# Implementing Defeasible Reasoning

John L. Pollock
Department of Philosophy
University of Arizona
Tucson, Arizona 85721
(e-mail: pollock@arizona.edu)

## 1. The Problems

In defeasible reasoning, arguments are constructed employing not only logical inference rules but also defeasible inference rules, or as I will call them, *defeasible reasons*. Defeasible reasons have defeaters. Given an agent that reasons defeasibly, we can collect together all of the arguments it has constructed into an inference graph, where the nodes are the conclusions, and the links between the nodes are of two sorts: support-links and defeat-connections. Figure 1 illustrates a simple inference graph, where '→' symbolizes inference relations, and '⇢' symbolizes defeat relations.

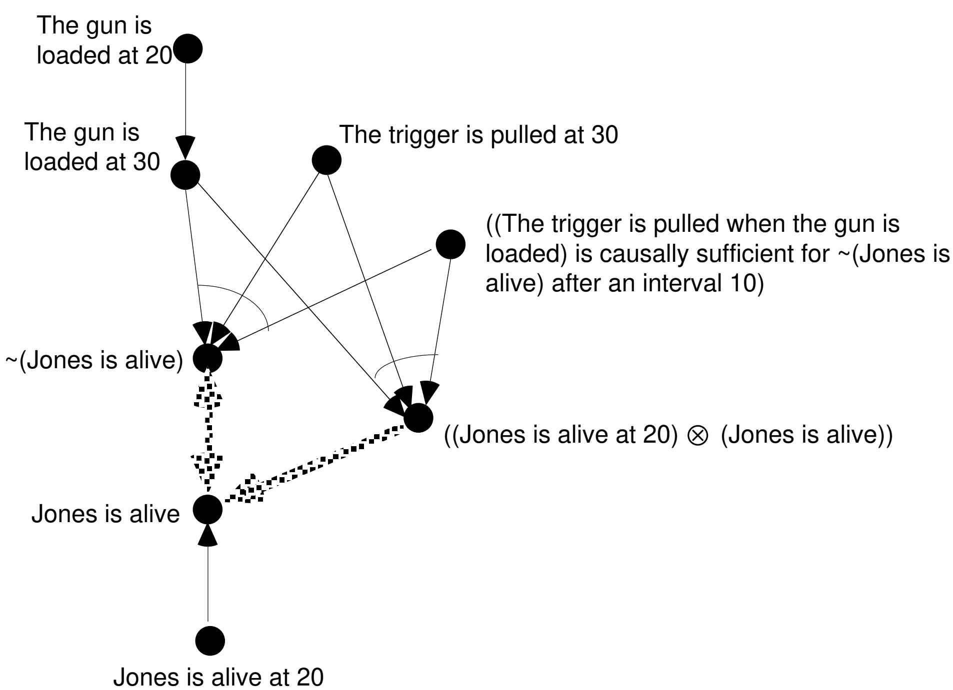


Figure 1. The inference graph for the Yale Shooting Problem

There are four main problems for implementing defeasible reasoning: (1) How are defeasible arguments to be constructed? (2) Some defeasible reasons are stronger reasons than others, and some information is input to the reasoner with greater confidence than other. How do we use these varying strengths to compute a strength for the conclusion of a defeasible argument? (3) Given a set of arguments of varying strengths, some of which support defeaters for others, how do we decide which arguments are defeated and which are undefeated? (4) What do we do about the fact that on almost any conception of defeasible reasoning, the set of warranted conclusions is not recursively enumerable? I am going to tell you how these problems are solved in OSCAR. Some of what I will tell you is covered in my recent book *Cognitive Carpentry*, and some of it postdates that. A lot of the more recent material is discussed in the *OSCAR Manual*, and I encourage you to download that and the implemented system to get more detail.

## 2. Two Concepts of Defeasibility

Let us suppose, for the moment, that we have an account of how to construct defeasible arguments, and we have a principle that determines which conclusions are defeated and which are undefeated given a set of arguments supporting all of them and possibly supporting defeaters for some of the arguments in the set. Consider an agent that reasons in accordance with these principles. Formally, we can regard the agent's reasoning as consisting of a set **B** of conclusions drawn thus far and an *update* function. *update*(**B**,**P**) is the set of beliefs to which the agent should move if its current set of beliefs is **B** and its current set of inputs is **P**. If we suppose the inputs are held fixed, we can define:

$$\mathbf{J}_0 = \mathbf{B};$$
$$\mathbf{J}_{n+1} = \textit{update}(\mathbf{J}_n,\varnothing).$$

Then $\mathbf{J}_n$ is the set of conclusions drawn after $n$ steps of reasoning. These conclusions, and the arguments supporting them, can be put into an inference graph that encodes the agent's current epistemic state. Let $\mathcal{G}_n$ be the inference-graph representing the agent's epistemic state after $n$ steps of reasoning. We can also construct another inference graph. This graph consists of all possible arguments that can be constructed given the agent's current inputs. Let us call this inference-graph $\mathcal{G}_\omega$. Let us say that a conclusion is *warranted* iff it is supported by an undefeated argument in $\mathcal{G}_\omega$. The warranted conclusions are in a sense the target at which the reasoner is aiming. Ideally, a reasoner would like to draw all and only warranted conclusions.

Unfortunately, this ideal is impossible. It was first observed 15 years ago (by both Israel and Reiter) that on almost any conception of defeasible or nonmonotonic reasoning, the set of warranted conclusions is not recursively enumerable. The basic idea behind this observation is a simple one. A necessary condition for a defeasibly supported conclusion to be warranted is that its negation not be a theorem of logic. Thus if we are reasoning in a rich enough formalism that logical consistency is undecidable, e.g., in first-order logic, then there can be no effective procedure for ensuring that a conclusion is warranted, and hence it is impossible to build a system that generates all and only warranted conclusions. In other words, the set of warranted conclusions is not recursively enumerable.

