Title: CSP 101 (cont.)

Material: Constraint Satisfaction, Dechter MIT Encyclopedia
CSP, An Overview, Pedro Meseguer

Foundations of Constraint Satisfaction
CSCE421/821, Fall 2001
`www.cse.unl.edu/~choueiry/F01-421-821/`

Berthe Y. Choueiry (Shu-we-ri)
Ferguson Hall, Room 104
`choueiry@cse.unl.edu`, Tel: (402)472-5444

Domain types

Given $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C}) \begin{cases} \mathcal{V} = \{V_1, V_2, \ldots, V_n\} \\ \mathcal{D} = \{D_{V_1}, D_{V_2}, \ldots, D_{V_n}\} \\ \mathcal{C} \text{ set of constraints} \end{cases}$

$$C_{V_a, V_b, \ldots, V_i} = \{\, (x, y, \ldots, z)\} \subseteq D_{V_a} \times D_{V_b} \times \ldots \times D_{V_i}$$

Domains:

$\longrightarrow$ Finite (discrete), enumeration works

$\longrightarrow$ Continuous, sophisticated algebraic techniques are needed

Terminology

Given $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C}) \ldots$

Query:

- Find a solution
- Find all solutions
- Find number of solutions
- Can you extend a partial solution, *etc.*

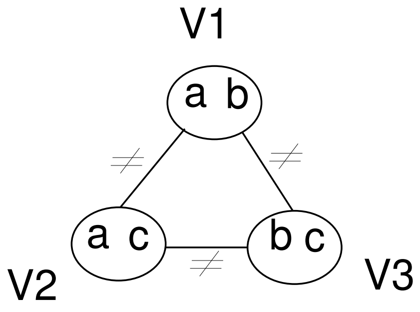
Constraints:

- *arity*: universal, unary, binary, ternary, ... global
- *scope*: set of variables to which the constraint applies
- *definition*: intension, extension
- *implementation*: predicate, set of tuples, binary matrix, etc.

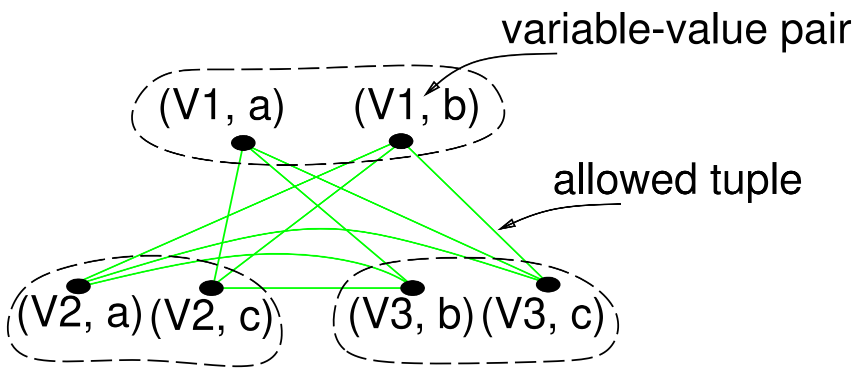
Representation:

Macrostructure $\mathcal{G}(\mathcal{P})$:

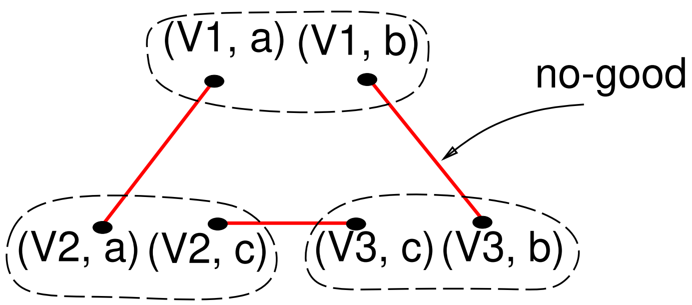
- constraint graph for binary constraints
- constraint network for non-binary constraints



Micro-structure $\mu(\mathcal{P})$:



Co-microstructure co-$\mu(\mathcal{P})$:



Complexity of CSP

Characterization:

Decision problem. In general, NP-complete by reduction from 3SAT

Steps for proving NP-completeness:

Given a problem Π in **NP**, show that some already know **NP-complete** problem Π' can be transformed to Π.

