, \beta \in arg$, $\langle \alpha, \beta \rangle$ is a link iff β defeats α.

I will construe the directionality so that it is the parents of a node that defeat it. In diagramming defeat graphs, I will write "$\alpha \leftarrow \beta$" when β is a parent of (defeats) α. It will be useful to talk about the defeat graphs both for the infinite set $\mathscr{A}$ and the finite sets $\mathscr{A}_i$.

As usual, a *branch* is any finite or infinite sequence $\{\xi_k\}$ such that for each i, $\langle \xi_i, \xi_{i+1} \rangle$ is a link. A *circular branch* is an infinite branch that repeats, i.e., there is a k such that for every $i \geqslant k$, there is a $j < k$ such that $\xi_i = \xi_j$. Collective defeat gives rise to circular branches. For example, the collective defeat diagrammed in Fig. 3 gives rise to the circular branch $\langle \alpha, \beta, \gamma, \beta, \gamma, \beta, \gamma, \ldots \rangle$. Unless they are defeated by other nodes not on the

$$\begin{tikzcd}
\alpha & \beta \arrow[l] \arrow[r, bend right] & \gamma \arrow[l, bend right]
\end{tikzcd}$$

Fig. 3. Circular branches.

branch, the nodes of a circular branch are all provisionally defeated. This is because they will all be in at level 0, but that will force them all out at level 1, and then in again at level 2, and so on.

Given any argument α, the nodes on the branches ascending from α can be put into correspondence with argument levels. If *arg* is finite, this enables us to construct an algorithm for computing defeat status relative to *arg* in terms of the structure of the defeat graph. The algorithm is as follows:

Algorithm 4.2.

Step 0. Mark all initial nodes (nodes having no parents) as undefeated.

Step 1. Apply the following two rules recursively until there are no more nodes to which they are applicable:

(a) If a node is unmarked and all of its parents are marked as defeated, mark the node as undefeated.

(b) If a node is unmarked and one of its parents is marked as undefeated, mark the node as defeated.

Step 2. Mark any remaining nodes as provisionally defeated.

(Remember that the parents of a node are the arguments defeating it.) The correctness of this algorithm is easily proven as follows.

Definition 4.3. An argument α *becomes stably in* (or *out*) at the nth level iff
(1) either $n = 0$ or α is out (or in, respectively) at the $(n - 1)$st level, and
(2) for all $m \geqslant n$, α is in (or out, respectively) at the mth level.

α is ultimately undefeated (or defeated) iff α becomes stably in (out, respectively) at some level. If we count the marking of initial nodes as the zeroth step of the recursion, then it is trivial to prove the following by induction on n:

> α is marked undefeated (or defeated) at the nth step of the recursion in Algorithm 4.2 iff α becomes stably in (or out, respectively) at the nth level.

The correctness of the algorithm follows.

The objective is now to show that the sequence of justification sets $\mathcal{J}(i)$ produced by a flag-based reasoner using this algorithm constitutes a defeasible enumeration of warrant. This cannot be proven without imposing further restrictions on the monotonic reasoner. What is required for $\mathcal{J}$ to be a defeasible enumeration of warrant is that for each proposition *P*, the status of *P* eventually stabilizes, i.e., aι some stage i of the monotonic reasoner, *P* becomes justified and subsequently remains justified, or *P* becomes unjustified and subsequently remains unjustified. For $\mathcal{J}$ to be a defeasible enumeration of warrant, it is sufficient (and perhaps necessary) that the following hold:

Condition 4.4. For any argument α, α is ultimately defeated (or ultimately undefeated, or provisionally defeated) relative to $\mathscr{A}$ iff there is a stage n of the monotonic reasoner such that α is ultimately defeated (or ultimately undefeated, or provisionally defeated, respectively) at every stage $\geqslant n$.

The only way Condition 4.4 can fail is for the status of α to cycle indefinitely at progressively later stages of the monotonic reasoner. This in turn can only happen if there is an infinite sequence $\{\xi_k\}$ of arguments produced by the monotonic reasoner all of which are relevant to the defeat-status of α. Obviously, there are only two ways to get infinitely many arguments connected to α:

(1) either α or some node that is an ancestor of α could be an infinite branch point in the defeat graph for $\mathscr{A}$;
(2) the defeat graph for $\mathscr{A}$ could contain an infinite branch having α as its initial node.

Let us consider these two possibilities separately.

Infinite branch points can lead to infinite cycling. To illustrate this possibility, suppose *input* $= \{P,R,S\}$, where P is a prima facie reason for Q, R is a prima facie reason for each of an infinite list of propositions D_i, where each D_i is a prima facie reason for $(P \otimes Q)$, and S is a prima facie reason for each proposition $(D_i \otimes (P \otimes Q))$. With this set of prima facie reasons, if the ordering of the arguments is such that those supporting D_i and inferring $(P \otimes Q)$ from it alternate with those supporting $(D_i \otimes (P \otimes Q))$, then Q will alternate indefinitely between being justified and being unjustified at the different stages. However, the mere existence of infinite branch points is not sufficient to guarantee infinite cycling. This is obvious when we realize that, for many argument systems, it will be possible to construct infinitely many variants for any given argument. For example, we may be able to construct notational variants, or add unnecessary steps. This can have the consequence that every branch point will be an infinite branch point. But this need not give rise to infinite cycling because the different branches are not independent—anything defeating one will defeat the other. Let us define:

Definition 4.5. An argument β is *parasitic on* an argument γ iff any defeater for γ is also a defeater for β.

A necessary (but not sufficient) condition for infinite cycling to result from an infinite branch point is that there is no finite set of parents of the branch point such that every other parent is parasitic on one of the parents in the finite set. I will call such a branch point a *non-redundantly infinite branch point*. If a branch point is not non-redundantly infinite, then defeating a finite set of its parents would defeat them all, and so the only way to get infinite cycling at the branch point would be to already have infinite cycling at one of the parents.

Noncircular infinite branches can also lead to infinite cycling. For example, infinite cycling would result if there were an infinite sequence of propositions Q_i all either in *input* or supported by arguments, and such that if we define D_i recursively by stipulating that

$$D_1 = (Q_0 \otimes P),$$
$$D_{i+1} = (Q_i \otimes D_i), \quad i \geq 1,$$

then (1) Q_0 is a prima facie reason for P, and (2) for each $i \geq 1$, Q_i is a prima facie reason for D_i. On the other hand, circular branches cannot lead to infinite cycling. They contain only finitely many different arguments, so once those arguments are all generated by the monotonic reasoner, they will be marked as provisionally defeated and will stay that way unless one of the nodes is defeated by other nodes not on the branch. If one of the nodes of the circular branch is defeated by a node ν not on the branch, that will lead to infinite cycling only if ν cycles infinitely. ν itself could be on another circular branch, and so on. Thus we might get a sequence of interacting circular branches as in Fig. 4. If the sequence is finite, there will still only be finitely many arguments involved, and so infinite cycling will not result. Infinite cycling would only be possible if the sequence of interacting circular branches were infinite, but then we could construct a noncircular infinite branch by just combining the top parts of each loop.

