CZ3005-ARTIFICIAL INTELLIGENCE

Lab Report

Zhang Xinye-SSP1

U1622118K

# Screenshots of the introduction

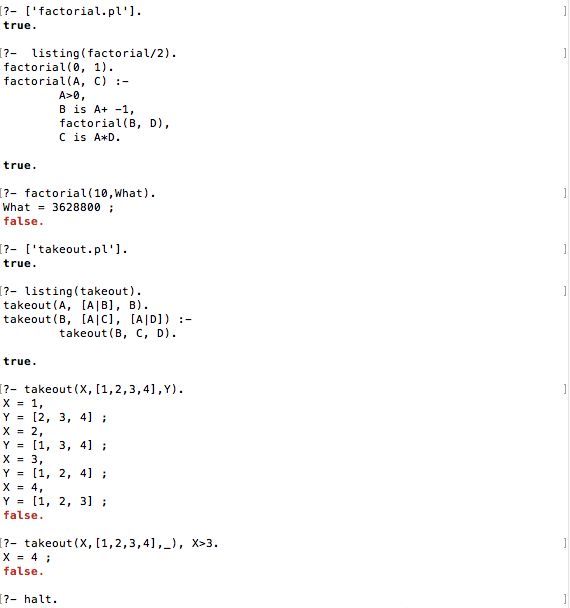


Figure 1. How to Run Prolog



Figure 2 Loading programs, editing programs



Figure 3Prolog as a Knowledge Base System

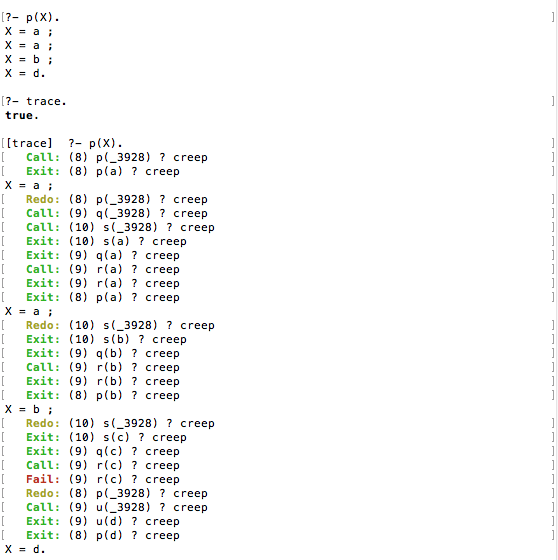


Figure 4 Prolog derivation trees, choices, and unification (1)

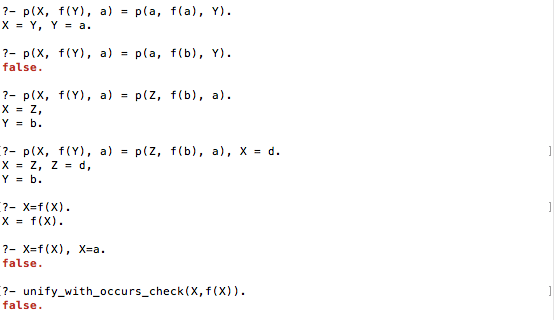


Figure 5 Prolog derivation trees, choices, and unification (2)

# Example 1: on Family Tree

## Implementation

1. % use tabling to avoid infinite recurtion.
2. :- use\_module(library(tabling)).
3. :- table parent\_of/2.
4. :- discontiguous 'parent\_of tabled'/2.
5. male(gf1).
6. male(u1).
7. male(u2).
8. male(u3).
9. male(yourself).
10. male(gs2).
11. male(gs3).
12. female(gm1).
13. female(a1).
14. female(a2).
15. female(a3).
16. female(gd1).
17. female(gd2).
18. brother(u1, u2).
19. brother(u2, u1).
20. brother(u2, u3).
21. brother(u3, u2).
22. brother(u1, u3).
23. brother(u3, u1).
24. brother(u1, a1).
25. brother(u1, a2).
26. brother(u1, a3).
27. brother(u2, a1).
28. brother(u2, a2).
29. brother(u2, a3).
30. brother(u3, a1).
31. brother(u3, a2).
32. brother(u3, a3).
33. brother(yourself, gs2).
34. brother(gs2, gs2).
35. brother(yourself, gd1).
36. brother(gs2, gd1).
37. brother(gs3, gd2).
38. sister(a1, a2).
39. sister(a2, a1).
40. sister(a2, a3).
41. sister(a3, a2).
42. sister(a1, a3).
43. sister(a3, a1).
44. sister(a1, u1).
45. sister(a1, u2).
46. sister(a1, u3).
47. sister(a2, u1).
48. sister(a2, u2).
49. sister(a2, u3).
50. sister(a3, u1).
51. sister(a3, u2).
52. sister(a3, u3).
53. sister(gd1, yourself).
54. sister(gd1, gs2).
55. sister(gd2, gs3).
56. parent\_of(gf1, u1).
57. parent\_of(gf1, u2).
58. parent\_of(gf1, u3).
59. parent\_of(gf1, a1).
60. parent\_of(gf1, a2).
61. parent\_of(gf1, a3).
62. parent\_of(gm1, u1).
63. parent\_of(gm1, u2).
64. parent\_of(gm1, u3).
65. parent\_of(gm1, a1).
66. parent\_of(gm1, a2).
67. parent\_of(gm1, a3).
68. parent\_of(u1, yourself).
69. parent\_of(a1, yourself).
70. parent\_of(u1, gs2).
71. parent\_of(a1, gs2).
72. parent\_of(u1, gd1).
73. parent\_of(a1, gd1).
74. parent\_of(u2, gs3).
75. parent\_of(u2, gd2).
76. parent\_of(a2, gs3).
77. parent\_of(a2, gd2).
78. father(X, Y) :- % X is father of Y
79. parent\_of(X, Y), % X is parent of Y and
80. male(X). % X is male
81. mother(X, Y) :- % X is mother of Y
82. parent\_of(X, Y), % X is parent of Y and
83. female(X). % X is female
84. son(X, Y) :- % X is son of Y
85. parent\_of(Y, X), % Y is parent of X and
86. male(X). % X is male
87. daughter(X, Y) :- % X is daughter of Y
88. parent\_of(Y, X), % Y is parent of X and
89. female(X). % X is female
90. grandfather(X, Y) :- % X is grandfather of Y
91. father(X, Z), % X is father of Z and
92. parent\_of(Z, Y). % Z is parent of Y
93. sibling(X, Y) :- % X is sibling of Y
94. brother(X, Y); % X is brother of Y or
95. brother(Y, X); % Y is brother of X or
96. sister(X, Y); % X is sister of Y or
97. sister(Y, X). % Y is sister of X
98. spouse(X,Y):- % X is spouse of Y
99. parent\_of(X,A), % X is parent of A and
100. male(X), % X is male and
101. parent\_of(Y,A), % Y is parent of A and
102. female(Y). % Y is female
103. uncle(X, Y):- % X is uncle of Y
104. parent\_of(P,Y), % P is parent of Y and
105. (brother(P,X); % P is brother of X or
106. brother(X,P)), % X is brother of P and
107. male(X). % X is male
108. aunt(X, Y):- % X is aunt of Y
109. parent\_of(P,Y), % P is parent of Y and
110. (sister(P,X); % P is sister of X or
111. sister(X,P)), % X is sister of P and
112. female(X). % X is female
113. cousin(X, Y) :- % X is cousin of Y
114. parent\_of(P1, X), % P1 is parent of X and
115. parent\_of(P2, Y), % P2 is parent of Y and
116. sibling(P1, P2). % P1 P2 are sibling

Because parent\_of/2 is defined recursively, infinite recursion is possible to occur. So that, I have used tabling package to memorize the previous searching result of parent\_of/2. In such a way, infinite recursion is dissolved.

## Queries

1. Spouse(X, Y).

?- spouse(X,Y).

X = gf1,

Y = gm1 **;**

X = gf1,

Y = gm1 **;**

X = gf1,

Y = gm1 **;**

X = gf1,

Y = gm1 **;**

X = gf1,

Y = gm1 **;**

X = gf1,

Y = gm1 **;**

X = u1,

Y = a1 **;**

X = u1,

Y = a1 **;**

X = u1,

Y = a1 **;**

X = u2,

Y = a2 **;**

X = u2,

Y = a2 **;**

**false.**

?- trace.

**true.**

[trace] ?- spouse(X,Y).