Familiar automated theorem provers for formal logic generate all and only valid conclusions. This is possible only because the set of valid conclusions for first-order logic is recursively enumerable. This means that an automated defeasible reasoner cannot look like an automated theorem prover. It must do something different. But what can it do? This has often seemed to be an insurmountable obstacle for the construction of an automated defeasible reasoner.

But notice—this mathematical result does not prevent human beings from reasoning defeasibly. How do we do it? The key to answering this question is to distinguish between two concepts of defeasibility. Human reasoning is defeasible in two different senses. It is *synchronically defeasible* in the sense that a conclusion can be warranted relative to one set of perceptual inputs, and unwarranted relative to a larger set of inputs. For instance, the initial set of inputs may support an inductive generalization that is defeated in light of the larger set of inputs. Human reasoning is also *diachronically defeasible* in the sense that we form beliefs provisionally on the basis of our current reasoning, but we may retract them later just as a result of further reasoning, without any new input. Beliefs that are reasonable to hold on the basis of our current reasoning are said to be *justified. Justification* concerns what we should do epistemically here and now; *warrant*, by contrast, concerns the ideal limit at which our reasoning aims. A conclusion can be justified at one stage of reasoning and unjustified at a later stage, without any additional input. This results from the fact that as reasoning progresses, defeaters may be inferred for earlier reasoning. Note that synchronic defeasibility concerns warrant, whereas diachronic defeasibility concerns justification.

Studies of defeasibility (or nonmonotonicity) in AI have focused on synchronic defeasibility. Although it is clear that human reasoning is also diachronically defeasible, it may well be wondered whether this is an inessential feature of human cognition. Could we build a rational agent whose reasoning was not diachronically defeasible but for which, relative to any set of inputs, the set of warranted propositions was the same as for human beings? The answer is that we could not. That would only be possible if the set of warranted conclusions were recursively enumerable, and it is not.

If a defeasible reasoner cannot work like a theorem prover, systematically grinding out conclusions that will be retained for all time, then how should it work? My proposal is that we take diachronic defeasibility seriously. 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". My proposal (Pollock [1989]) is that we understand this in the following sense:

> 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 undefeated and stays undefeated;
> (2) if $p$ is unwarranted then the reasoner will eventually reach a stage where $p$ is defeated and stays defeated.

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. Define:

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

I will say that the sequence of recursive sets $f(n)$ is a *defeasible enumeration* of $A$. Defeasibly enumerable sets are the same as those Gold [1965] calls "limiting recursive" and Putnam [1965] calls "trial and error". Both authors establish that a set is of this type iff it is $\Delta_2$ 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 recursively enumerable, then there is an effectively computable sequence of recursive 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 only 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.

My proposal regarding reasoning and warrant is that the set of warranted propositions should be defeasibly enumerable, and the rules for reasoning should be rules for successively approximating warrant in this way, i.e., they are rules for constructing a defeasible enumeration. More accurately:

> If $\mathbf{J}_n$ is the set of propositions justified after $n$ applications of *update* to the premises input, the reasoner is *d.e.-adequate* iff the sequence of sets $\mathbf{J}_n$ is a defeasible enumeration of the set of warranted propositions.

I propose d.e.-adequacy as the primary criterion of adequacy for a defeasible reasoner. It is shown in *Cognitive Carpentry* that if certain reasonable conditions are satisfied then the set of warranted propositions is defeasibly enumerable and hence d.e.-adequate reasoners are possible. OSCAR is based upon such a reasoner.

If a reasoner is d.e.-adequate, there will be cases in which it will never stop reasoning. Any given proposition may be concluded, retracted, and reinstated many times. Every warranted proposition will eventually reach a point where it is believed and is never subsequently retracted, and every unwarranted proposition will eventually reach a point where it is not believed and never subsequently becomes believed, 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 forever in a possibly fruitless search for defeating arguments. This, of course, is just the way human beings work. This highlights the distinction between synchronic and diachronic defeasibility. Although human reasoning is diachronically defeasible, and hence any particular reasoning might have to be retracted later, 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 have generally supposed that 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.

This has important consequences for the architecture of a defeasible reasoner. It must work by generating arguments and updating defeat statuses as new arguments are constructed. These are two largely independent processes, so we can think of a defeasible reasoner as consisting of two somewhat independent modules—an argument generator, and a defeat-status computer. Let us examine these modules. I will begin with the computation of defeat statuses.

# 3. The Weakest-Link Principle

Some reasons are better than others, and some premises are input with greater confidence than others. Let us suppose we can measure reason-strengths and the degrees of justification of premises with real-numbers. The strengths of the reasons used and the degrees of justification of the premises surely effect the degree of support of the conclusion of an argument. The degrees of support for the conclusions of arguments are in turn relevant to the adjudication of disputes and the computation of defeat statuses. For example, given arguments for two incompatible conclusions, the stronger argument ought to win. The question then is, how is the strength of an argument related to the strengths of the reasons used in it and the degrees of justification of its premises?

A common and seductive model would have it that argument strength can be modeled by the probability calculus. On this view, the strength a conclusion gains from the premises can be computed in accordance with the probability calculus from the strength of the reason (a conditional probability) and the probabilities of the premises. I, and numerous other authors, have argued against this view at length, but it has a remarkable ability to keep attracting new converts. There is a simple argument against the probabilistic model that I think is particularly telling. This model would make it impossible to be justified in believing a conclusion on the basis of a long argument from numerous uncertain premises. This is because as you conjoin premises, if degrees of support work like probabilities, the degree of support goes down. Suppose you have 100 independent premises, each highly probable, say, having probability .99. According to the probability calculus, the probability of the conjunction will only be .37, so we could never be justified in using these 100 premises conjointly in drawing a single conclusion. But this flies in the face of human practice. Consider an engineer building a bridge. The design of the bridge will turn upon at least 100 independent premises regarding such things as material strengths, soil conditions, expected weather, measurements of the distance to be spanned, etc. If the probabilistic model were correct, it would be impossible to build bridges, but clearly it is not. The probabilistic model cannot be an accurate model of the way humans reason. Of course, it is always possible to argue that humans don't reason correctly, and that a theoretically adequate reasoner would conform to the probabilistic model. But it is a little hard to see how a theoretically adequate reasoner could be an improvement on human beings if it could not build bridges.