Summarizing, we have the following simple lemma:

Lemma 4.6. *If the defeat graph for $\mathcal{A}$ contains no noncircular infinite branches and no nonredundantly infinite branch points, then Condition 4.4 holds.*

Although arrays of prima facie reasons and defeaters that will generate noncircular infinite branches and non-redundantly infinite branch points are a formal possibility, I doubt that they are a real possibility. That is, we cannot find real examples of prima facie reasons having these structures. (My only reason for saying this is that I have tried and failed.) Accordingly, my strategy will be to adopt some realistic assumptions about the structure of the set of

$$\begin{tikzcd}
\alpha & \xi_1 \arrow[l] \arrow[d] & \xi_2 \arrow[l] & \xi_3 \arrow[l] & \xi_4 \arrow[l] \\
& \xi_8 \arrow[r] & \xi_7 \arrow[r] & \xi_6 \arrow[r] & \xi_5 \arrow[u] & \mu_1 \arrow[l] \arrow[d] & \mu_2 \arrow[l] & \mu_3 \arrow[l] & \mu_4 \arrow[l] \\
& & & & & \mu_8 \arrow[r] & \mu_7 \arrow[r] & \mu_6 \arrow[r] & \mu_5 \arrow[u] & \cdots \arrow[l]
\end{tikzcd}$$

Fig. 4. Interacting circular branches.

prima facie reasons that preclude these possibilities, and impose them as constraints on the monotonic reasoner. It will then follow that $\mathscr{J}$ is a defeasible enumeration of warrant. Recall that, so far, we have made no assumptions about the monotonic reasoner except that it generates all arguments in the set $\mathscr{A}$.

Unless they are defeated by side branches, noncircular infinite branches lead to the provisional defeat of all of their nodes, but this defeat is odd because it does not arise from collective defeat. The only cases of provisional defeat that seem to arise in realistic systems of prima facie reasons and defeaters involve collective defeat. Accordingly, my first assumption will be that noncircular infinite branches are impossible:

Assumption 4.7. *The defeat graph for $\mathscr{A}$ contains no noncircular infinite branches.*

The second assumption to be adopted is:

Assumption 4.8. *For every proposition P and finite set X of propositions, there is a finite (possibly empty) set arg of arguments in $\mathscr{A}$ supporting P relative to X such that any other argument in $\mathscr{A}$ that supports P relative to X is parasitic on some member of arg.*

Both of these assumptions are finiteness assumptions. They tell us that for any finite *input* and finite supposition X, there is a limit to how much nondeductive reasoning we can do. Again, the only reason for making these assumptions is that I can think of no plausible counterexamples. Both assumptions ought to be provable for particular classes of arguments. Note, however, that we cannot expect to prove them as general theorems about the monotonic reasoner. Their truth will depend essentially on what arrays of prima facie reasons and defeaters are supplied for the use of the monotonic reasoner, and that will vary from application to application.

Assumption 4.7 precludes infinite cycling resulting from infinite branches. The role of Assumption 4.8 is to rule out non-redundantly infinite branch points. To see that it does this, consider any argument α in $\mathscr{A}$. α contains finitely many defeasible steps, and the parents of α are arguments supporting defeaters for those defeasible steps. If a defeasible step infers P from Γ relative to a supposition X, then an argument supporting a defeater for this step must support either $\sim P$ or $(\Pi\Gamma \otimes P)$ relative to a subset of X. Accordingly, there are only finitely many conclusions that a defeating argument can have, and hence by Assumption 4.8 there is a finite set arg of arguments in $\mathscr{A}$ supporting those conclusions and such that any other argument in $\mathscr{A}$ that supports one of those conclusions is parasitic on some member of arg . In other words, non-redundantly infinite branch points are impossible.

We now have the following theorem, which is the central theorem of the paper:

Theorem 4.9. *If Assumptions 4.7 and 4.8 hold, then $\mathcal{J}$ is a defeasible enumeration of warrant.*

Proof. This follows from the fact that if Assumptions 4.7 and 4.8 hold then the defeat graph for $\mathcal{A}$ contains no noncircular infinite branches and no non-redundantly infinite branch points, and hence by Lemma 4.6, Condition 4.4 holds and consequently $\mathcal{J}$ is a defeasible enumeration of warrant. In other words, given Assumptions 4.7 and 4.8, a flag-based reasoner using the Marking Algorithm 4.2 described above will be d.e.-adequate. □

5. The inference graph

In Section 4, an algorithm was described for computing $\mathcal{J}(i)$. This algorithm could, in principle, be used to construct a d.e.-adequate flag-based reasoner. However, implementing this algorithm directly would require too much memory because arguments are going to be repeated over and over again in the defeat graph as they recur as subarguments of larger arguments. Fortunately, the information that is represented in the defeat graph can be represented more efficiently in an *inference graph*, whose nodes are labeled with pairs of the form $\langle \Gamma, p \rangle$. The pair $\langle \Gamma, p \rangle$ signifies that p has been inferred from the supposition Γ. The inference graph can encode both arguments and defeat relations with two different kinds of links. Where ν and η are nodes, $\langle \nu, \eta \rangle$ is an *inference link* iff ν is labeled by a pair $\langle \Gamma, p \rangle$, η is labeled by a pair $\langle \Omega, q \rangle$, and $\langle \Gamma, p \rangle$ was inferred from a set $\{\langle \Omega_1, q_1 \rangle, \ldots, \langle \Omega_n, q_n \rangle\}$ of pairs where $\langle \Omega, q \rangle \in \{\langle \Omega_1, q_1 \rangle, \ldots, \langle \Omega_n, q_n \rangle\}$. The intent is for arguments to be encoded in the inference links. However, if we are to be able to recover the arguments from the inference links, we must take care in one respect. There can be more than one argument supporting $\langle \Gamma, p \rangle$. We must keep the inference links embodied in the different arguments separate. That can be done by having a different node labeled $\langle \Gamma, p \rangle$ for each argument supporting $\langle \Gamma, p \rangle$. Nodes and their inference links then become unambiguous. (Note that this will also allow us to associate unique strengths with nodes.) The *immediate inference ancestors* of a node are the nodes connected to it by inference links. Let us say that η is an *inference ancestor* of ν iff there is an inference branch connecting ν to η. A node is a *pf-node* iff, if its label is $\langle \Gamma, p \rangle$ and the set of labels of its immediate inference ancestors is $\{\langle \Omega_1, q_1 \rangle, \ldots, \langle \Omega_n, q_n \rangle\}$, then $\Gamma = \Omega_1 = \cdots = \Omega_n$, $\langle \{q_1, \ldots, q_n\}, p \rangle$ is a prima facie reason, and the monotonic reasoner inferred p from $\{q_1, \ldots, q_n\}$ on that basis. Let us say that μ is a *deductive ancestor* of ν iff μ is an inference ancestor of ν and the branch connecting them contains no pf-nodes.

$\langle \mu,\nu \rangle$ is a *defeat link* iff μ is a pf-node, and if its label is $\langle \Gamma,p \rangle$ and the set of labels of its immediate inference ancestors is $\{\langle \Gamma,q_1 \rangle, \ldots, \langle \Gamma,q_n \rangle\}$, and the label of ν is $\langle \Omega,q \rangle$, then $\Omega \subseteq \Gamma$ and q is either $\neg p$ or $((q_1 \,\&\, \cdots \,\&\, q_n) \otimes p)$. Defeat links encode undercutting and rebutting defeat.

Given the inference graph, replete with inference links and defeat links, we want an algorithm for determining which nodes are defeated, which are provisionally defeated, and which are undefeated. This can be done in an obvious way by translating the algorithm that was applied to the defeat graph into an algorithm that is applied to the inference graph.

Let us say that a node μ is *potentially defeated* iff there exists a node ν such that $\langle \mu,\nu \rangle$ is a defeat link. A node is *d-initial* iff neither it nor any of its inference ancestors is potentially defeated. d-initial nodes are guaranteed to be undefeated. They correspond to arguments that are initial nodes in the defeat graph. So the algorithm for computing defeat status from the inference graph will proceed as follows:

Algorithm 5.1.

Step 0. Mark all d-initial nodes as undefeated.

