Constraint Satisfaction Problems/Programming

A4B33ZUI, LS 2016/2017

Branislav Bošanský, Ondřej Vaněk, Karel Horak
{name.surname}@agents.fel.cvut.cz
Artificial Intelligence Center, Czech Technical University

State space search → CSP

- until now – no assumptions on the states
- CSP:
  - state → array of variables
  - context → domains for variables, set of constraints among the variables
  - goal → all variables have an assigned value, no constraint is violated
- alternative goals
  - optimization variant

Advantages/Disadvantages

- general enough to model many problems
- more efficient algorithms that exploit the structure
  - we can do better than DFS/BFS
  - advanced search techniques can be possibly reused in non-CSP problems
  - generic purpose CSP solvers
- not all problems can be modeled as CSP
- some formulations can be inefficient

Baseline Algorithm

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
    return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment given CONSTRAINTS[csp] then
            add {var = value} to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result ≠ failure then return result
            remove {var = value} from assignment
    return failure
```

Example

- map coloring, satisfiability, …
- sudoku, crypto-arithmetic
- scheduling requests to hotel rooms
  - list of requests (# of people, from, to)
  - set of rooms (# of beds)

Binary CSP

- we can do better than passively checking constraints
- unified structure of CSP instances
  - all constraints are binary
  - can we formalize all CSP problems with binary variables?
- constraints can be represented as a graph
  - nodes – variables
  - edges – constraints between variables

Consistency

- nodes
- edges / arcs

Consistency

- nodes
  - unary constraints are reflected in the domain of the variable
  - every value in the domain satisfies all unary constraints
- edges / arcs
  - directed $v_i \rightarrow v_j$
  - for every *x* from v_i domain there exists a value *y* from v_j domain that satisfies all constraints between v_i and v_j

Arc Consistency

- make each edge in the graph of constraints consistent

Arc consistency algorithm

```
function AC-3( csp) returns the CSP, possibly with reduced domains
   inputs: csp, a binary CSP with variables {X1, X2, ..., Xn}
   local variables: queue, a queue of arcs, initially all the arcs in csp

   while queue is not empty do
      (Xi, Xj) ← REMOVE-FIRST(queue)
      if REMOVE-INCONSISTENT-VALUES(Xi, Xj) then
         for each Xk in NEIGHBORS[Xi] do
            add (Xk, Xi) to queue
```

```
function REMOVE-INCONSISTENT-VALUES( Xi, Xj) returns true iff succeeds
   removed ← false
   for each x in DOMAIN[Xi] do
      if no value y in DOMAIN[Xj] allows (x,y) to satisfy the constraint Xi ↔ Xj
         then delete x from DOMAIN[Xi];  removed ← true
   return removed
```

Homework assignment 2