1. Show that Π is in NP
2. Select a known **NP-complete** problem Π'
3. Construct a transformation f from Π' to Π', and
4. Prove that f is a polynomial transformation

(check Chapter 3 of Garey & Johnson)

What is SAT?

SAT?

Given a sentence:

- *Sentence:* conjunction of clauses
  $(c_1 \vee \neg c_4 \vee c_5 \vee c_6) \wedge (c_2 \vee \neg c_3) \wedge (\neg c_4)$
- *Clause:* disjunction of literals
  $(c_2 \vee \neg c_3)$
- *Literal:* a term or its negation
  $c_1, \neg c_6$
- *Term:* boolean variable
  $c_1 = 1 \Leftrightarrow \neg c_1 = 0$

Question: Find an assignment of truth values to Boolean variables such the sentence is satisfied.

Problem reduction

Example: CSP into SAT *(proves nothing, just an exercise)*

Notation: variable-value pair = VVP

- VVP $\rightarrow$ term
- VVPs of a variable $\rightarrow$ disjunction of terms
- Constraint:

  tuple $\rightarrow$ conjunction of terms

  constraint $\rightarrow$ disjunction of these conjunctions

  $\longrightarrow$ transform into conjunctive normal form (CNF)

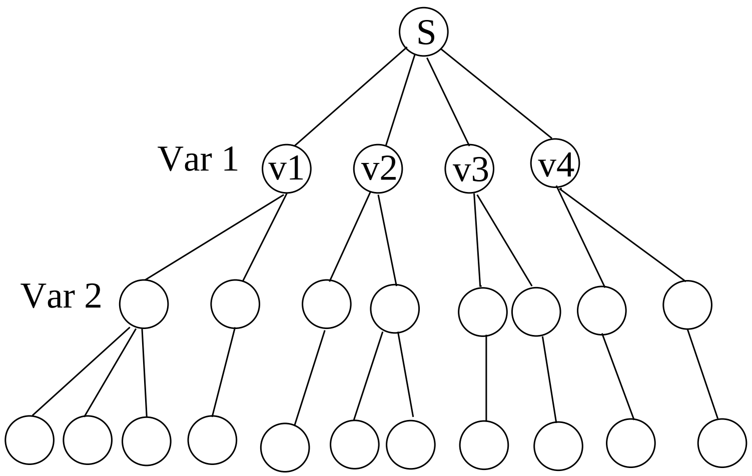
How to solve a CSP? search $\begin{cases} \text{Constructive, systematic} \\ \text{Iterative repair} \end{cases}$

Systematic search: $\rightarrow$ Starting from a root node

$\rightarrow$ Consider all values for a variable V_1

$\rightarrow$ For every value for V_1, consider all values for V_2

$\rightarrow$ etc..



For n variables, each of domain size d:

- Maximum depth? *fixed!*

- Maximum number of paths? *size of search space, size of CSP*

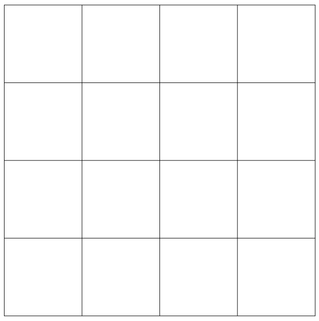
Before looking at search, consider

1. Importance of modeling/formulating to control the size of the search space

2. Preprocessing: consistency checking/filtering, constraint propagation to reduce size of search space

Importance of modeling

N-**queens**: formulation 1

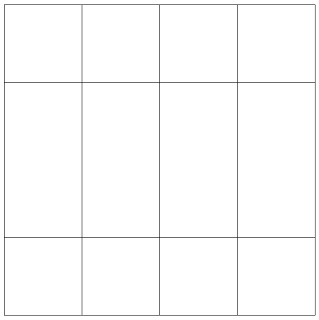


Variables?

Domains?

<u>Size</u> of CSP?