The next thing to notice is that this difficulty can be generalized. It applies to more than the probabilistic model. Any theory having the consequence that the degree of support provided by a deductive argument degrades as the number of premises increases will run into the same problem. The only alternative is to endorse the *Weakest Link Principle*, according to which a deductive argument is as good as its weakest link. More precisely:

> The degree of support of the conclusion of a deductive argument is the minimum of the degrees of justification of its premises.

The above formulation of the weakest link principle applies only to deductive arguments, but we can use it to obtain an analogous principle for defeasible arguments. If $P$ is a prima facie reason for $Q$, then we can use conditionalization to construct a simple defeasible argument for the conclusion $(P \supset Q)$, and this argument turns upon no premises:

| Suppose $P$
| Then (defeasibly) $Q$.

Therefore, $(P \supset Q)$.

As this argument has no premises, the degree of support of its conclusion should be a function of nothing but the strength of the prima facie reason. The next thing to notice is that any defeasible argument can be reformulated so that prima facie reasons are only used in subarguments of this form, and then all subsequent steps of reasoning are deductive. The conclusion of the defeasible argument is thus a deductive consequence of the premises together with a number of conditionals justified in this way. By the weakest link principle for deductive arguments, the degree of support of the conclusion should then be the minimum of (1) the degrees of justification of the premises used in the argument and (2) the strengths of the prima facie reasons. This yields *The Weakest Link Principle for Defeasible Arguments*:

> The degree of support of the conclusion of a defeasible argument is the minimum of the strengths of the defeasible reasons employed in it and the strengths of the input states to which it appeals.

The problem of computing degrees of support is thus computationally simple. This is the way they are computed in OSCAR.

# 4. Computing Defeat Statuses

Given a set of arguments some of which may support defeaters for inferences contained in others, we require an account of which of these arguments are defeated and which are undefeated. My analysis of defeat-status has evolved over the years, largely in an attempt to handle some intuitively complicated cases like the paradox of the preface and cases involving self-defeat (where an argument may defeat some of its own steps). I now believe that I have an adequate account of these phenomena, and my account is defended in *Cognitive Carpentry* (chapter three).

To formulate the analysis, let us begin by constructing inference-graphs in such a way that if the same conclusion is obtained by two different arguments, this is represented by two different nodes of the graph. In this way, we can take the nodes to have univocal defeat-statuses. In effect, each node encodes an argument for the conclusion represented at that node. Understanding inference-graphs in this way, the defeat-status analysis is as follows. We first define:

> A node of the inference-graph is *initial* iff its node-basis and list of node-defeaters is empty.

> $\sigma$ is a *partial status assignment* iff $\sigma$ is a function assigning "defeated" and "undefeated" to a subset of the nodes of an inference-graph in such a way that:
> 1. $\sigma$ assigns "undefeated" to all initial nodes;
> 2. $\sigma$ assigns "undefeated" to a node $n$ iff $\sigma$ assigns "undefeated" to all the members of the node-basis of $n$ and $\sigma$ assigns "defeated" to all node-defeaters of $n$; and
> 3. $\sigma$ assigns "defeated" to a node $n$ iff either some member of the node-basis of $n$ is assigned "defeated", or some node-defeaters of $n$ is assigned "undefeated".

σ is a *status assignment* iff σ is a partial status assignment and σ is not properly contained in any other partial status assignment

A node is undefeated iff every status assignment assigns "undefeated" to it; otherwise it is defeated.

It is shown in *Cognitive Carpentry* that this analysis is able to handle all of the problem cases that motivated it.

To illustrate the analysis, consider what I have termed "the lottery paradox paradox". The standard lottery paradox is generated by supposing that a proposition *R* describing the lottery (it is a fair lottery, has one million tickets, and so on) is justified. Given that *R* is justified, we get collective defeat for the proposition that any given ticket will not be drawn. But upon reflection, it can seem problematic how *R* can be justified. Normally, we will have only a defeasible reason for believing *R*. For instance, we may be told that it is true, or read it in a newspaper. Let $T_i$ be the proposition that ticket *i* will be drawn. In accordance with the standard reasoning involved in the lottery paradox, we can generate an argument supporting ~*R* by noting that the $\sim T_i$ jointly entail ~*R*. This is because if none of the tickets is drawn then the lottery is not fair. This is diagrammed in figure 2. The difficulty is now that ~*R* rebuts *R*. Thus these nodes defeat one another, which suggests that neither is defeated outright. In other words, the inference to *R* is provisionally defeated. This result is intuitively wrong. Obviously, if we consider examples of real lotteries (e.g., this week's New York State Lottery), it is possible to become justified in believing *R* on the basis described. This example illustrates the phenomenon of "self-defeat", wherein a node defeats one of its own inference-ancestors. Such a node turns out to be defeated automatically. In this case it is ~*R* that is defeated. This results from the fact that in figure 2, there are one million status assignments — one assigning "undefeated" to each $\sim T_i$ , but every such assignment assigns "defeated" to ~*R*.