Step 1. Apply the following two rules recursively until there are no more nodes to which they are applicable:

(a) If ν is marked as undefeated and $\langle \mu,\nu \rangle$ is a defeat link, then if μ is unmarked, mark it and all its inference descendants as defeated.

(b) If μ is unmarked and (1) for every ν such that $\langle \mu,\nu \rangle$ is a defeat link, ν is marked as defeated, and (2) every inference ancestor of μ is marked as undefeated, then mark μ as undefeated.

Step 2. Mark any remaining nodes as provisionally defeated.

The flag-based defeasible reasoner should now work as follows. The monotonic reasoner builds the inference graph one node at a time. As each node is constructed, defeat links between it and pre-existing nodes are computed and the defeat statuses of the nodes are recomputed. Rather than recomputing defeat status from scratch each time a new node is added to the inference graph, it is sometimes possible to update the markings by only examining those nodes whose status might be changed by the addition. This is at least true if the new node does not defeat any pre-existing node. In that case:

(1) if every node defeating the new node is marked as defeated and all of its immediate ancestors are marked as undefeated, we mark the new node as undefeated;

(2) if some node defeating the new node is marked as undefeated or at least one of its immediate ancestors is marked as defeated, we mark the new node as defeated;

(3) otherwise, mark it as provisionally defeated.

If the new node is a defeater for a pre-existing node, then it appears that all the node statuses must be recomputed from scratch because there is always the possibility of defeat cycles that can only be detected by starting from d-initial nodes.

6. An interest-driven defeasible reasoner

The flag-based reasoner described in Section 5 could actually be constructed, but it is impractical in one important respect. It operates by having the monotonic reasoner systematically generate all possible arguments in the class $\mathscr{A}$. This is the so-called "British Museum Algorithm". Any automated reasoner that is practical employs a more efficient control structure enabling it to focus its attention on arguments that are more intimately connected with the conclusions it is trying to establish. There are various ways of doing this, but my own preference is to begin with the interest-driven deductive reasoner OSCAR described in [30] and use it as the monotonic reasoner on which the defeasible reasoner is built.

Interest-driven reasoners are characterized by their use of two databases. One is the set *adoptions* of conclusions that have thus far been drawn, and the other is the set *interests* of conclusions the reasoner is trying to get. The conclusions that are of ultimate interest are inserted into *interests* at the beginning of the reasoning, but an interest-driven reasoner may also employ rules for adopting interest in additional conclusions during the course of its reasoning.

The desideratum for an interest-driven defeasible reasoner should no longer be d.e.-adequacy, because we only want to require the reasoner to discover the warranted propositions we assign as interests. The obvious proposal is to relativize d.e.-adequacy to interests:

Definition 6.1. A reasoner is *i.d.e.-adequate relative to* $\mathscr{A}$ iff, for any *input* and any proposition P in *interests*:

(1) if P is warranted *relative to* $\mathscr{A}$ then there is some n such that after n cycles of reasoning, P is marked as undefeated and that marking is never subsequently changed; and

(2) if P is not warranted *relative to* $\mathscr{A}$ then there is some n such that after n cycles of reasoning, P is marked as defeated and that marking is never subsequently changed.

An interest-driven defeasible reasoner makes no attempt to build the entire inference graph. It begins with certain initial interests, and then it attempts to perform just those inferences that are relevant to those interests and build just that part of the inference graph that records the relationships between those inferences. The relevant inferences are those involved in constructing arguments supporting the propositions in *interests*, and all those arguments supporting defeaters for those arguments, and defeaters for the defeating arguments, and so on. This will automatically be achieved if

(1) the monotonic reasoner is guaranteed to find all relevant arguments for anything in which it is interested, and
(2) the reasoner automatically adopts interest in defeaters for each defeasible inference it performs.

By virtue of these two conditions, if there is an argument supporting a desired conclusion, the reasoner will find it. If there is a defeating argument it will find that. Then it follows recursively that if there is a reinstating argument, the reasoner will find that, and so on.

However, condition (1) is not quite the condition that should be imposed on the monotonic reasoner. The difficulty is that every argument has infinitely many variations that result from adding redundant steps or from varying the way in which deductive inferences are made. It is not necessary for the monotonic reasoner to produce more than one of these infinite variations. It will suffice to produce just one on which all the others are parasitic. Accordingly, we have:

Definition 6.2. A monotonic reasoner is *interest-complete relative to $\mathscr{A}$* iff, for any *input* and any P and any argument α in $\mathscr{A}$ inferring P from *input* relative to a supposition X, if the reasoner is given *input* as premises and adopts interest in P relative to X, then the reasoner will construct an argument β inferring P from *input* relative to X which is such that any defeasible inference occurring in β also occurs in α and does so relative to the same or a less inclusive supposition.

Then we have the following theorem, which is the justifying theorem for defeasible reasoning:

Fundamental Theorem for Defeasible Reasoning. *If*
(1) *the defeat graph for $\mathscr{A}$ contains no noncircular infinite branches, and*
(2) *for every proposition P and finite set X of propositions, there is a finite (possibly empty) set $\mathscr{arg}$ of arguments in $\mathscr{A}$ supporting P relative to X such that any other argument in $\mathscr{A}$ that supports P relative to X is parasitic on some member of $\mathscr{arg}$,*

and if a flag-based defeasible reasoner
(3) *is based upon an interest-driven monotonic reasoner that is interest-complete relative to $\mathscr{A}$,*
(4) *adopts interest in defeaters for every defeasible inference it performs, and*

(5) *uses the Marking Algorithm 5.1 to compute defeat status, then it is i.d.e.-adequate relative to $\mathcal{A}$.*

It was observed in Section 3 that a defeasible reasoner may never stop reasoning. However, by restricting the monotonic reasoner to interests, the defeasible reasoner may often halt simply by running out of things to do. That cannot always be the case, but in many simple cases this is what will happen. This is illustrated in Appendix B. It follows from interest-completeness that if the reasoner halts, anything in *interest* that is unproven is unprovable, so members of *ultimate* are warranted iff they are justified at the stage at which the reasoner halts.

I have observed several times that flagging beliefs as provisionally defeated or ultimately defeated and then continuing to reason with them seems outrageously expensive. However, there is a way of alleviating this cost, at least to some extent. I assume that the monotonic reasoner uses some scheme for prioritizing potential inferences and stores them on an *inference queue*. This could be a simple "last in, first out" stack, but there is abundant reason to prefer more complex control structures. OSCAR uses a fairly sophisticated prioritizing scheme. Whatever prioritizing scheme is used, if it is modified in such a way that once beliefs are defeated, either provisionally or ultimately, inferences involving them are given low priority, then the reasoner will pursue the consequences of defeated reasoning "only when it has time". This looks much like what human beings do. As long as the prioritizing is done in such a way that even low priority inferences are eventually performed, the reasoner will remain i.d.e.-adequate.

This reasoner has actually been implemented, and is now being tested and refined. It is considered the latest version of OSCAR. Appendix A goes into a bit more detail regarding the implementation. Appendix B gives some examples of its operation.

7. An interrupt-driven defeasible reasoner

Etherington [6] makes the observation that for most nonmonotonic logics, little attention has been paid to the problem of updating beliefs (or extensions) in response to adding new premises (as opposed to simply starting over in the construction of the belief set or the computation of the extension). If a defeasible reasoner is to be used as the inference engine in a real-time system, e.g., a robot (or a human being), this is a serious problem. Such a system is embedded in an environment that is continually feeding it more information, and it must repeatedly update its beliefs in response to such inputs. It is totally impractical to require such a system to start over again from scratch every time it is given a new premise.

