CS 470 Spring 2011
Project 4

Colby Blair

Due May 10th, 2011

Abstract

There are many different approaches to implementing Artificially Intelligent agents. One approach is to use a symbolic approach. This report will use a logic-based area of the symbolic approach. This approach does not try to simulate the thought process of humans, but instead tries to find the basis for logical decision making. This seems to suggest that the Turing Test isn't the best measure for intelligence, as humans are imperfect and often act irrational.

This report uses Prolog to implement the logic-based symbolic family tree. This approach uses a knowledge base that the logic agent consults. Specific relationships are defined, and more general rules are made as well. This is a quick process, but seems to lack a core AI element (like learning) to be truely intelligent. It is, however, much quicker, and much more logical, than other methods.

Contents

List of Figures

1 Introduction

For this report, Prolog is used to create a family tree. For a logic based approach, Prolog is a good language. It doesn't do a lot of other things that other languages do, but it does relationships in the form of a knowledge base very well. For each possible input, there are possible outputs, much like other AI methods. If a query is defined in the knowledge base, the query is acknowledges with a success. Otherwise, it is a failure, which may just mean that the knowledge base is not complete.

All the rules in the knowledge base are tried until one succeeds. This equivaltes to Conjunctive Normal Form (CNF) statements.

$$(A \vee B) \wedge (C \vee D) \wedge ...$$

Figure 1: A CNF Statement

For this report, we will consider the following family trees:

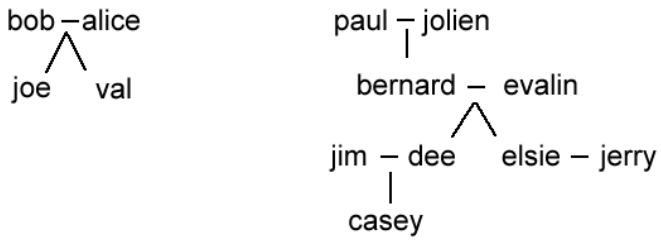

Figure 2: Family Tree

2 Example Knowledge Base Syntax

There was many ways to defined relationships for the family tree. The way this report chooses to define relationships between two specific people is mostly by common parents. Most of the family tree is constructed this way. Being married without children, and being male or female, are the only other

specific definition needed for each person. The full knowledge base is listed in Appendice A.

```
parent(jim,casey).
parent(dee,casey).

man(jim).
man(casey).
woman(dee).
```

Figure 3: Some specific definitions

The tree is then built with this concept. Rules that do not define specific people, but relationships between variables, then use this parent-child structure.

```
child(X,Y):- parent(Y,X).
sibling(X,Y):- parent(Z,X),parent(Z,Y).
```

Figure 4: Some definitions that use specific parent definitions

Consider the sibling rule in Figure 4. Prolog uses goal-based evaluation here to try every person who is a parent, assigns them to the 'Z' variable, and sees if some Z is a parent for both X and Y. This is an extremely powerful feature of Prolog, and simplifies the code. Anywhere in this knowledge base, 'Z' is used as the search variable for conditions.

3 Results

The results for searches are nice and quick:

```
| ?- uncle(jerry,casey).
true ?
yes
```

Figure 5: Uncle query

Even for some of the more intense questions:

```
| ?- related(jerry,casey).
true ?
yes
```

Figure 6: Related query

```
| ?- ancestor(paul,casey).
true ?
yes
```

Figure 7: Ancestor query

Simple queries work fine too:

```
| ?- sister(dee,elsie).
true ?
yes
```

Figure 8: Sisters query

But the relationship issues with sibling and married are still avoided:

```
| ?- married(jim,dee).
yes
| ?- married(jerry,elsie).
(1 ms) yes
| ?- married(dee,elsie).
no
```

Figure 9: Married and sibling query

4 Discussion

This knowledge base choose parents as the root way to define most relationships. This had pros and cons. There was a little more complexity with

married people, because you could not define them as having common parents, even though they would be parent-in-laws. This would be too close to the sibling definition. The result was to associate them by common children, and only define real parents in the scheme.

Another con to this knowledge base is that it makes very traditional assumptions on families. If one had common children with another, they would be married. Although sexes aren't checked here, they are checked under aunt and uncle definitions.

It is very easy to add new rules to the knowledge base. So much so, that almost every rule is based off the parent, man, or woman rules, or off other rules that are. The special cases are kept to a minumum. General queries always take the form of 'role(X,Y)', or 'X is the role of Y'.

Overall, Prolog is extremely powerful at this niche in compter science and AI. So much so, that it would be a great tool to keep in the toolbox for future projects. It defines relationships very quickly, something that can become long winded in a lower level language. It isn't as cryptic as Lisp, and has some real power with goal-oriented evaluation.

