GETTING STARTED WITH OPEN PROLOG

Welcome to Open Prolog. Open Prolog is a text-oriented Prolog application for the Macintosh. It combines the flexibility of an interpreter with the speed of compiled code.

This document will show you how to start Open Prolog and how to write and test a small Prolog program. It is assumed that the Open Prolog application is already in your system somewhere, on a disk or server volume. Follow each section below to complete the tutorial session. To learn about Prolog itself, you'll need a Prolog textbook. There are lots of good ones, just make sure it uses the 'Edinburgh syntax'.

If you've never written a Prolog program before, remember two important things when you are doing the tutorial. First, Prolog is *case sensitive*. That means that it distinguishes between identifiers on the basis of the case of the letters in them, as well as their spelling. So, the words *This* and *this* are different. (In fact, Prolog treats the former as a variable and the latter as an atom.) The second thing sounds strange: never put a space between a term's name and the bracket at the start of its parameter list. For instance, the term *this (here)* should correctly be written *this(here)*.

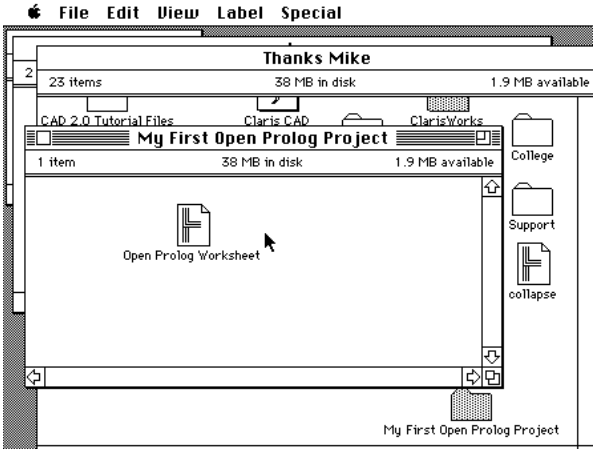
Make a Project Folder.

When you are using Open Prolog, the chances are that you'll use a number of files on one project. It is a good idea to open a separate folder for each project where you can keep the project's files together.

The first step is to create the folder and to put an Open Prolog Worksheet into it:

- Create a folder in the normal way, and call it anything you like.
- Find a document called 'Open Prolog Worksheet' somewhere. Copy it into your new folder. (If it's on the same volume as your folder, you can copy it by holding the Option key down while dragging it to the folder.)

Now you have created a project folder containing a Worksheet. The folder might look like this:

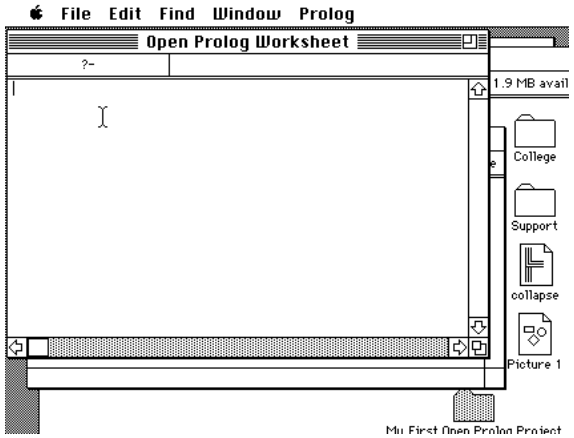


The Project Folder containing the Worksheet

Start Open Prolog

Having created a project folder, you can start Open Prolog by opening the Worksheet in the folder. This does three things: it starts up Open Prolog, it opens the Worksheet, and it makes the project folder the 'home folder'. The home folder is where Open Prolog will look for files later on.

When Open Prolog has started, you'll get something like this:



The Worksheet. This is where you enter commands and data to Open Prolog.
It doesn't matter if it contains some text already - simply delete it in the normal Macintosh way.

The Worksheet - 'Open Prolog Worksheet' - is a text window where you normally type commands and data to Open Prolog. If you want to write clauses (see later), it's best to put them in a separate file.

Later on, we'll use the Worksheet for entering commands and data.

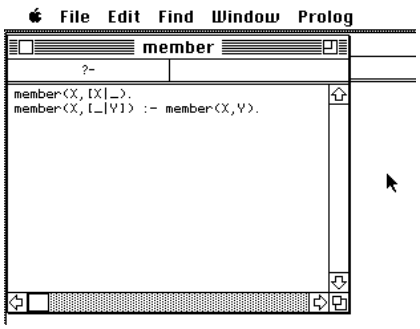
Write a Prolog Program

So far, you have created a project folder and opened Prolog in it. Now, you'll create a new Prolog file and type the member/2 program into it.