In OSCAR, there is no updating problem. If OSCAR is supplied a new premise after reasoning has already begun, OSCAR will quickly reach a point in its reasoning where its beliefs and arguments are the same as if that premise had been supplied at the beginning. This means that OSCAR can provide the inference engine for an interrupt-driven reasoner. Such a reasoner will receive additional premises from time to time after it has already begun reasoning. This will cause the reasoning to be interrupted at the end of a cycle, the new premises inserted into *input*, and then reasoning resumed. OSCAR's standard prioritizing scheme will have the result that if OSCAR is reasoning in some supposition other than the empty supposition, it will continue that reasoning until it exhausts the moves it can make within that supposition, and then it will return to the empty supposition and process the new members of *input*. However, different prioritizing schemes can be adopted depending upon how much precedence we want the new members of *input* to be given.

There is another way in which an interrupt-driven reasoner is important. Most problems with which a reasoner is actually presented will be simple and the reasoner will be able to perform all relevant reasoning and stop. This is illustrated in Appendix B. It has been observed, however, that on problems of sufficient complexity, reasoning may never terminate. Note that this is equally true for human reasoners. Nevertheless, such a reasoner may have to take action at specific times, even though reasoning has not terminated. This is just the point made in Section 2 that the reasoning must be interruptible. When the time comes to act, the reasoning will be interrupted and action taken based upon whatever the current set of beliefs is. It is always possible that if the reasoner had more time to reason, its beliefs would change in some crucial way, but the presumption behind defeasible reasoning is that at any stage of reasoning, if action must be taken then it is reasonable to act on the basis of the current set of beliefs.

The upshot of this is that a real-time defeasible reasoner must be interrupt-driven in two respects. It must be continually receptive to new inputs, and it must be prepared to interrupt its reasoning and act on its current set of beliefs whenever action is required. The intention is that OSCAR will supply the inference engine for such a real-time reasoner.

8. Conclusions

The general objective of the OSCAR project is the construction of a comprehensive theory of rational inference and its implementation in an automated reasoner. This project is a continuation of my epistemological work on defeasible reasoning that began as early as 1965, in my Ph.D. Dissertation. Defeasible reasoning has provided the principal logical tool that I have employed ever since for epistemological analysis. Most of the results of that analysis can be found in [24, 26, 29]. The general approach to defeasible reasoning that is presupposed by all my work is the argument-based approach that I sketched at the beginning of the paper. It has been developed in detail and defended in the papers cited in Section 1. Those papers propound a theory of warrant for defeasible reasoning. The objective of this paper is not further development of the theory of warrant, but the quite different question of how it is possible for a system to reason in a way that can be regarded as implementing that theory of warrant, particularly in the light of the fact that the set of warranted conclusions is not generally r.e.

The first main proposal of the paper is that the appropriate criterion to apply in evaluating defeasible reasoners is that of i.d.e.-adequacy. This makes it possible for reasoners to be adequate despite the fact that the set of warranted conclusions is not r.e. The paper then investigates how to construct an i.d.e.-adequate reasoner. It is argued that such a reasoner cannot stop reasoning from such a conclusion just because the arguments supporting it are defeated. Instead, it must mark conclusions as defeated or undefeated, but continue to reason from them. This suggests building the defeasible reasoner out of two largely autonomous modules—a monotonic reasoner that systematically produces arguments without worrying about their defeat status and a module that computes defeat status relative to the arguments that have been produced at any given stage of reasoning. The main result of the paper is that the sequence consisting of the sets of undefeated conclusions produced at each stage of operation of the monotonic reasoner constitutes a d.e.-approximation to warrant provided the monotonic reasoner satisfies certain plausible constraints. Two algorithms for evaluating defeat status are described. The first is based on the defeat graph, and arises directly out of the theory. The second is essentially a more practical reconstruction of the first, in terms of the inference graph. The inference graph encodes the same information as the defeat graph, but does so in a more efficient way. This is not yet sufficient to produce a truly practical defeasible reasoner, however. The d.e.-adequacy of the reasoner constructed in this way requires the totally impractical assumption that the monotonic reasoner will systematically produce all possible arguments (the British Museum Algorithm). This assumption is replaced by the more reasonable assumption that the monotonic reasoner is interest-driven. This requires a change to the adequacy condition, replacing d.e.-adequacy by its interest-relative analogue, i.d.e.-adequacy. It is then shown that, subject to reasonable assumptions about the interest-driven monotonic reasoner, either algorithm for evaluating defeat status will yield an i.d.e.-adequate reasoner. This final approach (using the inference graph algorithm) has been implemented in OSCAR.

There have been several earlier attempts to construct theories of reasoning (misleadingly called "proof theories") for defeasible reasoning and there have been a few attempts to implement such theories in automated defeasible reasoners. The most noteworthy are those of Nute and Lewis [20], Nute [19], Levesque [14], Ginsberg [8], Baker and Ginsberg [1], and Geffner [7]. There have also been some (often unimplemented) special-purpose reasoners like that described by Horty, Thomason, and Touretzky [10] for reasoning within defeasible inheritance hierarchies. These reasoners are based upon a wide variety of approaches to defeasible reasoning. For example, the Ginsberg system is based upon circumscription, the Levesque theory on autoepistemic logic, and the Horty, Thomason, and Touretzky theory on defeasible inheritance. Despite their variety, it is very simple to compare all of these theories simultaneously to the present theory. This is because all of these theories attempt to build a reasoner that is analogous to a traditional theorem prover. Such reasoners produce r.e. sets of conclusions. Because that is only possible if the underlying logic is decidable, such defeasible reasoners have typically been restricted to the propositional calculus or some other very weak logic. By contrast, because OSCAR seeks to provide an i.d.e. approximation to warrant rather than a recursive enumeration of warranted conclusions, it is applicable to the full predicate calculus, and indeed, OSCAR is deductively complete for the predicate calculus as well as being i.d.e.-adequate for defeasible reasoning.

OSCAR also stands in an interesting relationship to RMSs (reason maintenance systems). McDermott [18] constructs a general framework for RMSs, and emphasizes the distinction between the application program and the RMS itself. That is parallel to the distinction in OSCAR between the monotonic reasoner and the module computing defeat status on the basis of the inference graph. The latter module by itself is quite similar to a "justification-based" RMS (of which the original example was Doyle's TMS [5]). There are also some important differences, however. Standard justification-based RMSs have only one kind of link, corresponding to the inference links in the inference graph. They have no links corresponding to the defeat links, and correspondingly they are incapable of detecting undercutting defeat. They can only respond to inconsistencies, which is to say that they detect rebutting defeat. McDermott, however, constructs a more general kind of "nonmonotonic RMS" by adding dependencies between nodes that are formulated using a modal operator **L** meaning "it is definitely true that". The idea comes from McDermott and Doyle [18]. The result is not an RMS that computes defeat status in the same way OSCAR does, because McDermott explicitly opts for a credulous nonmonotonic logic. But it may be possible to use the same ideas to build a skeptical RMS that really does compute defeat statuses equivalently with OSCAR. In doing this, it should be noted that the literals upon which McDermott's RMS operates must be interpreted as expressing inferences (nodes of the inference graph) rather than propositions, but that seems to make no difference to the functionality.[6]

[6] It can also be observed that the assumption-based aspect of McDermott's RMS is unnecessary for this application. In addition, the introduction of the modal operator **L** introduces more expressive power than is required for formulating prima facie reasons. All prima facie reasons can be expressed in a single form: $\neg P \vee \neg \mathbf{L}\neg Q \vee \neg \mathbf{L}(P \otimes Q) \vee Q$ (P is a prima facie reason for Q). Accordingly, forms like ($\mathbf{L}P \vee P$) that give McDermott trouble ("odd loops") do not even arise in the formulation of prima facie reasons.

