```
% FILE: crypto.pro of crypto_hps
% TYPE: Prolog source
% LINE: The heuristic crypto problem solver
% DATE: 2019

% ------------------------------------------------------------
% LOAD FILES

:- consult('crypto.pro').
:- consult('combinatorial_sets.pro').
:- consult('lp.pro').

% ------------------------------------------------------------
% SOME SIMPLE "LOWER LEVEL" RECOGNITION UTILITIES

same(A,A).

adjacent(A,B) :- A is B+1.
adjacent(A,B) :- A is B-1.
% ------------------------------------------------------------
% SOME SIMPLE "LOWER LEVEL" ACTION UTILITIES

multiply([A,B],ex(A,*,B)).
multiply([H|T],ex(H,*,R)) :- multiply(T,R).

divide([A,B],ex(A,/,B)).
divide([H|T],ex(H,/,R)) :- divide(T,R).

add([A,B],ex(A,+,B)).
add([H|T],ex(H,+,R)) :- add(T,R).

subtract([A,B],ex(A,-,B)).
subtract([H|T],ex(H,-,R)) :- subtract(T,R).

% ------------------------------------------------------------
% SOME "HIGHER LEVEL" CRYPTO PROBLEM SOLVING PREDICATES

goal_is_zero :-
  value_of(crypto_problem,problem(_,goal(0))).

goal_is_one :-
  value_of(crypto_problem,problem(_,goal(1))).
```

```
goal_is_seven :-
  value_of(crypto_problem,problem(_,goal(7))).

goal_is_nonzero :-
  value_of(crypto_problem,problem(_,goal(G))),
  G \= 0.

zero_in_numbers :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),_)),
  contains([N1,N2,N3,N4,N5],0).

goal_in_numbers :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),goal(G))),
  contains([N1,N2,N3,N4,N5],G).

one_in_numbers :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),_)),
  contains([N1,N2,N3,N4,N5],1).

two_in_numbers :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),_)),
  contains([N1,N2,N3,N4,N5],2).

three_in_numbers :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),_)),
  contains([N1,N2,N3,N4,N5],3).


goal_M1_in_numbers :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),goal(G))),
  GM1 is G - 1,
  contains([N1,N2,N3,N4,N5],GM1).

goal_P4_in_numbers :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),goal(G))),
  GP4 is G + 4,
  contains([N1,N2,N3,N4,N5],GP4).

goal_X2_in_numbers :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),goal(G))),
  GX2 is G * 2,
  contains([N1,N2,N3,N4,N5],GX2).
```

```
pair_of_two :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),_)),
  comb(set(N1,N2,N3,N4,N5),comb(A,B),_),
  same(A,B),
  A = 2.

pair_in_numbers :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),_)),
  comb(set(N1,N2,N3,N4,N5),comb(A,B),_),
  same(A,B).

pair_in_numbers(value(V),rest(C,D,E)) :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),_)),
  comb(set(N1,N2,N3,N4,N5),comb(A,B),extras(C,D,E)),
  same(A,B), V=A.

numbers(N1,N2,N3,N4,N5) :-
  value_of(crypto_problem,problem(numbers(N1,N2,N3,N4,N5),_)).

goal(G) :-
  value_of(crypto_problem,problem(_,goal(G))).

numbers_other_than(these(A1,A2),those(B1,B2,B3)) :-
  numbers(N1,N2,N3,N4,N5),
  remove_elements([A1,A2],[N1,N2,N3,N4,N5],[B1,B2,B3]).

numbers_other_than(these(A1,A2,A3),those(B1,B2)) :-
  numbers(N1,N2,N3,N4,N5),
  remove_elements([A1,A2,A3],[N1,N2,N3,N4,N5],[B1,B2]).

numbers_other_than(these(A1,A2,A3,A4),those(B1)) :-
  numbers(N1,N2,N3,N4,N5),
  remove_elements([A1,A2,A3,A4],[N1,N2,N3,N4,N5],[B1]).

% ------------------------------------------------------------
% MORE LIST PROCESSORS

remove_elements([],List,List).
remove_elements([H|T],List,Reduced_List) :-
  member(H,List),
  remove(H,List,Partial_List),
  remove_elements(T,Partial_List,Reduced_List).
remove_elements([_|T],List,Reduced_List) :-
```

```
   remove_elements(T,List,Reduced_List).

% ------------------------------------------------------------
% CRYPTO HEURISTIC 1

situation1 :-
   goal_is_zero,
   zero_in_numbers.

action1 :-
   numbers(N1,N2,N3,N4,N5),
   multiply([N1,N2,N3,N4,N5],Expression),
   add_solution_to_KB(Expression).

% ------------------------------------------------------------
% CRYPTO HEURISTIC 2

situation2 :-
   goal_is_zero,
   pair_in_numbers.

action2 :-
   pair_in_numbers(value(V),rest(C,D,E)),
   subtract([V,V],X1),
   multiply([0,C,D,E],X2),
   substitute(X1,0,X2,Expression),
   add_solution_to_KB(Expression).

% ------------------------------------------------------------
% CRYPTO HEURISTIC 3

situation3 :-
   goal_is_nonzero,
   goal_in_numbers,
   zero_in_numbers.

action3 :-
   goal(G),
   numbers_other_than(these(0,G),those(A,B,C)),
   multiply([0,A,B,C],Zero_valued_expression),
   Expression = ex(G,+,Zero_valued_expression),
   add_solution_to_KB(Expression).
```

```
% ------------------------------------------------------------
% CRYPTO HEURISTIC 4

situation4 :-
   zero_in_numbers,
   pair_in_numbers,
   goal_is_one.

