

Project report - Satisfiability test of clauses and its application

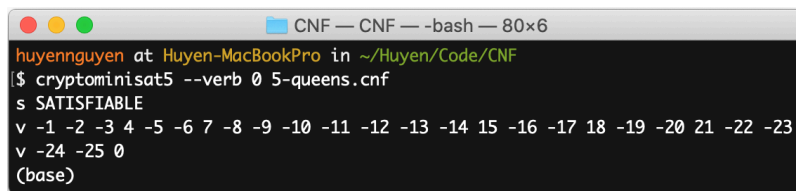
Huyen Nguyen

1. SAT Solver

The SAT Solver I used for this project is cryptominisat, installed on MacOS Catalina 10.15.4 (a UNIX machine).

Compiling cryptominisat using homebrew:
`brew install cryptominisat`

Command-line usage:
`cryptominisat5 --verb 0 5-queens.cnf`

A terminal window titled 'CNF — CNF — -bash — 80x6' shows the user 'huyennguyen' at 'Huyen-MacBookPro' in the directory '~/Huyen/Code/CNF'. The command '\$ cryptominisat5 --verb 0 5-queens.cnf' is executed, resulting in the output 's SATISFIABLE' followed by a list of variables: 'v -1 -2 -3 4 -5 -6 7 -8 -9 -10 -11 -12 -13 -14 15 -16 -17 18 -19 -20 21 -22 -23' and 'v -24 -25 0'. The prompt '(base)' is visible at the bottom.

```
huyennguyen at Huyen-MacBookPro in ~/Huyen/Code/CNF
$ cryptominisat5 --verb 0 5-queens.cnf
s SATISFIABLE
v -1 -2 -3 4 -5 -6 7 -8 -9 -10 -11 -12 -13 -14 15 -16 -17 18 -19 -20 21 -22 -23
v -24 -25 0
(base)
```

2. Encoding the n-queens problem into the DIMACS CNF format

2.1. Encoding the problem

The source code for encoding is at <https://gist.github.com/huyen-nguyen/f497216b6c2601d73fe3ca0f89f5c6d2>

We need to encode the following constraints 1) no two queens on the same row, 2) no two queens on the same column, and 3) no two queens on the same diagonal. (*)

We assign a proposition letter (variable) for each position on the board. Let P_i denote the propositional letter for position (i) of the board. If P_i is true, there is a queen on (i) ; otherwise, there is no queen on (i) . Let N be the size of queens; from now on, we name the cell from 1 (top left) to $N*N$ (bottom right).

The DIMACS CNF formula starts with some comments `c` about the size of n-queens problem, then followed by: `p cnf [N*N] [Number of clauses]`

We break each constraint down to the DIMACS CNF format using some helper functions, whose input is a list of propositions:

- **At least** one of the variables in the chosen list to be true: There is at least 1 queen from list of cells i, k, m :

$$P_i \vee P_k \vee P_m$$

In DIMACS CNF format: `i k m 0`

- **At most** one of the variables in the chosen list to be true: If any P_n is true from list of cells (for example i, k, m), then other P_n 's should be false.

$$\begin{aligned} P_i &\rightarrow \neg P_k \\ P_i &\rightarrow \neg P_m \\ P_m &\rightarrow \neg P_i \\ P_m &\rightarrow \neg P_k \\ P_k &\rightarrow \neg P_i \\ P_k &\rightarrow \neg P_m \end{aligned}$$

where $P_i \rightarrow \neg P_k$ is equivalent to $\neg P_i \vee \neg P_k$ or in DIMACS CNF format: $-i \ -k \ 0$

To generalize, any pair (i,k) of propositions from the list will be in format: $-i \ -k \ 0$

- Exactly one of the variables in the chosen list to be true: Combine at_most and at_least functions. The constraints will be encoded by both functions.

We formulate the general constraints (*) according to helper functions for encoding as:

- There is exactly 1 queen per row
- There is exactly 1 queen per column
- There is at most 1 queen per diagonal, whether it is positive or negative diagonal from top/left/right.

2.2. Live demo

The live demo is available at <https://huyen-nguyen.github.io/n-queens/>

The user can input a number to specify the number of queens (size of board), then click "Run" and the program will output:

- 1) DIMACS CNF formula of such n-queens problem
- 2) Result from SAT Solver of satisfiability of the given proposition

A chess board with corresponding queen positions is provided for visual reference. User can also copy the formula to clipboard or export this formula to a .cnf file.

N-queens Problem SAT Solver
Simple SAT solver for the *n-queens* problem. Just input the number of queens, voilà!

Number of Queens: Run SATISFIABLE

DIMACS CNF formula

```
c SAT Expression for N queens
c Chess board has 16 positions
p cnf 16 84
1 2 3 4 0
-1 -2 0
-1 -3 0
-1 -4 0
-2 -3 0
-2 -4 0
-3 -4 0
5 6 7 0 0
-5 -6 0
-5 -7 0
-5 -8 0
-6 -7 0
-6 -8 0
-7 -8 0
9 10 11 12 0
-9 -10 0
-9 -11 0
-9 -12 0
-10 -11 0
-10 -12 0
-11 -12 0
13 14 15 16 0
-13 -14 0
```

Output

```
s SATISFIABLE
c ----- FINAL TOTAL SEARCH STATS -----
c restarts : 1 (0.00 confs per restart)
c blocked restarts : 0 (0.00 per normal restart)
c decisions : 0 (0.00 % random)
c propagations : 0
c decisions/conflicts : 0.00
c conflicts : 0
c conf lits non-minin : 0 (0.00 lits/conf)
c conf lits final : 0.00
c cache hit re-learn cl : 0 (0.00 % of conf)
c red which : 0 (0.00 % of conf)
c props/decision : 0.00
c props/conflict : 0.00
c 0-depth assigns : 0 (0.00 % vars)
c [acc-substr] long subbySub: 0 subbyStr: 0 lits-rem-str: 0
c [cccl] new: 13 BP 0N: 0.00
c Conflicts in IIP : 0
c Mem used : 0.00 MB
v -1 2 -3 -4 -5 -6 -7 0 9 -10 -11 -12 -13 -14 15 -16 0
```