Laura Cabanová

N-queens KR project 2021-2022

I used prolog to solve N-queens puzzles. The task is: Place N queens on an NxN chess board, so that none of the queens is under attack. So, for instance lets have 8x8 chessboard and the queen is the queen is the piece in chess, which attacks all squares in the same column, which is called the file and in the same row, which is called the range and diagonals. For a purpose the colors of the chessboard do not matter so I can simply consider an NxN squares. For instance, if I will place a queen on a position in 3rd column and 4th row and then placing another queen in the same row, it will mean that the queen is under attack and since the queen are symmetric then both queens will be under attack. That’s why I considered to put into my NxN chess board the grey squares to indicate squares where no further queen can be placed. So, for the instance we can place the other queen in 8x8 chess board into 6th column and 5th row and then again, I can see where another queen can be placed (based again on where the queen can move I will put more grey squares and then I can place another queen and so on). Since each queen attacks in particular all squares in the same row, this means that I can place maximum N queens into NxN chess board, so N is really then the maximum – and this is also one possible solution in NxN board. For example for N = 0 is easy cause, simply, I will place none queen, if N = 1 is also easy but if N = 2 there is no solution. For N = 3 I found 3 solutions and also for N=4 was possible to find. I will use coordinates for this task to let know on which position the queen is. Furthermore, variable for row instead of numbers can be in 8x8 chess board Q1,…………….,Q2 so for instance having the 3rd queen in 4th row will look like Q3 = 4. So this representation also significantly reduces the space. So let´s Q\_i be an integer, denoting the row of the queen in column i. Since we are reasoning about integers, we use declarative integer arithmetic, also known as CLP(FD) or CLP(Z). In this task I used disequality of integers so then: A #\= B <-> and B are not equal. Lets write the code where n\_queens(N, Qs) <-> Qs are N safe queens, which means Qs is a list of N safe queens so we have N quens of NxN queues. We want to keep this as general as possible, so we want to describe when this holds and use the predicate to test, complete and generate solutions, so we stated the conditions that make this TRUE. So first of all the Qs must be a list of exactly the N elements, secondly queues must be the integers and each of these integers must be between 1 and N because as I mentioned each of these integers shell denoted the row of the queen in the respect column. And THIRD the queens must be safe, so safe queens shell be a predicate that it is true if and only if none of the queens is under attack. If there are no queens at all , then the queens are safe so safe\_queens holds for an empty list. Second if there is at least one queen then that queen must not attack the remaining queens and the remaining queen must have also be safe. So safe queens should be true if and only if none of the queens in the first argument is attack by the queen in the second argument. This also predicate that the queen in the second argument should not be attack by the queens in the first argument. For example if there are no queens in the first argument. Then I need to use the disequality to differ the queens in the Q and the Q0 so I need different integers, otherwise if there are the same they will be in the same row and then they would attack each other. But this is still not enough cause prolog can say that it is true if the 2x2 chess board the queens is in Q1 the other can be In Q2 – but that is for our task incorrect. We must make the predicate more specific. In particular, what is still missing we must consider also diagonal attacks (diagonal attacks mean that we can move also the queen diagonal to the left and to the right). Then we can say that Q9 and Q, D0 columns apart, do not attack each other diagonally if Q0 – Q is different than D0 and the Q0 – Q is different than negative D0. We can also use absolute value so in one step we can make either positive either negative D0. Now it is necessary to plug in this condition into right place in the previous predicate. First it is necessary to add one argument to safe\_queens(Qs, Q, 1) to track the number of columns between the queens that we are considering, e.g. the first of the remaining queen is 1 column apart from reference queen in a que, so if there are no queens remaining we do not care about this distance but if there is at least the one remaining queen then we need to put abs condition. Then the queen are safe. For example the solution would be : n\_queens(N, [2,4,1,3]) for N = 4. If I will write n\_queens(N, [2,4,C,D]) the system will tell us the solution for N = 4, C = 1, D =3.

This would be a solution for : N = 8, n\_queens(N, Qs), labeling([ff], Qs).

