Mathias Flink Brandt (mfli@itu.dk)
Simon Langhoff (siml@itu.dk)

N-Queens

Intelligent Systems Programming
Mandatory Assignment 2
April 9th, 2015

Using the provided library to build and represent a Binary Decision Diagram (BDD) we have implemented an interactive configurator for the N-Queens problem. Given a board of size *N* x *N*, our configurator allows the user to place a queen only on the board spaces that will eventually lead to a solution for the specific size configuration.

Construction of the Binary Decision Diagram

To solve the N-Queens problem using a binary decision diagram, we have created a BDD with *N* x *N* variables — one variable for each position on the chessboard. This allows us to specify if a queen can be placed at each position or not by restricting the value of the corresponding BDD variable to either `TRUE` or `FALSE`, respectively.

The rules of the N-Queens problem need to be transformed into a set of boolean expressions that can be validated by our BDD. We have defined the following set of rules:

1. No queen can be placed in a position that is immediately threatened by another, previously placed, queen.
2. There must be exactly one queen in each column. As a result, a total of *N* queens must be placed on the board.

To implement the first rule, we build four boolean expressions for each position on the board. The resulting expressions are conjugated, and applied as a rule in the BDD. When combined, these expressions will disallow placement of a queen on a position that is already being threatened by another queen. Consider the following 5 x 5 board:

```
  1 2 3 4 5
1 A B C D E
2 F G H I J
3 K L M N O
4 P Q R S T
5 U V W X Y
```

For example, based on position `G`, the rules will be defined as the following:
Max one queen per row: `(!G | !F) & (!G | !H) & (!G | !I) & (!G | !J)`
Max one queen per column: `(!G | !B) & (!G | !L) & (!G | !Q) & (!G | !V)`
Max one queen per diagonal (upwards, left-to-right): `(!G | !K) & (!G | !C)`
Max one queen per diagonal: `(!G | !A) & (!G | !M) & (!G | !S) & (!G | !Y)`

The second rule is implemented by constructing a boolean expression for each column. For example, for column 3 on the above board, the rule will be defined as the following:
`(C | H | M | R | W)`
These expressions are also conjugated, and applied as a rule in the BDD. The resulting expression will ensure that at least one queen has been placed in each column.

Restricting The Binary Decision Diagram

When a user places a queen on the board, the BDD must be restricted to reflect this placement. All the BDD variables that represent a queen's position on the board are restricted to a value of `TRUE`. Furthermore, all variables that represent a position on the board that is being threatened by any queen is restricted to a value of `FALSE`.

In addition, all solutions to the current configuration of the board are being checked to see if one or more positions are excluded in all of them. If this is the case, the specific position(s) will be restricted to a value of `FALSE`.

Lastly, the GUI is updated to prevent the user from placing a queen on an invalid position on the board. This is achieved by assigning a value of `-1` to all board positions that have been restricted to `FALSE` in the BDD. The provided library takes care of disabling these positions in the user interface.

Automatic Queen Placement

When the user has placed a queen on the board, and the BDD has been restricted accordingly, each column is examined for remaining available positions. If only a single position remains open, the algorithm automatically places a queen in that position, since this must inevitably be done at some point in the future. Since placing a queen will trigger this check, in some cases the algorithm will recursively complete the entire problem on its own.