**Call:** (8) spouse(\_4218, \_4220) ? creep

**Call:** (9) parent\_of(\_4218, \_4470) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), 'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, u1), user:'parent\_of tabled'(gf1, u1)) ? creep

**Exit:** (9) parent\_of(gf1, u1) ? creep

**Call:** (9) male(gf1) ? creep

**Exit:** (9) male(gf1) ? creep

**Call:** (9) parent\_of(\_4220, u1) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4220, u1), 'parent\_of tabled'(\_4220, u1)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, u1), user:'parent\_of tabled'(gf1, u1)) ? creep

**Exit:** (9) parent\_of(gf1, u1) ? creep

**Call:** (9) female(gf1) ? creep

**Fail:** (9) female(gf1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4220, u1), user:'parent\_of tabled'(\_4220, u1)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, u1), user:'parent\_of tabled'(gm1, u1)) ? creep

**Exit:** (9) parent\_of(gm1, u1) ? creep

**Call:** (9) female(gm1) ? creep

**Exit:** (9) female(gm1) ? creep

**Exit:** (8) spouse(gf1, gm1) ? creep

X = gf1,

Y = gm1 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, u2), user:'parent\_of tabled'(gf1, u2)) ? creep

**Exit:** (9) parent\_of(gf1, u2) ? creep

**Call:** (9) male(gf1) ? creep

**Exit:** (9) male(gf1) ? creep

**Call:** (9) parent\_of(\_4220, u2) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4220, u2), 'parent\_of tabled'(\_4220, u2)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, u2), user:'parent\_of tabled'(gf1, u2)) ? creep

**Exit:** (9) parent\_of(gf1, u2) ? creep

**Call:** (9) female(gf1) ? creep

**Fail:** (9) female(gf1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4220, u2), user:'parent\_of tabled'(\_4220, u2)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, u2), user:'parent\_of tabled'(gm1, u2)) ? creep

**Exit:** (9) parent\_of(gm1, u2) ? creep

**Call:** (9) female(gm1) ? creep

**Exit:** (9) female(gm1) ? creep

**Exit:** (8) spouse(gf1, gm1) ? creep

X = gf1,

Y = gm1 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, u3), user:'parent\_of tabled'(gf1, u3)) ? creep

**Exit:** (9) parent\_of(gf1, u3) ? creep

**Call:** (9) male(gf1) ? creep

**Exit:** (9) male(gf1) ? creep

**Call:** (9) parent\_of(\_4220, u3) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4220, u3), 'parent\_of tabled'(\_4220, u3)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, u3), user:'parent\_of tabled'(gf1, u3)) ? creep

**Exit:** (9) parent\_of(gf1, u3) ? creep

**Call:** (9) female(gf1) ? creep

**Fail:** (9) female(gf1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4220, u3), user:'parent\_of tabled'(\_4220, u3)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, u3), user:'parent\_of tabled'(gm1, u3)) ? creep

**Exit:** (9) parent\_of(gm1, u3) ? creep

**Call:** (9) female(gm1) ? creep

**Exit:** (9) female(gm1) ? creep

**Exit:** (8) spouse(gf1, gm1) ? creep

X = gf1,

Y = gm1 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, a1), user:'parent\_of tabled'(gf1, a1)) ? creep

**Exit:** (9) parent\_of(gf1, a1) ? creep

**Call:** (9) male(gf1) ? creep

**Exit:** (9) male(gf1) ? creep

**Call:** (9) parent\_of(\_4220, a1) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4220, a1), 'parent\_of tabled'(\_4220, a1)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, a1), user:'parent\_of tabled'(gf1, a1)) ? creep

**Exit:** (9) parent\_of(gf1, a1) ? creep

**Call:** (9) female(gf1) ? creep

**Fail:** (9) female(gf1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4220, a1), user:'parent\_of tabled'(\_4220, a1)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, a1), user:'parent\_of tabled'(gm1, a1)) ? creep

**Exit:** (9) parent\_of(gm1, a1) ? creep

**Call:** (9) female(gm1) ? creep

**Exit:** (9) female(gm1) ? creep

**Exit:** (8) spouse(gf1, gm1) ? creep

X = gf1,

Y = gm1 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, a2), user:'parent\_of tabled'(gf1, a2)) ? creep

**Exit:** (9) parent\_of(gf1, a2) ? creep

**Call:** (9) male(gf1) ? creep

**Exit:** (9) male(gf1) ? creep

**Call:** (9) parent\_of(\_4220, a2) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4220, a2), 'parent\_of tabled'(\_4220, a2)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, a2), user:'parent\_of tabled'(gf1, a2)) ? creep

**Exit:** (9) parent\_of(gf1, a2) ? creep

**Call:** (9) female(gf1) ? creep

**Fail:** (9) female(gf1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4220, a2), user:'parent\_of tabled'(\_4220, a2)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, a2), user:'parent\_of tabled'(gm1, a2)) ? creep

**Exit:** (9) parent\_of(gm1, a2) ? creep

**Call:** (9) female(gm1) ? creep

**Exit:** (9) female(gm1) ? creep

**Exit:** (8) spouse(gf1, gm1) ? creep

X = gf1,

Y = gm1 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, a3), user:'parent\_of tabled'(gf1, a3)) ? creep

**Exit:** (9) parent\_of(gf1, a3) ? creep

**Call:** (9) male(gf1) ? creep

**Exit:** (9) male(gf1) ? creep

**Call:** (9) parent\_of(\_4220, a3) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4220, a3), 'parent\_of tabled'(\_4220, a3)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gf1, a3), user:'parent\_of tabled'(gf1, a3)) ? creep

**Exit:** (9) parent\_of(gf1, a3) ? creep

**Call:** (9) female(gf1) ? creep

**Fail:** (9) female(gf1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4220, a3), user:'parent\_of tabled'(\_4220, a3)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, a3), user:'parent\_of tabled'(gm1, a3)) ? creep

**Exit:** (9) parent\_of(gm1, a3) ? creep

**Call:** (9) female(gm1) ? creep

**Exit:** (9) female(gm1) ? creep

**Exit:** (8) spouse(gf1, gm1) ? creep

X = gf1,

Y = gm1 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(u1, yourself), user:'parent\_of tabled'(u1, yourself)) ? creep

**Exit:** (9) parent\_of(u1, yourself) ? creep

**Call:** (9) male(u1) ? creep

**Exit:** (9) male(u1) ? creep

**Call:** (9) parent\_of(\_4220, yourself) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4220, yourself), 'parent\_of tabled'(\_4220, yourself)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(u1, yourself), user:'parent\_of tabled'(u1, yourself)) ? creep

**Exit:** (9) parent\_of(u1, yourself) ? creep

**Call:** (9) female(u1) ? creep

**Fail:** (9) female(u1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4220, yourself), user:'parent\_of tabled'(\_4220, yourself)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(a1, yourself), user:'parent\_of tabled'(a1, yourself)) ? creep

**Exit:** (9) parent\_of(a1, yourself) ? creep

**Call:** (9) female(a1) ? creep

**Exit:** (9) female(a1) ? creep

**Exit:** (8) spouse(u1, a1) ? creep

X = u1,

Y = a1 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(u1, gs2), user:'parent\_of tabled'(u1, gs2)) ? creep

**Exit:** (9) parent\_of(u1, gs2) ? creep

**Call:** (9) male(u1) ? creep

**Exit:** (9) male(u1) ? creep

**Call:** (9) parent\_of(\_4220, gs2) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4220, gs2), 'parent\_of tabled'(\_4220, gs2)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(u1, gs2), user:'parent\_of tabled'(u1, gs2)) ? creep

**Exit:** (9) parent\_of(u1, gs2) ? creep

**Call:** (9) female(u1) ? creep

**Fail:** (9) female(u1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4220, gs2), user:'parent\_of tabled'(\_4220, gs2)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(a1, gs2), user:'parent\_of tabled'(a1, gs2)) ? creep

**Exit:** (9) parent\_of(a1, gs2) ? creep

**Call:** (9) female(a1) ? creep

**Exit:** (9) female(a1) ? creep

**Exit:** (8) spouse(u1, a1) ? creep

X = u1,

Y = a1 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(u1, gd1), user:'parent\_of tabled'(u1, gd1)) ? creep

**Exit:** (9) parent\_of(u1, gd1) ? creep

**Call:** (9) male(u1) ? creep

**Exit:** (9) male(u1) ? creep

**Call:** (9) parent\_of(\_4220, gd1) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4220, gd1), 'parent\_of tabled'(\_4220, gd1)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(u1, gd1), user:'parent\_of tabled'(u1, gd1)) ? creep

**Exit:** (9) parent\_of(u1, gd1) ? creep

**Call:** (9) female(u1) ? creep

**Fail:** (9) female(u1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4220, gd1), user:'parent\_of tabled'(\_4220, gd1)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(a1, gd1), user:'parent\_of tabled'(a1, gd1)) ? creep

**Exit:** (9) parent\_of(a1, gd1) ? creep

**Call:** (9) female(a1) ? creep

**Exit:** (9) female(a1) ? creep

**Exit:** (8) spouse(u1, a1) ? creep

X = u1,

Y = a1 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(u2, gd2), user:'parent\_of tabled'(u2, gd2)) ? creep

**Exit:** (9) parent\_of(u2, gd2) ? creep

**Call:** (9) male(u2) ? creep

**Exit:** (9) male(u2) ? creep

**Call:** (9) parent\_of(\_4220, gd2) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4220, gd2), 'parent\_of tabled'(\_4220, gd2)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(a2, gd2), user:'parent\_of tabled'(a2, gd2)) ? creep

**Exit:** (9) parent\_of(a2, gd2) ? creep

**Call:** (9) female(a2) ? creep

**Exit:** (9) female(a2) ? creep

**Exit:** (8) spouse(u2, a2) ? creep

X = u2,

Y = a2 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4220, gd2), user:'parent\_of tabled'(\_4220, gd2)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(u2, gd2), user:'parent\_of tabled'(u2, gd2)) ? creep

**Exit:** (9) parent\_of(u2, gd2) ? creep

**Call:** (9) female(u2) ? creep

**Fail:** (9) female(u2) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(u2, gs3), user:'parent\_of tabled'(u2, gs3)) ? creep

**Exit:** (9) parent\_of(u2, gs3) ? creep

**Call:** (9) male(u2) ? creep

**Exit:** (9) male(u2) ? creep

**Call:** (9) parent\_of(\_4220, gs3) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4220, gs3), 'parent\_of tabled'(\_4220, gs3)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(a2, gs3), user:'parent\_of tabled'(a2, gs3)) ? creep

**Exit:** (9) parent\_of(a2, gs3) ? creep

**Call:** (9) female(a2) ? creep

**Exit:** (9) female(a2) ? creep

**Exit:** (8) spouse(u2, a2) ? creep

X = u2,

Y = a2 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4220, gs3), user:'parent\_of tabled'(\_4220, gs3)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(u2, gs3), user:'parent\_of tabled'(u2, gs3)) ? creep

