II
THE THEORETICAL BASIS

1. Practical Cognition

In its present incarnation, OSCAR is a programmable architecture for a rational agent, based upon a general-purpose defeasible reasoner. It is so designed that it can also be employed as a standalone reasoner for processing problems presented to it directly. The general architecture and the underlying theory are described in detail in my book *Cognitive Carpentry*. This chapter will present a brief overview of the theory.

On the conception of rationality embodied in OSCAR, a rational agent can be regarded as having four basic constituents:
- One or more mechanisms for proposing goals.
- A mechanism for evaluating the "goodness" of plans.
- A mechanism for searching for and adopting plans on the basis of their comparative evaluations by the plan evaluator.
- A mechanism for initiating action on the basis of adopted plans (together, possibly, with built-in or learned plan-schemas).

These mechanisms constitute a system of "practical cognition", to use traditional philosophical jargon. On any theory of rationality, plan evaluation and adoption will be based in part on what beliefs the agent has about its situation. Accordingly, an important part of a rational agent is a system of epistemic cognition producing such beliefs. Part of what makes OSCAR unique is that the bulk of the work involved in finding, evaluating, and choosing plans and directing action is done by epistemic cognition rather than by dedicated special-purpose modules devoted to practical cognition.

The OSCAR architecture begins with a *situation-evaluator*, which produces a (real-measurable) degree of liking for the agent's current situation. This is presumed to be sensitive to the agent's beliefs about its situation. The *likability* of a situation is the degree the agent would like it if it had true beliefs about all relevant aspects of the situation. The objective of the agent's reasoning is to put itself in situations that are more likable than the situations in which it would find itself if it did not take action. (For details, see chapter one of *Cognitive Carpentry*.)

Ideally, plans are evaluated in terms of the *expected likability* of their being adopted. This involves both reasoning about likabilities and reasoning about probabilities. Such reasoning is computationally difficult, so the OSCAR architecture also allows the use of shortcut procedures for producing approximate evaluations of plans. Such shortcut procedures are called *Q&I modules* ("quick and inflexible"). Q&I modules occur throughout rational cognition. A mark of rationality, however, is that when the output of Q&I modules conflicts with the output of explicit reasoning, the agent overrides the Q&I modules and adopts the conclusion of the reasoning.

Goals are also judged suitable or unsuitable on the basis of their expected likability. The function of goals is to direct the course of planning. The use of goals constitutes a control structure for planning. Without goals, the agent would have to survey plans at random. The chances of finding good plans in that way are miniscule. With the help of goals, planning is constrained, and because suitable goals have high expected likability, there is a (defeasible) presumption that plans having a high likelihood of achieving suitable goals will also have high expected likabilities. This provides a defeasible basis for adopting plans without going through the onerous process of computing their expected likabilities.

Shortcut procedures for the choice of goals are indispensible in any realistic agent, because the agent must be able to function in its environment before it acquires the general knowledge that is required for evaluating the suitability of goals. These take the form of Q&I modules called *optative dispositions*, which are dispositions to adopt goals. Some optative dispositions may be built-in from the beginning, and others can be acquired

by conditioning mechanisms. In human beings, such dispositions produce desires. Desiring something constitutes a defeasible ground for choosing it as a goal, but in an ideally rational agent, the belief that the object of desire does not have satisfactorily high expected likability should override the desire. (Human beings sometimes find this difficult.)

Traditional AI planning systems employ "planning algorithms" to literally "compute" plans. But this becomes problematic in a realistic rational agent. This is in part because planning must be based upon the agent's beliefs about its situation and about general features of the world, but such beliefs are defeasible. A rational agent must be prepared to withdraw beliefs that come in conflict with new beliefs acquired either through perception or simply by further reasoning from other beliefs. The core of OSCAR is a sophisticated defeasible epistemic reasoner. But the fact that epistemic reasoning is defeasible creates a problem for planning. If a rational agent retracts beliefs upon which planning was based, it must be prepared to modify its plans as well.

In fact, planning must be defeasible for two different reasons. First, we have noted that factual beliefs about the world may change, and this will can affect the evaluation of plans. But a second source of defeasibility results from the fact that the rationality of adopting a plan depends in part on what alternative plans are available. Plan adoption cannot await the production of all possible alternative plans, because there are infinitely many of them. Plan adoption must instead proceed on the basis of what plans have been discovered so far, and when new plans are found that are better than previously adopted plans, a rational agent must be prepared to change its mind and adopt the new plans in place of the old plans.

In principle, the defeasibility of planning could be accommodated by constructing a "defeasible practical reasoner", dedicated to reasoning defeasibly about plans and actions. However, the OSCAR architecture is based upon the observation that a simpler alternative is available to us. We must already have a defeasible epistemic reasoner, and it can be made to serve double duty as a defeasible practical reasoner through a technical trick called "doxastification". Corresponding to the adoption of a plan is the "epistemic judgment" (i.e., belief) that *it should be an adopted plan*. This judgment is epistemic in name only. It requires no "objective fact" to anchor it or give it truth conditions. It is merely a computational device whose sole purpose is to allow us to use defeasible epistemic reasoning to accomplish defeasible practical reasoning. Let us abbreviate $\ulcorner \sigma$ should be an adopted plan$\urcorner$ (where this is a practical 'should' — not a moral 'should') as $\ulcorner \sigma$ is *adoptable*$\urcorner$. In OSCAR, planning proceeds by epistemic reasoning about plan adoptability. The details of such reasoning are described in chapter seven of *Cognitive Carpentry*.

