Knowledge Representation and Reasoning: Constraint Satisfaction Problems

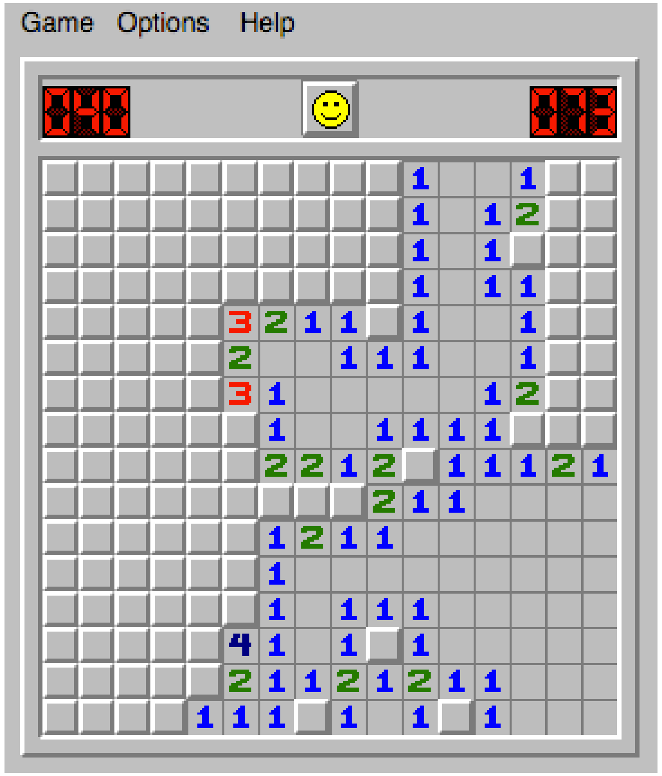
Chapter 5

Outline of the lecture

- ♢ Introduction
- ♢ Constraint Networks
- ♢ CSPs: the Logical View
- ♢ Assignments, Consistency, Solutions
- ♢ Backtracking

Constraint Satisfaction Example: Minesweeper

// Reasoning Prob.



Constraint Satisfaction Example: Sudoku

5			4		2		6	
		9		5		1		
		8		9	1	7		5
							2	6
1	2	5	3	4	6	8	9	7
6	9							
9		6	1	3		2		
		3		2		6		
	4		8		5			1

Conditions imposed, have to satisfy in order to solve

Constraint Satisfaction: Car Sequencing

CSPs: a general class of problems

◇ General problem: find an arrangement agreeing with a set of constraints
— distribution of mines and non-mines giving the right numbers
— ways to fill in squares so that rows, columns and blocks are all permutations of $(1, \ldots, 9)$
— order of cars so that every assembly job gets done smoothly

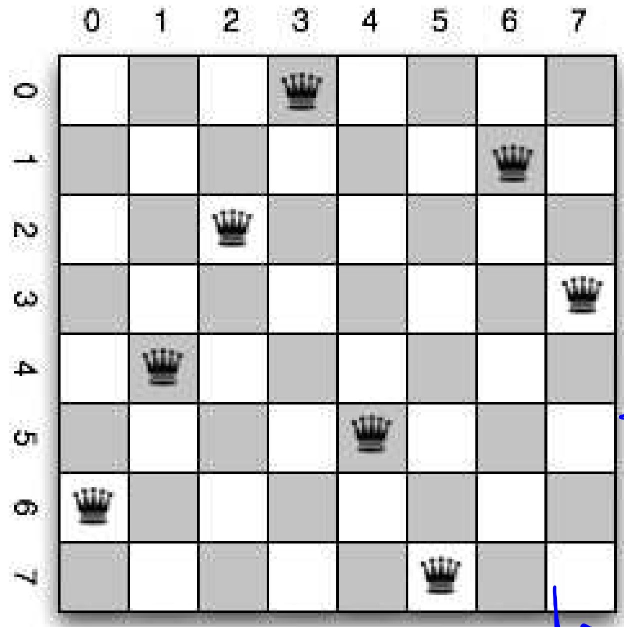
◇ Situation can be described by a set of variables

◇ Constraint is a condition the variables must meet

◇ Problem: find assignments of values satisfying all constraints

◇ May want any solution, all solutions, a good/best solution, . . .

Example: 8 queens problem



the bigger the board the more the solutions

- Variables: positions of the 8 queens
- Domains: squares of the board
- Constraints: no 2 queens in the same row, column or diagonal

92 solns

Binary constraint network

◊ A constraint network is a triple $\langle V, D, C \rangle$

◊ V a finite set of variables $v_1, \ldots, v_n$

◊ D a set of [finite] sets $D_{v_1}, \ldots, D_{v_n}$

◊ C a set of binary relations $\{C_{u,v} \mid u, v \in V, u \neq v\}$
$C_{u,v} \subseteq D_u \times D_v$ [Constrains between u & v = set of pairs (state u, state v)]

E.g. $V = \{a, b\}$. Suppose $D_a = \{1, \ldots, 10\}$ and $D_b = \{8, \ldots, 20\}$.
If we require $a > b$ then $C_{a,b}$ is the set $\{(9, 8), (10, 8), (10, 9)\}$.

