THE OSCAR PROJECT

July, 1996

The OSCAR Manual

Operating manual and annotated code for OSCAR — the general-purpose defeasible reasoner and architecture for a rational agent.

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

I
LOADING AND RUNNING OSCAR

OSCAR is the programmable architecture for rational agents developed in *Cognitive Carpentry*. It is designed to provide the vehicle for experiments in artificial-agent building. It can also serve as a standalone defeasible and deductive reasoner. This manual explains how to use OSCAR, and explains the details of its construction.

OSCAR was developed and written in Macintosh Common LISP™, and should run in any Common LISP environment, on any kind of hardware. The program is contained in twelve files (the names may be different on the website):

Tools. — This contains the definitions of functions used as tools throughout the program.
Syntax — This defines the syntax of the formulas used by OSCAR.
Trees — This contains the code for computing defeat-statuses.
OSCAR_3.31.lisp — This contains the main code defining OSCAR.
Rules — This contains a set of inference rules for the propositional calculus.
Args — This contains the functions that format and print the arguments found in the course of OSCAR's reasoning.
Pcompiler — This compiles human-readable problems into OSCAR-readable problems.
Probs — This contains a set of problems involving defeasible reasoning and deductive reasoning in the propositional and predicate calculi. The defeasible reasoning problems are keyed to figures in chapter three of *Cognitive Carpentry*.
graphics — This contains the code for the graphics display used in the Macintosh version.
Rmacros — This contains the macros used for producing reason-schemas.
PC1 — This defines the reason-schemas used in chapter seven for reasoning about perception, time, and causation.
PC2 — This contains the sample problems referenced in chapter seven.

These files can be downloaded from Pollock's web page at http://www.u.arizona.edu/~pollock/.

To run OSCAR, begin by copying these files into a directory on your computer. The following line appears close to the beginning of the file *OSCAR.lsp*:

```
(setf oscar-pathname
       (parse-namestring
         "Little Mac:Nisus Miscellaneous:COGNITIVE CARPENTRY:OSCAR FOLDER:"))
```

This must be edited to reflect the location of the files on your computer. Once that is done, OSCAR can be loaded by starting LISP and then loading *OSCAR.lsp.* The latter will load the other files automatically if they are not already loaded. You may have to compile the files to achieve reasonable speed, although that is not necessary in Macintosh Common LISP™.

Load OSCAR by loading the file "OSCAR_3.31.lisp". This creates the package OSCAR, and the program loads in that package. To run OSCAR you must then enter the package by typing "(in-package OSCAR)" in the Listener window.

Once OSCAR is loaded, the problems contained in the problem-set that has been loaded can be run by typing "(test)". "(test n)" will run just problem number *n*, and "(test n t)" will run all the problems beginning with problem number *n*. "(test :skip i j k ...)" will run all the problems in the problem-set except for problems numbered i, j, k, ... "(test n :skip i j k ...)" will run all the problems beginning with problem n except for problems numbered i, j, k, ... It will be explained in appendix two how to construct your own problem-sets.

For the sake of portability, OSCAR is designed to display its output in the Listener window rather than in a separate window of its own. The nature of the display is controlled by the following switches:

(proof-on) — All relevant arguments found during the reasoning are displayed at the end of the problem. If OSCAR is interrupted during reasoning, it can be instructed to show all relevant arguments found so far by typing “(show-arguments)”. (default is "off".)
(proof-off) — Arguments are not displayed.
(display-on) — The reasoning is displayed as it proceeds. (default is "off".)
(display-off) — The reasoning is not displayed. This also turns off trace-mode.
(trace-on) — This turns on display-mode and also gives a detailed trace of what operations are being performed. This is primarily of use for debugging, or understanding the code. (default is "off".)
(trace-off) — This turns off trace-mode, but leaves display-mode on.
(trace-from n) — This turns on trace-mode when conclusion number n is retrieved from the *inference-queue*.
(logic-on) — This turns on the use of logical inference rules in additional to the substantive reason-schemas contained in the problem. (default is "on".)
(logic-off) — This turns off the use of logical inference rules.
(reductio-on) — This turns on the use of *reductio-ad-absurdum*. (default is "on".)
(reductio-off) — This turns off the use of *reductio-ad-absurdum*.
(pause-on) — This makes the reasoner pause for a keystroke between problems. (default is "off".)
(pause-off) — This turns off the pause.
(log-on) — The reasoner will display all the reasoning after it finishes, and will attempt to indicate which reasoning was actually used in getting the solution. (The latter is made complicated by the fact that interests that guide the reasoning may be generated in more than one way.) (default is "off".)
(log-off) — This turns off the log.

