Giovanni Filomeno (K12315325) and Daniele Bielli (K12340304)

1. Problem Description

The N-Queens problem is a classical mathematical problem of placing N queens on an $N \times N$ chessboard such that no two queens are mutually attacking, i.e., no two queens share the same row, column, or a diagonal. That is, a solution requires that no two queens share the same row, column, or diagonal.

1.1. Declaration of variables

The problem contains 8×8 variables that must be defined at the beginning of the code. The notation for the variable has been defined as qRC where R represents the row number and C the column number. For simplicity, the variables are displayed into a 4 columns table:

(declare-fun q11 () Bool)	(declare-fun q31 () Bool)	(declare-fun q51 () Bool)	(declare-fun q71 () Bool)
(declare-fun q12 () Bool)	(declare-fun q32 () Bool)	(declare-fun q52 () Bool)	(declare-fun q72 () Bool)
(declare-fun q13 () Bool)	(declare-fun q33 () Bool)	(declare-fun q53 () Bool)	(declare-fun q73 () Bool)
(declare-fun q14 () Bool)	(declare-fun q34 () Bool)	(declare-fun q54 () Bool)	(declare-fun q74 () Bool)
(declare-fun q15 () Bool)	(declare-fun q35 () Bool)	(declare-fun q55 () Bool)	(declare-fun q75 () Bool)
(declare-fun q16 () Bool)	(declare-fun q36 () Bool)	(declare-fun q56 () Bool)	(declare-fun q76 () Bool)
(declare-fun q17 () Bool)	(declare-fun q37 () Bool)	(declare-fun q57 () Bool)	(declare-fun q77 () Bool)
(declare-fun q18 () Bool)	(declare-fun q38 () Bool)	(declare-fun q58 () Bool)	(declare-fun q78 () Bool)
(declare-fun q21 () Bool)	(declare-fun q41 () Bool)	(declare-fun q61 () Bool)	(declare-fun q81 () Bool)
(declare-fun q22 () Bool)	(declare-fun q42 () Bool)	(declare-fun q62 () Bool)	(declare-fun q82 () Bool)
(declare-fun q23 () Bool)	(declare-fun q43 () Bool)	(declare-fun q63 () Bool)	(declare-fun q83 () Bool)
(declare-fun q24 () Bool)	(declare-fun q44 () Bool)	(declare-fun q64 () Bool)	(declare-fun q84 () Bool)
(declare-fun q25 () Bool)	(declare-fun q45 () Bool)	(declare-fun q65 () Bool)	(declare-fun q85 () Bool)
(declare-fun q26 () Bool)	(declare-fun q46 () Bool)	(declare-fun q66 () Bool)	(declare-fun q86 () Bool)
(declare-fun q27 () Bool)	(declare-fun q47 () Bool)	(declare-fun q67 () Bool)	(declare-fun q87 () Bool)
(declare-fun q28 () Bool)	(declare-fun q48 () Bool)	(declare-fun q68 () Bool)	(declare-fun q88 () Bool)

1.2. Row constraints

For the row constraints the following logic has been applied: every positive Boolean returns 1 and the sum overall the row has to be one since there is no solution with an empty row or column.

(assert (= 1 (+ (if q11 1 0) (if q12 1 0) (if q13 1 0) (if q14 1 0) (if q15 1 0) (if q16 1 0) (if q17 1 0) (if q18 1 0))))
(assert (= 1 (+ (if q21 1 0) (if q22 1 0) (if q23 1 0) (if q24 1 0) (if q25 1 0) (if q26 1 0) (if q27 1 0) (if q28 1 0))))
(assert (= 1 (+ (if q31 1 0) (if q32 1 0) (if q33 1 0) (if q34 1 0) (if q35 1 0) (if q36 1 0) (if q37 1 0) (if q38 1 0))))
(assert (= 1 (+ (if q41 1 0) (if q42 1 0) (if q43 1 0) (if q44 1 0) (if q45 1 0) (if q46 1 0) (if q47 1 0) (if q48 1 0))))
(assert (= 1 (+ (if q51 1 0) (if q52 1 0) (if q53 1 0) (if q54 1 0) (if q55 1 0) (if q56 1 0) (if q57 1 0) (if q58 1 0))))
(assert (= 1 (+ (if q61 1 0) (if q62 1 0) (if q63 1 0) (if q64 1 0) (if q65 1 0) (if q66 1 0) (if q67 1 0) (if q68 1 0))))
(assert (= 1 (+ (if q71 1 0) (if q72 1 0) (if q73 1 0) (if q74 1 0) (if q75 1 0) (if q76 1 0) (if q77 1 0) (if q78 1 0))))
(assert (= 1 (+ (if q81 1 0) (if q82 1 0) (if q83 1 0) (if q84 1 0) (if q85 1 0) (if q86 1 0) (if q87 1 0) (if q88 1 0))))