Although RMSs are quite close, at least in spirit, to OSCAR's defeat status computation based on the inference graph, it is important to realize that the defeat status computation is only part of OSCAR. RMSs are not themselves reasoners. The idea of building an i.d.e.-adequate reasoner by cycling between the monotonic reasoner and the defeat status computation on the inference graph, and the proof that this will work, takes us far beyond an RMS.

Appendix A. Implementation

A.1. The monotonic reasoner

The monotonic reasoner incorporated into the defeasible OSCAR is based upon the interest-driven deductive version of OSCAR described in [30]. Without going into details, let me indicate roughly how this reasoner works. It is based upon the simple idea that when trying to infer a specified conclusion from a set of premises, the reasoner not only works forwards from the premises but also backwards from the conclusion. The latter is just goal reduction. The reasoner begins with the set of premises *input* and a set *ultimate* of conclusions in which it is ultimately interested. The reasoner keeps two distinct databases —*adoptions* and *interests*. The former comprises the propositions believed at any given time, and the latter comprises those propositions the reasoner is trying to infer at any point. The reasoner reasons forwards from *adoptions* to new conclusions which are then inserted into *adoptions*, and backwards from *interests* to new interests. The bases for adopting new interests are recorded in *forset*. A distinction is made between those reason schemata of use in forwards reasoning (from adoptions to adoptions) and those of use in backwards reasoning (from interests to interests). Then the three basic rules governing interest-driven reasoning are:

Rule A.1 (*R-Infer*). *If $\langle \Gamma, p \rangle$ is an instance of a forwards reason schema, and for some supposition X and for some q in Γ, q is adopted relative to X and all the other members of Γ have already been adopted, then adopt p relative to X.*

Rule A.2 (*Interest-adoption*). *If $\langle \Gamma, p \rangle$ is an instance of a backwards reason schema, and for some supposition X, the system adopts interest in p relative to X, then adopt interest in the members of Γ relative to X and record the basis for the interest by inserting $\langle \Gamma, p, X \rangle$ into forset. If all the members of Γ have already been adopted relative to X, then adopt p relative to X.*

Rule A.3 (*I-Infer*). *If the system adopts p relative to X and p is in interest, then for any member $\langle \Gamma, q, X \rangle$ of forset, if $p \in \Gamma$ and the other members of Γ are already adopted relative to X, then adopt q relative to X.*

In addition, OSCAR embodies various structural rules like *conditionalization* and *reductio ad absurdum* that govern suppositional reasoning. OSCAR is complete for the predicate calculus.

For incorporation into the defeasible reasoner, the deductive reasoner is modified in four main respects.

First, a supposition is now taken to inherit all of the adoptions made relative to less inclusive suppositions, and adoptions are made explicitly in only the minimal suppositions relative to which they can be inferred. By contrast, in the deductive reasoner, when a new supposition was made, all the adoptions from less inclusive suppositions were automatically moved into the set of adoptions for the new supposition, but if a new conclusion was subsequently inferred in the new supposition from adoptions drawn from a less inclusive supposition, that new conclusion was not automatically adopted in the less inclusive supposition. Even though this sometimes required repeating the same reasoning in two or more suppositions, it made the deductive reasoner run faster. But for the i.d.e.-adequacy of the defeasible reasoner it is important to ensure that conclusions are always available simultaneously in the minimal suppositions possible and all more inclusive suppositions.

The second modification allows the reasoner to employ substantive reasons in its reasoning in addition to formal principles of logic. When the reasoner is given a problem, this now consists of a set of premises, a desired conclusion, a set of prima facie reason schemata, and a set of conclusive reason schemata. For example, it might be informed that $\ulcorner Fx \urcorner$ is a prima facie reason for $\ulcorner Gx \urcorner$, and that $\ulcorner Rx \urcorner$ is a conclusive reason for $\ulcorner Sx \urcorner$. These reasons are given in the form of a triple

$$\langle \textit{premise-set}, \textit{conclusion}, \textit{variables} \rangle ,$$

where *premise-set* is a set of formulas, *conclusion* is a formula, and *variables* is the set of variables occurring free in the schema. The reasoner then uses these substantive reasons in its reasoning, both backwards and forwards, just as it previously used principles of deductive reasoning. One complication is that in an interest-driven reasoner it must be specified whether a reason is to be used for backwards reasoning, forwards reasoning, or both. It is important that this be done properly, or relevant arguments may not be constructible by the reasoner. To take a very simple example, suppose *input* = $\{P\}$, P is a reason for Q, and Q is a reason for R, and *ultimate* = $\{R\}$. If P is designated as only a backwards reason for Q, and Q is designated as only a forwards reason for R, the reasoner will be unable to infer R from P. With any other combination of designations, the reasoner will be able to infer R from P. For instance, if P is designated as a forwards reason for Q, and Q is designated as a backwards reason for R, then the reasoner will adopt interest in Q on the basis of interest in R, and will then infer Q from P and then R from Q. I do not have a general theory to propose regarding how directions are to be assigned to reasons. More theoretical work is needed on this question. However, to avoid combinatorial explosion, a necessary condition for a forwards reason is that any variables free in the conclusion are also free in the premises, and a necessary condition for a backwards reason is that any variables free in the premises are also free in the conclusion. For now I have also adopted these as sufficient conditions for the designation. When presented with a problem, the reasoner assigns directions to the reasons on this bases. This has the consequence that most substantive reasons will be designated as both backwards and forwards reasons. Although OSCAR is complete for the predicate calculus, it is unclear whether this handling of substantive reasons makes OSCAR interest-complete.

The third modification concerns *reductio ad absurdum* and is required by the introduction of prima facie reasons into the reasoning. Reasoning by *reductio ad absurdum* appears to be essential in the predicate calculus (see the discussion in [30]), but it is not generally valid for defeasible reasoning. If a contradiction is inferred at the end of a defeasible argument, rather than justifying an inference to the negation of the premise, this will normally have the effect of defeating some of the defeasible reasoning. Thus the rule of *reductio ad absurdum* must be qualified. If we suppose $\neg P$ with the intent of inferring P from it, only deductive reasoning can be allowed within this supposition. Of the substantive reasons, the conclusive reasons can still be used within a reductio-supposition, but the prima facie reasons cannot. The results of defeasible reasoning within less inclusive suppositions can, however, still be used in the reasoning, because the contradiction in the larger supposition cannot defeat reasoning in less inclusive suppositions.

The final modification concerns the rules for interest-cancellation. In the deductive reasoner, once a conclusion is obtained, the reasoner cancels interest in both it and (except in reductio-suppositions) its negation. In a defeasible reasoner, however, this is inappropriate. Having constructed one argument for a conclusion, a reasoner cannot thereby ignore the possibility of others, because the first argument might get defeated while other arguments are not. Instead, we simply impose a restriction against circular reasoning (i.e., a proposition P cannot be inferred from prior conclusions already inferred, in part, from P). Similarly, having inferred P, the defeasible reasoner cannot thereby cancel interest in $\neg P$, because inferring the latter may just be a matter of acquiring a rebutting defeater. Accordingly, the monotonic reasoner keeps track of whether any defeasible inferences have been made in inferring a conclusion. If the reasoning has been entirely deductive, then the deductive interest-cancellation rules are employed, but otherwise not.

A further modification may ultimately be required, but has not yet been implemented. This involves changing the prioritizing scheme so that when interest is adopted in defeaters, they receive a lower priority than other interests. Implementing this may involve rather large changes, because much of OSCAR's current prioritizing scheme involves built-in structures in suppositions.