action4 :-
   %pair_in_numbers(value(V),rest(A,B)),
   numbers_other_than(these(V,V,0),those(A,B)),
   divide([V,V],X1),
   multiply([0,A,B],X2),
   add([X1,X2], Expression),
   add_solution_to_KB(Expression).

% ------------------------------------------------------------
% CRYPTO HEURISTIC 5
situation5 :-
   zero_in_numbers,
   one_in_numbers,
   goal_M1_in_numbers.

action5 :-
   goal(G),
   GM1 is G -1,
   numbers_other_than(these(1,GM1,0),those(A,B)),
   multiply([0,A,B],Zero_valued_expression),
   add([GM1,1], One),
   add([One,Zero_valued_expression], Expression),
   add_solution_to_KB(Expression).


% ------------------------------------------------------------
% CRYPTO HEURISTIC 6
situation6 :-
   pair_of_two,
   goal_P4_in_numbers,
   zero_in_numbers.

action6 :-
   goal(G),
   GP4 is G + 4,
```

```
   numbers_other_than(these(V,V,GP4,0),those(A)),
   add([V,V],Four),
   multiply([0,A],Zero_valued_expression),
   subtract([GP4,Four], SG),
   add([SG,Zero_valued_expression], Expression),
   add_solution_to_KB(Expression).

% ------------------------------------------------------------
% CRYPTO HEURISTIC 7
situation7 :-
   two_in_numbers,
   goal_X2_in_numbers,
   zero_in_numbers.

action7 :-
   goal(G),
   GX2 is G * 2,
   numbers_other_than(these(2,GX2,0),those(A,B)),
   multiply([0,A,B],Zero_valued_expression),
   divide([GX2,2], SG),
   add([SG,Zero_valued_expression], Expression),
   add_solution_to_KB(Expression).

% ------------------------------------------------------------
% CRYPTO HEURISTIC 8
situation8 :-
   pair_of_two,
   goal_is_seven,
   zero_in_numbers,
   three_in_numbers.


action8 :-
   numbers_other_than(these(2,2,3,0),those(A)),
   multiply([0,A],Zero_valued_expression),
   add([2,2],Four),
   add([Four,3], SG),
   add([SG,Zero_valued_expression],Expression),
   add_solution_to_KB(Expression).

% ------------------------------------------------------------
% THE RULE BASE
```

```
rule(1,situation1,action1).
rule(2,situation2,action2).
rule(3,situation3,action3).
rule(4,situation4,action4).
rule(5,situation5,action5).
rule(6,situation6,action6).
rule(7,situation7,action7).
rule(8,situation8,action8).


% ------------------------------------------------------------
% SOLVE AN INTERNALIZED PROBLEM (IT MAY HAVE BEEN RANDOMLY
% GENERATED OR SPECIFICALLY ESTABLISHED) HEURISTICALLY,
% PLACING ITS SOLUTION IN THE KB.

solve_problem_heuristically :-
rule(Number,Situation,Action),
write('considering rule '),write(Number),write(' ...'),nl,
Situation,
write('application of rule '),write(Number),
Action.
solve_problem_heuristically :-
add_solution_to_KB(none).

add_solution_to_KB(Expression) :-
undeclare(solution),
declare(solution,solution(Expression)).
add_solution_to_KB(Expression) :-
declare(solution,solution(Expression)).

% ------------------------------------------------------------
% DISPLAY THE SOLUTION -- ASSUMING THAT THE PROBLEM HAS BEEN SOLVED

display_solution :-
value_of(solution,solution(none)),
write('NO SOLUTION found with this rule base.'),nl.
display_solution :-
value_of(solution,solution(Expression)),
write(' produces solution: '), display_result(Expression),nl.
display_solution.

display_result(ex(A,O,B)) :-
number(A),number(B),
```

```
write('( '),write(A),write(' '),write(O),write(' '),write(B),write(' )').
display_result(ex(A,O,B)) :-
number(A), B = ex(A1,O1,B1),
write('( '),write(A),write(' '),write(O),write(' '),
display_result(ex(A1,O1,B1)),write(' )').
display_result(ex(A,O,B)) :-
number(B), A = ex(A1,O1,B1),
write('( '),display_result(ex(A1,O1,B1)),write(' '),write(O),write(' '),
write(B),write(' )').
display_result(ex(A,O,B)) :-
A = ex(A1,O1,B1),B = ex(A2,O2,B2),
write('( '),display_result(ex(A1,O1,B1)),write(' '),write(O),write(' '),
display_result(ex(A2,O2,B2)),write(' )').

% ------------------------------------------------------------
% SUBSTITUTION CODE

substitute( New, Old, ex( Old, O, Z ), ex( New, O, Z ) ).
substitute( New, Old, ex( X, O, Old ), ex( X, O, New ) ).
substitute( New, Old, ex( X, O, Z ), ex( Y, O, Z ) ) :-
substitute( New, Old, X, Y ).
substitute( New, Old, ex( X, O, Y ), ex( X, O, Z ) ) :-
substitute( New, Old, Y, Z ).

% ------------------------------------------------------------
% CRYPTO PROBLEM SOLVER -- SOLVE A RANDOM PROBLEM

solve_one :-
generate_random_crypto_problem,
display_problem,
solve_problem_heuristically,
display_solution.

% ------------------------------------------------------------
% CRYPTO PROBLEM SOLVER -- SOLVE A SPECIFIC PROBLEM

solve(problem(numbers(N1,N2,N3,N4,N5),goal(G))) :-
add_crypto_problem_to_KB(N1,N2,N3,N4,N5,G), % froom crypto.pro of crypto_rpg
display_problem,
solve_problem_heuristically,
display_solution.

% ------------------------------------------------------------
```

```
% RANDOM CRYPTO PROBLEM SOLVING

demo(1) :-
solve_one.
demo(N) :-
demo(1),
M is N-1,
demo(M).
```