N-**queens**: formulation 2



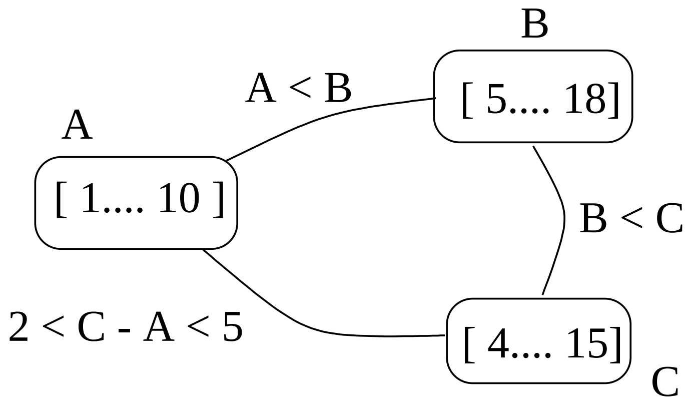
variables?

domains?

<u>size</u> of csp?

Constraint checking

→ arc-consistency



1- B: [5 .. 14]
   C: [6 .. 15]

2- A: [2 .. 10]
   C: [6 .. 14]

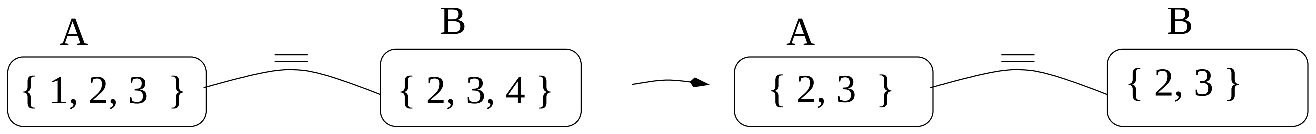
3- B: [5 .. 13]

⟶ 3-consistency, k-consistency ($k \leq n$)

⟶ Constraint filtering, constraint checking, etc..

eliminate non-acceptable tuples prior to search

→ Warning: arc-consistency does not solve the problem

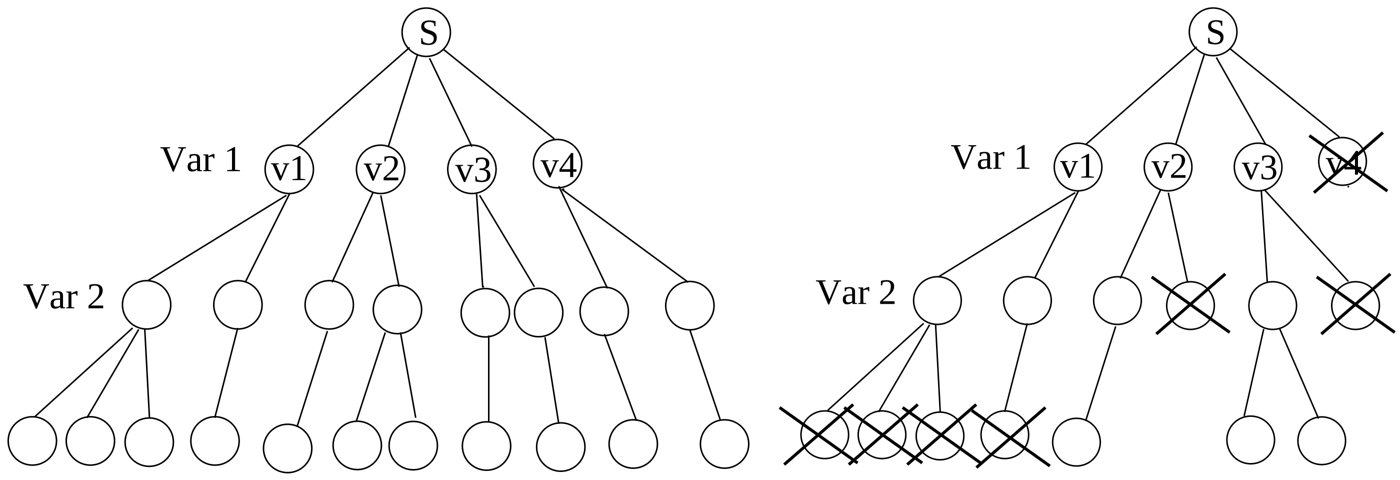


(A=2) ∧ (B=3) still isn't a solution!