**Exit:** (9) parent\_of(u2, gs3) ? creep

**Call:** (9) female(u2) ? creep

**Fail:** (9) female(u2) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, u1), user:'parent\_of tabled'(gm1, u1)) ? creep

**Exit:** (9) parent\_of(gm1, u1) ? creep

**Call:** (9) male(gm1) ? creep

**Fail:** (9) male(gm1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, u2), user:'parent\_of tabled'(gm1, u2)) ? creep

**Exit:** (9) parent\_of(gm1, u2) ? creep

**Call:** (9) male(gm1) ? creep

**Fail:** (9) male(gm1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, u3), user:'parent\_of tabled'(gm1, u3)) ? creep

**Exit:** (9) parent\_of(gm1, u3) ? creep

**Call:** (9) male(gm1) ? creep

**Fail:** (9) male(gm1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, a1), user:'parent\_of tabled'(gm1, a1)) ? creep

**Exit:** (9) parent\_of(gm1, a1) ? creep

**Call:** (9) male(gm1) ? creep

**Fail:** (9) male(gm1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, a2), user:'parent\_of tabled'(gm1, a2)) ? creep

**Exit:** (9) parent\_of(gm1, a2) ? creep

**Call:** (9) male(gm1) ? creep

**Fail:** (9) male(gm1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(gm1, a3), user:'parent\_of tabled'(gm1, a3)) ? creep

**Exit:** (9) parent\_of(gm1, a3) ? creep

**Call:** (9) male(gm1) ? creep

**Fail:** (9) male(gm1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(a1, yourself), user:'parent\_of tabled'(a1, yourself)) ? creep

**Exit:** (9) parent\_of(a1, yourself) ? creep

**Call:** (9) male(a1) ? creep

**Fail:** (9) male(a1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(a1, gs2), user:'parent\_of tabled'(a1, gs2)) ? creep

**Exit:** (9) parent\_of(a1, gs2) ? creep

**Call:** (9) male(a1) ? creep

**Fail:** (9) male(a1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(a1, gd1), user:'parent\_of tabled'(a1, gd1)) ? creep

**Exit:** (9) parent\_of(a1, gd1) ? creep

**Call:** (9) male(a1) ? creep

**Fail:** (9) male(a1) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(a2, gd2), user:'parent\_of tabled'(a2, gd2)) ? creep

**Exit:** (9) parent\_of(a2, gd2) ? creep

**Call:** (9) male(a2) ? creep

**Fail:** (9) male(a2) ? creep

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4218, \_4462), user:'parent\_of tabled'(\_4218, \_4462)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(a2, gs3), user:'parent\_of tabled'(a2, gs3)) ? creep

**Exit:** (9) parent\_of(a2, gs3) ? creep

**Call:** (9) male(a2) ? creep

**Fail:** (9) male(a2) ? creep

**Fail:** (8) spouse(\_4218, \_4220) ? creep

**false.**

1. Uncle(X, yourself).

?- uncle(X, yourself).

X = u2 **;**

X = u3 **;**

X = u2 **;**

X = u3 **;**

X = u2 **;**

X = u3.

?- trace.

**true.**

[trace] ?- uncle(X, yourself).

**Call:** (8) uncle(\_4020, yourself) ? creep

**Call:** (9) parent\_of(\_4240, yourself) ? creep

^ **Call:** (10) tabling:start\_tabling(user:parent\_of(\_4232, yourself), 'parent\_of tabled'(\_4232, yourself)) ? creep

**Call:** (16) 'parent\_of tabled'(\_4232, yourself) ? creep

**Exit:** (16) 'parent\_of tabled'(u1, yourself) ? creep

**Redo:** (16) 'parent\_of tabled'(\_4232, yourself) ? creep

**Exit:** (16) 'parent\_of tabled'(a1, yourself) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(u1, yourself), user:'parent\_of tabled'(u1, yourself)) ? creep

**Exit:** (9) parent\_of(u1, yourself) ? creep

**Call:** (9) brother(u1, \_4020) ? creep

**Exit:** (9) brother(u1, u2) ? creep

^ **Call:** (9) not(parent\_of(u2, yourself)) ? creep

**Call:** (10) parent\_of(u2, yourself) ? creep

^ **Call:** (11) tabling:start\_tabling(user:parent\_of(u2, yourself), 'parent\_of tabled'(u2, yourself)) ? creep

**Call:** (17) 'parent\_of tabled'(u2, yourself) ? creep

**Fail:** (17) 'parent\_of tabled'(u2, yourself) ? creep

^ **Fail:** (11) tabling:start\_tabling(user:parent\_of(u2, yourself), user:'parent\_of tabled'(u2, yourself)) ? creep

**Fail:** (10) parent\_of(u2, yourself) ? creep

^ **Exit:** (9) not(user:parent\_of(u2, yourself)) ? creep

**Call:** (9) male(u2) ? creep

**Exit:** (9) male(u2) ? creep

**Exit:** (8) uncle(u2, yourself) ? creep

X = u2 **;**

**Redo:** (9) brother(u1, \_4020) ? creep

**Exit:** (9) brother(u1, u3) ? creep

^ **Call:** (9) not(parent\_of(u3, yourself)) ? creep

**Call:** (10) parent\_of(u3, yourself) ? creep

^ **Call:** (11) tabling:start\_tabling(user:parent\_of(u3, yourself), 'parent\_of tabled'(u3, yourself)) ? creep

**Call:** (17) 'parent\_of tabled'(u3, yourself) ? creep

**Fail:** (17) 'parent\_of tabled'(u3, yourself) ? creep

^ **Fail:** (11) tabling:start\_tabling(user:parent\_of(u3, yourself), user:'parent\_of tabled'(u3, yourself)) ? creep

**Fail:** (10) parent\_of(u3, yourself) ? creep

^ **Exit:** (9) not(user:parent\_of(u3, yourself)) ? creep

**Call:** (9) male(u3) ? creep

**Exit:** (9) male(u3) ? creep

**Exit:** (8) uncle(u3, yourself) ? creep

X = u3 **;**

**Redo:** (9) brother(u1, \_4020) ? creep

**Exit:** (9) brother(u1, a1) ? creep

^ **Call:** (9) not(parent\_of(a1, yourself)) ? creep

**Call:** (10) parent\_of(a1, yourself) ? creep

^ **Call:** (11) tabling:start\_tabling(user:parent\_of(a1, yourself), 'parent\_of tabled'(a1, yourself)) ? creep

**Call:** (17) 'parent\_of tabled'(a1, yourself) ? creep

**Exit:** (17) 'parent\_of tabled'(a1, yourself) ? creep

^ **Exit:** (11) tabling:start\_tabling(user:parent\_of(a1, yourself), user:'parent\_of tabled'(a1, yourself)) ? creep

**Exit:** (10) parent\_of(a1, yourself) ? creep

^ **Fail:** (9) not(user:parent\_of(a1, yourself)) ? creep

**Redo:** (9) brother(u1, \_4020) ? creep

**Exit:** (9) brother(u1, a2) ? creep

^ **Call:** (9) not(parent\_of(a2, yourself)) ? creep

**Call:** (10) parent\_of(a2, yourself) ? creep

^ **Call:** (11) tabling:start\_tabling(user:parent\_of(a2, yourself), 'parent\_of tabled'(a2, yourself)) ? creep

**Call:** (17) 'parent\_of tabled'(a2, yourself) ? creep

**Fail:** (17) 'parent\_of tabled'(a2, yourself) ? creep

^ **Fail:** (11) tabling:start\_tabling(user:parent\_of(a2, yourself), user:'parent\_of tabled'(a2, yourself)) ? creep

**Fail:** (10) parent\_of(a2, yourself) ? creep

^ **Exit:** (9) not(user:parent\_of(a2, yourself)) ? creep

**Call:** (9) male(a2) ? creep

**Fail:** (9) male(a2) ? creep

**Redo:** (9) brother(u1, \_4020) ? creep

**Exit:** (9) brother(u1, a3) ? creep

^ **Call:** (9) not(parent\_of(a3, yourself)) ? creep

**Call:** (10) parent\_of(a3, yourself) ? creep

^ **Call:** (11) tabling:start\_tabling(user:parent\_of(a3, yourself), 'parent\_of tabled'(a3, yourself)) ? creep

**Call:** (17) 'parent\_of tabled'(a3, yourself) ? creep

**Fail:** (17) 'parent\_of tabled'(a3, yourself) ? creep

^ **Fail:** (11) tabling:start\_tabling(user:parent\_of(a3, yourself), user:'parent\_of tabled'(a3, yourself)) ? creep

**Fail:** (10) parent\_of(a3, yourself) ? creep

^ **Exit:** (9) not(user:parent\_of(a3, yourself)) ? creep

**Call:** (9) male(a3) ? creep

**Fail:** (9) male(a3) ? creep

**Redo:** (8) uncle(\_4020, yourself) ? creep

**Call:** (9) brother(\_4020, u1) ? creep

**Exit:** (9) brother(u2, u1) ? creep

^ **Call:** (9) not(parent\_of(u2, yourself)) ? creep

**Call:** (10) parent\_of(u2, yourself) ? creep

^ **Call:** (11) tabling:start\_tabling(user:parent\_of(u2, yourself), 'parent\_of tabled'(u2, yourself)) ? creep

^ **Fail:** (11) tabling:start\_tabling(user:parent\_of(u2, yourself), user:'parent\_of tabled'(u2, yourself)) ? creep

