Assignment 5 - Machine Problem 5

Matej Shanel CIS-530A Fall 2018

October 7, 2018

Problem 1. Greedy Best-First search
Implement Greedy search using Best-First-Search applied with functions for h only. Show the evolution of the emphOPEN and *CLOSED* lists. Sample output will be given in which your programs output will be compared automatically by the grader.

Solution The solution to this problem is located in folder *./problem_1/*. The executable file is named *mp5-1.py* and it uses *aig.py* for graph parsing.

Sample input file is located in the *input* folder and the sample output files are located in the *output* folder. Each of the sample output files has the number of heuristic used in its name.

Problem 2. First-Order Predicate Calculus (*FOPC*) a.k.a. First-Order Logic (*FOL*) and Constraint Satisfaction Problems (*CSP*)
Derive a predicate *N-Queens-Solution (N, QueenList)* such that *QueenList* contains a list of *N queen rank positions* on consecutive files that satisfies the constraints of N queens.

How would you incorporate algorithms such as *forward checking* and heuristics such as *MRV* or *LCV* into this solution?

Solution Before deriving the solution itself, it is important to state, that I assume, that the classic mathematical, programming and *FOL* operators work as usual. For my predicate definition I have used following:

- comparison operators (specifically $<$ and $\leq$)
- function absolute value (used as $|x|$)
- list item accessing (used as $list[x]$)
- subtraction and equality ($x - y$ and $x = y$ respectively)

Let's now derive the solution. To achieve the required effect, we must make sure, that no two queens can attack each other. However, the attacking condition is symmetrical - meaning that if queen A can attack queen B, it also means, that queen B can attack queen A.

This allows us to check every pair just once by performing the check only for all unchecked queens. From this observation the precondition of the formula is created:

$$\forall x, y \in \mathbb{N}.((0 \leq x < N - 1) \land (x < y < N))$$

Also each queen must be positioned on chessboard. This can be assured by checking following condition for every single queen:

$$0 < QueenList[x] \leq N$$

Now we need to figure out, in which cases two queens can attack each other. This can be simplified to these three cases, when those two queens:

- are in the same file
- have the same rank
- are on the same diagonal

The first case when the queens would be in the same file is solved by the problem definition, since the *QueenList* allows only *one queen per file*.

The second case will be covered if there are no two equal numbers in the *QueenList*. This can be easily checked by adding following statement:

$$\neg(QueenList[x] = QueenList[y])$$

The last condition regarding diagonals is bit more tricky. However, we can notice, that two squares in different files are on the same diagonal, if and only if the distance of the two files equals the distance of the two ranks.

The distance in this space can be easily computed as the absolute value of subtraction of two numbers. However, we know, that the file x will be also lesser than the file y and therefore we can omit the absolute value from that side of equation. The final condition then looks like this:

$$\neg(|QueenList[x] - QueenList[y]| = y - x)$$

Now the only thing remaining is to put all the parts together to result in final definition:

$$\begin{aligned}
&N - QueensSolution(N, QueenList) \Leftrightarrow \\
&\quad \forall x, y \in \mathbb{N}.(((0 \leq x < N) \land (x < y < N)) \Rightarrow \\
&\qquad ((0 < QueenList[x] \leq N)\land \\
&\qquad (\neg(QueenList[x] = QueenList[y]))\land \\
&\qquad (\neg(|QueenList[x] - QueenList[y]| = y - x))))
\end{aligned}$$

To incorporate mechanisms like *FC*, *MRV* or *LCV* into this solution, we would need to turn this problem into *CSP* one. We could accomplish this by turning files into variables and ranks into values.

After that the rules on the right side of implication in my predicate would become the constraints in between variables. At this point, we would be able to apply all mentioned mechanisms to this problem.

Problem 3. Learning *CLIPS*

Implement rules in *CLIPS* and use them with facts describing the *Wumpus World* example from § 7.5.4, p. 257-259 *R&N 3e*, to prove the sentence $P_{1,2}$ using *CLIPS* inference rather than *PL-FC-Entails*.

Solution The solution for this problem is located in *./problem_3/*. In this folder there is the *mp5_4.clp* file containing the rules and constructs. There are also two *.bat* sample input files and corresponding *.txt* output files.

My solution might seem like an overkill, because it should be able to make inference for *any given situation* in the *Wumpus World*. I have written it in this way, because the specification was *once again* ambiguous and I thought our solution is supposed to handle any situation, not just the dummy one.

The requested scenario and corresponding inferences can be seen in sample files *00*. I have also included more advanced example with example in *figure 7.3*, p. 239 *R&N 3e*, which can be found in sample files *01*.

The reader is encouraged to try out more situations to see, whether my solution is working. The documentation is provided in file *mp5_4.clp* in form of comments.