Systematic search (I) ***Back-checking***

Systematic search generates d^n possibilities

Are all possibilities acceptable?

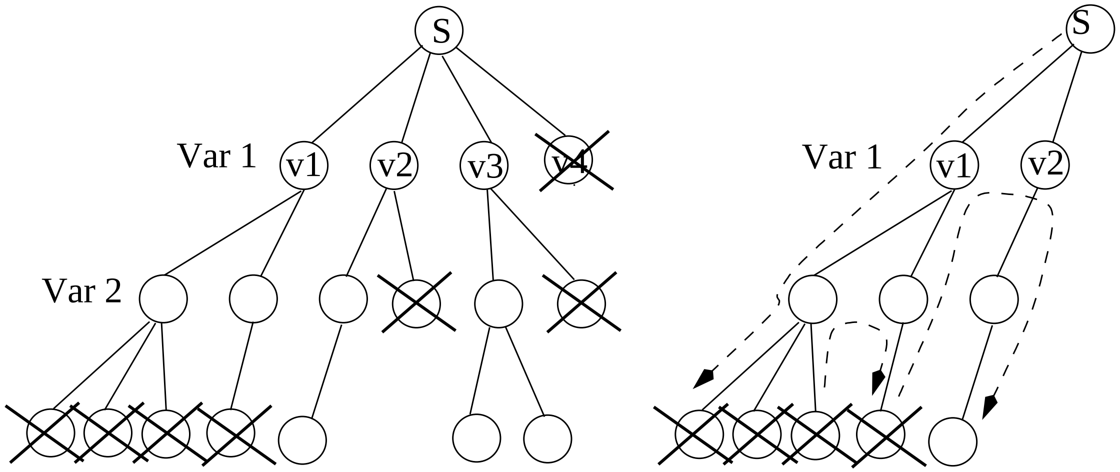


→ Expand a partial solution only when consistent

⟶ **early pruning**

Systematic search (II) ***Chronological backtracking***

What if only <u>one</u> solution is needed?



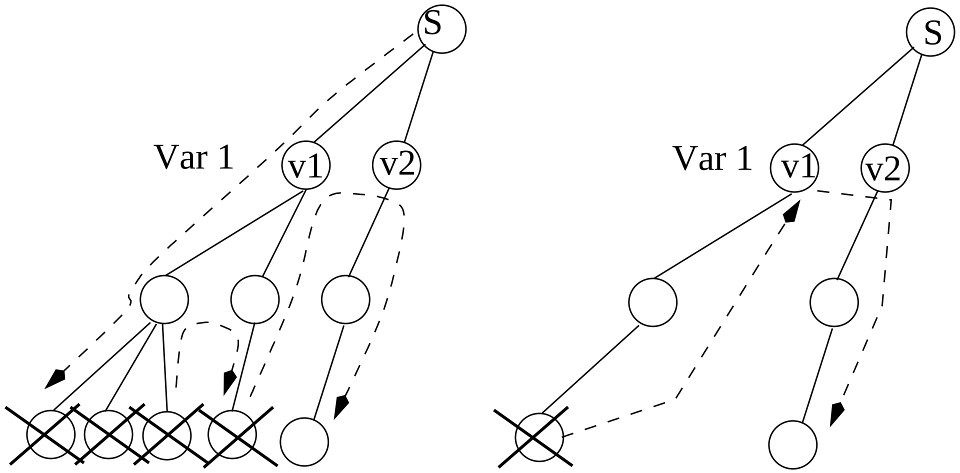
$\longrightarrow$ **Depth-first search & chronological backtracking**

$\rightarrow$ DFS: soundness? completeness?

Systematic search (III) — ***Intelligent backtracking***

What if the reason for failure was higher up in the tree?

Backtrack to source of conflict!!