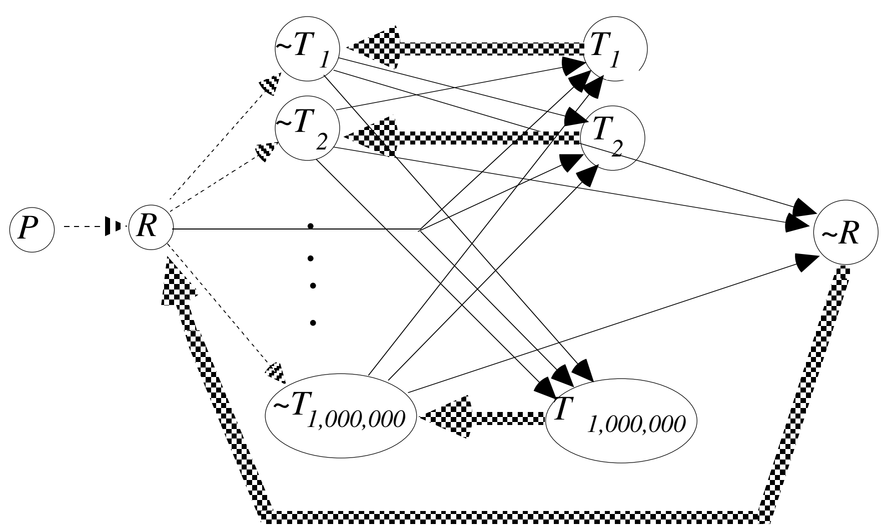


**Figure 2.** The Lottery Paradox Paradox.

Thus far I have required inference-graphs to be such that different arguments for the same conclusion are represented by different nodes. This made it possible to formulate the defeat-status analysis rather simply. However, for the purpose of implementing defeasible reasoning, this is an inefficient representation of the reasoning, because it leads to needless duplication. If we have two arguments supporting a single conclusion, then any further reasoning from that conclusion will generate two different nodes. If we have two arguments for each of two conclusions, and another inference proceeds from those two conclusions, the latter will have to be represented by four different nodes in the inference-graph, and so on. This is illustrated in figure 3, where *P* and *Q* are each inferred in two separate ways, and then *R* is inferred from *P* and *Q*.

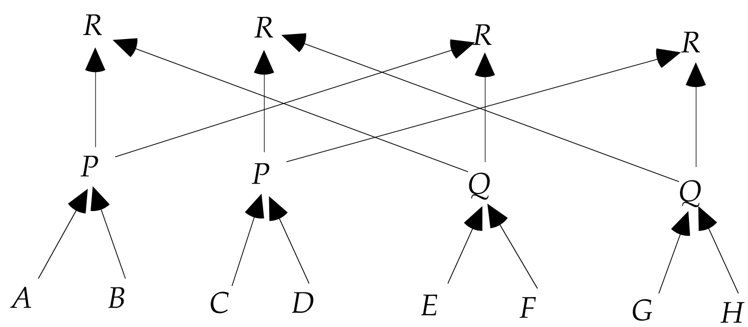


**Figure 2.** An inference-graph with multiple arguments for a single conclusion.

A more efficient representation of reasoning would take the inference-graph to be an and/or graph rather than a standard graph. In an and/or graph, nodes are linked to *sets* of nodes rather than individual nodes. This is represented diagramatically by connecting the links with arcs. In an and/or inference-graph, when we have multiple arguments for a conclusion, the single node representing that conclusion will be tied to different bases by separate groups of links. This is illustrated in figure 3 by an and/or inference-graph encoding the same reasoning as the standard inference-graph in figure 2.

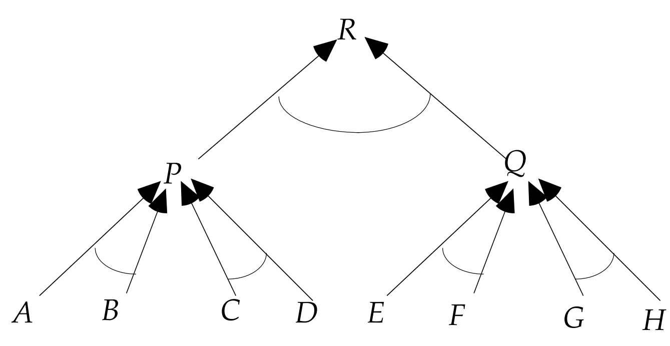


**Figure 3.** An and/or Inference-graph.

In an and/or inference-graph, a *support-link* will be a *set* of supporting-arrows connected with an arc.

We can construct a defeat-status analysis for and/or inference-graphs that is equivalent to that for simple inference-graphs. The trick is to take the defeat-status of a node to be its undefeated-degree-of-support rather than just "defeated" or "undefeated". If all arguments supporting a node are defeated, then the undefeated-degree-of-support is 0. Otherwise, it is the maximum of the strengths of the undefeated arguments. The analysis of defeat-status for and/or inference-graphs is then as follows:

A node of the inference-graph is *initial* iff its node-basis and list of node-defeaters is empty.

$\sigma$ is a *partial status assignment* iff $\sigma$ is a function assigning real numbers between 0 and 1 to a subset of the nodes of an inference-graph and to the support-links of those nodes in such a way that:
1. $\sigma$ assigns its node-strength to any initial node;
2. If $\sigma$ assigns a value $\alpha$ to a defeat-node for a support-link and assigns a value less than or equal to $\alpha$ to some member of the link-basis, then $\sigma$ assigns 0 to the link;
3. Otherwise, if $\sigma$ assigns values to every member of the link-basis of a link and every link-defeater for the link, $\sigma$ assigns to the link the minimum of the strength of the link-rule and the numbers $\sigma$ assigns to the members of the link-basis.
4. If every support-link of a node is assigned 0, the node is assigned 0;
5. If some support-link of a node is assigned a value greater than 0, the node is assigned the maximum of the values assigned to its support-links.
6. If every support-link of a node that is assigned a value is assigned 0, but some support-link of the node is not assigned a value, then the node is not assigned a value.