In OSCAR, action is directed on the basis of beliefs about plan adoptibility. The belief that a plan is adoptable, together with the belief that the preconditions for one of its plan steps are satisfied, constitutes a reason for believing that the action prescribed by the plan step is "executable". Note that this again involves doxastification. Acts judged executable are then sent to a module called *the ACTION-INTIATOR*, which initiates their execution.

The upshot of this is that, in OSCAR, the bulk of the computational work involved in practical cognition is relegated to epistemic cognition. Accordingly, the core of OSCAR is a sophisticated system of epistemic cognition, based upon a general-purpose defeasible reasoner. Practical cognition functions by passing queries to epistemic cognition. These queries comprise the list of *ultimate-epistemic-interests*. The introduction of a query initiates epistemic cognition aimed at answering it, and queries are stored along with instructions for what to do with an answer. OSCAR begins with an interest in finding suitable goals. This is encoded as a permanent member of *ultimate-epistemic-interests*. When such a goal is found, the corresponding instruction is to insert a new query into *ultimate-epistemic-interests* regarding the discovery of adoptable plans aimed at achieving the goal. When such a plan is found, a query is inserted regarding the satisfaction of the preconditions of its plan steps. When that query is answered affirmatively, the corresponding instruction is to pass the prescribed action to the ACTION-INTIATOR. Thus most of the work of practical cognition is performed by passing queries to *ultimate-epistemic-interests* with appropriate instructions for what to do with answers to the queries. In many cases, those instructions take the form of inserting new queries into *ultimate-epistemic-interests*.
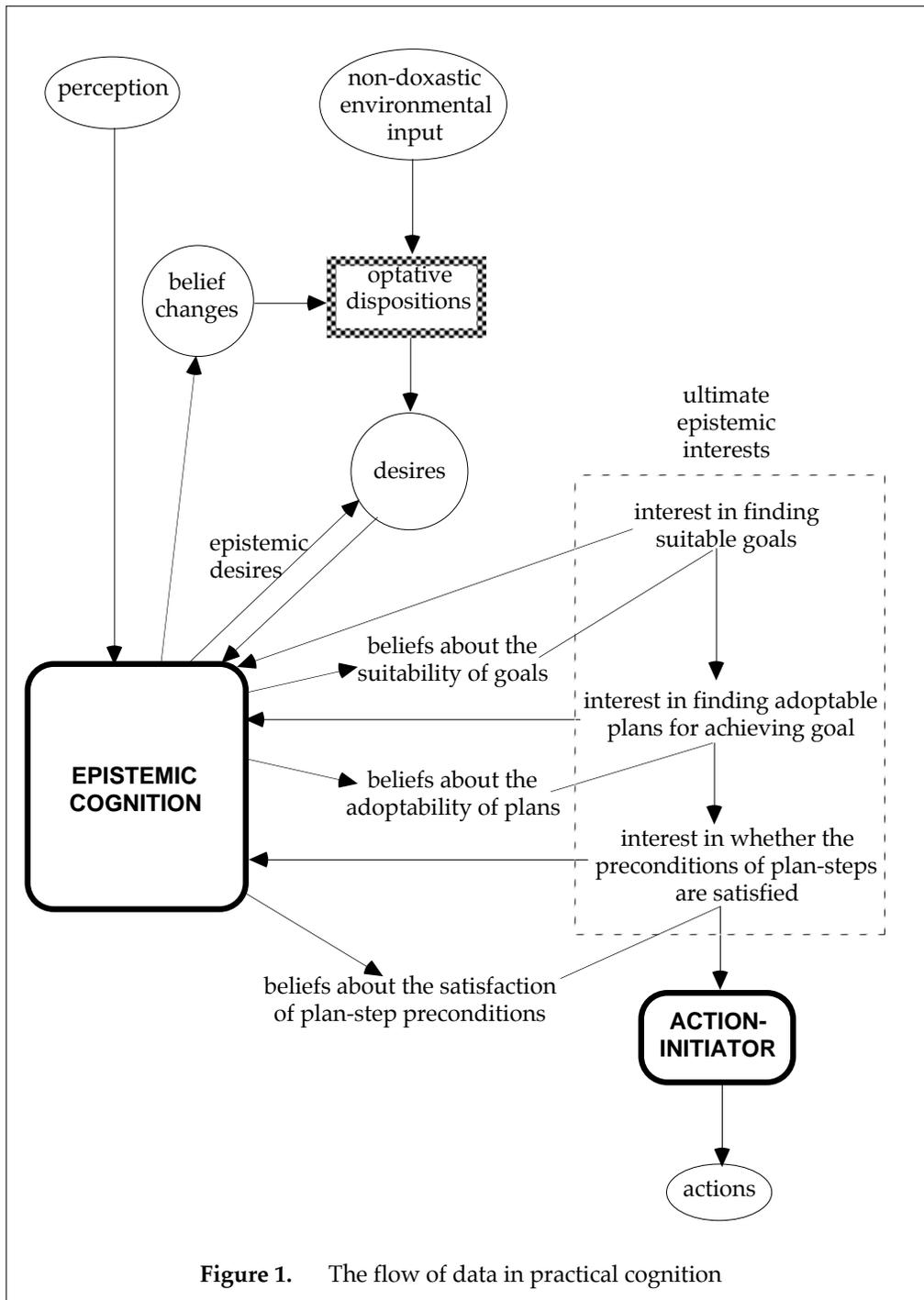
Figure 1.   The flow of data in practical cognition

In light of the doxastification of practical reasoning, OSCAR consists of a system of epistemic cognition taking input from both perception and desire formation, and passing actions to the ACTION-INTIATOR. This architecture is diagrammed in figure 1. In effect, the architecture has two levels. The lowest level consists of the system of epistemic cognition, together with various hooks to practical cognition and action. This level is fully implemented. The higher level consists of the system of practical cognition, and it is implemented *in* the lower level via doxastification. This is not yet fully implemented.

