Discrete Mathematics

Solving Puzzles with SAT Solver

Shin Hong

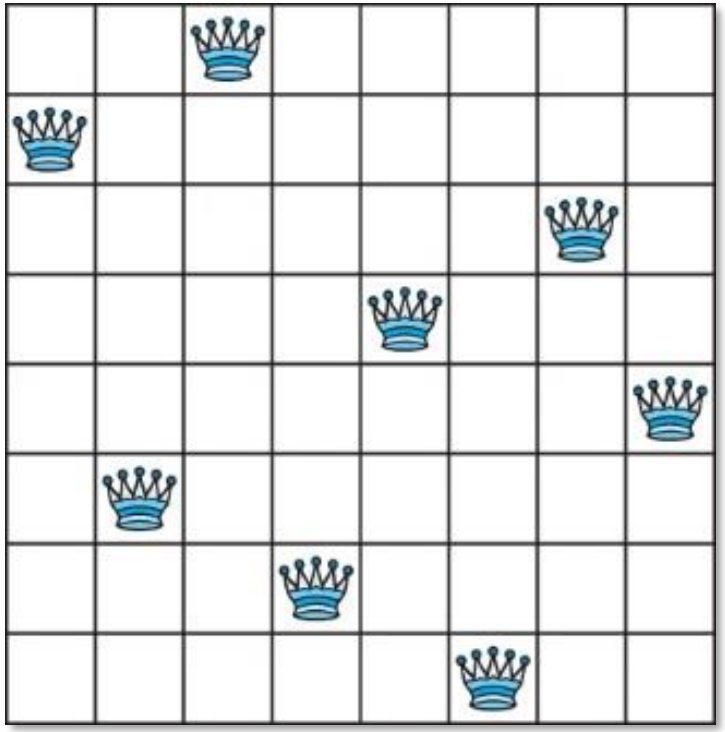
Logic Programming with SAT Solver

- Write a program by specifying the conditions that an expected output must satisfy, and uses a constraint solver to find the output
  - instead of constructing a procedure

- A constraint solver is a program that quickly finds a solution or checks a property of a given logic formula
  - SAT solvers check if a propositional formula is satisfiable and finds a solution if there is. SMT solvers treat certain subclasses of predicate formulas.
  - SAT/SMT solvers are widely used for real-world applications as their scalability and efficiency have been dramatically improved since mid 2000
    - e.g., Software Package Management module of Eclipse

N-Queen Problem (1/2)

- Place *N* queens on a *N*-by-*N* checkboard such that no two queens are placed on the same vertical, horizontal or diagonal line



N-Queen Problem (2/2)

$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$
$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$
$p_{3,1}$	$p_{3,2}$	$p_{3,3}$	$p_{3,4}$
$p_{4,1}$	$p_{4,2}$	$p_{4,3}$	$p_{4,4}$

- Proposition variable $p_{i,j}$ indicates whether a queen is placed at the i-th row and at the j-th column
- The condition for a solution is represented as a propositional formula over $p_{1,1}$ to $p_{N,N}$

$$Q_{RE} = \bigwedge_{i=1..n} \bigvee_{j=1..n} p_{i,j}$$

$$Q_{RU} = \bigwedge_{i=1..n} \bigwedge_{j=1..n-1} \bigwedge_{k=j+1..n} \neg(p_{i,j} \wedge p_{i,k})$$

$$Q_{CE} = \bigwedge_{j=1..n} \bigvee_{i=1..n} p_{i,j}$$

$$Q_{CU} = \bigwedge_{i=1..n} \bigwedge_{j=1..n-1} \bigwedge_{k=j+1..n} \neg(p_{j,i} \wedge p_{k,i})$$

$$Q_{DU1} = \bigwedge_{s=3..n+1} \bigwedge_{i=2..n} \bigwedge_{k=1..i-1} \neg(p_{i,s-i} \wedge p_{k,s-k})$$

$$Q_{DU2} = \bigwedge_{s=n+2..2n-1} \bigwedge_{j=2..n-1} \bigwedge_{k=i+1..n} \neg(p_{s-j,j} \wedge p_{s-k,k})$$

$$Q_{DD1} = \bigwedge_{d=0..n-2} \bigwedge_{i=1..n-1} \bigwedge_{k=i+1..n} \neg(p_{i+d,i} \wedge p_{k+d,k})$$

$$Q_{DD2} = \bigwedge_{d=0..n-2} \bigwedge_{i=1..n-1} \bigwedge_{k=i+1..n} \neg(p_{i,i+d} \wedge p_{k,k+d})$$

SAT Solving with Z3 (1/2)

- Microsoft Z3: https://github.com/Z3Prover/z3
  - tutorial: https://www.philipzucker.com/z3-rise4fun/guide.html
  - interactive demo: https://www.philipzucker.com/z3-rise4fun/

- Input example: $(p \to q) \land (q \to \neg p) \land \neg(p \lor q)$

```
$ cat prop.txt
; prop.txt
(declare-const p Bool)
(declare-const q Bool)
(assert (and (=> p q)
             (=> q (not p))
             (not (or p q))))
(check-sat)
(get-model)
```

```
$ z3 prop.txt
sat
(model
  (define-fun q () Bool
    false)
  (define-fun p () Bool
    false)
)
```

Application: Sudoku Puzzle (1/3)

- A Sudoku puzzle is represented as a 9x9 grid with nine 3x3 subgrids called subgrids
- Each cell has a number in 1 to 9
- The puzzle is solved by assigning a number to each cell so that every row, every column, and every of a subgrid contains each of the 9 numbers

	2	9				4		
			5			1		
	4							
				4	2			
6							7	
5								
7			3					5
	1			9				
							6	

Application: Sudoku Puzzle (2/3)

- $p(i,j,n)$ holds when row i and column j has n

	2	9				4		
			5			1		
	4							
				4	2			
6							7	
5								
7			3					5
	1			9				
							6	

- each cell is assigned with exactly one number

$$\bigwedge_{i=1}^{9}\bigwedge_{j=1}^{9}\bigwedge_{n=1}^{8}\bigwedge_{m=n+1}^{9}\neg(p(i,j,n)\wedge p(i,j,m))$$

$$\bigwedge_{i=1}^{9}\bigwedge_{j=1}^{9}\bigvee_{n=1}^{9}p(i,j,n)$$

- each pre-assigned cell contains the given number
  - E.g., $p(1,2,2)\wedge p(1,3,9)\wedge \ \ldots \ \wedge p(2,4,5)\wedge\cdots p(9,8,6)$

Application: Sudoku Puzzle (3/3)

- each row has every number between 1 and 9

$$\bigwedge_{i=1}^{9}\bigwedge_{n=1}^{9}\bigvee_{j=1}^{9} p(i,j,n)$$

- each column has every number between 1 and 9

$$\bigwedge_{j=1}^{9}\bigwedge_{n=1}^{9}\bigvee_{i=1}^{9} p(i,j,n)$$

- each subgrid has every number between 1 and 9

$$\bigwedge_{r=0}^{2}\bigwedge_{s=0}^{2}\bigwedge_{n=1}^{9}\bigvee_{i=1}^{3}\bigvee_{j=1}^{3} p(3r+i,3s+j,n)$$

	2	9				4		
			5			1		
	4							
				4	2			
6							7	
5								
7			3					5
	1			9				
							6	

Homework 1

- Write a program that finds a solution of a given puzzle problem using the Z3 SAT solver

- Collaborative work (70%): by 11:59 PM, Oct 7 (Thur)
  - Anti-King Sudoku
  - Nondango

- Individual work (30%): by 11:59 PM, Oct 11 (Mon)
  - Gappy