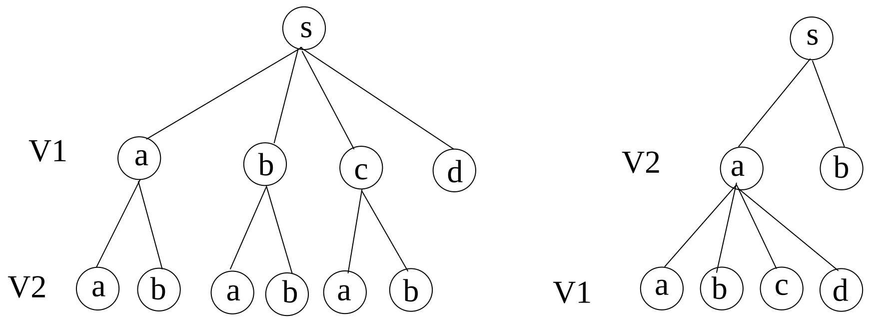
→ Backjumping, conflict-directed backjumping, etc.

Systematic search (IV) — ***Ordering heuristics***

Which variable to expand first?

Exp: $V_1, V_2,\ D_{V_1} = \{a, b, c, d\}, D_{V_2} = \{a, b\}$

Sol: $\{(V_1 = c), (V_2 = a)\}$ and $\{(V_1 = c), (V_2 = b)\}$

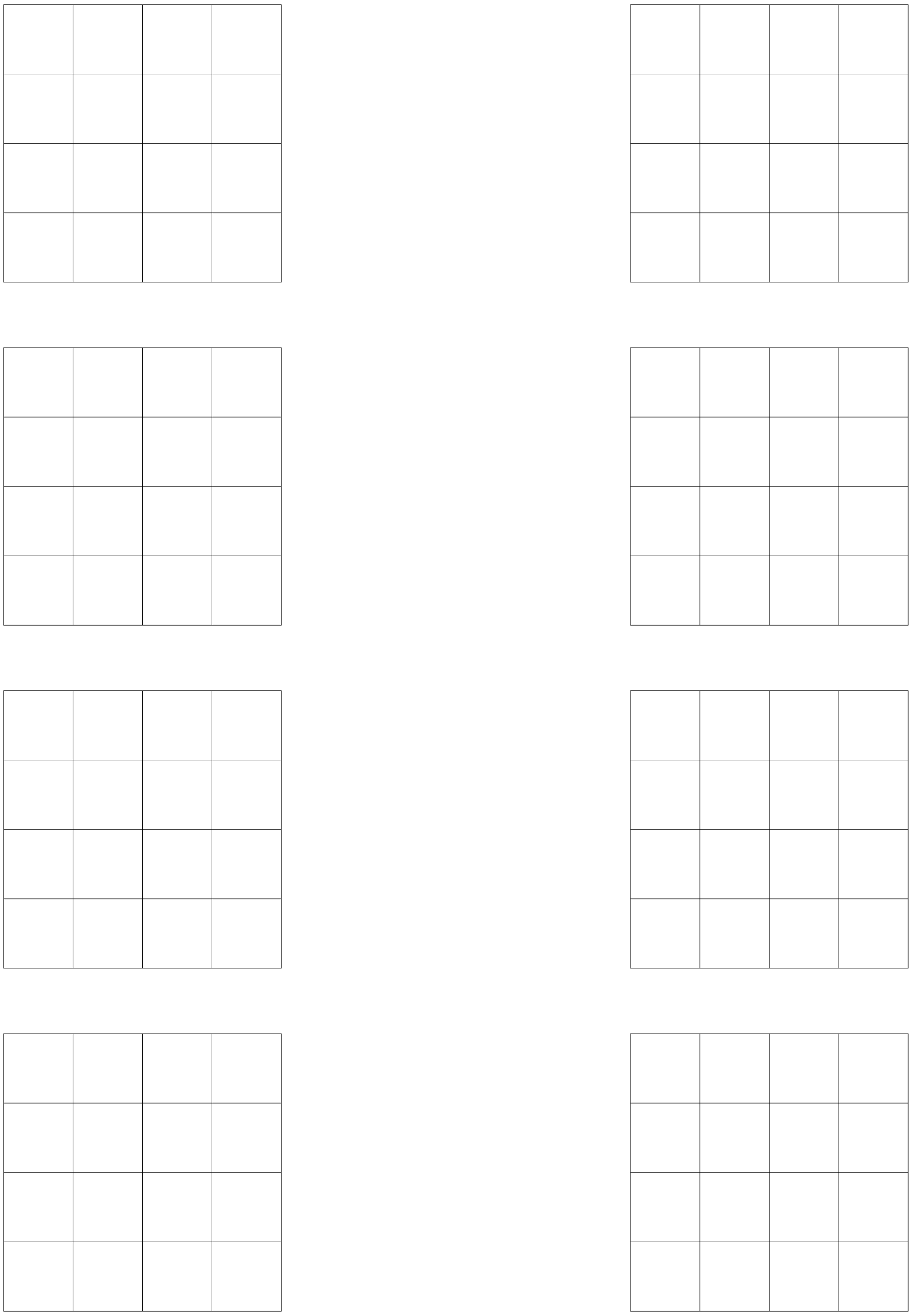


Heuristics:
- most <u>constrained</u> variable first (reduce branching factor)
- most <u>promising</u> value first (find quickly first solution)

Strategies for { variable ordering, value ordering } could be { static, dynamic }

Systematic search (V) ***Forward checking***

Why not filter right away effects of an action?



Backtrack search

- Constructive
- In general sound and complete
- Back-checking
- Backtracking: chronological vs. intelligent
- Ordering heuristics:
  static, dynamic variable, dymanic variable-value pairs,