It should be noted that figure 1 indicates a further connection between epistemic cognition and practical cognition in the form of an arrow labeled "epistemic desires".

This reflects the fact that epistemic cognition will be unable to answer many of the questions posed by practical cognition just by thinking. If I want to know what time it is, I cannot just think about it — I have to look at a clock. Generally, answering practically-motivated questions requires investigating the world empirically. This requires *taking action*. The actions can range from focusing my eyes to examining the contents of my immediate surroundings to asking someone else a question to engaging in a multi-year research project involving high-energy linear accelerators. These are activities of the sort directed by practical cognition. So we get a loop from practical cognition through epistemic cognition and back to practical cognition. The mechanism for this involves epistemic cognition producing new goals for practical cognition to try to achieve — goals of acquiring certain kinds of knowledge. Goals are encoded in desires, and I will refer to these epistemically-driven desires as *epistemic desires*. Given an epistemic desire, practical cognition can produce reasoning about how to acquire the desired knowledge. That practical cognition may pose further questions for epistemic cognition, which may produce further epistemic desires, and so on.

2. Epistemic Cognition

In OSCAR epistemic cognition is performed jointly by a general-purpose defeasible reasoner and an array of special-purpose Q&I modules. Although an interface to the Q&I modules has been provided, no Q&I modules have been incorporated into the present implementation.

OSCAR's defeasible reasoner is based upon seven fundamental ideas. These are (1) an argument-based account of defeasible reasoning, (2) an analysis of defeat-status given a set of interrelated arguments, (3) a general adequacy criterion for automated defeasible reasoners, called "i.d.e.-adequacy", (4) the claim that a general-purpose defeasible reasoner must have the kind of structure I have called "flag-based", (5) an algorithm for computing defeat-status, (6) an interest-driven monotonic reasoner, and (7) an account of undefeated degrees of support, degrees of interest, and their interactions. I will discuss each of these in turn.

2.1 *Argument-Based Defeasible Reasoning*
The basic conception of defeasible reasoning embodied in OSCAR can, in its most general form, be regarded as the received view in philosophical epistemology. It is the argument-based approach pioneered by Roderick Chisholm and myself in the late 60's and early 70's (see Chisholm [1966] and [1977], and Pollock [1967], [1970], [1971], and [1974]), and developed further by myself (Pollock [1986], [1987], [1990c], [1991], [1992], [1994]) a number of people in the intervening years (e.g., Kyburg [1983], Loui [1987], and Nute [1988] and [1990]). The basic idea is that reasoning consists of the construction of arguments, where reasons are the atomic links in arguments. Defeasibility arises from the fact that some reasons are subject to defeat. I call these "prima facie reasons", and the considerations that defeat them are "defeaters". Although this is somewhat controversial, I have long argued and will here assume that there are only two kinds of defeaters. A "rebutting defeater" is a reason for denying the conclusion of the reason. An "undercutting defeater" attacks the connection between the premises and the conclusion, and can be regarded as a reason for denying that the premises wouldn't be true unless the conclusion were true. It will be convenient to abbreviate ⌜It is false that P wouldn't be true unless Q were true⌝ as ⌜$(P \otimes Q)$⌝. Reasons that are not defeasible are "conclusive".

A convenient way to encode arguments is as "inference-graphs". The nodes of an inference-graph represent premises and conclusions, and the links between the nodes represent dependency relations. The following is a simple inference-graph:
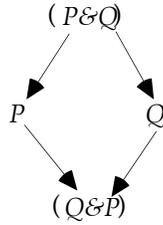
The "node-basis" of a node is the set of nodes from which it is inferred. We can combine all of the reasoning of the reasoner into a single global inference-graph, and that will be the central data-structure used by the reasoner. Defeat relations can be encoded by adding a separate category of "defeat links" to the inference-graph. It is important to emphasize that, for now, I am assuming that if the same conclusion is obtained by two different arguments, this is represented by two different nodes. 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. A more efficient form of inference-graph will be introduced in chapter three.

Because OSCAR engages in "suppositional reasoning", wherein conclusions are drawn relative to suppositions, the nodes of the inference-graph will be taken to encode sequents rather than formulas. A sequent is a pair $\langle \Gamma, P \rangle$ where P is a formula and Γ is a set of formulas (the supposition). This will be discussed further in section 2.5.

2.2 Defeat Status

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). The 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.