**Fail:** (10) parent\_of(u2, yourself) ? creep

^ **Exit:** (9) not(user:parent\_of(u2, yourself)) ? creep

**Call:** (9) male(u2) ? creep

**Exit:** (9) male(u2) ? creep

**Exit:** (8) uncle(u2, yourself) ? creep

X = u2 **;**

**Redo:** (9) brother(\_4020, u1) ? creep

**Exit:** (9) brother(u3, u1) ? creep

^ **Call:** (9) not(parent\_of(u3, yourself)) ? creep

**Call:** (10) parent\_of(u3, yourself) ? creep

^ **Call:** (11) tabling:start\_tabling(user:parent\_of(u3, yourself), 'parent\_of tabled'(u3, yourself)) ? creep

^ **Fail:** (11) tabling:start\_tabling(user:parent\_of(u3, yourself), user:'parent\_of tabled'(u3, yourself)) ? creep

**Fail:** (10) parent\_of(u3, yourself) ? creep

^ **Exit:** (9) not(user:parent\_of(u3, yourself)) ? creep

**Call:** (9) male(u3) ? creep

**Exit:** (9) male(u3) ? creep

**Exit:** (8) uncle(u3, yourself) ? creep

X = u3 **;**

^ **Redo:** (10) tabling:start\_tabling(user:parent\_of(\_4232, yourself), user:'parent\_of tabled'(\_4232, yourself)) ? creep

^ **Exit:** (10) tabling:start\_tabling(user:parent\_of(a1, yourself), user:'parent\_of tabled'(a1, yourself)) ? creep

**Exit:** (9) parent\_of(a1, yourself) ? creep

**Call:** (9) brother(a1, \_4020) ? creep

**Fail:** (9) brother(a1, \_4020) ? creep

**Redo:** (8) uncle(\_4020, yourself) ? creep

**Call:** (9) brother(\_4020, a1) ? creep

**Exit:** (9) brother(u1, a1) ? creep

^ **Call:** (9) not(parent\_of(u1, yourself)) ? creep

**Call:** (10) parent\_of(u1, yourself) ? creep

^ **Call:** (11) tabling:start\_tabling(user:parent\_of(u1, yourself), 'parent\_of tabled'(u1, yourself)) ? creep

**Call:** (17) 'parent\_of tabled'(u1, yourself) ? creep

**Exit:** (17) 'parent\_of tabled'(u1, yourself) ? creep

**Redo:** (17) 'parent\_of tabled'(u1, yourself) ? creep

**Fail:** (17) 'parent\_of tabled'(u1, yourself) ? creep

^ **Exit:** (11) tabling:start\_tabling(user:parent\_of(u1, yourself), user:'parent\_of tabled'(u1, yourself)) ? creep

**Exit:** (10) parent\_of(u1, yourself) ? creep

^ **Fail:** (9) not(user:parent\_of(u1, yourself)) ? creep

**Redo:** (9) brother(\_4020, a1) ? creep

**Exit:** (9) brother(u2, a1) ? creep

^ **Call:** (9) not(parent\_of(u2, yourself)) ? creep

**Call:** (10) parent\_of(u2, yourself) ? creep

^ **Call:** (11) tabling:start\_tabling(user:parent\_of(u2, yourself), 'parent\_of tabled'(u2, yourself)) ? creep

^ **Fail:** (11) tabling:start\_tabling(user:parent\_of(u2, yourself), user:'parent\_of tabled'(u2, yourself)) ? creep

**Fail:** (10) parent\_of(u2, yourself) ? creep

^ **Exit:** (9) not(user:parent\_of(u2, yourself)) ? creep

**Call:** (9) male(u2) ? creep

**Exit:** (9) male(u2) ? creep

**Exit:** (8) uncle(u2, yourself) ? creep

X = u2 **;**

**Redo:** (9) brother(\_4020, a1) ? creep

**Exit:** (9) brother(u3, a1) ? creep

^ **Call:** (9) not(parent\_of(u3, yourself)) ? creep

**Call:** (10) parent\_of(u3, yourself) ? creep

^ **Call:** (11) tabling:start\_tabling(user:parent\_of(u3, yourself), 'parent\_of tabled'(u3, yourself)) ? creep

^ **Fail:** (11) tabling:start\_tabling(user:parent\_of(u3, yourself), user:'parent\_of tabled'(u3, yourself)) ? creep

**Fail:** (10) parent\_of(u3, yourself) ? creep

^ **Exit:** (9) not(user:parent\_of(u3, yourself)) ? creep

**Call:** (9) male(u3) ? creep

**Exit:** (9) male(u3) ? creep

**Exit:** (8) uncle(u3, yourself) ? creep

X = u3.

## parent\_of(X,Y) details

1. Trace, and AND-OR tree

?- parent\_of(X,Y).

X = maryalice,

Y = jerry **;**

X = maryalice,

Y = stuart **;**

X = maryalice,

Y = kather **;**

X = warren,

Y = jerry **;**

X = warren,

Y = stuart **;**

X = warren,

Y = kather.

?- trace.

**true.**

[trace] ?- parent\_of(X,Y).

**Call:** (8) parent\_of(\_3804, \_3806) ? creep

^ **Call:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), 'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(maryalice, jerry), user:'parent\_of tabled'(maryalice, jerry)) ? creep

**Exit:** (8) parent\_of(maryalice, jerry) ? creep

X = maryalice,

Y = jerry **.**

[trace] ?- parent\_of(X,Y).

**Call:** (8) parent\_of(\_3804, \_3806) ? creep

^ **Call:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), 'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(maryalice, jerry), user:'parent\_of tabled'(maryalice, jerry)) ? creep

**Exit:** (8) parent\_of(maryalice, jerry) ? creep

X = maryalice,

Y = jerry **;**

^ **Redo:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), user:'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(maryalice, stuart), user:'parent\_of tabled'(maryalice, stuart)) ? creep

**Exit:** (8) parent\_of(maryalice, stuart) ? creep

X = maryalice,

Y = stuart **;**

^ **Redo:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), user:'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(maryalice, kather), user:'parent\_of tabled'(maryalice, kather)) ? creep

**Exit:** (8) parent\_of(maryalice, kather) ? creep

X = maryalice,

Y = kather **;**

^ **Redo:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), user:'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(warren, jerry), user:'parent\_of tabled'(warren, jerry)) ? creep

**Exit:** (8) parent\_of(warren, jerry) ? creep

X = warren,

Y = jerry **;**

^ **Redo:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), user:'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(warren, stuart), user:'parent\_of tabled'(warren, stuart)) ? creep

**Exit:** (8) parent\_of(warren, stuart) ? creep

X = warren,

Y = stuart **;**

^ **Redo:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), user:'parent\_of tabled'(\_3804, \_3806)) ? creep

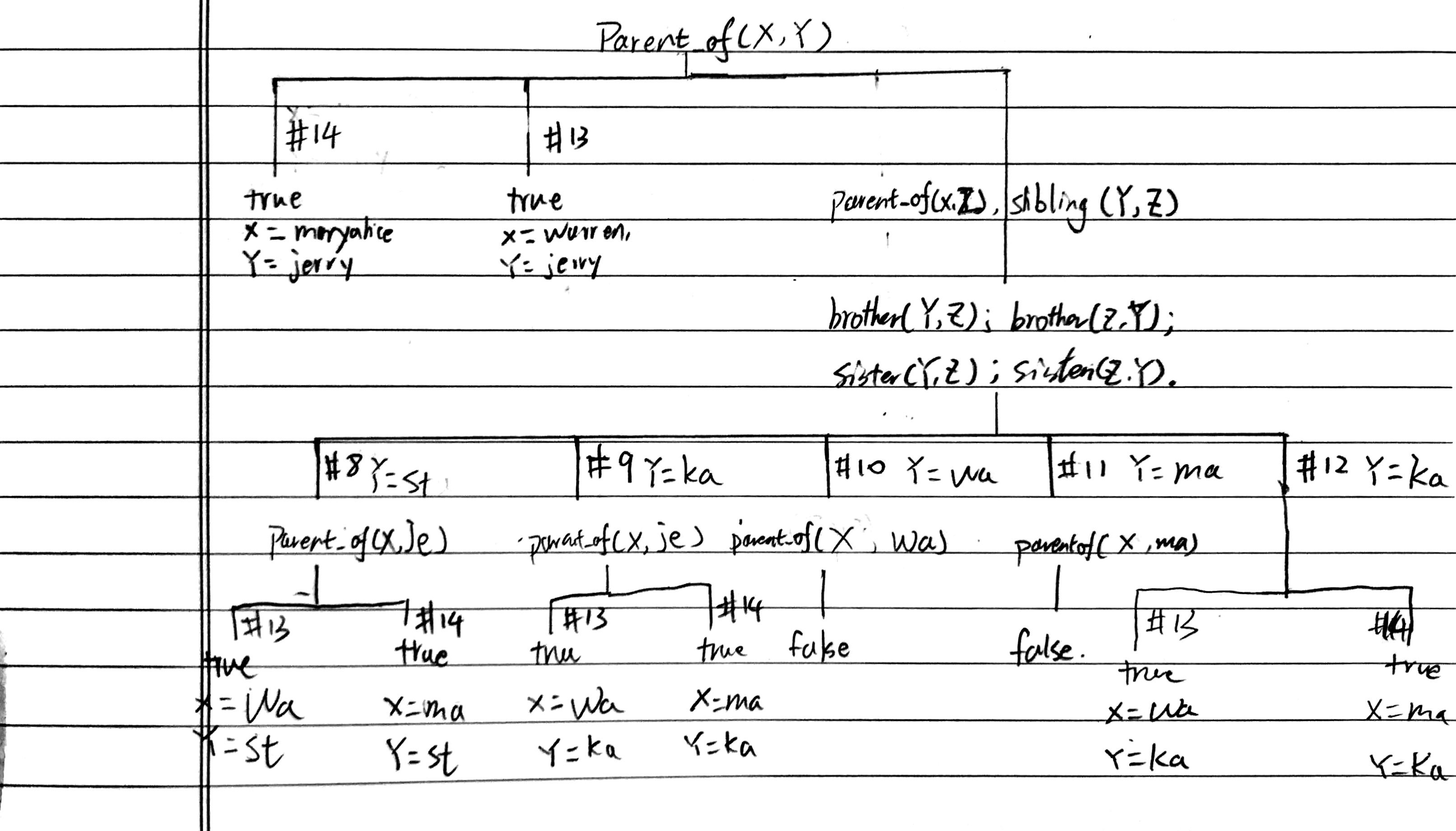
^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(warren, kather), user:'parent\_of tabled'(warren, kather)) ? creep

**Exit:** (8) parent\_of(warren, kather) ? creep

X = warren,

Y = kather.

a)



b)

After reordering, the result is:

?- parent\_of(X,Y).