$\sigma$ is a *status assignment* iff $\sigma$ is a partial status assignment and $\sigma$ is not properly contained in any other partial status assignment

It is possible to design a reasonably efficient generate-and-test algorithm for computing status assignments for and/or graphs. This proceeds by first computing assignments to as many nodes and support-links as possible on the basis of the assignments to the initial nodes. If some support-links remain unassigned, assignments are made to them arbitrarily and tested for consistency. Reasoning with and/or graphs is then more efficient than reasoning with simple inference-graphs. However the algorithm for computing defeat-statuses in and/or inference-graphs also encounters a residual efficiency problem. When it has to try arbitrary assignments to support-links whose assignments cannot be computed from already-assigned nodes and links, it will consider all possible combinations of assignments to each of the support-links that must be treated in that way. This has the result that if we present multiple copies of the same problem to the algorithm simultaneously, the number of arbitrary assignments it considers will increase exponentially rather than linearly.

There is a better way of computing defeat-statuses. The different parts of the composite problem are really separate subproblems, and it should be possible to separate the computations, resulting in a linear increase in computational difficulty rather than an exponential one. Note that in realistic cases, this will be important, because the inference-graph of a real agent will generate a large base-assignment consisting of the assignment-closure of an assignment of "undefeated" to the initial nodes, and then a possibly large number of small independent problems consisting of collective defeat and kindred phenomena, each generating multiple extensions to the base assignment. It should be possible to handle each of these sets of multiple extensions independently, rather than lumping them all together.

Let us define the inference/defeat-ancestors of a support-link to be the set of inference-nodes and support-links that can be reached by working backwards via support-links and defeat-links. Precisely:

*node* is an *inference/defeat-ancestor* of *link* iff either (1) *node* is a member of the support-link-basis of *link* or of a support-link that is an inference-defeat-ancestor of *link*, or (2) *node* is a defeater for *link* or for a support-link that is an inference-defeat-ancestor of *link*.

A support-link *L* is an *inference/defeat-ancestor* of *link* iff *L* is a support-link for an inference-node that is an inference/defeat-ancestor of *link*.

The inference/defeat-ancestors of a node are its support-links and their inference/defeat-ancestors.

In computing the defeat-status of a node or link, the only nodes and links that are relevant are its inference/defeat-ancestors. When we are presented with different subproblems embedded in a single inference-graph, we can divide the inference-graph into regions as in figure 4. Here *D* is the set of nodes and links assigned statuses by the base-assignment. The "triangle-sets" *A, B,* and *C* represent isolated subproblems. They are characterized by the fact that they are minimal sets *X* such that every inference/defeat-ancestor of a member of *X* is in either *X* or *D*. For the purpose of computing the defeat-statuses of the members of the triangle-sets, we can treat the $A \cup D$, $B \cup D$, and $C \cup D$ as separate inference-graphs. This will yield the same value for the defeat-status as a computation based upon the entire inference-graph.

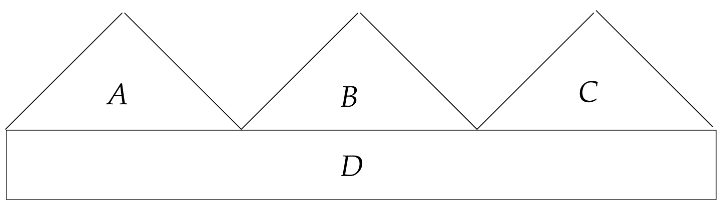


**Figure 4.** Triangle-sets.

The construction of triangle-sets can be iterated, because there may be further triangle-sets that sprout above the existing ones. Notice, however, that higher-level triangle-sets must be relativized to assignments to the lower-level triangle-sets rather than to the lower-level triangle-sets themselves. This is because different assignments to a lower-level triangle-set may leave different sets of nodes unassigned and candidates for inclusion in a higher-level triangle-set.

A simple way to conceptualize this is in terms of an *assignment-tree*. The base of an assignment-tree is the base-assignment. That divides into a set of tree-nodes representing triangle-sets. Each of those tree-nodes gives rise to a set of assignments. Those assignments divide in turn into a new set of tree-nodes representing triangle-sets, and so on, as diagrammed in figure 5. This can be accommodated formally by taking an *assignment-tree* to be a data-structure consisting of an assignment and the list of resulting triangle-sets, and taking a *triangle-set* to be a data-structure consisting of a set of inference-nodes and support-links (its domain) and the list of assignment-trees generated by the assignment-closures of maximal-consistent assignments to the inference-nodes and support-links in the triangle-sets. For the purpose of evaluating the defeat-statuses of the nodes of the inference-graph, we can construct the assignment-tree *global-assignment-tree* generated by the base-assignment. Then an inference-node *node* is undefeated iff the inference-graph contains an argument for *node* which is such that no assignment in *global-assignment-tree* assigns 0 or "unassigned" to any nodes or links in the argument

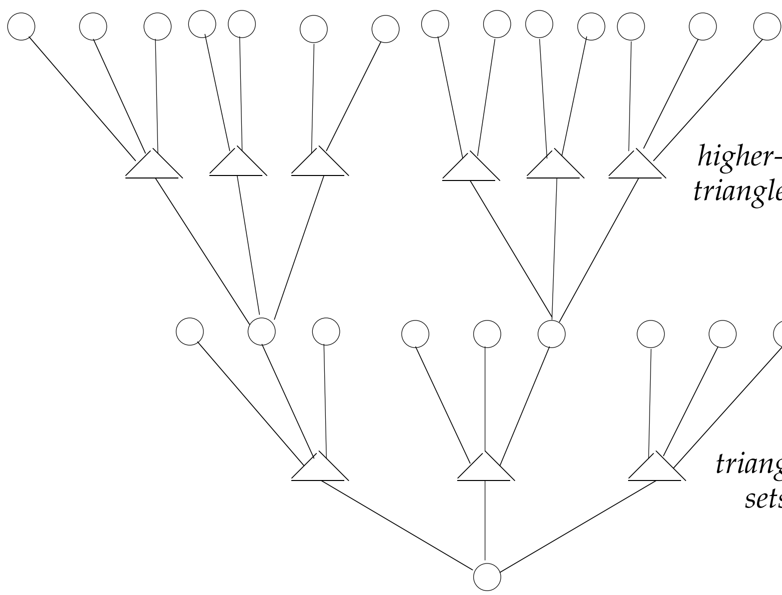


**Figure 5.** An assignment-tree.

The generate-and-test algorithm for computing status-assignments for and/or graphs can be modifed to apply to assignment-trees, and this constitutes an efficient way of computing defeat-statuses for an inference-graph that has already been constructed. However, that is an artificial problem. A rational agent must construct its inference-graph one support-link at a time, and update the defeat-statuses with the addition of each link. The result of updating the assignment-tree with the addition of each link should be the same as the result of computing the assignment-tree as above for the completed inference-graph.