- Choose New from the File menu of Open Prolog. You'll get a window entitled 'Untitled ◊ 0'.
- Type the following clauses exactly (there is an error in one of the clauses):

```
member(X,[X|_).
member(X,[_|Y]) :- member(X,Y).
```

- Now, save the window by choosing Save from the File Menu. You'll be asked for a file name - use the name 'member' in this example (any name will do except one containing colons or quotation marks).



The new file 'member' containing the definition of member/2.

Now, you have created a file containing some (incorrect) Prolog clauses. Open Prolog must now be told to read the clauses from the file.

Consult, correct and reconsult your Program.

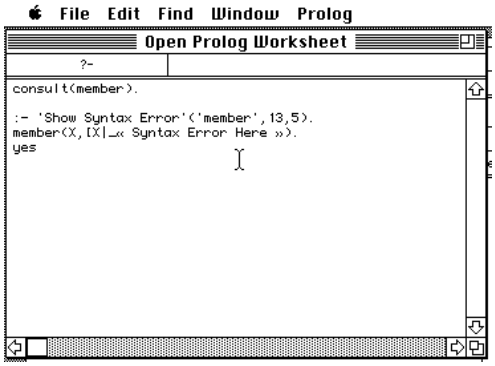
To get Open Prolog to load (i.e. to 'consult') the clauses in the file 'member',do the following:

- Bring the Worksheet to the front (click on it, or select it from the Windows menu).
- Type the following onto the Worksheet - don't type RETURN afterwards.

```
consult(member).
```

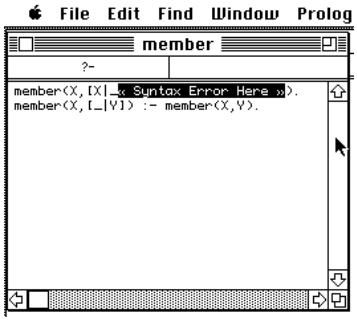
- Instead of typing the RETURN key at the end, type the ENTER key (if you have typed RETURN by mistake, click the mouse somewhere on the line to position the flashing caret on it). Using the ENTER key this way tells Open Prolog that the line contains a piece of information - in this case, a command. You'll see the words being boldfaced to show they are being actioned.

We know there's an error in your program, and you'll get an error message on the worksheet:



The Worksheet after consulting 'member'. It shows a syntax error has been detected.

The line that reads :- 'Show Syntax Error' etc. is actually another command that Open Prolog can use to show you where the error seems to be. To execute it, click the mouse on the line (to position the flashing caret on it), then type the ENTER key. After a short time, the file will be opened, and the error position will be highlighted as follows:



The file 'member' opened, and the location of the error highlighted.

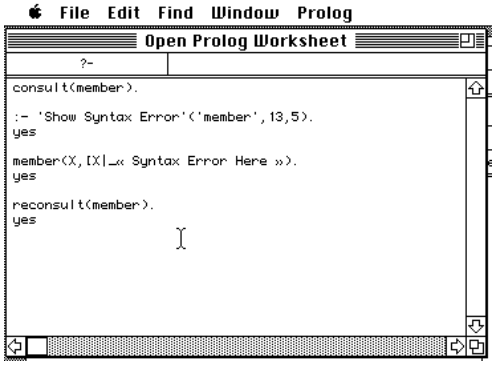
You can remove the syntax error message just by typing the DELETE key.

Correct the error (a square bracket is missing).

Now, to load the corrected version of the clauses into Open Prolog's database you should firstly save the file again. Then type the following command on the worksheet:

```
reconsult(member).
```

Notice that it is `reconsult` rather than `consult`. There are no errors this time, so the clauses are now loaded. Here is the Worksheet now:



The Worksheet is beginning to look a bit cluttered…

If the Worksheet becomes too cluttered, simply delete the text you don't need. You can keep command lines, etc. that you expect to re-use later on - it's up to you.

You can go through the edit-reconsult cycle as often as you please.

Now that the clauses have been typed into member properly, the next time you use Open Prolog you just consult the file straight off. In a complex project, you might have many files - there is no limit on the number of files you can consult from, although only eight can be open as windows at a time.