1.3. Column constraints

Same logic have been applied for the columns

(assert (= 1 (+ (if q11 1 0) (if q21 1 0) (if q31 1 0) (if q41 1 0) (if q51 1 0) (if q61 1 0) (if q71 1 0) (if q81 1 0))))
(assert (= 1 (+ (if q12 1 0) (if q22 1 0) (if q32 1 0) (if q42 1 0) (if q52 1 0) (if q62 1 0) (if q72 1 0) (if q82 1 0))))
(assert (= 1 (+ (if q13 1 0) (if q23 1 0) (if q33 1 0) (if q43 1 0) (if q53 1 0) (if q63 1 0) (if q73 1 0) (if q83 1 0))))
(assert (= 1 (+ (if q14 1 0) (if q24 1 0) (if q34 1 0) (if q44 1 0) (if q54 1 0) (if q64 1 0) (if q74 1 0) (if q84 1 0))))
(assert (= 1 (+ (if q15 1 0) (if q25 1 0) (if q35 1 0) (if q45 1 0) (if q55 1 0) (if q65 1 0) (if q75 1 0) (if q85 1 0))))
(assert (= 1 (+ (if q16 1 0) (if q26 1 0) (if q36 1 0) (if q46 1 0) (if q56 1 0) (if q66 1 0) (if q76 1 0) (if q86 1 0))))
(assert (= 1 (+ (if q17 1 0) (if q27 1 0) (if q37 1 0) (if q47 1 0) (if q57 1 0) (if q67 1 0) (if q77 1 0) (if q87 1 0))))
(assert (= 1 (+ (if q18 1 0) (if q28 1 0) (if q38 1 0) (if q48 1 0) (if q58 1 0) (if q68 1 0) (if q78 1 0) (if q88 1 0))))

1.4. Diagonal Constraints

Regarding the diagonal, a set of conjunction and disjunction have been used. The reason for the different declaration w.r.t. to columns/rows is that in this case we have to ensure that two queen do not share the same diagonal, but we do not impose any presence on a specific diagonal.

(assert (not (or (and q11 q22) (and q11 q33) (and q11 q44) (and q11 q55) (and q11 q66) (and q11 q77))))
(assert (not (or (and q12 q23) (and q12 q34) (and q12 q45) (and q12 q56) (and q12 q67))))
(assert (not (or (and q13 q24) (and q13 q35) (and q13 q46) (and q13 q57))))
(assert (not (or (and q13 q22))))
(assert (not (or (and q14 q25) (and q14 q36) (and q14 q47))))
(assert (not (or (and q14 q23) (and q14 q32))))
(assert (not (or (and q15 q26) (and q15 q37))))
(assert (not (or (and q15 q24) (and q15 q33) (and q15 q42))))
(assert (not (or (and q16 q27))))
(assert (not (or (and q16 q25) (and q16 q34) (and q16 q43) (and q16 q52))))
(assert (not (or (and q17 q26) (and q17 q35) (and q17 q44) (and q17 q53) (and q17 q62))))
(assert (not (or (and q18 q27) (and q18 q36) (and q18 q45) (and q18 q54) (and q18 q63) (and q18 q72))))
(assert (not (or (and q21 q32) (and q21 q43) (and q21 q54) (and q21 q65) (and q21 q76))))
(assert (not (or (and q22 q33) (and q22 q44) (and q22 q55) (and q22 q66) (and q22 q77))))
(assert (not (or (and q23 q34) (and q23 q45) (and q23 q56) (and q23 q67))))
(assert (not (or (and q23 q32))))
(assert (not (or (and q24 q35) (and q24 q46) (and q24 q57))))
(assert (not (or (and q24 q33) (and q24 q42))))
(assert (not (or (and q25 q36) (and q25 q47))))
(assert (not (or (and q25 q34) (and q25 q43) (and q25 q52))))
(assert (not (or (and q26 q37))))
(assert (not (or (and q26 q35) (and q26 q44) (and q26 q53) (and q26 q62))))
(assert (not (or (and q27 q36) (and q27 q45) (and q27 q54) (and q27 q63) (and q27 q72))))
(assert (not (or (and q28 q37) (and q28 q46) (and q28 q55) (and q28 q64) (and q28 q73))))
(assert (not (or (and q31 q42) (and q31 q53) (and q31 q64) (and q31 q75))))
(assert (not (or (and q32 q43) (and q32 q54) (and q32 q65) (and q32 q76))))
(assert (not (or (and q33 q44) (and q33 q55) (and q33 q66) (and q33 q77))))
(assert (not (or (and q33 q42))))
(assert (not (or (and q34 q45) (and q34 q56) (and q34 q67))))
(assert (not (or (and q34 q43) (and q34 q52))))

