APPENDIX 2

domains

file = file
symbollist = symbol*
integerlist = integer*
condition = condition(integer,symbol)
conditionlist = condition*
database = general

over(integer)
notover(integer)
facts(integer,symbol,symbol)
routen(integerlist)
path(integerlist)

predicates

action(integer)
showmenu
ask
open
click
filter
infer(integer,string)
checkfilter
answers(integer,symbol,symbollist)
member(symbol,symbollist)
checkanswers(integer,symbol)
questions(integer,string)
noninteger,conditionlist)
checkquestion(integer)
questionlist(integerlist)
process(integerlist)
shuttle(integerlist)
display(integerlist)
trace(integerlist)
askquestion(integer)
showanswers

showpath
initialize

clauses

questionlist([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]),
showmenu: shiftwindow(1).cursor(0,0),
questionist(L). process(L). ask.

process([]).
process([H|T]) :- askquestion(H), process(T).

askquestion(N) :- questions(N,O), checkquestion(N), write(N,".*",O), nl

askquestion().

checkquestion(N) :- qcon(N,[|]).
checkquestion(N) :- qcon(N,[con(A,B)]). facts(A,B).

questions(1. "The size of your hat."
questions(2. "The state of the hat's brims ").
questions(3. "The state of your hat ").
questions(4. "The color of the silk band ").
questions(5. "The state of your hat's elastic ").
questions(6. "The elastic has been replaced ").
questions(7. "Clean cut hair is present in your hat ").
questions(8. "Grizzled hair is present in your hat ").
questions(9. "Dust is present in your hat ").
questions(10. "Moisture is present in your hat ").
questions(11. "Tallow stains are present in your hat ").
questions(12. "The hair has adhesive ").
questions(13. "The hair has odour ").
questions(14. "The type of odour present ").
questions(15. "The type of dust present ").
questions(16. "The stains are covered with ink ").
questions(17. "Your hat is brushed ").

qcon(1,[|]).
qcon(2,[|]).
qcon(3,[|]).
qcon(4,[|]).
qcon(5,[|]).
qcon(6,[con(5,broken)]). qcon(7,[|]).
qcon(8,[con(7,no)]). qcon(9,[|]).
qcon(10,[|]).
qcon(11,[|]).
qcon(12,[con(7,yes)]). qcon(13,[con(12,yes)]). qcon(14,[con(13,yes)]).
qcon(15,[con(9,yes)]).
qcon(16,[|]).
qcon(17,[|]).
filter: - retract(route(L)), shuffle(L).
filter: - !, assertz(path([0])).

shuffle(L):- route(L), retract(route(L)), shuffle(L).
shuffle(L):- assertz(path(L)), !, filter.

ask: - shiftwindow(2), clearwindow,
    write("Your Choice is (0: INFER: -1: TRACE: -2: QUIT): ").
readin(Choice), Choice > -3. Choice < 18, action(Choice)
ask: - !

showpath:- path(X), ddisplay(X), fail.
showpath:- !.

ddisplay([0]):- retractall(_).
ddisplay(D):- nl, open, tracedown(D), close

tracedown([]) :- nl
tracedown([HIT]):- H>17, infer(H,S),
    write("I infer that " S " <Because.>"). nl
    tracedown(T), !.
tracedown([HIT]):- H>0, H<18, facts(H,..yes). !,
    questions(H,S), upper_lower(S,SS),
    write("You agreed that *.SS.* <and.>"). nl
    tracedown(T)
tracedown([HIT]):- H>0, H<18, facts(H,..no), !,
    questions(H,S), upper_lower(S,SS),
    write("You refuted that *.SS.* <and>"). nl.
    tracedown(T)
tracedown([HIT]):- H>0, H<18, facts(H,..A),
    questions(H,S), upper_lower(S,SS),
    write("You told me that *.SS.* is *.A.* <and.>"). nl.
    tracedown(T)
tracedown([HIT]):- H<0, infer(H,S),
    write("It is my assumption that *.S.* <and.>"). nl
    tracedown(T)

action(-2):- !

action(0):- shiftwindow(3), cleanwindow, checkinfer

action(-1):- shiftwindow(4), cleanwindow.
filter, I, showpath, file_str("trace.dat",Trace).
display(Trace), clearwindow.

action(C):- clearwindow, checkquestion(C), answers(C,_,L).
    write(C,"",",",L,"",": ").
    readin(An). upper_lower(An,A),
    checkanswer(C,A), answers(C,B,_,_), reракти(facts(_,_,_)).
    assertz(facts(C,B,A), showanswers, showmenu).

showanswers:-- shiftwindow(5), clearwindow, facts(_,_,B), A<>0.
    write(A,"",",",B), nl. fail.
    showanswers.