X = maryalice,

Y = kather **;**

X = maryalice,

Y = stuart **;**

X = maryalice,

Y = jerry **;**

X = warren,

Y = kather **;**

X = warren,

Y = stuart **;**

X = warren,

Y = jerry.

?- trace.

**true.**

[trace] ?- parent\_of(X,Y).

**Call:** (8) parent\_of(\_3804, \_3806) ? creep

^ **Call:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), 'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(maryalice, kather), user:'parent\_of tabled'(maryalice, kather)) ? creep

**Exit:** (8) parent\_of(maryalice, kather) ? creep

X = maryalice,

Y = kather **;**

^ **Redo:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), user:'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(maryalice, stuart), user:'parent\_of tabled'(maryalice, stuart)) ? creep

**Exit:** (8) parent\_of(maryalice, stuart) ? creep

X = maryalice,

Y = stuart **;**

^ **Redo:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), user:'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(maryalice, jerry), user:'parent\_of tabled'(maryalice, jerry)) ? creep

**Exit:** (8) parent\_of(maryalice, jerry) ? creep

X = maryalice,

Y = jerry **;**

^ **Redo:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), user:'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(warren, kather), user:'parent\_of tabled'(warren, kather)) ? creep

**Exit:** (8) parent\_of(warren, kather) ? creep

X = warren,

Y = kather **;**

^ **Redo:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), user:'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(warren, stuart), user:'parent\_of tabled'(warren, stuart)) ? creep

**Exit:** (8) parent\_of(warren, stuart) ? creep

X = warren,

Y = stuart **;**

^ **Redo:** (9) tabling:start\_tabling(user:parent\_of(\_3804, \_3806), user:'parent\_of tabled'(\_3804, \_3806)) ? creep

^ **Exit:** (9) tabling:start\_tabling(user:parent\_of(warren, jerry), user:'parent\_of tabled'(warren, jerry)) ? creep

**Exit:** (8) parent\_of(warren, jerry) ? creep

X = warren,

Y = jerry.

The order of the result are different after reordering, but the result set is the same. As Prolog uses DFS, so the order of each rule’s appearance, could change the order of the result. For example, in the order of #8 before #9, stuart appears first in the result, however, in the order of #8 after #9, stuart appears after kather, which means the search engine will give a result as soon as it hit one.

# Exercise 1: The Smart Phone Rivalry

## 1. First Order Logic

company(sumSum).

company(appy).

smartPhonetTechnology(galacticaS3).

developed(galacticaS3, sumSum).

boss(stevey).

competitor(sumSum, appy).

steal(stevey, galacticaS3).

∀ Tech smartPhonetTechnology(Tech) ⇒ business(Tech)

∀ Comp competitor(Comp, appy) ∨competitor(appy, Comp) ⇒ rival(Comp)

∀ X, Biz, CompA boss(X) ∧steal(X, Biz) ∧business(Biz) ∧developed(Biz, CompA) ∧rival(CompA) ⇒ unethical(X)

## 2. Implementation

company(sumSum).

company(appy).

smartPhonetTechnology(galacticaS3).

developed(galacticaS3, sumSum).

boss(stevey).

competitor(sumSum, appy).

steal(stevey, galacticaS3).

business(Tech):-smartPhonetTechnology(Tech).

rival(Comp):- competitor(Comp, appy);competitor(appy, Comp).

unethical(X):-boss(X),steal(X, Biz),business(Biz),developed(Biz, CompA),rival(CompA).

## 3. Trace

?- unethical(stevey).

**true .**

?- trace.

**true.**

[trace] ?- unethical(stevey).

**Call:** (8) unethical(stevey) ? creep

**Call:** (9) boss(stevey) ? creep

**Exit:** (9) boss(stevey) ? creep

**Call:** (9) steal(stevey, \_4196) ? creep

**Exit:** (9) steal(stevey, galacticaS3) ? creep

**Call:** (9) business(galacticaS3) ? creep

**Call:** (10) smartPhonetTechnology(galacticaS3) ? creep

**Exit:** (10) smartPhonetTechnology(galacticaS3) ? creep

**Exit:** (9) business(galacticaS3) ? creep

**Call:** (9) developed(galacticaS3, \_4196) ? creep

**Exit:** (9) developed(galacticaS3, sumSum) ? creep

**Call:** (9) rival(sumSum) ? creep

**Call:** (10) competitor(sumSum, appy) ? creep

**Exit:** (10) competitor(sumSum, appy) ? creep

**Exit:** (9) rival(sumSum) ? creep

**Exit:** (8) unethical(stevey) ? creep

**true .**

# Exercise 2: The Royal Family

## 1. Old Succession Rule

### Implementation

offspring(prince, charles).

offspring(princess, ann).

offspring(prince, andrew).

offspring(prince, edward).

older(charles, ann).

older(ann, andrew).

older(andrew, edward).

male(A):- offspring(prince,A).

female(A):- offspring(princess,A).

is\_older(X, Y):- older(X, Y).

is\_older(A, B):- older(A, X),is\_older(X, B).

in\_order(X, Y) :- offspring(prince, X), offspring(princess, Y).

in\_order(X, Y) :- offspring(prince, X), offspring(prince, Y), is\_older(X, Y).

in\_order(X, Y) :- offspring(princess, X), offspring(princess, Y), is\_older(X, Y).

successors(X, Y) :- insert\_sort(X, Y).

insert\_sort(X, Y) :- i\_sort(X, [], Y).

i\_sort([], Acc, Acc).

i\_sort([H|T], Acc, Y) :- insert(H, Acc, NewAcc), i\_sort(T, NewAcc, Y).

insert(X, [], [X]).

insert(X, [Y|T], [X, Y|T]) :- in\_order(X, Y).

insert(X, [Y|T], [Y|NewT]) :- not(in\_order(X, Y)), insert(X, T, NewT).

successionList(SuccessionList):- findall(Y,offspring(\_,Y),OffspringList),successors(OffspringList,SuccessionList).

### Trace

[trace] ?- successionList(SuccessionList).

**Call:** (8) successionList(\_3808) ? creep

^ **Call:** (9) findall(\_4014, offspring(\_4012, \_4014), \_4036) ? creep

**Call:** (14) offspring(\_4012, \_4014) ? creep

**Exit:** (14) offspring(prince, charles) ? creep

**Redo:** (14) offspring(\_4012, \_4014) ? creep

**Exit:** (14) offspring(princess, ann) ? creep

**Redo:** (14) offspring(\_4012, \_4014) ? creep

**Exit:** (14) offspring(prince, andrew) ? creep

**Redo:** (14) offspring(\_4012, \_4014) ? creep

**Exit:** (14) offspring(prince, edward) ? creep

^ **Call:** (14) call('$bags':'$destroy\_findall\_bag') ? creep

^ **Exit:** (14) call('$bags':'$destroy\_findall\_bag') ? creep

^ **Exit:** (9) findall(\_4014, user:offspring(\_4012, \_4014), [charles, ann, andrew, edward]) ? creep