A.2. The defeasible reasoner

The defeasible reasoner results from making several additions to the monotonic reasoner. When a proposition is adopted in a particular supposition, three new operations are performed: *adopt-interest-in-defeaters*, *make-nodes*, and *update-defeat-statuses*. In addition, whenever the defeat status of a member of *ultimate* changes, the reasoner announces this fact, and if the reasoning terminates, all relevant arguments are displayed. As previously observed, it cannot be assumed that the reasoning will terminate, but the reasoner can also be interrupted at any point and asked to display all relevant arguments found so far. The three new operations mentioned above are as follows.

When the monotonic reasoner performs a defeasible inference, *adopt-interest-in-defeaters* leads it to adopt interest in defeaters for that inference.

The operation *make-nodes* builds the inference graph. The inference graph consists of a list *nodes* of nodes, where each node k is a structure containing slots for the following information:

- node proposition: a formula p,
- node supposition: a set of formulas,
- node basis: the set of formulas from which p is inferred,
- node justification: a string describing the reason for the inference,
- immediate ancestors: a set of nodes,
- node ancestors: a set of nodes,
- immediate descendants: a set of nodes,
- pf-node: t or *nil* depending upon whether k is a pf-node,
- node defeaters: a set of nodes,
- node defeatees: a set of nodes,
- defeat status: 0 if undefeated, 1 if defeated, *nil* if provisionally defeated.

When a proposition is adopted relative to a supposition, *make-node* adds a node to the inference graph encoding this adoption. When a node is added to the inference graph, its defeat relations to other nodes are computed. Then *update-defeat-statuses* recomputes the defeat statuses of all nodes using Algorithm 5.1. If this results in a change to the status of any member of *ultimate*, this is announced.

To some extent, it is arbitrary when these last two operations are performed, because they do not affect the performance of the monotonic reasoner. They could all be postponed until the operator queries the reasoner regarding the status of the members of *ultimate*, or until the reasoning terminates. This would be more efficient, but for aesthetic reasons I have chosen instead to have the reasoner perform these operations as it goes along and supply a running commentary on the status of the members of *ultimate*.

Appendix B. Examples

This appendix gives some examples of the operation of OSCAR. For simplicity, all but one of the examples are formulated in the propositional calculus, but that is not an essential feature of OSCAR. OSCAR can deal with problems in the full predicate calculus, and on deductive problems it is complete. The final (predicate calculus) example was explicitly chosen to illustrate what happens when the reasoner does not halt. The defeasible structure in that example is the same as in the previous example except that first-order inferences must be performed to find the defeaters. The reasoner quickly does the requisite reasoning, but the first-order structure of the premises forces the reasoner to continue searching for additional defeaters, which it never finds. In the following examples, the diagrams of the inference graphs are done by hand, but the rest is an actual printout of the operation of the program. The times are for the program running in Allegro Common LISP on a MacIIx. Figure B.1 shows the symbols that are used in the diagrams of the inference graphs.

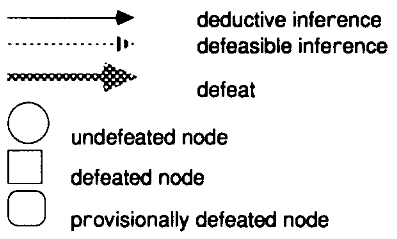
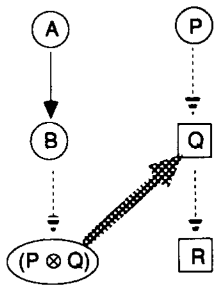


Fig. B1. Inference graph symbols.

Example 1. This is a case of undercutting defeat.

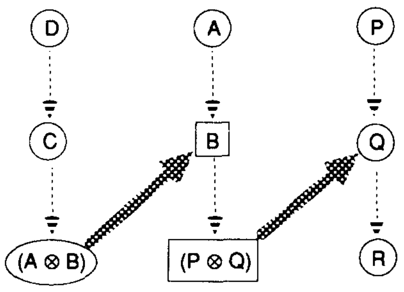
```
GIVEN: P
       A
DESIRED CONCLUSIONS:
       R

  PF-REASONS:
     P |⇒ Q
     Q |⇒ R
     B |⇒ (P ⊗ Q)
  CON-REASONS:
     A |⇒ B
```



```
••••••••••••••••••••••••••••••
A defeasible argument has been constructed for:
     R
••••••••••••••••••••••••••••••
    R is undefeated.
••••••••••••••••••••••••••••••
    R is defeated.
••••••••••••••••••••••••••••••

All the available inferences have been performed.
FINAL STATUS:
    R is unwarranted
••••••••••••••••••••••••••••••

This reasoning took 1.283 sec

==============================
The following relevant arguments were constructed:
••••••••••••••••••••••••••••••
ARGUMENT #1
This is an ultimately defeated argument for:
  R

1. P   given
4. Q   pf reason from 1
5. R   pf reason from 4

Line #4 of argument #1 is directly defeated by the
following argument:
••••••••••••••••••••••••••••••
ARGUMENT #2
This is an ultimately undefeated argument for:
  (P ⊗ Q)

2. A   given
3. B   conclusive reason from 2
6. (P ⊗ Q)   pf reason from 3
```

Example 2. This is a case of defeat and reinstatement.

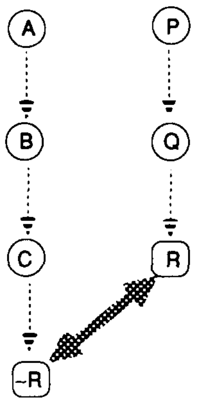
```
GIVEN: P
       A
       D
DESIRED CONCLUSIONS:
         R

  PF-REASONS:
     C ‖⇒ (A ⊗ B)
     D ‖⇒ C
     B ‖⇒ (P ⊗ Q)
     A ‖⇒ B
     Q ‖⇒ R
     P ‖⇒ Q
```



```
••••••••••••••••••••••••••••••
A defeasible argument has been constructed for:
     R
••••••••••••••••••••••••••••••
     R is justified..
••••••••••••••••••••••••••••••
     R is unjustified.
••••••••••••••••••••••••••••••
     R is justified..
••••••••••••••••••••••••••••••

All the available inferences have been performed.
FINAL STATUS:
     R is warranted
••••••••••••••••••••••••••••••

This reasoning took 2.033 sec

==============================
The following relevant arguments were constructed:
••••••••••••••••••••••••••••••
ARGUMENT #1
This is an undefeated argument for:
  R

1. P   given
6. Q   pf reason from 1
7. R   pf reason from 6

Line #6 of argument #1 is defeated by the
following argument:

••••••••••••••••••••••••••••••
ARGUMENT #2
This is a defeated argument for:
  (P ⊗ Q)

2. A   given
5. B   pf reason from 2
8. (P ⊗ Q)   pf reason from 5

Line #5 of argument #2 is directly defeated by the
following argument:
••••••••••••••••••••••••••••••
ARGUMENT #3
This is an undefeated argument for:
  (A ⊗ B)

3. D   given
4. C   pf reason from 3
9. (A ⊗ B)   pf reason from 4
```

Example 3. This is a case of collective rebutting defeat.