5 Full Knowlede Base Definition Explanations

5.1 parent

Like mentioned before, parents are one of the few definitions that use names only:

```
parent(jim, casey).
parent(dee, casey).
```

Figure 10: Parent definition

5.2 gender

Simple enough:

```
man(jim).
man(casey).
woman(dee).
```

Figure 11: gender definition

5.3 child

The reverse of parent. You are a child of someone if they are a parent of you:

```
child(X,Y):- parent(Y,X).
```

Figure 12: child definition

5.4 mother

You are a mother of someone if you are a parent and a woman:

```
mother(X,Y)    :- parent(X,Y),woman(X).
```

Figure 13: mother definition

5.5 father

You are a father of someone if you are a parent and a man:

```
father(X,Y)    :- parent(X,Y),man(X).
```

Figure 14: father definition

5.6 sister

You are a sister of someone if you are a sibling and a woman:

```
sister(X,Y)    :- sibling(X,Y),woman(X).
```

Figure 15: sister definition

5.7 brother

You area a brother os someone if you are a sibling and a man:

```
brother(X,Y)   :- sibling(X,Y),man(X).
```

Figure 16: brother definition

5.8 sibling

You are a sibling of someone if you have common parents:

```
sibling(X,Y):- parent(Z,X),parent(Z,Y).
```

Figure 17: sibling definition

5.9 aunt

You (X) are an aunt of someone if you are either a). a sister of some Z, and that Z is a parent of Y, or b). married to some Z, and that Z is an uncle to Y.

```
aunt(X,Y)      :-          (sister(X,Z),parent(Z,Y));
                           (married(X,Z),uncle(Z,Y)).
```

Figure 18: aunt definition

5.10 uncle

You (X) are a uncle of someone if a). you are a brothr of some Z, and that Z is a parent of Y, or b). you are married to some Z, and they are an aunt to some y.

```
uncle(X,Y)    :-    (brother(X,Z),parent(Z,Y));
                    (married(X,Z),aunt(Z,Y)).
```

Figure 19: uncle definition

5.11 related

This report tried to use ancestor and decendent to implement 'related', but got stack overflows. This was probably an issue with infinite recursion. The solution was to use rules like ancestor and decendent. If you (X) are related to Y, then you are a parent, child, or married to someone that is a parent, child, or married to, someone who... through recursion, is eventual a parent, child, or married to some relation to you.

```
related(X,Y)  :- parent(X,Y).
related(X,Y)  :- parent(X,Z),related(Z,Y).
related(X,Y)  :- child(X,Y).
related(X,Y)  :- child(X,Z),related(Z,Y).
related(X,Y)  :- married(X,Y).
related(X,Y)  :- married(X,Z),related(Z,Y).
```

Figure 20: related definition

5.12 decendent

You (X) are a decendent if you are a child of someone who is a child of someone... recursively, until someone who is a child of the ancestor (Y).

```
decendent(X,Y):- child(X,Y).
decendent(X,Y):- child(X,Z),decendent(Z,Y).
```

Figure 21: decendent definition

5.13 ancestor

You (X) are an ancestor of someone if you are a parent to someone who is a parent... recursively, until someone who is a parent of a child (Y).

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

Figure 22: ancestor definition

Appendices

A The full Knowledge Base

```
%(X,Y) - X is a <> of Y

%specific definitions
parent(bob,joe).
parent(alice,joe).
parent(bob,val).
parent(alice,val).
parent(paul,bernard).
parent(jolien,bernard).
parent(bernard,dee).
parent(evalin,dee).
parent(bernard,elsie).
parent(evalin,elsie).
parent(jim,casey).
parent(dee,casey).

%specific definitions
man(joe).
man(bob).
man(jerry).
man(jim).
man(casey).
man(paul).
woman(val).
woman(alice).
woman(elsie).
woman(dee).
woman(jolien).

%non-sex specific statements
child(X,Y):- parent(Y,X).
sibling(X,Y):- parent(Z,X),parent(Z,Y).
```

```
married(X,Y)   :- parent(X,Z),parent(Y,Z).        %have common children
        married(jerry,elsie).                     %don't have children

%sex specific statements
father(X,Y)    :- parent(X,Y),man(X).
mother(X,Y)    :- parent(X,Y),woman(X).
sister(X,Y)    :- sibling(X,Y),woman(X).
brother(X,Y)   :- sibling(X,Y),man(X).
aunt(X,Y)      :- (sister(X,Z),parent(Z,Y));(married(X,Z),uncle(Z,Y)).
uncle(X,Y)     :- (brother(X,Z),parent(Z,Y));(married(X,Z),aunt(Z,Y)).

%others
ancestor(X,Y)  :- parent(X,Y).
ancestor(X,Y)  :- parent(X,Z),ancestor(Z,Y).
decendent(X,Y):- child(X,Y).
decendent(X,Y):- child(X,Z),decendent(Z,Y).
related(X,Y)   :- parent(X,Y).
related(X,Y)   :- parent(X,Z),related(Z,Y).
related(X,Y)   :- child(X,Y).
related(X,Y)   :- child(X,Z),related(Z,Y).
related(X,Y)   :- married(X,Y).
related(X,Y)   :- married(X,Z),related(Z,Y).
```