INTELLIGENT SYSTEMS PROGRAMMING

Configuration project

Mikkel Gaub(mikg)
Malthe Ettrup Kirkbro(maek)
Mads Frederik Madsen(mfrm)

Overview

The objective of this program is to determine if a given queen placement allows for n queens to be placed on a board of size $n \times n$ board without being able to attack each other. This is accomplished by maintaining n^2 variables in a BDD, each representing whether or not a queen has been placed at the corresponding board location. A rule for each board location is then applied, enforcing that no queen can be placed in a position vulnerable to attack from any other queen – the *board rule*. Another rule of the board ensures that a single queen is placed in each row – the *queens rule*.

Board rule

The board rule is constructed by conjoining the `NAND`-operator between each variable and any other variable in its attack path. As shown in figure , this produces a BDD for each board position, preventing a queen from being in a position allowing it to attack another queen.
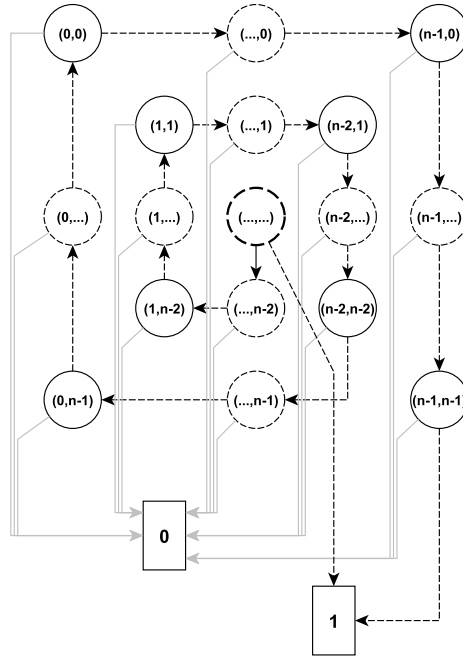

Figure 1: ROBDD of the *board rule* of a single queen placement in the center of a $n \times n$ chessboard. The variable ordering has been altered to give the simplest representation. The included variables are the places on the board, which are reachable from the queen's placement.

Queens rule

The queens rule ensures that n queens are placed on the board, as each of the n rows of the board must contain a queen. This rule is constructed by conjoining the disjunction of each position in the row for each row. As seen in figure this essentially guarantees that at least n queens are placed on the board. When this rule is applied on variables already constrained by the board rule, the objective is solved. We can then count the number of paths through the BDD and thus determine if only one solution is possible, at which point the program automatically places the remaining queens.
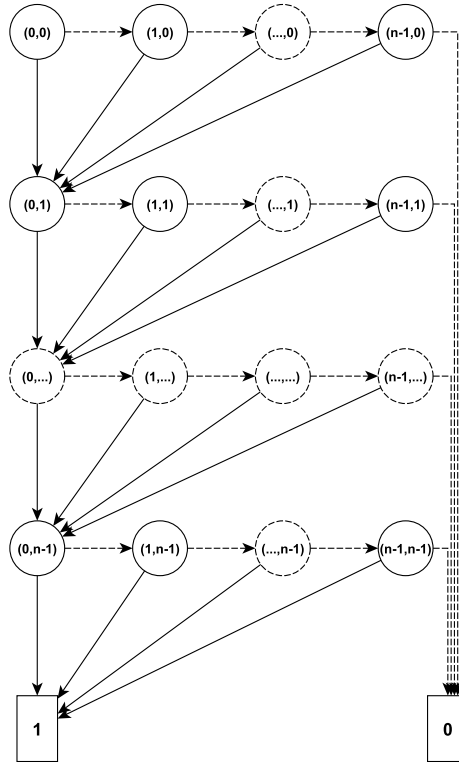

Figure 2: ROBDD of the *queens rule* of a $n \times n$ chessboard.