σ is a *partial status assignment* iff σ is a function assigning "defeated" and "undefeated" to a subset of the nodes of an inference-graph in such a way that:
1. σ assigns "undefeated" to all initial nodes;
2. σ assigns "undefeated" to a node n iff σ assigns "undefeated" to all the members of the node-basis of n and σ assigns "defeated" to all node-defeaters of n; and
3. σ 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 this treatment of defeat-status, suppose we have the inference graph diagrammed in figure 2, where defeasible inferences are indicated by dashed arrows, deductive inferences by solid arrows, and defeat links by arrows of the form '⤳▶'. $\alpha, \varepsilon,$ and ν are initial nodes, so they must be assigned "undefeated". It follows that μ, κ, λ, and β must be assigned "undefeated". Then ζ, ξ, and χ must be assigned "defeated", in which
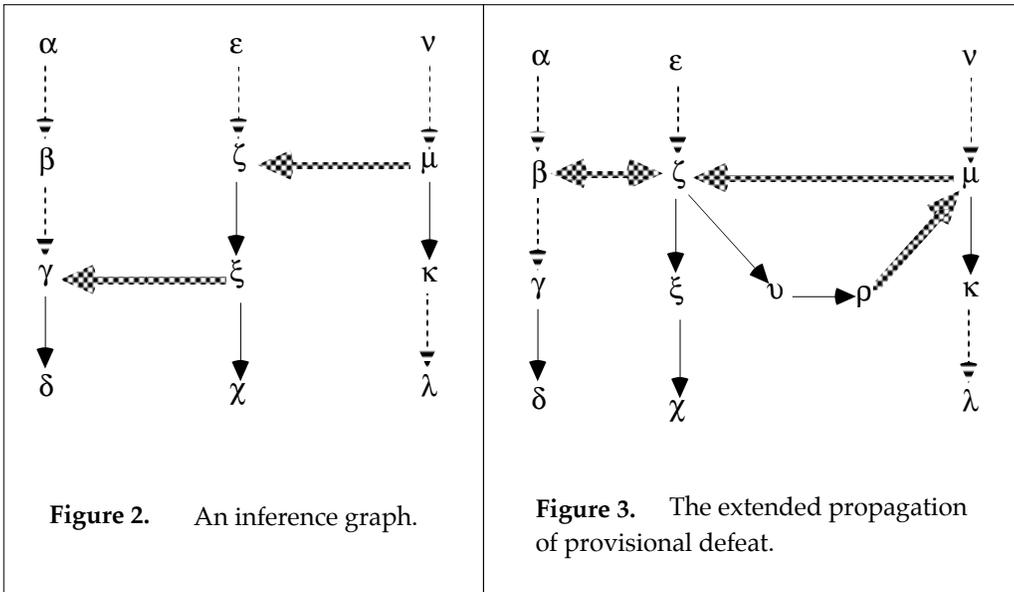
Figure 2. An inference graph.

Figure 3. The extended propagation of provisional defeat.
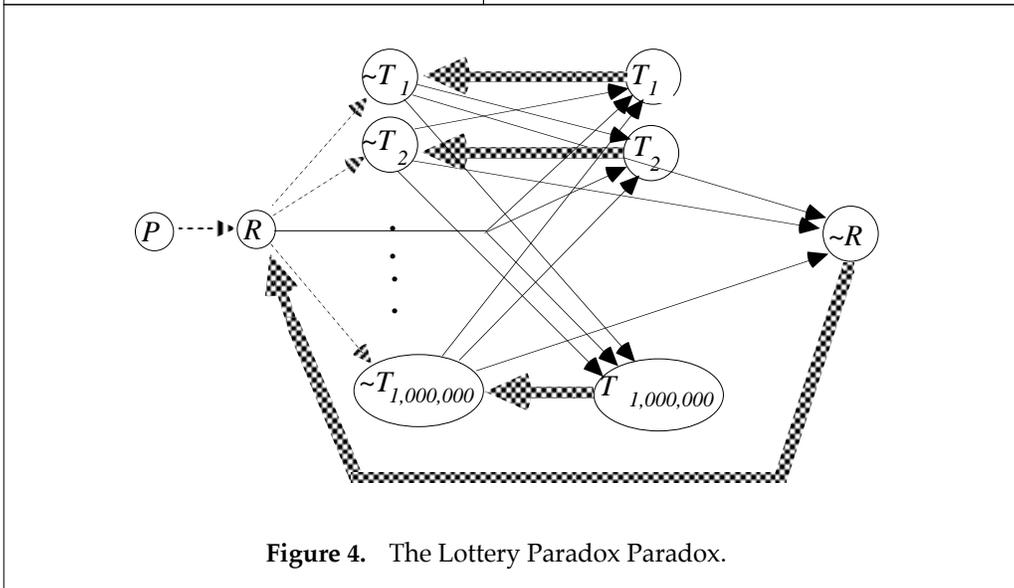

Figure 4. The Lottery Paradox Paradox.

case γ and δ must be assigned "undefeated". Thus there is just one status assignment for this inference graph.

This analysis has the consequence that a distinction can be made between two kinds of defeated nodes. Sometimes a node is defeated "outright" by another node that is undefeated. In that case, if the defeated node is a defeater for a third node, the third node remains undefeated. But sometimes a node is defeated "collectively". This happens when two or more otherwise undefeated nodes are defeaters for each other. Then they are all defeated. But if one of them is a defeater for another node, that further node is also defeated, despite the fact that its defeater is defeated. Defeated nodes that retain the power to defeat other nodes are "provisionally" defeated. This is illustrated in figure 3. Here there are two status assignments, one assigning "undefeated" to α, ε, ν, β, μ, γ, δ, κ, and λ, and "defeated" to the rest, and the other assigning "undefeated" to α, ε, ν, ζ, ξ, χ, υ, and ρ, and "defeated" to the rest.