The variable **time-limit** is either an integer or NIL (the default value is 5). If it is an integer and the reasoning on a problem takes more than that many minutes, the reasoner aborts and goes on to the next problem.

To give a brief illustration, it is argued in *Cognitive Carpentry* that the paradox of the preface has the form of problem #17 in *Problems.lsp.* Running that problem by typing “(test 17)”, with display-off and proof-on, yields the following output:

```
******************************************************************************************
******************************************************************************************
Problem #17
Figure 18 -- the paradox of the preface, using logic.
Given premises:
   P1    justification = 1
   P2    justification = 1
   P3    justification = 1
   S    justification = 1
   T    justification = 1
Ultimate epistemic interests:
   (Q1 & (Q2 & Q3))    interest = 1

  FORWARDS PRIMA FACIE REASONS
   pf-reason 1:   {P1} ||=> Q1   strength = 1
   pf-reason 2:   {P2} ||=> Q2   strength = 1
   pf-reason 3:   {P3} ||=> Q3   strength = 1
   pf-reason 4:   {S} ||=> R   strength = 1
   pf-reason 5:   {T} ||=> ~(Q1 & (Q2 & Q3))    strength = 1
   pf-reason 6:   {S1} ||=> (T @ ~(Q1 & (Q2 & Q3)))   strength = 1
   pf-reason 7:   {S2} ||=> (T @ ~(Q1 & (Q2 & Q3)))   strength = 1
   pf-reason 8:   {S3} ||=> (T @ ~(Q1 & (Q2 & Q3)))   strength = 1
   FORWARDS CONCLUSIVE REASONS
     con-reason 4:   {R , Q1 , Q3} ||=> S2   strength = 1
     con-reason 5:   {R , Q2 , Q3} ||=> S1   strength = 1
     con-reason 6:   {R , Q1 , Q2} ||=> S3   strength = 1



================= ULTIMATE EPISTEMIC INTERESTS ==================
  Interest in (Q1 & (Q2 & Q3))
  is answered affirmatively by node 16
---------------------------------------------------

Elapsed time = 1.10 sec

=========================================================================
ARGUMENT #1
This is an undefeated argument of strength 1 for:
     (Q1 & (Q2 & Q3))
 which is of ultimate interest.

 1.  P1     given
 12.  Q1     pf-reason 1 from { 1 }
 2.  P2     given
 9.  Q2     pf-reason 2 from { 2 }
 3.  P3     given
 8.  Q3     pf-reason 3 from { 3 }
 15.  (Q2 & Q3)     adjunction from { 9 , 8 }
 16.  (Q1 & (Q2 & Q3))     adjunction from { 12 , 15 }

This argument is defeated by arguments #2, #3, #4

This argument defeats arguments #2, #3, #4
=========================================================================
ARGUMENT #2
This is a defeated argument for:
     ~Q1

 3.  P3     given
 8.  Q3     pf-reason 3 from { 3 }
 2.  P2     given
 9.  Q2     pf-reason 2 from { 2 }
 15.  (Q2 & Q3)     adjunction from { 9 , 8 }
 5.  T     given
 6.  ~(Q1 & (Q2 & Q3))     pf-reason 5 from { 5 }
 24.  (~Q1 v ~(Q2 & Q3))     DM from { 6 }
 25.  (Q1 -> ~(Q2 & Q3))     disj-simp from { 24 }
 27.  ~Q1     modus-tollens1 from { 15 , 25 }

This argument is defeated by arguments #1, #3, #4, #5, #6, #7

This argument defeats arguments #1, #3, #4, #5, #6
=========================================================================
ARGUMENT #3
This is a defeated argument for:
     ~Q3

 2.  P2     given
 9.  Q2     pf-reason 2 from { 2 }
 1.  P1     given
 12.  Q1     pf-reason 1 from { 1 }
 5.  T     given
 6.  ~(Q1 & (Q2 & Q3))     pf-reason 5 from { 5 }
 24.  (~Q1 v ~(Q2 & Q3))     DM from { 6 }
 25.  (Q1 -> ~(Q2 & Q3))     disj-simp from { 24 }
 26.  ~(Q2 & Q3)     modus-ponens1 from { 12 , 25 }
 28.  (~Q2 v ~Q3)     DM from { 26 }
 29.  (Q2 -> ~Q3)     disj-simp from { 28 }
 30.  ~Q3     modus-ponens1 from { 9 , 29 }

This argument is defeated by arguments #1, #2, #4, #5, #6, #7

This argument defeats arguments #1, #2, #4, #5, #7
===========================================================================
ARGUMENT #4
This is a defeated argument for:
     ~Q2

 3.  P3     given
 8.  Q3     pf-reason 3 from { 3 }
 1.  P1     given