Performing a query

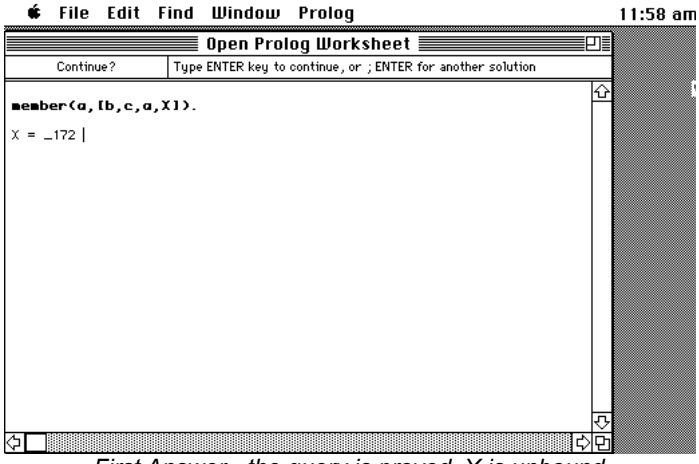
Now that you have the member predicate defined, let's see how to make a query.

Suppose we wish to see whether the query `member(a,[b,c,a,X])` can be proved. Just by looking at it, you can see that there are two ways it can be proved. The first is because the atom `a` is already in the list, and the second is because the variable `X` could be unified with `a`.

On the worksheet, enter the following command (i.e. type it and follow it by ENTER):

```
member(a,[b,c,a,X]).
```

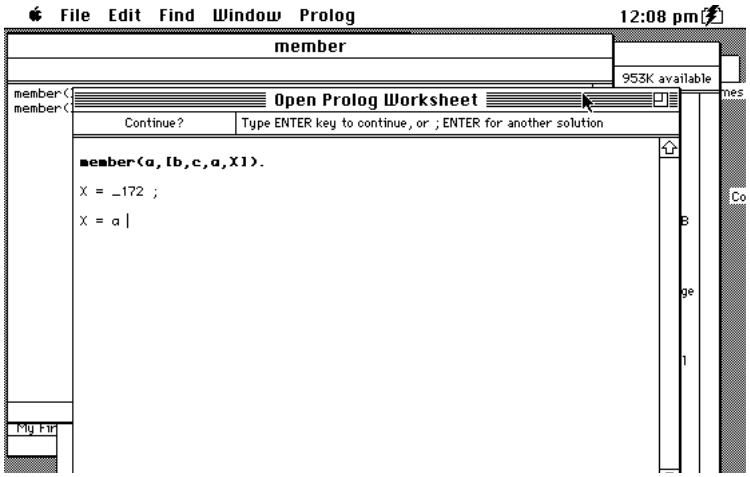
You'll get the following result:



First Answer - the query is proved, X is unbound

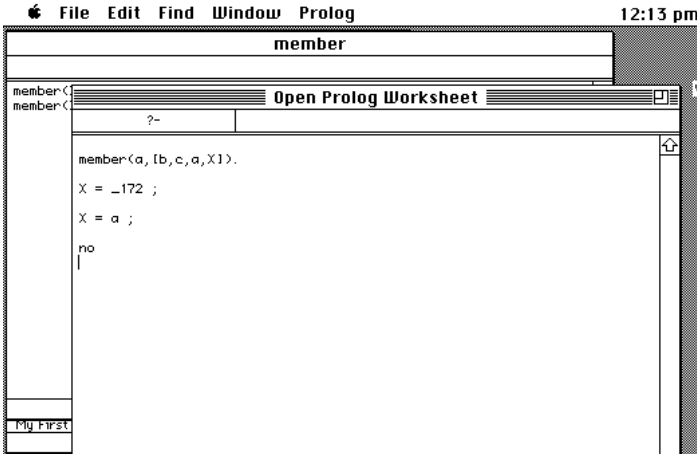
Notice three things about the first answer: The query is highlighted, indicating that it is still active. Secondly, the query has been proved, and Open Prolog is showing you the values it assigned to the variables to make the proof. In this case, there is only one variable - X- and it has been bound to another variable, called `_172`. This means that is was

not necessary to instantiate `X` to prove the query. The third thing to notice is that the interpreter is waiting for you to signal that it may continue. If the answer provided is satisfactory, you can simply type the ENTER key. If you wish to have Open Prolog search for an alternative answer, then type a semicolon followed by ENTER. This will cause the interpreter to backtrack to look for an alternative way of proving the query. If you do it now, you'll get:



Second Answer - the query is proved, X is bound to a.

Now, you may again ask for an alternative way of proving the query. Of course, in this simple example, we know there is no other way of satisfying the query, so the answer will be '`no`':



Third Answer - having requested another answer, none could be found, so the answer is 'no'

As soon as you indicate that you are finished a query by typing ENTER, or when no [further] solution is found, the query ends and the highlighting disappears from it.

Trace and debug a query.