**Call:** (9) insert\_sort([charles, ann, andrew, edward], \_3808) ? creep

**Call:** (10) insert\_sort\_helper([charles, ann, andrew, edward], [], \_3808) ? creep

**Call:** (11) insert(charles, [], \_4094) ? creep

**Exit:** (11) insert(charles, [], [charles]) ? creep

**Call:** (11) insert\_sort\_helper([ann, andrew, edward], [charles], \_3808) ? creep

**Call:** (12) insert(ann, [charles], \_4100) ? creep

**Call:** (13) in\_order(ann, charles) ? creep

**Call:** (14) offspring(prince, ann) ? creep

**Fail:** (14) offspring(prince, ann) ? creep

**Redo:** (13) in\_order(ann, charles) ? creep

**Call:** (14) offspring(prince, ann) ? creep

**Fail:** (14) offspring(prince, ann) ? creep

**Redo:** (13) in\_order(ann, charles) ? creep

**Call:** (14) offspring(princess, ann) ? creep

**Exit:** (14) offspring(princess, ann) ? creep

**Call:** (14) offspring(princess, charles) ? creep

**Fail:** (14) offspring(princess, charles) ? creep

**Fail:** (13) in\_order(ann, charles) ? creep

**Redo:** (12) insert(ann, [charles], \_4100) ? creep

^ **Call:** (13) not(in\_order(ann, charles)) ? creep

**Call:** (14) in\_order(ann, charles) ? creep

**Call:** (15) offspring(prince, ann) ? creep

**Fail:** (15) offspring(prince, ann) ? creep

**Redo:** (14) in\_order(ann, charles) ? creep

**Call:** (15) offspring(prince, ann) ? creep

**Fail:** (15) offspring(prince, ann) ? creep

**Redo:** (14) in\_order(ann, charles) ? creep

**Call:** (15) offspring(princess, ann) ? creep

**Exit:** (15) offspring(princess, ann) ? creep

**Call:** (15) offspring(princess, charles) ? creep

**Fail:** (15) offspring(princess, charles) ? creep

**Fail:** (14) in\_order(ann, charles) ? creep

^ **Exit:** (13) not(user:in\_order(ann, charles)) ? creep

**Call:** (13) insert(ann, [], \_4084) ? creep

**Exit:** (13) insert(ann, [], [ann]) ? creep

**Exit:** (12) insert(ann, [charles], [charles, ann]) ? creep

**Call:** (12) insert\_sort\_helper([andrew, edward], [charles, ann], \_3808) ? creep

**Call:** (13) insert(andrew, [charles, ann], \_4124) ? creep

**Call:** (14) in\_order(andrew, charles) ? creep

**Call:** (15) offspring(prince, andrew) ? creep

**Exit:** (15) offspring(prince, andrew) ? creep

**Call:** (15) offspring(princess, charles) ? creep

**Fail:** (15) offspring(princess, charles) ? creep

**Redo:** (15) offspring(prince, andrew) ? creep

**Fail:** (15) offspring(prince, andrew) ? creep

**Redo:** (14) in\_order(andrew, charles) ? creep

**Call:** (15) offspring(prince, andrew) ? creep

**Exit:** (15) offspring(prince, andrew) ? creep

**Call:** (15) offspring(prince, charles) ? creep

**Exit:** (15) offspring(prince, charles) ? creep

**Call:** (15) is\_older(andrew, charles) ? creep

**Call:** (16) older(andrew, charles) ? creep

**Fail:** (16) older(andrew, charles) ? creep

**Redo:** (15) is\_older(andrew, charles) ? creep

**Call:** (16) older(andrew, \_4134) ? creep

**Exit:** (16) older(andrew, edward) ? creep

**Call:** (16) is\_older(edward, charles) ? creep

**Call:** (17) older(edward, charles) ? creep

**Fail:** (17) older(edward, charles) ? creep

**Redo:** (16) is\_older(edward, charles) ? creep

**Call:** (17) older(edward, \_4134) ? creep

**Fail:** (17) older(edward, \_4134) ? creep

**Fail:** (16) is\_older(edward, charles) ? creep

**Fail:** (15) is\_older(andrew, charles) ? creep

**Redo:** (15) offspring(prince, charles) ? creep

**Fail:** (15) offspring(prince, charles) ? creep

**Redo:** (15) offspring(prince, andrew) ? creep

**Fail:** (15) offspring(prince, andrew) ? creep

**Redo:** (14) in\_order(andrew, charles) ? creep

**Call:** (15) offspring(princess, andrew) ? creep

**Fail:** (15) offspring(princess, andrew) ? creep

**Fail:** (14) in\_order(andrew, charles) ? creep

**Redo:** (13) insert(andrew, [charles, ann], \_4124) ? creep

^ **Call:** (14) not(in\_order(andrew, charles)) ? creep

**Call:** (15) in\_order(andrew, charles) ? creep

**Call:** (16) offspring(prince, andrew) ? creep

**Exit:** (16) offspring(prince, andrew) ? creep

**Call:** (16) offspring(princess, charles) ? creep

**Fail:** (16) offspring(princess, charles) ? creep

**Redo:** (16) offspring(prince, andrew) ? creep

**Fail:** (16) offspring(prince, andrew) ? creep

**Redo:** (15) in\_order(andrew, charles) ? creep

**Call:** (16) offspring(prince, andrew) ? creep

**Exit:** (16) offspring(prince, andrew) ? creep

**Call:** (16) offspring(prince, charles) ? creep

**Exit:** (16) offspring(prince, charles) ? creep

**Call:** (16) is\_older(andrew, charles) ? creep

**Call:** (17) older(andrew, charles) ? creep

**Fail:** (17) older(andrew, charles) ? creep

**Redo:** (16) is\_older(andrew, charles) ? creep

**Call:** (17) older(andrew, \_4140) ? creep

**Exit:** (17) older(andrew, edward) ? creep

**Call:** (17) is\_older(edward, charles) ? creep

**Call:** (18) older(edward, charles) ? creep

**Fail:** (18) older(edward, charles) ? creep

**Redo:** (17) is\_older(edward, charles) ? creep

**Call:** (18) older(edward, \_4140) ? creep

**Fail:** (18) older(edward, \_4140) ? creep

**Fail:** (17) is\_older(edward, charles) ? creep

**Fail:** (16) is\_older(andrew, charles) ? creep

**Redo:** (16) offspring(prince, charles) ? creep

**Fail:** (16) offspring(prince, charles) ? creep

**Redo:** (16) offspring(prince, andrew) ? creep

**Fail:** (16) offspring(prince, andrew) ? creep

**Redo:** (15) in\_order(andrew, charles) ? creep

**Call:** (16) offspring(princess, andrew) ? creep

**Fail:** (16) offspring(princess, andrew) ? creep

**Fail:** (15) in\_order(andrew, charles) ? creep

^ **Exit:** (14) not(user:in\_order(andrew, charles)) ? creep

**Call:** (14) insert(andrew, [ann], \_4108) ? creep

**Call:** (15) in\_order(andrew, ann) ? creep

**Call:** (16) offspring(prince, andrew) ? creep

**Exit:** (16) offspring(prince, andrew) ? creep

**Call:** (16) offspring(princess, ann) ? creep

**Exit:** (16) offspring(princess, ann) ? creep

**Exit:** (15) in\_order(andrew, ann) ? creep

**Exit:** (14) insert(andrew, [ann], [andrew, ann]) ? creep

**Exit:** (13) insert(andrew, [charles, ann], [charles, andrew, ann]) ? creep

**Call:** (13) insert\_sort\_helper([edward], [charles, andrew, ann], \_3808) ? creep

**Call:** (14) insert(edward, [charles, andrew, ann], \_4154) ? creep

**Call:** (15) in\_order(edward, charles) ? creep

**Call:** (16) offspring(prince, edward) ? creep

**Exit:** (16) offspring(prince, edward) ? creep

**Call:** (16) offspring(princess, charles) ? creep

**Fail:** (16) offspring(princess, charles) ? creep

**Redo:** (15) in\_order(edward, charles) ? creep

**Call:** (16) offspring(prince, edward) ? creep

**Exit:** (16) offspring(prince, edward) ? creep

**Call:** (16) offspring(prince, charles) ? creep

**Exit:** (16) offspring(prince, charles) ? creep

**Call:** (16) is\_older(edward, charles) ? creep

**Call:** (17) older(edward, charles) ? creep

**Fail:** (17) older(edward, charles) ? creep

**Redo:** (16) is\_older(edward, charles) ? creep

**Call:** (17) older(edward, \_4164) ? creep

**Fail:** (17) older(edward, \_4164) ? creep

**Fail:** (16) is\_older(edward, charles) ? creep

**Redo:** (16) offspring(prince, charles) ? creep

**Fail:** (16) offspring(prince, charles) ? creep

**Redo:** (15) in\_order(edward, charles) ? creep

**Call:** (16) offspring(princess, edward) ? creep

**Fail:** (16) offspring(princess, edward) ? creep

**Fail:** (15) in\_order(edward, charles) ? creep

**Redo:** (14) insert(edward, [charles, andrew, ann], \_4154) ? creep

^ **Call:** (15) not(in\_order(edward, charles)) ? creep