 12.  Q1     pf-reason 1 from { 1 }
 5.  T     given
 6.  ~(Q1 & (Q2 & Q3))     pf-reason 5 from { 5 }
 24.  (~Q1 v ~(Q2 & Q3))     DM from { 6 }
 25.  (Q1 -> ~(Q2 & Q3))     disj-simp from { 24 }
 26.  ~(Q2 & Q3)     modus-ponens1 from { 12 , 25 }
 28.  (~Q2 v ~Q3)     DM from { 26 }
 29.  (Q2 -> ~Q3)     disj-simp from { 28 }
 31.  ~Q2     modus-tollens1 from { 8 , 29 }

This argument is defeated by arguments #1, #2, #3, #5, #6, #7

This argument defeats arguments #1, #2, #3, #6, #7
===========================================================================
ARGUMENT #5
This is an undefeated argument of strength 1 for:
     (T @ ~(Q1 & (Q2 & Q3)))

 4.  S     given
 7.  R     pf-reason 4 from { 4 }
 1.  P1     given
 12.  Q1     pf-reason 1 from { 1 }
 3.  P3     given
 8.  Q3     pf-reason 3 from { 3 }
 13.  S2     con-reason 4 from { 12 , 7 , 8 }
 11.  (T @ ~(Q1 & (Q2 & Q3)))     pf-reason 7 from { 13 }

This argument is defeated by arguments #2, #3

This argument defeats arguments #2, #3, #4
===========================================================================
ARGUMENT #6
This is an undefeated argument of strength 1 for:
     (T @ ~(Q1 & (Q2 & Q3)))

 4.  S     given
 7.  R     pf-reason 4 from { 4 }
 1.  P1     given
 12.  Q1     pf-reason 1 from { 1 }
 2.  P2     given
 9.  Q2     pf-reason 2 from { 2 }
 14.  S3     con-reason 6 from { 12 , 7 , 9 }
 11.  (T @ ~(Q1 & (Q2 & Q3)))     pf-reason 8 from { 14 }

This argument is defeated by arguments #2, #4

This argument defeats arguments #2, #3, #4
===========================================================================
ARGUMENT #7
This is an undefeated argument of strength 1 for:
     (T @ ~(Q1 & (Q2 & Q3)))

 4.  S     given
 7.  R     pf-reason 4 from { 4 }
 2.  P2     given
 9.  Q2     pf-reason 2 from { 2 }
 3.  P3     given
 8.  Q3     pf-reason 3 from { 3 }
 10.  S1     con-reason 5 from { 9 , 7 , 8 }
 11.  (T @ ~(Q1 & (Q2 & Q3)))     pf-reason 6 from { 10 }

This argument is defeated by arguments #3, #4

This argument defeats arguments #2, #3, #4
============================================================================


Cumulative size of arguments = 24
Size of inference-graph = 37 of which 0 were unused suppositions.
64% of the inference-graph was used in the argument.
28 interests were adopted.
28 i-lists left uncancelled.
11 suppositions were made.
```

Chapter two will give a brief presentation of the theory underlying OSCAR, chapter three will present the pseudocode in detail, and subsequent chapters will discuss the implementation of particular varieties of reasoning within the OSCAR architecture. The codification of syntax, rules of inference, and problem-sets will be discussed in appendices.

This manual should prepare the reader for undertaking independent projects, using OSCAR as the inference engine and reasoning environment. As will be explained in subsequent chapters, OSCAR can be programmed by providing reason-schemas and permanent-ultimate-epistemic-interests, and this can provide the basis for systems of data-analysis and decision-making. OSCAR is being distributed in order to encourage researchers to use it in their investigations and provide feedback on its design.

OSCAR is based upon the theory of rationality described in *Cognitive Carpentry*. In recent years, many people have become dubious of the possibility of success of traditional AI, in the form of the attempt to build artificial rational agents. It is my contention that previous failures have stemmed largely from a failure to base AI systems on an adequate theoretical foundation, in the form of an implementable philosophical theory of rationality. The point of *Cognitive Carpentry* was to provide such a foundation. My contention was and is that the theory developed there provides an adequate foundation for much of AI. But as they say, the proof of the pudding is in the eating. Bon Appetit!