To illustrate the analysis with an ultimately more interesting example, 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 $\sim R$ by noting that the $\sim T_i$ jointly entail $\sim R$. This is because if none of the tickets is drawn then the lottery is not fair. This is diagrammed in figure 4. The difficulty is now that $(\sim R)$ 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. Oviously, 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 $(\sim R)$ that is defeated. This results from the fact that in figure 4, there are one million status assignments — one assigning "undefeated" to each $\sim T_i$, but every such assignment assigns "defeated" to $(\sim R)$.

A conclusion is "justified" at a particular stage of reasoning iff it is supported by some undefeated argument. However, further reasoning might produce further relevant arguments that change the status of a conclusion from justified to unjustified, or vice versa. Given a set of premises and an array of reasons and rules of inference, we can say that a proposition is "warranted" iff the inference-graph produced by the set of all possible arguments built from those building blocks contains an undefeated node supporting the conclusion. Warranted propositions are the "ultimately reasonable" conclusions a defeasible reasoner is striving to identify.

2.3 I.D.E.-Adequacy

Perhaps the greatest problem facing the designers of automated defeasible reasoners is that the set of warranted conclusions resulting from a set of premises and reason schemas is not in general recursively enumerable. This was first observed informally by David Israel and Raymond Reiter in 1980. I gave a more formal proof of it in my [1992]. This has the consequence that a defeasible reasoner cannot look like a theorem prover, just systematically grinding out its conclusions in a mechanical way. Something more sophisticated is required. In my [1992], I urged that we take defeasibility seriously, and allow a defeasible reasoner to draw conclusions tentatively, sometimes retracting them later, and perhaps reinstating them still later, and so on. The set of conclusions drawn by such a reasoner cannot constitute a recursive enumeration of the set of warranted propositions, but I have proposed instead that it should systematically approximate the set of warranted conclusions in the following sense:

> The rules for reasoning should be such that if the reasoner is interested in a proposition p then:
> (1) if p is warranted, the reasoner will eventually reach a stage where p is justified and stays justified;
> (2) if p is unwarranted, the reasoner will eventually reach a stage where p is unjustified and stays unjustified.

If these conditions are satisfied, the sequence of sets of justified conclusions at different stages of reasoning constitutes a "defeasible enumeration" of the subset of warranted propositions in which the reasoner is interested, and the reasoner is said to be "i.d.e.-adequate".

Because the set of warranted propositions is not recursively enumerable, an i.d.e.-adequate reasoner may never stop reasoning. In simple cases it may stop, by running out of things to do, but in complicated cases it will go on reasoning forever. The significance of such a reasoner, embedded in a rational agent, is that it tells the agent what to believe "given the current state of its reasoning", and if the agent has to take action, it does so on the basis of what it believes at that time, even if there is no guarantee that it would not change its beliefs later if it had more time to reason. This involves the conception of defeasible conclusions as "innocent until proven guilty". In other words, they are pefectly

reasonable beliefs, and it is reasonable to act upon them, despite the fact that the agent cannot "prove" that they will never have to be retracted.

2.4 Flag-Based Reasoners

In the attempt to build an i.d.e.-adequate defeasible reasoner, a natural first inclination is to suppose that when a conclusion is defeated, the reasoner should stop reasoning from it unless or until it is subsequently reinstated. It turns out, however, that such a proposal cannot work. Consider two long arguments, and suppose the final step of each defeats an early step of the other. It follows from the analysis of defeat-status that both arguments are defeated. They undergo "collective defeat". But a reasoner that stopped reasoning from a conclusion once it was defeated would never discover the collective defeat, because having completed one of the arguments, it would cease developing the other one. For this reason, an i.d.e.-adequate reasoner must continue reasoning from conclusions even when they are defeated. This suggests that such a reasoner should simply flag conclusions as defeated, and go on reasoning. Defeat-status may affect the priorities that are assigned to performing various inference steps, so that undefeated conclusions are given some precedence over defeated ones, but inferences from defeated conclusions must still go forth. I call such a reasoner a "flag-based" reasoner. We can think of a flag-based reasoner as consisting of two largely autonomous modules — a "monotonic reasoner" that reasons and builds the inference-graph, without paying much attention (except in prioritizing inferences) to defeat-status, and a module that computes or recomputes defeat-statuses as new inferences are made. Such a reasoner has the form of a simple loop:

```
(loop
    (draw-a-conclusion)
    (recompute-defeat-statuses))
```

In my [1992], I was able to prove that subject to some reasonable assumptions about the structure of prima facie reasons provided to the reasoner and the adequacy of the monotonic reasoner, the resulting flag-based reasoner will be i.d.e.-adequate.

2.5 Interest-Driven Reasoning

OSCAR's monotonic reasoner is based upon the deductive reasoner described in my [1990a]. This is an "interest-driven suppositional reasoner" for first-order logic. The reference to suppositional reasoning just means that it can accommodate natural deduction rules like conditionalization, reasoning by cases, and *reductio-ad-absurdum*. For this purpose, the nodes of the *inference-graph* are taken to encode *sequents* rather than formulas. A sequent is a pair $\langle X,P \rangle$, where X is a set of formulas (the supposition of the sequent) and P is an individual formula. The sense in which the reasoner is interest-driven is that it reasons both backwards from the desired conclusions and forwards from the given premises. Reasoning backwards can be regarded as deriving interests from interests. This is related to backwards chaining and forwards chaining, but it is not quite the same thing. The difference lies in the fact that different rules of inference and different reason schemas are employed for backwards reasoning and for forwards reasoning. The motivation for this is that natural rules of inference are often very reasonable when applied in one direction but combinatorially explosive when applied in the other. For instance the rule of *addition* tells us to infer the disjunction $(P \lor Q)$ from the disjunct P. As a rule of backwards reasoning, this is eminently reasonable. It tells us that if we want to establish a disjunction, one way to do that is to try to get the first disjunct. But as a rule of forwards reasoning it would be catastrophic. It would have the reasoner infer every disjunction involving any conclusion it obtains.