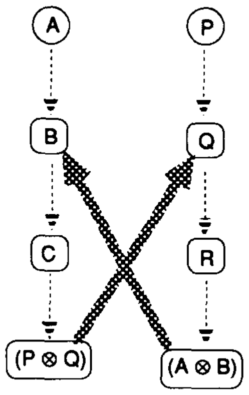
```
GIVEN: P
       A
DESIRED CONCLUSIONS:
        R

  PF-REASONS:
     A |⇒ B
     B |⇒ C
     C |⇒ ~R
     P |⇒ Q
     Q |⇒ R
```



```
••••••••••••••••••••••••••••••
A defeasible argument has been constructed for:
     R
••••••••••••••••••••••••••••••
     R is undefeated.
••••••••••••••••••••••••••••••
     R is provisionally defeated
••••••••••••••••••••••••••••••

All the available inferences have been performed.
FINAL STATUS:
     R is unwarranted
••••••••••••••••••••••••••••••

This reasoning took 1.183 sec

The following relevant arguments were constructed:
••••••••••••••••••••••••••••••
ARGUMENT #1
This is a provisionally defeated argument for:
   R

1. P    given
5. Q    pf reason from 1
6. R    pf reason from 5

Line #6 of argument #1 is directly defeated by the
following argument:
••••••••••••••••••••••••••••••
ARGUMENT #2
This is a provisionally defeated argument for:
  ~R

2. A    given
3. B    pf reason from 2
4. C    pf reason from 3
7. ~R   pf reason from 4

Line #7 of argument #2 is directly defeated by
argument #1.
```

Example 4. This is a case of hidden collective undercutting defeat (as in Fig. 2).

```
GIVEN: A
       P
DESIRED CONCLUSIONS:
        C

  PF-REASONS:
     A |⇒ B
     B |⇒ C
     C |⇒ (P ⊗ Q)
     P |⇒ Q
     Q |⇒ R
     R |⇒ (A ⊗ B)
```

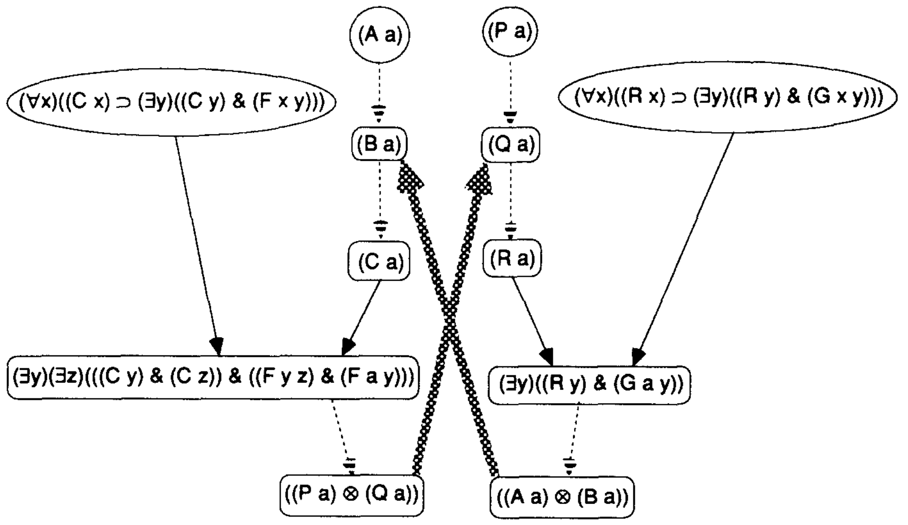


```
●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
A defeasible argument has been constructed for:
     C
●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
     C is undefeated.
●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
     C is provisionally defeated
●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●

All the available inferences have been performed.
FINAL STATUS:
     C is unwarranted
●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●

This reasoning took 1.617 sec
══════════════════════════════
The following relevant arguments were constructed:
●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
ARGUMENT #1
This is a provisionally defeated argument for:
   C

1. A    given
5. B    pf reason from 1
6. C    pf reason from 5

Line #5 of argument #1 is defeated by the
following argument:
```

```
●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
ARGUMENT #2
This is a provisionally defeated argument for:
   (A ⊗ B)

2. P    given
3. Q    pf reason from 2
4. R    pf reason from 3
8. (A ⊗ B)    pf reason from 4

Line #3 of argument #2 is directly defeated by the
following argument:
●●●●●●●●●●●●●●●●●●●●●●●●●●●●●●
ARGUMENT #3
This is a provisionally defeated argument for:
   (P ⊗ Q)

1. A    given
5. B    pf reason from 1
6. C    pf reason from 5
7. (P ⊗ Q)    pf reason from 6

Line #5 of argument #3 is directly defeated by
argument #2.
```

Example 5. This has the same defeasible structure as the previous example, but the reasoner must perform first-order inferences to find the defeaters. The first-order structure of the problem prevents the reasoner from halting. Eventually, it is halted manually by the operator, and then all relevant arguments that have been discovered are displayed.

```
GIVEN: (P a)
       (A a)
       (∀x)[(C x) ⊃ (∃y)[(C y) & (F x y)]]
       (∀x)[(R x) ⊃ (∃y)[(R y) & (G x y)]]
```


(A a)
(P a)
(∀x)((C x) ⊃ (∃y)((C y) & (F x y)))
(∀x)((R x) ⊃ (∃y)((R y) & (G x y)))
(B a)
(Q a)
(C a)
(R a)
(∃y)(∃z)(((C y) & (C z)) & ((F y z) & (F a y)))
(∃y)((R y) & (G a y))
((P a) ⊗ (Q a))
((A a) ⊗ (B a))

```
DESIRED CONCLUSIONS:
        (C a)

  PF-REASONS:
      (A x) |⇒ (B x)  (variables = x)
      (B x) |⇒ (C x)  (variables = x)
      (P x) |⇒ (Q x)  (variables = x)
      (Q x) |⇒ (R x)  (variables = x)

  CONCLUSIVE-REASONS:
      (∃ y)(∃ z)([(C y) & (C z)] & [(F y z) & (F x y)]) |⇒ ((P x) ⊗ (Q x))  (variables = x)
      (∃ y)[(R y) & (G x y)] |⇒ ((A x) ⊗ (B x))  (variables = x)
```

```
•••••••••••••••••••••••••••••••••
A defeasible argument has been constructed for:
      (C a)
•••••••••••••••••••••••••••••••••
      (C a) is undefeated.
•••••••••••••••••••••••••••••••••
      (C a) is defeated.
•••••••••••••••••••••••••••••••••
      (C a) is provisionally defeated
•••••••••••••••••••••••••••••••••
THE REASONER HAS NOT YET HALTED, BUT AFTER
580 INFERENCES, THE CURRENT STATUS IS:
      (C a) is unjustified
•••••••••••••••••••••••••••••••••
THE REASONER HAS NOT YET HALTED, BUT AFTER
997 INFERENCES, THE CURRENT STATUS IS:
      (C a) is unjustified
•••••••••••••••••••••••••••••••••
THE REASONER HAS NOT YET HALTED, BUT AFTER
1514 INFERENCES, THE CURRENT STATUS IS:
      (C a) is unjustified
•••••••••••••••••••••••••••••••••
THE REASONER HAS NOT YET HALTED, BUT AFTER
2041 INFERENCES, THE CURRENT STATUS IS:
      (C a) is unjustified
•••••••••••••••••••••••••••••••••
THE REASONER HAS NOT YET HALTED, BUT AFTER
2421 INFERENCES, THE CURRENT STATUS IS:
      (C a) is unjustified
•••••••••••••••••••••••••••••••••
THE REASONER HAS NOT YET HALTED, BUT AFTER
2807 INFERENCES, THE CURRENT STATUS IS:
      (C a) is unjustified
•••••••••••••••••••••••••••••••••
THE REASONER HAS NOT YET HALTED, BUT AFTER
3221 INFERENCES, THE CURRENT STATUS IS:
      (C a) is unjustified
•••••••••••••••••••••••••••••••••
THE REASONER HAS NOT YET HALTED, BUT AFTER
3575 INFERENCES, THE CURRENT STATUS IS:
      (C a) is unjustified
•••••••••••••••••••••••••••••••••
THE REASONER HAS NOT YET HALTED, BUT AFTER
3927 INFERENCES, THE CURRENT STATUS IS:
      (C a) is unjustified
•••••••••••••••••••••••••••••••••
THE REASONER HAS NOT YET HALTED, BUT AFTER
4329 INFERENCES, THE CURRENT STATUS IS:
      (C a) is unjustified
•••••••••••••••••••••••••••••••••
THE REASONER HAS NOT YET HALTED, BUT AFTER
4668 INFERENCES, THE CURRENT STATUS IS:
      (C a) is unjustified
•••••••••••••••••••••••••••••••••
```

```
        At this point the reasoning was
        aborted by the operator.
=================================
The following relevant arguments were constructed:
•••••••••••••••••••••••••••••••••
ARGUMENT #1
This is a provisionally defeated argument for:
    (C a)

 2.  (A a)   given
 5.  (B a)   pf reason from 2
 6.  (C a)   pf reason from 5

Node #5 in argument #1 is defeated by the following
argument:
•••••••••••••••••••••••••••••••••
ARGUMENT #2
This is a provisionally defeated argument for:
    ((A a) ⊗ (B a))

 1.  (P a)   given
 7.  (Q a)   pf reason from 1
 4.  (∀x)((R x) ⊃ (∃y)((R y) & (G x y)))   given
 8.  (R a)   pf reason from 7
 9.  (∃y)((R y) & (G a y))   all-detachment from 8,4
 13.  ((A a) ⊗ (B a))   conclusive reason from 9

Node #7 in argument #2 is defeated by the following
argument:
•••••••••••••••••••••••••••••••••
ARGUMENT #3
This is a provisionally defeated argument for:
    ((P a) ⊗ (Q a))

 2.  (A a)   given
 5.  (B a)   pf reason from 2
 3.  (∀x)((C x) ⊃ (∃y)((C y) & (F x y)))   given
 6.  (C a)   pf reason from 5
 16.  (∃y)((C y) & (F a y))   all-detachment from 6,3
 17.  ((C @@y) & (F a @@y))   EI from 16
 18.  (C @@y)   simplification from 17
 20.  (∃y)((C y) & (F @@y y))   all-detachment from 18,3
 38.  ((C @@@@y) & (F @@y @@@@y))   EI from 20
 39.  (C @@@@y)   simplification from 38
 19.  (F a @@y)   simplification from 17
 40.  (F @@y @@@@y)   simplification from 38
 41.  (∃z)(((C @@y) & (C z)) & ((F @@y z) & (F a @@y)))
     EG from 40,19,39,18
 43.  (∃y)(∃z)(((C y) & (C z)) & ((F y z) & (F a y)))   EG from 41
 44.  ((P a) ⊗ (Q a))   conclusive reason from 43

Node #5 in argument #2 is defeated by node #13 (argument
#3):
```