- Looking ahead:
  Partial (Forward checking, FC), directional arc-consistency,
  Full (a.k.a Maintaining Arc-consistency, MAC),

CSP: a decision problem (NP-complete)

1- Modeling:
— abstraction and reformulation

2- Preprocessing techniques:
— eliminate non-acceptable tuples prior to search

3- Systematic search:
— potentially d^n paths of fixed lengths
— back-checking, backtracking, ordering heuristics, etc.

4- Search 'hybrids':
— Mixing consistency checking with search: look-ahead strategies

Non-systematic search

Iterative repair, local search

modifies a global but **inconsistent** solution

to decrease the number of violated constraints

Examples: Simulated annealing, hill climbing, taboo,

GSAT, WalkSAT, Genetic Algorithms, etc.

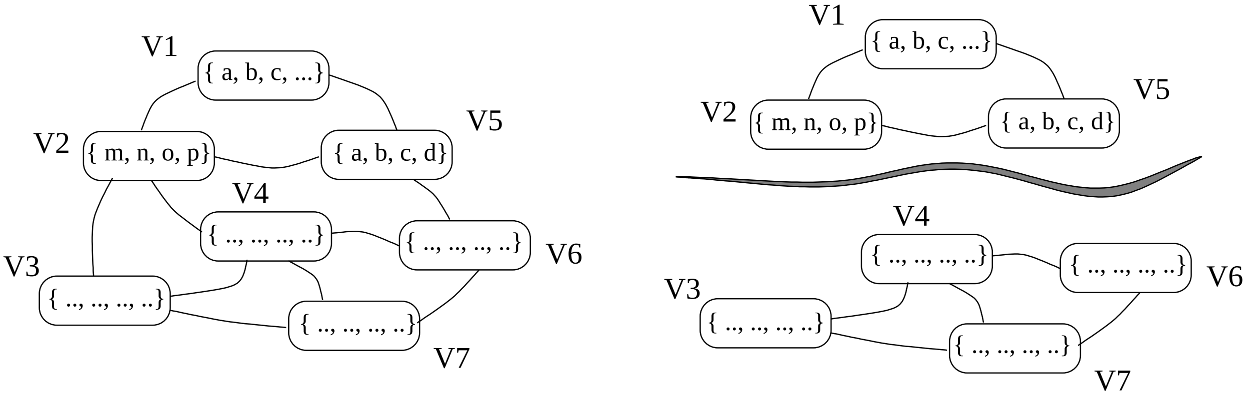
$\longrightarrow$ Incomplete & not sound