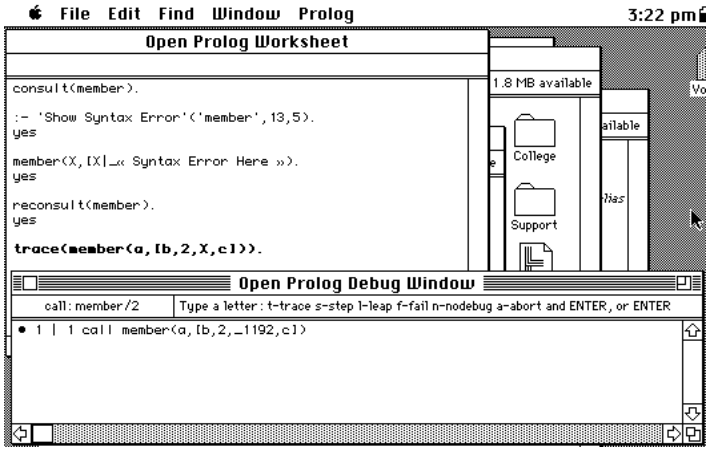
Let's see how to trace through the evaluation of a query.

Suppose we wish to see how the query `member(a,[b,2,X,c])` is processed.

On the worksheet, enter the following command (i.e. type it and follow it by ENTER):

```
trace(member(a,[b,2,X,c])).
```

The trace handler will open up a new window called 'Open Prolog Debug Window'. You can resize it to suit yourself.

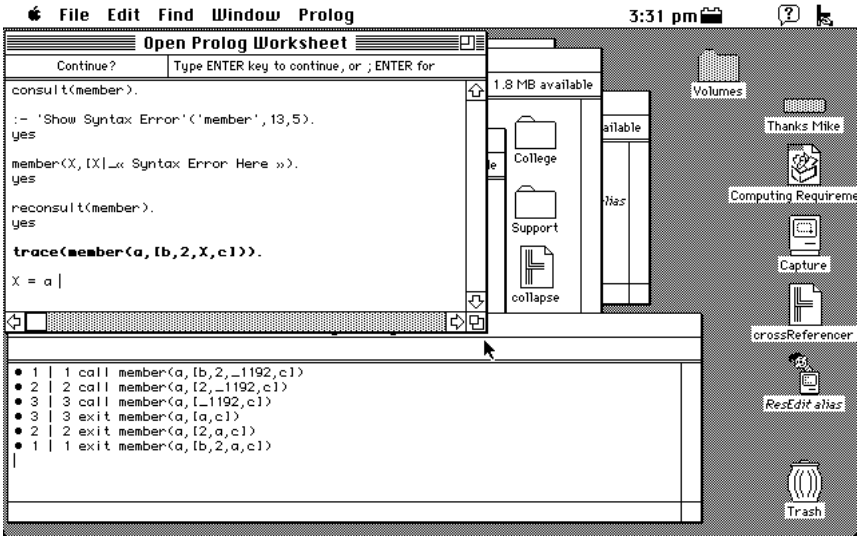


The Debug Window. Notice the information on the status line at the top of the window.

By simply typing the ENTER key you can step along the execution of the query. Step over a call by typing ‘s’ followed by ENTER, ‘f’ fails the call, ‘n’ switches off debugging, etc.

If you discover an error in your program, you can edit the file, save it and reconsult it in the normal way. Be aware, however, that if execution has begun on a clause that is superseded by reconsulted clauses, the old clause will continue in use, so that the program the debugger is running may not quite correspond to the updated program text.

Here is the complete trace of the query:



The Debug Window contains the complete sequence of calls and exits for the simple query.

Tracing is analogous to ‘single-stepping’ in a conventional language. You can also set breakpoints in Prolog, but they are called ‘spy points’. Look in the file Open Prolog Info for more information.

Printing your Documents, Sharing your Data

You can print your Open Prolog documents by using Page Setup and Print under the File menu. You can also print a selection of a document.

Open Prolog documents are of type TEXT, so that any word processor can read and change them. If you do use a word processor, make sure to save documents as TEXT-only, because Open Prolog can’t read it otherwise.

You can cut-and-paste between Open Prolog and other text-handling applications, such as word processors and mail programs.

Summary

If you have followed this tutorial, you should be able to: set up Open Prolog projects; to edit, consult and reconsult program files; to run and debug programs, and many other things.

You should now be able to undertake more ambitious projects, drawn, say, from the many excellent textbooks on Prolog. Open Prolog uses the so-called 'Edinburgh Syntax' which is the standard nowadays for Prolog.

More Information

Look into some of the files that accompany Open Prolog, particularly Open Prolog Info, for more detailed and advanced information on Open Prolog.

Finally

Open Prolog is not 'bullet-proof' software. If you come across bugs or infelicities, please mail the developer at `brady@cs.tcd.ie`, who will try to address the problems. Be nice to him.