checkanswer(C,A):- answers(C,_,List), member(A,List).
checkanswer(C,?):- beep, action(C).

checkinfer:-- write("I infer that..."). nl.
    write("----------"). nl. nl.
    infer(N,Inference), getbacktrack(G).
    N>0, write(Inference), nl.
    cutbacktrack(G), fail.

checkinfer:-- nl. nl. nl. write("No (more) inference(s) can be made for this (these) fact(s) "). nl.
    write("Press any key to continue ").
    readchar( ), showanswers, shiftwindow(1). ask

infer(-1,"you own a hat ").
infer(-2,"your hat had no loop ").
infer(-3,"your hat had no disk ").
infer(-4,"you are married ").
infer(-5,"your hat had sick band ").
infer(-6,"your hat had excellent lining ").
infer(18,"your hat is no good now "). :-
    facts(A.elastic,broken).
    assertz(route((18,A))).
    facts(B.color,discolored).
    assertz(route(18,B))).
infer(19,"your hat was worn outside "). :-
    facts(A.dust,_,yes).
    facts(B.dust_type,_,gritty).
    assertz(route(19,A,B))).
infer(20,"your hat was worn inside the house "). :-
    facts(A.dust,_,yes).
    facts(B.dust_type,_,fluffy brown).
    assertz(route(20,A,B))).
infer(21,"you bought your hat long ago "). :-
    facts(A.state,old).
    facts(B.brims,curled).
    assertz(route(21,A,B))).
infer(40, "you use tallow.") :- facts(A, tallow_stains, yes).
assertz(facts([39,A,B]).)

infer(41, "It is improbable that you have gas laid on in your house ") :- infer(A, "you use tallow.").
assertz(facts([41,A]).)

infer(42, "your head capacity is small.") :- facts(A, size, small).
assertz(facts([42,A]).)

infer(43, "your intellect is not very high.") :- infer(A, "your head capacity is small.").
assertz(facts([43,A]).)

infer(44, "your hat is probably new or has not been used much.") :- facts(A, brims, straight).
assertz(facts([44,A]).)

infer(44, "your hat is probably new or has not been used much.") :- facts(A, color, colored).
assertz(facts([44,A]).)

infer(44, "your hat is probably new or has not been used much.") :- facts(A, elastic, new).
assertz(facts([44,A]).)

infer(45, "your wife is far from loving ") :- infer(A, "you are named ").

answers(1, size, [big, small])
answers(2, brims, [curled, straight])
answers(3, state, [new, old])
answers(4, color, [colored, discolored])
answers(5, elastic, [new, broken])
answers(6, replaced, [yes, no])
answers(7, hair, [yes, no])
answers(8, grizzled, [yes, no])
answers(9, dust, [yes, no])
answers(10, moisture, [yes, no])
answers(11, tallow_stains, [yes, no])
answers(12, adhesive, [yes, no])
answers(13, odour, [yes, no])
answers(14, odour_type, [limestone, unknown])
answers(15, dust_type, [fluffy, brown, gritty])
answers(16, inked_stains, [yes, no])
answers(17, brushed, [yes, no])

member(H, [HI, J])
member(H, [IT]) :- member(H, T).

initialize :- assertz(facts(0, size, nil)).
assertz(facts(0, brims, nil)).
assertz(facts(0, state, nil)).
assertz(facts(0, color, nil)).
assertz(facts(0, elastic, nil)).
assertz(facts(0, replaced, nil)).
assertz(facts(0, hair, nil)).

276
assertz(facts(0, grizzled, nil)).
assertz(facts(0, dust, nil)).
assertz(facts(0, moisture, nil)).
assertz(facts(0, tallow_stains, nil)).
assertz(facts(0, adhesive, nil)).
assertz(facts(0, odour, nil)).
assertz(facts(0, odour_type, nil)).
assertz(facts(0, dust_type, nil)).
assertz(facts(0, inked_stains, nil)).
assertz(facts(0, brushed, nil)).

open! :- openappend(tfile, "trace.dat"), writedevice(tfile).
closet :- closefile(tfile), writedevice(screen).

goal

initialize, openwrite(tfile, "trace.dat"), closet,
makewindow(4.78, 47.0, "The Trace of my Inferences", 0.0, 25.80),
makewindow(3.30, 78.0, "My Inference", 0.0, 25.80),
makewindow(1.30, 36.0, "Please give me facts about your hat", 0.0, 22.55),
makewindow(2.47, 36.0, "Enter Your Choice Here", 22.0, 3.80),
makewindow(5.36, 78.0, "Your Answers", 1.55, 21.25),
showmenu