Project 2 Report

We have only one fluent called matrix_state:

- Syntax: matrix_state(NeoLocation, HostagesState, S).
- Semantics: NeoLocation is a list containing the X and Y location of Neo in situation S. HostagesState is a list having a number of elements equal to the number of hostages in KB, where each element of HostagesState can have one of three values; (n,c,d), 'n' means not carried or dropped, 'c' means carried, and 'd' means dropped. S is either s0 or result(A, Sn) which is the situation resulting from applying action A in situation Sn.

We have one helper predicate to do iterative deepening search called goalHelper:

- Syntax: goalHelper(NeoLocation, HostagesState, D, S).
- Semantics: NeoLocation, HostagesState, and S are as defined above. D is the depth limit. Initially, goalHelper is called inside goal(S) with D equal to zero, and we increment it by 1 in goalHelper to find all possible solutions.

Concerning the implementation of the successor-state axioms, we have only one successor-state axiom written in multiple rules with different arguments and conditions:

- For the drop action to be valid, NeoLocation must be the same as the telephone booth location, and HostagesState must contain at least one 'c' element in the previous situation.

- For the carry action to be valid, NeoLocation must be the same as one hostage's location, and the number of 'c' elements in HostagesState to be less than the capacity value in the previous situation.
- For the up action to be valid, the X part of NeoLocation must be greater than zero in the previous situation.
- For the down action to be valid, the X part of NeoLocation must be less than the grid height – 1 in the previous situation.
- For the left action to be valid, the Y part of NeoLocation must be greater than zero in the previous situation.
- For the right action to be valid, the Y part of NeoLocation must be less than the grid width – 1 in the previous situation.

We have two rules for the goal(S) predicate, one for generating a plan, while the other, when given a plan, checks if it is valid or not:

- First rule: S must be a variable, NeoLocation is instantiated to be in the same location as the telephone booth, and HostagesState is instantiated to have a number of elements equal to the number of hostages in KB and all have the value 'd'. We call a helper predicate inside this rule to do iterative deepening search.
- Second rule: S must not be a variable, NeoLocation is instantiated to be in the same location as the telephone booth, and HostagesState is instantiated to have a number of elements equal to the number of hostages in KB and all have the value 'd'. We call matrix_state with the instantiated NeoLocation, HostagesState, and S.

First running example:

- KB: grid(3,3). neo_loc(1,0). hostages_loc([[2,2]]). booth(0,2). capacity(1).
- goal(s) Output:
  S = result(drop, result(up, result(up, result(carry, result(down, result(right, result(right, s0))))))) .

Second running example:

- KB: grid(3,3). neo_loc(2,1). hostages_loc([[1,2],[0,0]]). booth(0,1). capacity(2).
- goal(s) Output:
  S = result(drop, result(right, result(carry, result(up, result(left, result(left, result(carry, result(up, result(right, s0)))))))) .

Resources:

- https://www.swi-prolog.org/