One of the virtues of the assignment-tree algorithm is that it leads to an efficient procedure for updating the defeat-statuses of the nodes of the inference-graph in response to adding new support-links. When a link is added to the inference-graph, the only pre-existing nodes and links that can be affected are the inference/defeat-descendants of the support-link-target of the new link. The latter concept is defined as follows:

> The *inference/defeat-descendants* of a support-link consist of its target and all of the inference/defeat-descendants of its target; the *inference/defeat-descendants* of an inference-node consist of (1) all the consequent-links of the node and all the inference/defeat-descendants of those consequent-links, and (2) all the node-defeatees of the node and all the inference/defeat-descendants of those node-defeatees.

When we add new links, the changes to the inference-graph are confined to the new links and their inference/defeat-descendants. Call these the *affected-nodes* and *affected-links*. If the affected-nodes have no defeatees, we can just compute the assignments to the affected-nodes and affected-links at any point in the assignment tree at which they become computable, and add those assignments to the assignment-tree. Otherwise, we must update *global-assignment-tree*. In the unusual case in which some of the affected-nodes and affected-links occur in the base assignment, then we must recompute the entire assignment-tree. But normally that will not be the case. Then we can just walk up through the tree until we come to the first triangle-sets containing affected-links. We need only recompute the parts of the tree that branch above such triangle sets. This makes the updating problem much easier.

# 5. Constructing Defeasible Arguments

Thus far I have simply assumed that we have an automated reasoner capable of producing defeasible arguments. Now let us turn to the question of how to construct such a reasoner. Most automated deductive reasoners are based upon resolution-refutation, or some variant thereof. However, an automated defeasible reasoner cannot be so based. The difficulty is that resolution refutation proceeds by *reductio-ad-absurdum*. Given a set of premises and a desired conclusion, the conclusion is negated, added to the premises, and the reasoner tries to derive a contradiction from the enlarged set of premises. However, if the reasoner is allowed to use defeasible reasons in the course of its reasoning, then inferring a contradiction does not establish the falsity of one of the premises. All it does is defeat the defeasible inferences. For example, suppose I am in a windowless room, and Jones, whom I regard as reliable, tells me that it is raining outside. This gives me a defeasible reason for thinking it is raining. But then Smith, whom I regard as equally reliable, tells me that it is not raining. This gives me a defeasible reason for thinking it is not raining. From these contradictory conclusions I cannot infer that Jones and Smith did not after all tell me what they told me. All that follows is that my defeasible conclusions about the weather are defeated.

Apparently resolution-refutation is not a viable candidate for the form of a defeasible reasoner. A better choice would seem to be natural deduction, and this is the course followed in OSCAR. OSCAR uses a system of natural deduction that is, among other things, complete for first-order logic. Natural deduction is based upon two central ideas. First, reasoning is bi-directional. It begins with some initial premises, and some desired conclusions. The premises are encoded as nodes of the inference-graph, and the desired conclusions are adopted as "interests". Forwards reasoning draws new conclusions from the premises, and backwards reasoning adopts new interests on the basis of the given interests. Reasoning proceeds forwards from conclusions and backwards from interests until the two chains of reasoning come together, and then the interests are "discharged" by the conclusions, allowing the drawing of new conclusions that satisfy the remaining interests. The second central idea is that backwards reasoning can lead to the making of suppositions. For example, given an interest in a condition $(p \supset q)$, the natural-deduction strategy is to suppose $p$ and adopt interest in $q$ relative to that supposition. Thus the objects about which natural deduction reasons are sequents rather than formulas, where sequents are pairs consisting of a supposition (a set of formulas) and a formula. I will write a sequent affirming $p$ relative to the supposition $X$ as "$p/X$".

Suppositions are principally of interest in deductive reasoning, and play little role in defeasible reasoning. But bi-directional reasoning is of great importance in the way in which OSCAR constructs defeasible arguments. Bi-directional reasoning is accomplished in OSCAR by having two kinds of reasons—forwards reasons and backwards reasons. Forwards reasons generate conclusions and backwards reasons generate interests. In deductive reasoning, we can for the most part think of forwards reasons as leading from conclusions instantiating the premises of the reason to a conclusion instantiating the conclusion of the reason. We can similarly think of backwards reasons as leading from an interest in the conclusion of the reason to interests in the premises of the reason. It is natural to regard introduction rules as backwards reasons and elimination rules as forwards reasons. For example, *simplification* tells us that from (P&Q) we can infer P, and this is best regarded as a forwards reason. *Adjunction*, on the other hand, tells us that we can infer (P&Q) from P and Q, and this is best regarded as a backwards reason.

This suggests regarding reasons as having premises and a conclusion, and backwards reasons lead from interests to interests while forwards reasons lead from conclusions to conclusions. However, there is one standard inference rule that does fit this simple model. That is *reductio-ad-absurdum*. The form of *reductio* used in OSCAR is as follows:

> Given an interest in p/X, suppose ~p. Then for every conclusion q/Y where $Y \subseteq X \cup \{\sim p\}$, adopt interest in ~q/$X \cup \{\sim p\}$.