We can think of interest-driven reasoning as consisting of three operations: (1) we reason forwards from previously drawn conclusions to new conclusions; (2) we reason backwards from interests to interests; (3) when we have reasoned backwards to a set of sequents as interests, and forwards to the same set of sequents as conclusions, then we *discharge interest* and conclude the sequent that led to those interests. For example, suppose we are given the premises P and $(P \to Q)$ and are interested in the conclusion $(Q \lor R)$. From the latter, we could reason backwards to interest in Q and in R (using *addition* as

our rule of inference). From the premises we could reason forwards to Q (using *modus ponens*). We then have Q both as a conclusion and an interest, so we can discharge interest. Our interest in Q derived from our interest in $(Q \vee R)$, so when we discharge interest, we conclude $(Q \vee R)$, and we are through.

This reasoning proceeds in accordance with three procedures:

REASON-FORWARDS

> If a set of sequents X is a forwards reason for a sequent S, some member of X is newly concluded, and the other members of X have already been concluded, then conclude S.

REASON-BACKWARDS

> If interest is adopted in a sequent S, and a set X of sequents is a backwards reason for S, then adopt interest in any members of X that have not already been concluded. If every member of X has been concluded, conclude S.

DISCHARGE-INTEREST

> If interest was adopted in the members of X as a way of getting the sequent S, and some member of X is concluded and the other members of X have already been concluded, then conclude S.

However, as formulated, these procedures are not adequate for some interest-driven reasoning. To see why, first note that, as a theoretical matter, there can be not only the "simple" backwards reasons discussed above, but also *generalized backwards reasons*. In a generalized backwards reason, the premises are segregated into two sets:

$$\frac{\overline{p_1,...,p_n}}{\dfrac{q_1,...,q_m}{r}}$$

where the premises in the first set (the *forwards premises*) function in a forwards direction and the premises in the second set (the *backwards premises*) function in a backwards direction. In other words, given interest in r, the reasoner reasons backwards to interest in $q_1,...,q_m$, but only if the reasoner has already established $p_1,...,p_n$.

A simple example of a generalized backwards reason would be the following:

$$\frac{\overline{(\forall x)(Fx \supset Gx)}}{\dfrac{Fa}{Ga}}$$

The effect of this reason will be that whenever the reasoner adopts interest in Ga, it will look at all universal generalizations $(\forall x)(Fx \supset Gx)$ that it has already concluded, and adopt interest in their antecedents as a way of establishing Ga.

We can regard generalized backwards reasons as the most general form of backwards reasons, henceforth calling them simply "backwards reasons". The simple backwards reasons discussed earlier represent the special case in which the set of forwards premises is empty. There can also be "degenerate backwards-reasons" in which there are forwards-premises but no backwards-premises. Given an interest in the conclusion, if the forwards-premises have already been established then the conclusion is drawn, but no interests are adopted.

Backwards-reasons can be defined easily within OSCAR using the macro DEF-BACKWARDS-REASON. For instance, we can define the preceding reason as follows:

```
(def-backwards-reason backwards-mp
   :conclusion  Ga
```

```
:forwards-premises  (∀x)(Fx ⊃ Gx)
:backwards-premises  Fa
:variables  F G a)
```

Forwards-reasons can also be generalized by allowing the addition of backwards-premises. Given a forwards-reason with both forwards- and backwards-premises, once the forwards-premises are established, interest is adopted in the backwards-premises, and when they are established the conclusion is drawn. This differs from generalized backwards-reasons only in that there need not be any interest in the conclusion for it to be drawn. Forwards-reasons can be defined using a macro DEF‑FORWARDS‑REASON which is analogous to DEF‑BACKWARDS‑REASON.

To illustrate the importance of this generalization of interest-driven reasoning, consider the following formulation of *reductio-ad-absurdum*:

$$\frac{\langle\Gamma\cup\{\sim p\},(q\ \&\ \sim q)\rangle}{\langle\Gamma,p\rangle}$$

This rule cannot be regarded as a simple backwards reason. Used for backwards reasoning it would have us adopt interest in infinitely many contradictions. This is of considerable significance, because there is reason to believe that the use of some form of this rule for backwards reasoning is essential in a deductive reasoner that is complete for first-order logic.[1] This problem can be solved by adopting the following variant of the rule as a generalized backwards reason:

$$\frac{\langle\Gamma\cup\{\sim p\},q\rangle}{\langle\Gamma\cup\{\sim p\},\sim q\rangle}$$
$$\langle\Gamma,p\rangle$$

This can be defined as follows:

```
(def-backwards-reason reductio
    :conclusion  p
    :forwards-premises  q
    :backwards-premises  ~q
    :discharge  ~p
    :variables  p q)
```

The effect of this is that, in trying to derive a contradiction from the supposition $\Gamma\cup\{\sim p\}$, the reasoner only looks for contradictions one of whose conjuncts have already been concluded. This avoids infinite branching, and when combined with other rules of inference, turns out to be sufficient to enable the construction of a deductive reasoner complete for first-order logic.

