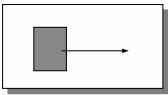# FAMILY: A family database

Tim Menzies

Lane Department of Computer Science, West Virginia University, PO Box 6109, Morgantown, WV, 26506-6109, USA;
`http://tim.menzies.us`; tim@menzies.us

Wp ref: `~menzies/src/pl/prod/family.pl`, January 18, 2003.

Abstract   A simple family database written in Prolog is described.

Contents

List of Figures

1 Introduction

The standard introductory Prolog example. Not sure where it came from originally but I think it was Bratko [1].

Fig. 1 Applications within the current release of PROD. $Y \to X$ indicates that Y has to first load X.

2 The facts

2.1 The boys

```
1 male(alan).
2 male(david).
3 male(donald).
4 male(eric).
5 male(james).
6 male(john).
7 male(scott).
```

2.2 The girls

```
8  female(elizabeth).
9  female(emily).
10 female(florence).
11 female(jess).
12 female(katherine).
13 female(louise).
14 female(rebecca).
```

2.3 The parents

Definition of the **parent(Parent, Child)** relationship:

```
15 parent(james, david).
16 parent(james, louise).
17 parent(james, john).
18 parent(james, alan).
19 parent(florence, david).
20 parent(florence, louise).
21 parent(florence, john).
22 parent(florence, alan).
23 parent(david, scott).
24 parent(louise, rebecca).
25 parent(louise, eric).
26 parent(louise, katherine).
27 parent(alan, elizabeth).
28 parent(alan, emily).
29 parent(david,jessie).
30 parent(louise,donald).
```

3 The rules

A mother is a female parent.

```
31 mother(Mother, Child):-
32     parent(Mother, Child),
33     female(Mother).
```

A father is a male parent.

```
34 father(Father, Child):-
35     parent(Father, Child),
36     male(Father).
```

A son is a male that has a parent.

```
37 son(Parent, Son):-
38     parent(Parent, Son),
39     male(Son).
```

A daughter is a female that has a parent.

```
40 daughter(Parent, Daughter):-
41     parent(Parent, Daughter),
42     female(Daughter).
```

A grandparent is the parent of a parent.

```
43 grandparent(GrandParent, GrandChild):-
44     parent(GrandParent, Parent),
45     parent(Parent, GrandChild).
```

An ancestor is a parent.

```
46 ancestor(X,Y) :- parent(X,Y).
```

Also, an ancestor is also the parent of someone who is an ancestor.

```
47 ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
```

4 Demonstrations

To demo the code, we need to:

– Write a demo predicate that shows off our code in action. This is accomplished using the **egfamily/0** predicate shown below.

– Trap the output to a file. This is accomplished using the **Demos/1** predicate.

– Include that file in this document. This is accomplished using the following latex magic:

```
\SRC{egfamily.spy}{The family
        database in action.}
```

After that include, we can show the output in Figure 2 and reference that figure using

```
\fig{egfamily.spy}
```

---
egfamily.spy

```
% output from ':- demos(egfamily).'


::[1]:::::::::::::::::::::::::
The parent of david is james.
The parent of david is florence.


::[2]:::::::::::::::::::::::::
The father of david is james.

::[3]:::::::::::::::::::::::::
The mother of david is florence.

::[4]:::::::::::::::::::::::::
A son of james is david.
A son of james is john.
A son of james is alan.

::[5]:::::::::::::::::::::::::
A daughter of florence is louise.


::[6]:::::::::::::::::::::::::
A grandparent of scott is florence.
A grandparent of jessie is florence.
A grandparent of rebecca is florence.
A grandparent of eric is florence.
A grandparent of katherine is florence.
A grandparent of donald is florence.
A grandparent of elizabeth is florence.
A grandparent of emily is florence.

::[7]:::::::::::::::::::::::::
An ancestor of david is florence.
An ancestor of louise is florence.
An ancestor of john is florence.
An ancestor of alan is florence.
An ancestor of scott is florence.
An ancestor of jessie is florence.
An ancestor of rebecca is florence.
An ancestor of eric is florence.
An ancestor of katherine is florence.
An ancestor of donald is florence.
An ancestor of elizabeth is florence.
An ancestor of emily is florence.

% runtime = 0 sec(s)
```

Fig. 2 The family database in action.

4.1 **Egfamily/0**

```
94 egfamily:- % see Figure 2
95     forall(egfamily0(N,Goal,String,Results),
96         egfamily1(N,Goal,String,Results)).
97
98 egfamily1(N,Goal,String,Results) :-
99     format('\n::[~w]:::::::::::::::::::::::::::::\n',[N]),
100    forall(Goal,
101        format(String,Results)).
```

4.2 The test suite for **Egfamily/0**

```
102 egfamily0(1,parent(X,david),
103     'The parent of david is ~w.\n',[X]).
104 egfamily0(2,father(Dad,david),
105     'The father of david is ~w.\n',[Dad]).
106 egfamily0(3,mother(Mom,david),
107     'The mother of david is ~w.\n',[Mom]).
108 egfamily0(4,son(james,Son),
109     'A son of james is ~w.\n',[Son]).
110 egfamily0(5,daughter(florence,Daugh),
111     'A daughter of florence is ~w.\n',[Daugh]).
112 egfamily0(6,grandparent(florence,GP),
113     'A grandparent of ~w is florence.\n',[GP]).
114 egfamily0(7,ancestor(florence,Anc),
115     'An ancestor of ~w is florence.\n',[Anc]).
```

4.3 `Demos/1`

`Demos/1` runs a goal `G` and catches the output to the file `G.spy`. Also, just so you know what is going on, it runs the goal `G` a second time and sends the output to the screen.

```
116 demos(G) :-
117     sformat(Out,'~w.spy',G),
118     freshFile(Out),
119     tell(Out),
120     format('% output from '':- demos(~w).''\n\n',G),
121         T1 is cputime,
122     ignore(forall(G,true)),
123     T2 is (cputime - T1),
124     format('\n% runtime = ~w sec(s)\n',[T2]),
125         told,
126     format('% output from '':- demos(~w).''\n',G),
127         ignore(forall(G,true)),
128     format('\n% runtime = ~w sec(s)',[T2]).
```

`Demos/1` needs a helper predicate. `FreshFile/1` makes sure that no one else has scribbled, or is currently scribbling, on our output file.

```
129 freshFile(X) :-
130     (current_stream(X,_,S) -> close(S)      ; true),
131     (exists_file(X)        -> delete_file(X) ; true).
```

4.4 Using `Demos/1`.

Next, we need to run the demo code as follows:

```
?- demos(egfamily).
```

Once that is done, then when this document will include the output in Figure 2.

5 Bugs

None known.

References

1. I. Bratko. *Prolog Programming for Artificial Intelligence. (third edition).* Addison-Wesley, 2001.

A License

This software is distributed under the GNU General Public License.

A.1 nowarranty.txt

FAMILY comes with ABSOLUTELY NO WARRANTY: for more details type 'warranty'.

This is free software, and you are welcome to redistribute it under certain conditions: for more details, type 'conditions'.

A.2 warranty.txt

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; version 2 (see http://www.gnu.org/copyleft/gpl.html or type 'conditions').

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, US.

A.3 conditions.txt

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The 'Program', below, refers to any such program or work, and a 'work based on the Program' means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term 'modification'.) Each licensee is addressed as 'you'.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

Fig. 3 This document is part of the PROD delivery and documentation tool for Prolog applications. To find out more about PROD, the best place to start is memo #2.