

# **Constraint Satisfaction Problems/Programming**

**B4B36ZUI, LS 2020**

**Jaromír Janisch ,Martin Schaefer**

Artificial Intelligence Center, Czech Technical University in Prague

# 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 (COP)

# 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  $\{X_1, X_2, \dots, X_n\}$ 
local variables: queue, a queue of arcs, initially all the arcs in csp

while queue is not empty do
     $(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\textit{queue})$ 
    if REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) then
        for each  $X_k$  in NEIGHBORS[ $X_i$ ] do
            add  $(X_k, X_i)$  to queue

```

---

```

function REMOVE-INCONSISTENT-VALUES( $X_i, X_j$ ) returns true iff succeeds
    removed  $\leftarrow$  false
    for each  $x$  in DOMAIN[ $X_i$ ] do
        if no value  $y$  in DOMAIN[ $X_j$ ] allows  $(x, y)$  to satisfy the constraint  $X_i \leftrightarrow X_j$ 
            then delete  $x$  from DOMAIN[ $X_i$ ]; removed  $\leftarrow$  true
    return removed

```

# Example