This is a backwards reason, but it must be regarded as having two different kinds of premises. The conclusion is p, and the premises are q and ~q/{~p}. However, the reason does not instruct the reasoner that when interest is adopted in p, then interest should be adopted in both q/{~p} and ~q/{~p}. Rather, the instruction is that if q/{~p} has already been concluded, then interest should be adopted in ~q/{~p}. We can express this by saying that *reductio-ad-absurdum* has two premises—one a *forwards-premise* and the other a *backwards-premise*. In deploying a backwards reason, the reasoner does not adopt interest in the forwards-premises. Only the backwards-premises give rise to interests, and they only do so when conclusions have already been drawn that instantiate the forwards-premises.

We can distinguish between three different kinds of backwards-reasons. *Simple backwards-reasons* have only backwards-premises—no forwards premises. In formal logic, most backwards-reasons are simple backwards-reasons, but my experience has been that in defeasible reasoning, there are few simple backwards-reasons. *Mixed backwards-reasons* are backwards-reasons having both forwards- and backwards-premises. There can also be *degenerate backwards-reasons* that have only forwards-premises. Given an interest in the conclusion of a degenerate backwards-reason, if the reasoner draws conclusions instantiating the forwards-premises, then a conclusion will be drawn instantiating the conclusion of the reason, but no new interests will be adopted. For example, the principle of temporal projection that I will be talking about in my lecture later this week can be stated informally as follows:

> *TEMPORAL-PROJECTION*
> If P is temporally-projectible and $\Delta t < \log(.5)/\log(\rho)$ then believing P-at-t is a defeasible reason of strength $2\cdot(\rho^{\Delta t} - .5)$ for the agent to believe P-at-(t+$\Delta$t).

This reason is implemented as a degenerate backwards-reason. Given an interest in "p at time" the reasoner checks to see whether it has a conclusion of the form "p at $time_0$" where $time_0$ < time. If so it infers "p at time", but it does not adopt interest in "p at $time_0$" for all possible choices of $time_0$. That would be a computational nightmare. The reasoner simply becomes alert for such conclusions, without actively trying to establish them. Temporal projection is implemented in OSCAR as follows:

```
(def-backwards-reason TEMPORAL-PROJECTION
  :conclusions  "(p at time)"
  :condition  (and (temporally-projectible p) (numberp time))
  :forwards-premises
     "(p at time0)"
  :backwards-premises
      "(time0 < time)"
      "((time* - time0) < 693)"
  :variables p time0 time
```

```
:defeasible? T
:strength (- (* 2 (expt *temporal-decay* (- time time0))) 1)
:description
"It is defeasibly reasonable to expect temporally projectible truths to remain unchanged.")
```

Forwards-reasons can also have both backwards- and forwards-premises. *Simple forwards-reasons* have only forwards-premises. Given conclusions instantiating the forwards-reasons, a conclusion is drawn instantiating the conclusion of the reason. *Mixed forwards-reasons* have both forwards- and backwards-premises. Given conclusions instantiating the forwards-reasons, interest is adopted in the backwards-premises, and when those interests are discharged then a conclusion is drawn instantiating the conclusion of the reason. In principle there can also be *degenerate forwards-reasons* having only backwards-premises. These would be forwards-reasons for which there is an automatic interest in the backwards-premises, and when those interests are discharged then a conclusion is drawn instantiating the conclusion of the reason. However, I have not found degenerate forwards-reasons useful.

OSCAR adds another device to forwards-premises that has proven very useful. A forwards-premise can be designated a *clue*. When a conclusion instantiating a clue is obtained, this will satisfy the forwards-premise and typically initiate interest in the backwards-premises, but when a conclusion instantiating the conclusion of the reason is finally drawn, the clue is not counted among the premises from which the inference is made. The function of clues is simply to direct the reasoning.

The use of both forwards- and backwards-premises in forwards- and backwards-reasons is a very powerful device for controlling the reasoning and avoiding combinatorial explosions. To illustrate, consider three reasons related to the material I will be discussing later in the week. Informally, they can be expressed as follows:

> <u>*PERCEPTION*</u>
> Having a percept at time t with the content P is a prima facie reason for the agent to believe P-at-t.

The inference rule *PERCEPTION* will have some strength r, although this may vary from implementation to implementation. The value of r should correspond roughly to the reliability of an agent's system of perceptual input in the circumstances in which it normally functions. *PERCEPTUAL-RELIABILITY* constitutes a defeater by informing us that under the present circumstances, perception is not as reliable as it is normally assumed to be:

> <u>*PERCEPTUAL-RELIABILITY*</u>
> Where R is projectible, r is the strength of *PERCEPTION* , and s < r, "R-at-t, and the probability is less than or equal to s of P's being true given R and that I have a percept with content P" is a defeater for *PERCEPTION*.

Notice, however, that this should not prevent our using percepts to draw conclusions with a weaker level of justification. The probability recorded in *PERCEPTUAL-RELIABILITY* should function merely to weaken the strength of the perceptual inference rather than completely block it. This can be accomplished by supplementing *PERCEPTION* with the following rule:

> <u>*DISCOUNTED-PERCEPTION*</u>
> Where R is projectible, r is the strength of *PERCEPTION* , and s < r, having a percept at time t with the content P and the belief "R-at-t, and the probability is less or equal to s of P's being

true given R and that I have a percept with content P" is a prima facie reason of strength s for the agent to believe P-at-t.