Reasoning forwards builds the *inference-graph,* and reasoning backwards builds the *interest-graph.* When the two graphs meet, interests are discharged, with the result that sequents that are of interest (as recorded in the *interest-graph*) are concluded and incorporated into the *inference-graph.* In the agent-architecture, queries posed by practical cognition form termini in the *interest-graph.* When OSCAR is treated as a standalone defeasible reasoner, these queries are supplied by the operator. When such queries are answered, the agent does something with the answers. What the agent does is determined by the purpose of the query. This is accomplished by storing instructions for what to do with an answer along with the query in *ultimate-epistemic-interests*.

_____

[1] See the discussion of this in my [1990a].

As a deductive reasoner, this monotonic reasoner turns out to be surprisingly efficient, as is documented in my [1990a]. However, the original motivation for developing it was to incorporate it into a system of defeasible reasoning. This is because standard deductive reasoners based upon resolution refutation cannot serve that purpose. Such systems, in effect, always reason by *reductio-ad-absurdum*, but *reductio-ad-absurdum* is invalid for reasoning involving prima facie reasons. If a contradiction is obtained from some premises by reasoning via prima facie reasons, that does not support an inference to the negation of the premises. Instead it defeats the defeasible inferences.

In deductive reasoning, all reasons are equally good (i.e., perfect). But defeasible reasons can vary in strength. This has important consequences for the behavior of a defeasible reasoner. For instance, given an argument for a conclusion and a stronger argument for its negation, the stronger argument wins. A common view in epistemology has been that conclusion strengths should behave like probabilities, but I have argued against that in chapter three of *Cognitive Carpentry*. OSCAR instead computes argument-strengths in terms of the "weakest link" principle. According to this principle, the strength of an argument, and the degree of support it confers on its conclusion, is the minimum of the (stipulated) degrees of justification of the input premises used and the strengths of the reasons used.

One of the main roles of degrees of support lies in the adjudication of disputes. Given an argument for P and another argument for $\sim P$, the stronger argument wins, i.e., the conclusion of the stronger argument is justified and the weaker argument is defeated. If the arguments are equally strong, they defeat each other collectively, and neither conclusion is justified. More generally, given nodes α and β of the *inference-graph* where the sequent supported by α has the syntactical form of a rebutting or undercutting defeater for β, α defeats β iff the degree of support for α is at least as great as the degree of support for β.

Figure 5.   The monotonic reasoner.
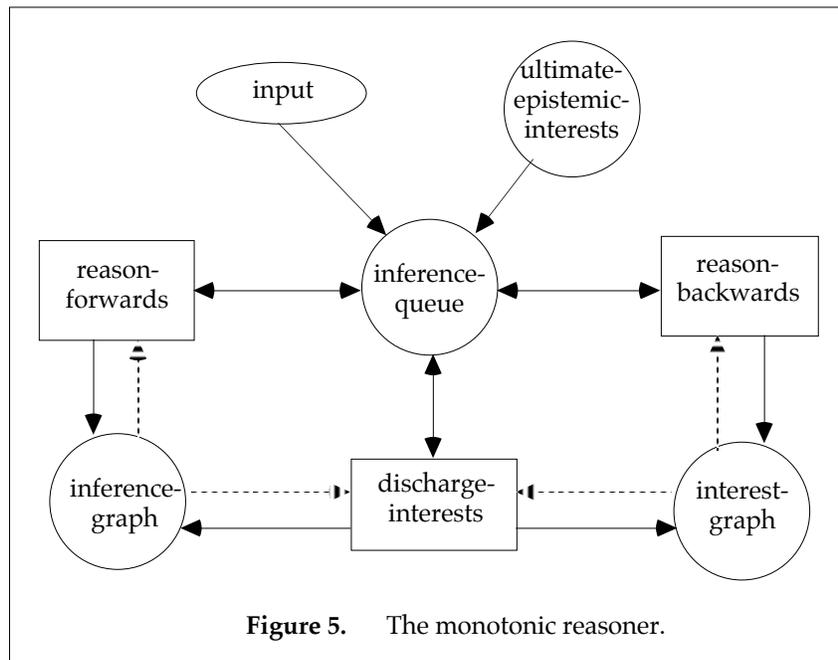
Interests also come in degrees. A rational agent has practical goals, and this leads to queries regarding how to satisfy those goals being sent to the epistemic reasoner. The reasoner thus becomes interested in answering those queries, and that initiates backwards

reasoning. But some of the goals will typically be more important than others, and this is reflected by differing degrees of interest in the queries deriving from those goals. One effect of those degrees of interest is to prioritize the backwards reasoning — preference is given to answering more important questions first.

OSCAR imposes an important constraint on undefeated degrees of support and degrees of interest. Basically, more important questions require better answers. In other words, when the agent reasoners backwards to a certain question and forwards to an answer, the answer is only deemed adequate if the undefeated degree of support is at least as great as the degree of interest.

The basic control structure for the inference operations is the *inference-queue*. This is an ordered queue of inferences (either backwards or forwards) waiting to be performed. The details of the ordering can be changed without changing the OSCAR architecture, but it is presumed that the ordering is sensitive to degrees of support and degrees of interest. The general structure of the monotonic reasoner is diagrammed in figure 5, where a solid arrow indicates that an operation alters the contents of a data-structure and a dashed arrow indicates that an operation retrieves information from a data-structure.

Combining figures 1 and 5 gives us figure 6, which diagrams the entire OSCAR architecture.