**Call:** (16) in\_order(edward, charles) ? creep

**Call:** (17) offspring(prince, edward) ? creep

**Exit:** (17) offspring(prince, edward) ? creep

**Call:** (17) offspring(princess, charles) ? creep

**Fail:** (17) offspring(princess, charles) ? creep

**Redo:** (16) in\_order(edward, charles) ? creep

**Call:** (17) offspring(prince, edward) ? creep

**Exit:** (17) offspring(prince, edward) ? creep

**Call:** (17) offspring(prince, charles) ? creep

**Exit:** (17) offspring(prince, charles) ? creep

**Call:** (17) is\_older(edward, charles) ? creep

**Call:** (18) older(edward, charles) ? creep

**Fail:** (18) older(edward, charles) ? creep

**Redo:** (17) is\_older(edward, charles) ? creep

**Call:** (18) older(edward, \_4170) ? creep

**Fail:** (18) older(edward, \_4170) ? creep

**Fail:** (17) is\_older(edward, charles) ? creep

**Redo:** (17) offspring(prince, charles) ? creep

**Fail:** (17) offspring(prince, charles) ? creep

**Redo:** (16) in\_order(edward, charles) ? creep

**Call:** (17) offspring(princess, edward) ? creep

**Fail:** (17) offspring(princess, edward) ? creep

**Fail:** (16) in\_order(edward, charles) ? creep

^ **Exit:** (15) not(user:in\_order(edward, charles)) ? creep

**Call:** (15) insert(edward, [andrew, ann], \_4138) ? creep

**Call:** (16) in\_order(edward, andrew) ? creep

**Call:** (17) offspring(prince, edward) ? creep

**Exit:** (17) offspring(prince, edward) ? creep

**Call:** (17) offspring(princess, andrew) ? creep

**Fail:** (17) offspring(princess, andrew) ? creep

**Redo:** (16) in\_order(edward, andrew) ? creep

**Call:** (17) offspring(prince, edward) ? creep

**Exit:** (17) offspring(prince, edward) ? creep

**Call:** (17) offspring(prince, andrew) ? creep

**Exit:** (17) offspring(prince, andrew) ? creep

**Call:** (17) is\_older(edward, andrew) ? creep

**Call:** (18) older(edward, andrew) ? creep

**Fail:** (18) older(edward, andrew) ? creep

**Redo:** (17) is\_older(edward, andrew) ? creep

**Call:** (18) older(edward, \_4182) ? creep

**Fail:** (18) older(edward, \_4182) ? creep

**Fail:** (17) is\_older(edward, andrew) ? creep

**Redo:** (17) offspring(prince, andrew) ? creep

**Fail:** (17) offspring(prince, andrew) ? creep

**Redo:** (16) in\_order(edward, andrew) ? creep

**Call:** (17) offspring(princess, edward) ? creep

**Fail:** (17) offspring(princess, edward) ? creep

**Fail:** (16) in\_order(edward, andrew) ? creep

**Redo:** (15) insert(edward, [andrew, ann], \_4138) ? creep

^ **Call:** (16) not(in\_order(edward, andrew)) ? creep

**Call:** (17) in\_order(edward, andrew) ? creep

**Call:** (18) offspring(prince, edward) ? creep

**Exit:** (18) offspring(prince, edward) ? creep

**Call:** (18) offspring(princess, andrew) ? creep

**Fail:** (18) offspring(princess, andrew) ? creep

**Redo:** (17) in\_order(edward, andrew) ? creep

**Call:** (18) offspring(prince, edward) ? creep

**Exit:** (18) offspring(prince, edward) ? creep

**Call:** (18) offspring(prince, andrew) ? creep

**Exit:** (18) offspring(prince, andrew) ? creep

**Call:** (18) is\_older(edward, andrew) ? creep

**Call:** (19) older(edward, andrew) ? creep

**Fail:** (19) older(edward, andrew) ? creep

**Redo:** (18) is\_older(edward, andrew) ? creep

**Call:** (19) older(edward, \_4188) ? creep

**Fail:** (19) older(edward, \_4188) ? creep

**Fail:** (18) is\_older(edward, andrew) ? creep

**Redo:** (18) offspring(prince, andrew) ? creep

**Fail:** (18) offspring(prince, andrew) ? creep

**Redo:** (17) in\_order(edward, andrew) ? creep

**Call:** (18) offspring(princess, edward) ? creep

**Fail:** (18) offspring(princess, edward) ? creep

**Fail:** (17) in\_order(edward, andrew) ? creep

^ **Exit:** (16) not(user:in\_order(edward, andrew)) ? creep

**Call:** (16) insert(edward, [ann], \_4156) ? creep

**Call:** (17) in\_order(edward, ann) ? creep

**Call:** (18) offspring(prince, edward) ? creep

**Exit:** (18) offspring(prince, edward) ? creep

**Call:** (18) offspring(princess, ann) ? creep

**Exit:** (18) offspring(princess, ann) ? creep

**Exit:** (17) in\_order(edward, ann) ? creep

**Exit:** (16) insert(edward, [ann], [edward, ann]) ? creep

**Exit:** (15) insert(edward, [andrew, ann], [andrew, edward, ann]) ? creep

**Exit:** (14) insert(edward, [charles, andrew, ann], [charles, andrew, edward, ann]) ? creep

**Call:** (14) insert\_sort\_helper([], [charles, andrew, edward, ann], \_3808) ? creep

**Exit:** (14) insert\_sort\_helper([], [charles, andrew, edward, ann], [charles, andrew, edward, ann]) ? creep

**Exit:** (13) insert\_sort\_helper([edward], [charles, andrew, ann], [charles, andrew, edward, ann]) ? creep

**Exit:** (12) insert\_sort\_helper([andrew, edward], [charles, ann], [charles, andrew, edward, ann]) ? creep

**Exit:** (11) insert\_sort\_helper([ann, andrew, edward], [charles], [charles, andrew, edward, ann]) ? creep

**Exit:** (10) insert\_sort\_helper([charles, ann, andrew, edward], [], [charles, andrew, edward, ann]) ? creep

**Exit:** (9) insert\_sort([charles, ann, andrew, edward], [charles, andrew, edward, ann]) ? creep

**Exit:** (8) successionList([charles, andrew, edward, ann]) ? creep

SuccessionList = [charles, andrew, edward, ann] **.**

## 2. New Succession Rule

## Implementation

offspring(prince, charles).

offspring(princess, ann).

offspring(prince, andrew).

offspring(prince, edward).

older(charles, ann).

older(ann, andrew).

older(andrew, edward).

male(A):- offspring(prince,A).

female(A):- offspring(princess,A).

is\_older(X, Y):- older(X, Y).

is\_older(A, B):- older(A, X),is\_older(X, B).

insert\_sort(X, Y) :- insert\_sort\_helper(X, [], Y).

insert\_sort\_helper([], Acc, Acc).

insert\_sort\_helper([H|T], Acc, Y) :- insert(H, Acc, NewAcc), insert\_sort\_helper(T, NewAcc, Y).

insert(X, [], [X]).

insert(X, [Y|T], [X, Y|T]) :- is\_older(X, Y).

insert(X, [Y|T], [Y|NewT]) :- not(is\_older(X, Y)), insert(X, T, NewT).

successionList(SuccessionList):-

findall(Y,offspring(\_,Y),OffspringList),insert\_sort(OffspringList,SuccessionList).

There are two rules- “is\_older/2” and “in\_order/2” to determine the order of offsprings, which is by order of birth, and old succession rule respectively. In the case of new succession rule, the order succession is only determined by the order of birth. So that, the rule “in\_order/2” could be removed. In addition, in the process of insertion sort, the comparator rule should be changed to “is\_older/2” instead of “in\_order/2”.

The rule “is\_older/2” implements the idea of transitive clauses which are used to make relations which enables “is\_older/2” transitivity property. In FOL:

## Trace and Result

?- successionList(SuccessionList).

SuccessionList = [charles, ann, andrew, edward] **.**

?- trace.

**true.**

[trace] ?- successionList(SuccessionList).

**Call:** (8) successionList(\_3808) ? creep

^ **Call:** (9) findall(\_4014, offspring(\_4012, \_4014), \_4036) ? creep

**Call:** (14) offspring(\_4012, \_4014) ? creep

**Exit:** (14) offspring(prince, charles) ? creep

**Redo:** (14) offspring(\_4012, \_4014) ? creep

**Exit:** (14) offspring(princess, ann) ? creep

**Redo:** (14) offspring(\_4012, \_4014) ? creep

**Exit:** (14) offspring(prince, andrew) ? creep

**Redo:** (14) offspring(\_4012, \_4014) ? creep

**Exit:** (14) offspring(prince, edward) ? creep

^ **Call:** (14) call('$bags':'$destroy\_findall\_bag') ? creep

^ **Exit:** (14) call('$bags':'$destroy\_findall\_bag') ? creep

^ **Exit:** (9) findall(\_4014, user:offspring(\_4012, \_4014), [charles, ann, andrew, edward]) ? creep