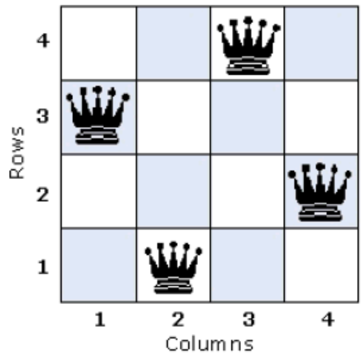
Constraint network: notes

- A constraint $C_{u,v}$ is the allowed pairs of assignments to u and v
- These are arbitrary relations: they need not have an intuitive reading
  unrestricted
  using/based on what one feels to be true even without conscious reasoning
- Sometimes require domains to be finite (FD problem)
  Sometimes allow domains to be infinite (e.g. integers, reals)
- Extension to non-binary constraints is simple. [can be transformed to binary]
- SAT is the special case where all domains have just 2 values.
  size blows up
- Linear programming is the special case where domains are the real numbers and all constraints are linear inequalities.

Cost: exponentially

in most cases we do not do that

Queens problem again



◇ Variables: $V = \{v_1, v_2, v_3, v_4\}$. Row of queen in each column

◇ Domains: For all v, $D_v = \{1, 2, 3, 4\}$

◇ Constraints: For $1 \leq i < j \leq 4$

$$C_{v_i,v_j} = \{(d, d') \in D \times D \ : \ d \neq d', \ \underline{|d - d'| \neq |i - j|}\}$$

no two queens on the same diagonal

e.g. $C_{v_1,v_3} = \{(1,2), (1,4), (2,1), (2,3), (3,2), (3,4), (4,1), (4,3)\}$

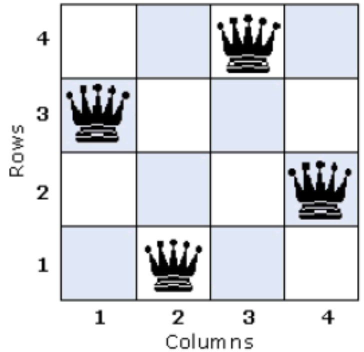
◇ Assignment above is $v_1 \leftarrow 3, \ v_2 \leftarrow 1, \ v_3 \leftarrow 4, \ v_4 \leftarrow 2$

The logical view

- In interesting cases, problems have logical descriptions
- Interpretation of logic: assign a relation to each predicate, and a function to each function symbol.
- Makes formulae true or false.
- Interpreted over finite domain, need to specify value of each function f for each choice of arguments.
  — E.g. decide that $f(a) = 3$
- So term $f(a)$ corresponds to a decision variable
  — Has a set of possible values (its domain)
  — Is assigned a value from this domain on any interpretation
- Constraints can be written as logical formulae
  — Succinct and readable formulation
- Solutions to the CSP are exactly models of the theory

interpretations to make the theory true.

From a logical point of view



Given: monadic function symbol $q(\_)$.

Find: interpretation satisfying

$$\forall x \forall y([q(x) = q(y)] \rightarrow [x = y])$$

$$\forall x \forall y([abs(q(x) - q(y)) = abs(x - y)] \rightarrow [x = y])$$

over the domain $\{1, 2, 3, 4\}$

Consistency

Definition (Consistency). Let $\langle V, D, C\rangle$ be a constraint network.
Let a be a partial assignment.

/ some variables instead of all variables /

a is inconsistent if there are variables u, v in V and a constraint $C_{u,v}$ in C such that $a(u)$ and $a(v)$ are defined, and $(a(v), a(u)) \notin C_{u,v}$

In that case, a violates the constraint $C_{u,v}$

◇ Consistency is <u>local</u>: inconsistent a already violates a constraint.

Solution

Definition (Solution). Let $\gamma = \langle V, D, C \rangle$ be a constraint network.

a is a solution to γ if it is a total consistent assignment for γ.

If a solution to γ exists, then γ is solvable. Otherwise it is unsolvable or over-constrained.

A partial assignment a can be extended to a solution if there is a solution which agrees with a wherever a is defined.

Not every consistent partial assignment can be extended to a solution

eg. [empty or No assignments]

Consistent ≠ Being part of solution

Searching for solutions

◇ Search: Systematic enumeration of partial assignments
— If a complete assignment is found, that's a solution
— If the search space is exhausted, there are no [more] solutions

◇ Backtracking: Pruning of inconsistent partial assignments (and all their extensions)

◇ Inference: Reasoning about a partial assignment, to tighten constraints and reduce domains for its extensions

◇ There is a tradeoff: reduction in number of search nodes vs runtime needed for inference

Pure Backtracking

function BACKTRACK(γ, a) **returns** solution, or "inconsistent"

    if a is inconsistent with γ **then return** "inconsistent"
    if a is total **then return** a
    select variable v for which a is not defined
    for each d in D_v **do**
        $a' \leftarrow a \cup \{(v, d)\}$
        $a'' \leftarrow$ BACKTRACK(γ, a')
        if $a'' \neq$ "inconsistent" **then return** a'
        return "inconsistent"
    end

call: BACKTRACK(γ, $\{\}$)

/ naive backtracking

Pure Backtracking: notes

◇ Informal version:

Recursively instantiate variables one by one, backing up out of a search branch if the partial assignment is inconsistent.

◇ Better that exhaustive search: avoids enumerating many inconsistent (partial) assignments by detecting them early

◇ **Advantages:**

Very simple to implement

Very fast (per node of the search tree)

Complete (always gives a decision)

◇ **Disadvantages:**

Does no reasoning except detecting actual inconsistency // No intelligent reasoning

Cannot look further ahead than the current state

Summary

◊ Constraint networks consist of variables associated with (usually finite) domains and constraints which are [binary] relations specifying allowed pairs (or tuples) of values.

◊ A partial assignment maps some variables to values; a total assignment does so for all variables. A partial assignment is consistent if it does not violate any constraint. A consistent total assignment is a solution.

◊ The constraint satisfaction problem (CSP) consists in finding a solution for a constraint network. Applications are everywhere!

◊ Backtracking instantiates variables one by one, cutting branches when inconsistent partial assignments occur.