(assert (not (or (and q35 q46) (and q35 q57))))
(assert (not (or (and q35 q44) (and q35 q53) (and q35 q62))))
(assert (not (or (and q36 q47))))
(assert (not (or (and q36 q45) (and q36 q54) (and q36 q63) (and q36 q72))))
(assert (not (or (and q37 q46) (and q37 q55) (and q37 q64) (and q37 q73))))
(assert (not (or (and q38 q47) (and q38 q56) (and q38 q65) (and q38 q74))))
(assert (not (or (and q41 q52) (and q41 q63) (and q41 q74))))
(assert (not (or (and q42 q53) (and q42 q64) (and q42 q75))))
(assert (not (or (and q43 q54) (and q43 q65) (and q43 q76))))
(assert (not (or (and q43 q52))))
(assert (not (or (and q44 q55) (and q44 q66) (and q44 q77))))
(assert (not (or (and q44 q53) (and q44 q62))))
(assert (not (or (and q45 q56) (and q45 q67))))
(assert (not (or (and q45 q54) (and q45 q63) (and q45 q72))))
(assert (not (or (and q46 q57))))
(assert (not (or (and q46 q55) (and q46 q64) (and q46 q73))))
(assert (not (or (and q47 q56) (and q47 q65) (and q47 q74))))
(assert (not (or (and q48 q57) (and q48 q66) (and q48 q75))))
(assert (not (or (and q51 q62) (and q51 q73))))
(assert (not (or (and q52 q63) (and q52 q74))))
(assert (not (or (and q53 q64) (and q53 q75))))
(assert (not (or (and q53 q62))))
(assert (not (or (and q54 q65) (and q54 q76))))
(assert (not (or (and q54 q63) (and q54 q72))))
(assert (not (or (and q55 q66) (and q55 q77))))
(assert (not (or (and q55 q64) (and q55 q73))))
(assert (not (or (and q56 q67))))
(assert (not (or (and q56 q65) (and q56 q74))))
(assert (not (or (and q57 q66) (and q57 q75))))
(assert (not (or (and q58 q67) (and q58 q76))))
(assert (not (or (and q61 q72))))
(assert (not (or (and q62 q73))))
(assert (not (or (and q63 q74))))
(assert (not (or (and q63 q72))))
(assert (not (or (and q64 q75))))
(assert (not (or (and q64 q73))))
(assert (not (or (and q65 q76))))
(assert (not (or (and q65 q74))))
(assert (not (or (and q66 q77))))
(assert (not (or (and q66 q75))))
(assert (not (or (and q67 q76))))

2. Output

Satisfiable. Solution (only True values):

q32 = True

q74 = True

q61 = True

q45 = True

q58 = True

q26 = True

q87 = True

q13 = True

3. Visual representation of Output

		×					
					×		
	×						
				×			
							×
×							
			×				
						×	

4. Appendix: Python code for automatic code generation

To automatically generates the code, a python code has been used.

In [ ]:

```python
def generate_smtlib_for_n_queens(n):
    smtlib_script = ""

    # Declare variables
    smtlib_script += "; Declare variables\n"
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            smtlib_script += f"(declare-fun q{i}{j} () Bool)\n"

    # Row constraints
    smtlib_script += "\n; Row constraints\n"
    for i in range(1, n + 1):
        smtlib_script += "(assert (= 1 (+ "
        smtlib_script += " ".join(f"(if q{i}{j} 1 0)" for j in range(1, n + 1))
        smtlib_script += ")))\n"

    # Column constraints
    smtlib_script += "\n; Column constraints\n"
    for j in range(1, n + 1):
        smtlib_script += "(assert (= 1 (+ "
        smtlib_script += " ".join(f"(if q{i}{j} 1 0)" for i in range(1, n + 1))
        smtlib_script += ")))\n"

    # Diagonal constraints
    smtlib_script += "\n; Diagonal constraints\n"
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            # Primary diagonals
            primary = []
            for d in range(1, n - max(i, j)):
                primary.append(f"(and q{i}{j} q{i + d}{j + d})")
            if primary:
                smtlib_script += f"(assert (not (or {' '.join(primary)})))\n"

            # Secondary diagonals
            secondary = []
            for d in range(1, min(n - i, j - 1)):
                secondary.append(f"(and q{i}{j} q{i + d}{j - d})")
            if secondary:
                smtlib_script += f"(assert (not (or {' '.join(secondary)})))\n"

    # Solve command
    smtlib_script += "\n; Solve the problem\n(check-sat)\n(get-model)\n"

    return smtlib_script

# Generate SMTLib script for 8-Queens problem
smtlib_code = generate_smtlib_for_n_queens(8)
smtlib_code
```