3. The Status of the OSCAR Project

The system described here is currently running, and available to anyone who wants to experiment wth it. It is to be emphasized that this is an *architecture* for a rational agent. No attempt has been made to provide a complete functional description of a rational agent. The difference is an important one. An architecture is a framework within which cognition operates. Different parts of the framework can be filled out in various ways, resulting in somewhat different rational agents that still have the same architecture. The architecture itself can be viewed as a programmable computational device. It is programmed by providing computational modules and data structures for the varous functional constituents. In particular, contents must be provided for the list of *permanent-ultimate-epistemic-interests*, the lists of prima facie and conclusive reasons, the relation used for prioritizing the *inference-queue*, and a list of Q&I modules. It is not far-fetched to regard this as a matter of programming a general computational device. It would be unreasonable to stipulate the contents of these databases as a fixed part of the OSCAR architecture, because it is unlikely that there is just one good way of constructing such databases. In human beings, it is done one way, but there may be other ways of doing it that are, for some purposes, preferable. In particular, the best choices for these databases may depend heavily on the hardware used for implementing the architecture. Different hardware may be better suited to different kinds of computations. For instance, human beings are very good at pattern matching, but seem to find unification (two-way pattern matching) unnatural. On the other hand, unification plays an important role in most contemporary systems of automated reasoning, and it is easily computed on a digital computer. This suggests that, just maybe, a system of reason schemas employing unification might be better suited for an implementation of the OSCAR architecture on a digital computer than it is for certain kinds of biological implementations.

The next stage of the OSCAR project will explore the programming of OSCAR to implement particular kinds of reasoning. In particular, this archiecture will provide the basis for an attempt to implement my [1990] theory of probabilistic and inductive reasoning, and to implement the defeasible approach to planning and acting described in my [1992a] and in more detail in *Cognitive Carpentry*.
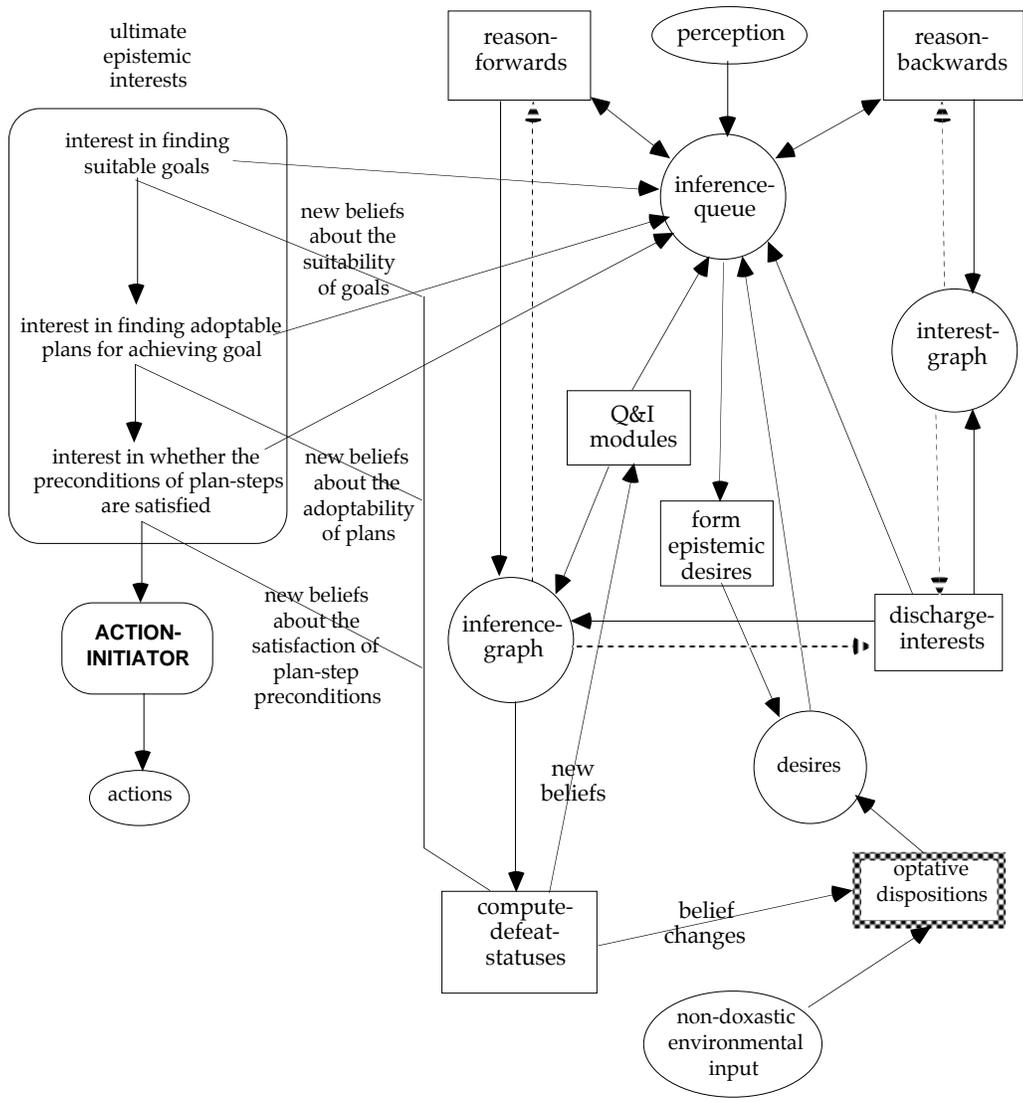
Figure 6. The OSCAR Architecture