Advantage: anytime algorithm

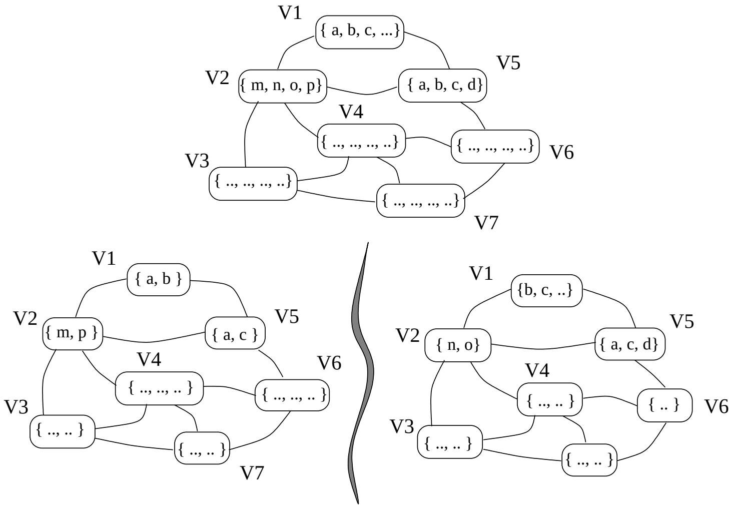
Shortcoming: Cannot tell that no solution exist

Decomposition $\begin{cases} \text{conjunctive} \\ \text{disjunctive} \\ \text{partition constraints (DB)} \end{cases}$

Conjunctive



Disjunctive



⟶ Properties of disjunctive decompositions

Properties of disjunctive decompositions
(ignore for now)

$$\mathcal{P} \longrightarrow \{\ \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_i\ \}$$

Conjunctive vs. disjunctive

Consistent	No constraint is removed
Simplifying	Size($\mathcal{P}_i$) $<$ Size($\mathcal{P}$)
Semi-complete	At least one solution is kept
Complete	No solution is lost
Redundant	Solutions replicated in $\{\mathcal{P}_i\}$
Reducible	$\sum_i$ Size($\mathcal{P}_i$) may be $<$ Size($\mathcal{P}$)

Deep analysis

Uncover particular properties, *e.g.*

- bound the required level of consistency (islands of tractability)
- predict ease/difficulty of solving a given instance

- **C-graph structure, topology**

  tree, DAGs, chordal, etc.

- **Constraints type**

  subsets of Allen's relations, all-diffs, functional,
  monotonic, row-convex, etc.

- **Order parameter** (phase transition)

Research directions

Preceding (i.e., search, backtrack, iterative repair, V/V/ordering, consistency checking, decomposition, symmetries & interchangeability, deep analysis) + ...

- **Evaluation of algorithms**:
  worst-case analysis, empirical studies
  random problems?
- **Cross-fertilization**:
  SAT, DB, math. programming,
  interval math, planning, etc.
- **Modeling & Reformulation**
- **Multi-agents**:
  Distribution and negotiation
  → decomposition & alliance formation

Multi-agent

- Problem:

    decomposed, replicated

- Agent:

    broker, solver, problem, solver+problem

- Architecture and authority:

    hierarchical, egalitarian

    priority-based (e.g., vote)

- Communication:

    agent-to-agent, group, broadcast

- Interaction strategies:

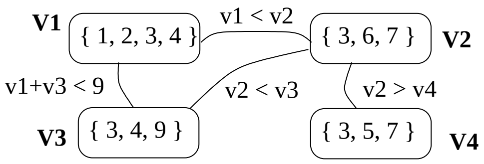
    cooperative, competitive

    transparent, secretive

CSP in a nutshell (I)

Definition: $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$

$$\begin{cases} \mathcal{V} = \{V_1, V_2, \ldots, V_n\} \\ \\ \mathcal{D} = \{D_{V_1}, D_{V_2}, \ldots, D_{V_n}\} \\ \\ \mathcal{C} \ \text{ set of constraints} \end{cases}$$



Constraint graph.
Finite domains.
Binary constraints, universal constraints.

Examples: map coloring, puzzles, resource allocation, temporal reasoning, product configuration, databases, spreadsheets, graphical layouts, graphical user-interfaces, bioinformatics, etc.

CSP in a nutshell (II)

Solution technique: Search
- constructive
- iterative repair

Enhancing search:
- intelligent backtrack
- variable/value ordering
- consistency checking
- forward-checking
- symmetries (interchangeability)
- decomposition

Deep analysis:
- graph structure
- constraints semantics
- phase transition

Open:
- k-ary constraints
- evaluation of algorithms
- cross-fertilization (mathematical program.)
- reformulation and approximation
- architectures (multi-agent, negotiation)