**Call:** (9) insert\_sort([charles, ann, andrew, edward], \_3808) ? creep

**Call:** (10) insert\_sort\_helper([charles, ann, andrew, edward], [], \_3808) ? creep

**Call:** (11) insert(charles, [], \_4094) ? creep

**Exit:** (11) insert(charles, [], [charles]) ? creep

**Call:** (11) insert\_sort\_helper([ann, andrew, edward], [charles], \_3808) ? creep

**Call:** (12) insert(ann, [charles], \_4100) ? creep

**Call:** (13) is\_older(ann, charles) ? creep

**Call:** (14) older(ann, charles) ? creep

**Fail:** (14) older(ann, charles) ? creep

**Redo:** (13) is\_older(ann, charles) ? creep

**Call:** (14) older(ann, \_4110) ? creep

**Exit:** (14) older(ann, andrew) ? creep

**Call:** (14) is\_older(andrew, charles) ? creep

**Call:** (15) older(andrew, charles) ? creep

**Fail:** (15) older(andrew, charles) ? creep

**Redo:** (14) is\_older(andrew, charles) ? creep

**Call:** (15) older(andrew, \_4110) ? creep

**Exit:** (15) older(andrew, edward) ? creep

**Call:** (15) is\_older(edward, charles) ? creep

**Call:** (16) older(edward, charles) ? creep

**Fail:** (16) older(edward, charles) ? creep

**Redo:** (15) is\_older(edward, charles) ? creep

**Call:** (16) older(edward, \_4110) ? creep

**Fail:** (16) older(edward, \_4110) ? creep

**Fail:** (15) is\_older(edward, charles) ? creep

**Fail:** (14) is\_older(andrew, charles) ? creep

**Fail:** (13) is\_older(ann, charles) ? creep

**Redo:** (12) insert(ann, [charles], \_4100) ? creep

^ **Call:** (13) not(is\_older(ann, charles)) ? creep

**Call:** (14) is\_older(ann, charles) ? creep

**Call:** (15) older(ann, charles) ? creep

**Fail:** (15) older(ann, charles) ? creep

**Redo:** (14) is\_older(ann, charles) ? creep

**Call:** (15) older(ann, \_4116) ? creep

**Exit:** (15) older(ann, andrew) ? creep

**Call:** (15) is\_older(andrew, charles) ? creep

**Call:** (16) older(andrew, charles) ? creep

**Fail:** (16) older(andrew, charles) ? creep

**Redo:** (15) is\_older(andrew, charles) ? creep

**Call:** (16) older(andrew, \_4116) ? creep

**Exit:** (16) older(andrew, edward) ? creep

**Call:** (16) is\_older(edward, charles) ? creep

**Call:** (17) older(edward, charles) ? creep

**Fail:** (17) older(edward, charles) ? creep

**Redo:** (16) is\_older(edward, charles) ? creep

**Call:** (17) older(edward, \_4116) ? creep

**Fail:** (17) older(edward, \_4116) ? creep

**Fail:** (16) is\_older(edward, charles) ? creep

**Fail:** (15) is\_older(andrew, charles) ? creep

**Fail:** (14) is\_older(ann, charles) ? creep

^ **Exit:** (13) not(user:is\_older(ann, charles)) ? creep

**Call:** (13) insert(ann, [], \_4084) ? creep

**Exit:** (13) insert(ann, [], [ann]) ? creep

**Exit:** (12) insert(ann, [charles], [charles, ann]) ? creep

**Call:** (12) insert\_sort\_helper([andrew, edward], [charles, ann], \_3808) ? creep

**Call:** (13) insert(andrew, [charles, ann], \_4124) ? creep

**Call:** (14) is\_older(andrew, charles) ? creep

**Call:** (15) older(andrew, charles) ? creep

**Fail:** (15) older(andrew, charles) ? creep

**Redo:** (14) is\_older(andrew, charles) ? creep

**Call:** (15) older(andrew, \_4134) ? creep

**Exit:** (15) older(andrew, edward) ? creep

**Call:** (15) is\_older(edward, charles) ? creep

**Call:** (16) older(edward, charles) ? creep

**Fail:** (16) older(edward, charles) ? creep

**Redo:** (15) is\_older(edward, charles) ? creep

**Call:** (16) older(edward, \_4134) ? creep

**Fail:** (16) older(edward, \_4134) ? creep

**Fail:** (15) is\_older(edward, charles) ? creep

**Fail:** (14) is\_older(andrew, charles) ? creep

**Redo:** (13) insert(andrew, [charles, ann], \_4124) ? creep

^ **Call:** (14) not(is\_older(andrew, charles)) ? creep

**Call:** (15) is\_older(andrew, charles) ? creep

**Call:** (16) older(andrew, charles) ? creep

**Fail:** (16) older(andrew, charles) ? creep

**Redo:** (15) is\_older(andrew, charles) ? creep

**Call:** (16) older(andrew, \_4140) ? creep

**Exit:** (16) older(andrew, edward) ? creep

**Call:** (16) is\_older(edward, charles) ? creep

**Call:** (17) older(edward, charles) ? creep

**Fail:** (17) older(edward, charles) ? creep

**Redo:** (16) is\_older(edward, charles) ? creep

**Call:** (17) older(edward, \_4140) ? creep

**Fail:** (17) older(edward, \_4140) ? creep

**Fail:** (16) is\_older(edward, charles) ? creep

**Fail:** (15) is\_older(andrew, charles) ? creep

^ **Exit:** (14) not(user:is\_older(andrew, charles)) ? creep

**Call:** (14) insert(andrew, [ann], \_4108) ? creep

**Call:** (15) is\_older(andrew, ann) ? creep

**Call:** (16) older(andrew, ann) ? creep

**Fail:** (16) older(andrew, ann) ? creep

**Redo:** (15) is\_older(andrew, ann) ? creep

**Call:** (16) older(andrew, \_4152) ? creep

**Exit:** (16) older(andrew, edward) ? creep

**Call:** (16) is\_older(edward, ann) ? creep

**Call:** (17) older(edward, ann) ? creep

**Fail:** (17) older(edward, ann) ? creep

**Redo:** (16) is\_older(edward, ann) ? creep

**Call:** (17) older(edward, \_4152) ? creep

**Fail:** (17) older(edward, \_4152) ? creep

**Fail:** (16) is\_older(edward, ann) ? creep

**Fail:** (15) is\_older(andrew, ann) ? creep

**Redo:** (14) insert(andrew, [ann], \_4108) ? creep

^ **Call:** (15) not(is\_older(andrew, ann)) ? creep

**Call:** (16) is\_older(andrew, ann) ? creep

**Call:** (17) older(andrew, ann) ? creep

**Fail:** (17) older(andrew, ann) ? creep

**Redo:** (16) is\_older(andrew, ann) ? creep

**Call:** (17) older(andrew, \_4158) ? creep

**Exit:** (17) older(andrew, edward) ? creep

**Call:** (17) is\_older(edward, ann) ? creep

**Call:** (18) older(edward, ann) ? creep

**Fail:** (18) older(edward, ann) ? creep

**Redo:** (17) is\_older(edward, ann) ? creep

**Call:** (18) older(edward, \_4158) ? creep

**Fail:** (18) older(edward, \_4158) ? creep

**Fail:** (17) is\_older(edward, ann) ? creep

**Fail:** (16) is\_older(andrew, ann) ? creep

^ **Exit:** (15) not(user:is\_older(andrew, ann)) ? creep

**Call:** (15) insert(andrew, [], \_4126) ? creep

**Exit:** (15) insert(andrew, [], [andrew]) ? creep

**Exit:** (14) insert(andrew, [ann], [ann, andrew]) ? creep

**Exit:** (13) insert(andrew, [charles, ann], [charles, ann, andrew]) ? creep

**Call:** (13) insert\_sort\_helper([edward], [charles, ann, andrew], \_3808) ? creep

**Call:** (14) insert(edward, [charles, ann, andrew], \_4166) ? creep

**Call:** (15) is\_older(edward, charles) ? creep

**Call:** (16) older(edward, charles) ? creep

**Fail:** (16) older(edward, charles) ? creep

**Redo:** (15) is\_older(edward, charles) ? creep

**Call:** (16) older(edward, \_4176) ? creep

**Fail:** (16) older(edward, \_4176) ? creep

**Fail:** (15) is\_older(edward, charles) ? creep

**Redo:** (14) insert(edward, [charles, ann, andrew], \_4166) ? creep

^ **Call:** (15) not(is\_older(edward, charles)) ? creep

**Call:** (16) is\_older(edward, charles) ? creep

**Call:** (17) older(edward, charles) ? creep

**Fail:** (17) older(edward, charles) ? creep

**Redo:** (16) is\_older(edward, charles) ? creep

**Call:** (17) older(edward, \_4182) ? creep

**Fail:** (17) older(edward, \_4182) ? creep

**Fail:** (16) is\_older(edward, charles) ? creep

^ **Exit:** (15) not(user:is\_older(edward, charles)) ? creep

**Call:** (15) insert(edward, [ann, andrew], \_4150) ? creep

**Call:** (16) is\_older(edward, ann) ? creep

**Call:** (17) older(edward, ann) ? creep

**Fail:** (17) older(edward, ann) ? creep

**Redo:** (16) is\_older(edward, ann) ? creep

**Call:** (17) older(edward, \_4194) ? creep

**Fail:** (17) older(edward, \_4194) ? creep

**Fail:** (16) is\_older(edward, ann) ? creep

**Redo:** (15) insert(edward, [ann, andrew], \_4150) ? creep

^ **Call:** (16) not(is\_older(edward, ann)) ? creep

**Call:** (17) is\_older(edward, ann) ? creep

**Call:** (18) older(edward, ann) ? creep

**Fail:** (18) older(edward, ann) ? creep

**Redo:** (17) is\_older(edward, ann) ? creep

**Call:** (18) older(edward, \_4200) ? creep

**Fail:** (18) older(edward, \_4200) ? creep

**Fail:** (17) is\_older(edward, ann) ? creep

^ **Exit:** (16) not(user:is\_older(edward, ann)) ? creep

**Call:** (16) insert(edward, [andrew], \_4168) ? creep

**Call:** (17) is\_older(edward, andrew) ? creep

**Call:** (18) older(edward, andrew) ? creep

**Fail:** (18) older(edward, andrew) ? creep

**Redo:** (17) is\_older(edward, andrew) ? creep

**Call:** (18) older(edward, \_4212) ? creep

**Fail:** (18) older(edward, \_4212) ? creep

**Fail:** (17) is\_older(edward, andrew) ? creep

**Redo:** (16) insert(edward, [andrew], \_4168) ? creep

^ **Call:** (17) not(is\_older(edward, andrew)) ? creep

**Call:** (18) is\_older(edward, andrew) ? creep

**Call:** (19) older(edward, andrew) ? creep

**Fail:** (19) older(edward, andrew) ? creep

**Redo:** (18) is\_older(edward, andrew) ? creep

**Call:** (19) older(edward, \_4218) ? creep

**Fail:** (19) older(edward, \_4218) ? creep

**Fail:** (18) is\_older(edward, andrew) ? creep

^ **Exit:** (17) not(user:is\_older(edward, andrew)) ? creep

**Call:** (17) insert(edward, [], \_4186) ? creep

**Exit:** (17) insert(edward, [], [edward]) ? creep

**Exit:** (16) insert(edward, [andrew], [andrew, edward]) ? creep

**Exit:** (15) insert(edward, [ann, andrew], [ann, andrew, edward]) ? creep

**Exit:** (14) insert(edward, [charles, ann, andrew], [charles, ann, andrew, edward]) ? creep

**Call:** (14) insert\_sort\_helper([], [charles, ann, andrew, edward], \_3808) ? creep

**Exit:** (14) insert\_sort\_helper([], [charles, ann, andrew, edward], [charles, ann, andrew, edward]) ? creep

**Exit:** (13) insert\_sort\_helper([edward], [charles, ann, andrew], [charles, ann, andrew, edward]) ? creep

**Exit:** (12) insert\_sort\_helper([andrew, edward], [charles, ann], [charles, ann, andrew, edward]) ? creep

**Exit:** (11) insert\_sort\_helper([ann, andrew, edward], [charles], [charles, ann, andrew, edward]) ? creep

**Exit:** (10) insert\_sort\_helper([charles, ann, andrew, edward], [], [charles, ann, andrew, edward]) ? creep

**Exit:** (9) insert\_sort([charles, ann, andrew, edward], [charles, ann, andrew, edward]) ? creep

**Exit:** (8) successionList([charles, ann, andrew, edward]) ? creep

SuccessionList = [charles, ann, andrew, edward] **.**