These reasons are implemented in OSCAR as follows:

```
(def-forwards-reason PERCEPTION
   :forwards-premises "(p at time)" (:kind :percept)
   :conclusions "(p at time)"
   :variables p time
   :defeasible? t
   :strength .9
   :description "When information is input, it is defeasibly reasonable to believe it.")

(def-backwards-undercutter PERCEPTUAL-RELIABILITY
   :defeatee 8perception
   :forwards-premises
   "((the probability of p given ((I have a percept with content p) & R)) <= s)"
     (:condition (s < 0.99))
   "(R at time0)"
      (:condition (and (projectible R) (time0 < time)))
      (:clue? t)
   :backwards-premises "(R at time)"
   :variables p time R time0 s
   :defeasible? t
   :description "When perception is unreliable, it is not reasonable to accept its representations.")

(def-forwards-reason DISCOUNTED-PERCEPTION
   :forwards-premises
   "((the probability of p given ((I have a percept with content p) & R)) <= s)"
     (:condition (s < 0.99))
   "(p at time)"
     (:kind :percept)
   "(R at time0)"
      (:condition (and (projectible R) (time0 < time)))
      (:clue? t)
   :backwards-premises "(R at time)"
   :conclusions "(p at time)"
   :variables p time R time0 s
   :strength  s
   :defeasible? t
   :description "When information is input, it is defeasibly reasonable to believe it.")
```

Consider how these reasons work in a simple case. Suppose the reasoner has a percept at time 10 to the effect that Fred is red, and an interest in whether Fred is red at time 10. *PERCEPTION* provides a defeasible reason for drawing this conclusion. But suppose further the reasoner is given that I am wearing red tinted-glasses at time 1, and the probability of Fred being red given that it looks red but I am wearing red-tinted glasses is $\leq$ .8. Then by *PERCEPTUAL-RELIABILITY*, the inference (of strength .98) to the conclusion that Fred is red should be defeated. But *DISCOUNTED-PERCEPTION* provides a reason (of strength .6) for concluding anew that Fred is red. The way the latter reasoning proceeds is by noting first that the forwards-premises for

*PERCEPTUAL-RELIABILITY* are satisfied. Notice that the second premise is a clue. Noting that I am wearing red tinted-glasses at 1 is a clue that I may still be wearing red tinted-glasses at 10, and so an interest in the latter is adopted (as the backwards-premise of *PERCEPTUAL-RELIABILITY*). The reasoner uses *TEMPORAL-PROJECTION* to infer that I am still be wearing red tinted-glasses at 10, thus establishing the defeat. The defeater is symbolized:

(((it appears to me that (the color of Fred is red)) at 10) ⊗ ((the color of Fred is red) at 10))

But exactly the same reasoning works in *DISCOUNTED-PERCEPTION* to provide a new, albeit weaker, reason for concluding that Fred is red at 10. Here is a printout of OSCAR's reasoning:

```
 # 1
  ((the probability of (the color of Fred is red) given ((I have a percept with content (the color of Fred is red)) & I am
wearing red tinted glasses)) <= 0.8)
  given
                                    # 1
                                    interest: ((the color of Fred is red) at 10)
                                    This is of ultimate interest
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
It appears to me that ((the color of Fred is red) at 10)
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
  # 2
  It appears to me that ((the color of Fred is red) at 10)
  # 3
  ((the color of Fred is red) at 10)
  Inferred by:
          support-link #2 from { 2 } by PERCEPTION  defeaters: { 6 }
  This node is inferred by discharging interest #1
                                    # 2
                                    interest: (((it appears to me that (the color of Fred is red)) at 10) ⊗ ((the color of Fred is
                                      red) at 10))
                                    Of interest as a defeater for support-link 2 for node 3
         ==========================================
         Justified belief in ((the color of Fred is red) at 10)
         with undefeated-degree-of-support 0.98
         answers #<Query 1: ((the color of Fred is red) at 10)>
         ==========================================
  # 4
  (I am wearing red tinted glasses at 1)
  given
                                    # 5
                                    interest: (I am wearing red tinted glasses at 10)
                                    For interest 1 by DISCOUNTED-PERCEPTION
                                    For interest 2 by PERCEPTUAL-RELIABILITY
                                    This interest is discharged by node 5
  # 5
  (I am wearing red tinted glasses at 10)
  Inferred by:
          support-link #4 from { 4 } by TEMPORAL-PROJECTION
  This discharges interest 5
  # 3
  ((the color of Fred is red) at 10)
  Inferred by:
          support-link #5 from { 1 , 2 , 5 } by DISCOUNTED-PERCEPTION
          support-link #2 from { 2 } by PERCEPTION defeaters: { 6 }
  This node is inferred by discharging interest #1
  # 6
```

```
(((it appears to me that (the color of Fred is red)) at 10) ⊗ ((the color of Fred is red) at 10))
Inferred by:
            support-link #6 from { 1 , 5 } by PERCEPTUAL-RELIABILITY
defeatees: { link 2 for node 3 }
This node is inferred by discharging interest #2
          vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
          The undefeated-degree-of-support of #<Node 3> has decreased to 0.6
          vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
```

Notice how efficient the reasoning is. No conclusions are drawn that are not essential for the problem. This is a direct result of the judicious assignment of directionality to the premises of the reasons and the use of clues.

# 6. Conclusions and Ongoing Research

OSCAR is more than an automated defeasible reasoner. OSCAR at least purports to be a complete architecture for a rational agent, but the core of the architecture is the defeasible reasoner. The use of this reasoner makes it simple to formulate and implement reasons aimed at enabling an agent to perform various kinds of reasoning, and the resulting reasoning can be made very efficient. I will illustrate this later in the week when I talk about an implemented solution to the Frame Problem and the Yale Shooting Problem. My current research is aimed at building a planning system within OSCAR. Given that OSCAR can reason its way around the Frame Problem and the Yale Shooting Problem, a planner can be constructed that reasons about the effects of actions rather than precompiling them in a STRIPS-like representation. The use of OSCAR-style reasons will then allow us to reconstruct existing planning algorithms as reasoning schemas rather than computational algorithms, thus freeing planning from the syntactical constraints of the STRIPS representation. Of even greater significance, I hope that this will facilitate the construction of a decision-theoretic planner based upon OSCAR's defeasible reasoner.