References

[1] A. Baker and M.L. Ginsberg, A theorem prover for prioritized circumscription, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 463–467.
[2] R.M. Chisholm, *Perceiving* (Cornell University Press, Ithaca, NY, 1957).
[3] R.M. Chisholm, *Theory of Knowledge* (Prentice Hall, Englewood Cliffs, NJ, 1st ed., 1966).
[4] R.M. Chisholm, *Theory of Knowledge* (Prentice Hall, Englewood Cliffs, NJ, 2nd ed., 1977).
[5] J. Doyle, A truth maintenance system, *Artif. Intell.* **12** (1979) 231–272.
[6] D.W. Etherington, Formalizing nonmonotonic reasoning systems, *Artif. Intell.* **31** (1987) 41–85.
[7] H. Geffner, Conditional entailment: closing the gap between defaults and conditionals, in: *Proceedings Third International Workshop on Nonmonotonic Reasoning*, South Lake, Tahoe, CA (1990) 58–72.
[8] M.L. Ginsberg, A circumscriptive theorem prover, *Artif. Intell.* **39** (1989) 209–230.
[9] E.M. Gold, Limiting recursion, *J. Symbolic Logic* **30** (1965) 28–48.
[10] J.F. Horty, R.H. Thomason and D.S. Touretzky, A skeptical theory of inheritance in nonmonotonic semantic networks, *Artif. Intell.* **42** (1990) 311–348.
[11] D. Israel, What's wrong with non-monotonic logic? in: *Proceedings AAAI-80*, Stanford, CA (1980) 99–101.
[12] H.E. Kyburg, *The Logical Foundations of Statistical Inference* (Reidel, Dordrecht, Netherlands, 1974).
[13] H.E. Kyburg, The reference class, *Philos. Sci.* **50** (1983) 374–397.
[14] H.J. Levesque, All I know: a study in autoepistemic logic, *Artif. Intell.* **42** (1990) 263–309.
[15] F. Lin and Y. Shoham, Argument systems: a uniform basis for nonmonotonic reasoning, Tech. Rept. No. STAN-CS-89-1243, Department of Computer Science, Stanford University, Stanford, CA (1990).
[16] R.P. Loui, Defeat among arguments: a system of defeasible inference, *Comput. Intell.* **3** (1987) 100–106.
[17] D. McDermott, A general framework for reason maintenance, *Artif. Intell.* **50** (1991) 289–329.
[18] D. McDermott and J. Doyle, Non-monotonic logic I, *Artif. Intell.* **13** (1980) 41–72.
[19] D. Nute, Basic defeasible logic, in: L. Farinas-del-Cerro and M. Pentonnen, eds., *Intensional Logics for Programming* (1990).
[20] D. Nute and M. Lewis, A users manual for d-Prolog, ACMC Research Report 01-0016, The University of Georgia, Athens, GA (1986).
[21] J. Pollock, Implication and analyticity, Ph.D. Dissertation, University of California, Berkeley, CA (1965).
[22] J. Pollock, Criteria and our knowledge of the material world, *Philos. Rev.* **76** (1967) 28–60.
[23] J. Pollock, The structure of epistemic justification, in: American Philosophical Quarterly Monograph Series **4** (Blackwell, Oxford, 1970) 62–78.
[24] J. Pollock, *Knowledge and Justification* (Princeton University Press, Princeton, NJ, 1974).
[25] J. Pollock, Reasons and reasoning, APA Symposium, Denver, CO, Unpublished (1979).
[26] J. Pollock, *Contemporary Theories of Knowledge* (Rowman and Littlefield, Totowa, NJ, 1986).
[27] J. Pollock, Defeasible reasoning, *Cogn. Sci.* **11** (1987) 481–518.
[28] J. Pollock, *How to build a person* (Bradford/MIT Press, Cambridge, MA, 1989).
[29] J. Pollock, *Nomic Probability and the Foundations of Induction* (Oxford University Press, New York, 1990).
[30] J. Pollock, Interest-driven suppositional reasoning, *J. Autom. Reasoning* **4** (1990) 419–462.
[31] J. Pollock, A theory of defeasible reasoning, *Int. J. Intell. Syst.* **6** (1991) 33–54.
[32] J. Pollock, Self-defeating arguments, *Minds Mach.* **1** (1991) 367–392.
[33] H. Putnam, Trial and error predicates and the solution to a problem of Mostowski, *J. Symbolic Logic* **30** (1965) 49–57.
[34] R. Reiter, A logic for default reasoning, *Artif. Intell.* **13** (1980) 81–132.
[35] D.S. Touretzky, J.F. Horty and R.H. Thomason, A clash of intuitions: the current state of nonmonotonic multiple inheritance systems, in: *Proceedings IJCAI-87*, Milan, Italy (1987) 476–482.