Constraint Processing

- Constraint Satisfaction: algorithms *Research+CSCE421/821*
- Constraint Languages: $\begin{cases} \text{functional} \\ \text{logic} \\ \text{object-oriented} \end{cases}$ *Visitor*

Berthe Y. Choueiry (Shu-we-ri)
Computer Science and Engineering
`choueiry@cse.unl.edu`, Tel: (402)472-5444

Outline

- Definition
- Practical usefulness
- Underlying 'science'
- Commercial successes & job market

What is this about?

Context: You are a senior in college

Problem: You need to register in 4 courses for Spring'2000

Possibilities: Many courses offered in Math, CSE, EE, *etc.*

Constraints: restrict the choices you can make

- *Unary:* Courses have prerequisites you have/don't have
  Courses/instructors you like/dislike
- *Binary:* Courses are scheduled at the same time
- *n-ary:* In CompEng, 4 courses from 5 tracks such as at least 3 tracks are covered

It is about **You**: you have choices, restricted by constraints

$\longrightarrow$ Make the right decisions

Constraint Satisfaction

Given

- A set of variables *4 courses at UNL*
- For each variable, a set of choices (values)
- A set of constraints that restrict the combinations of values the variables can take at the same time

Questions

- Does a solution exist? *classical decision problem*
- How two or more solutions differ? How to change specific choices without perturbing the solution?
- If there is no solution, what are the sources of conflicts? Which constraints should be retracted?
- *etc.*

Constraint Processing is about

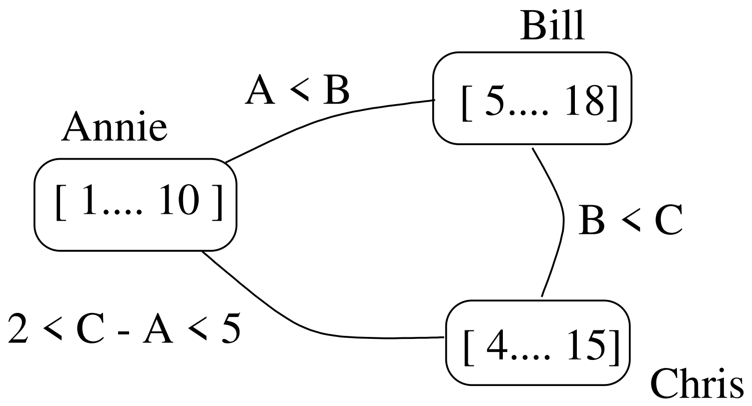
- solving a decision problem
- while allowing the user to state arbitrary constraints in an expressive way and
- providing concise and high-level feedback about alternatives and conflicts

Power of Constraints Processing

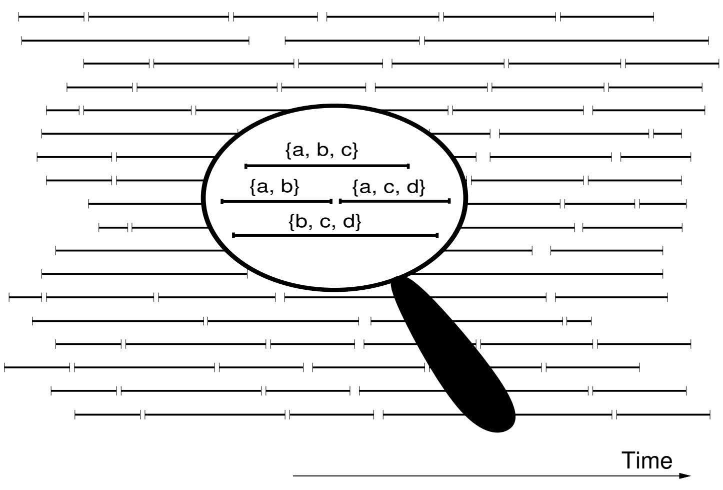
- flexibility & expressiveness of representations
- interactivity, users can $\left\{ \begin{array}{c} \text{relax} \\ \text{reinforce} \end{array} \right\}$ constraints

Related areas: AI, OR, Algorithmic, DB, Prog. Languages, *etc.*

Example II: Temporal reasoning



Example III: Resource Allocation



Practical Applications

Adapted from E.C. Freuder

- Radio resource management (RRM)
- Databases (computing joins, view updates)
- Temporal and spatial reasoning
- Planning, scheduling, resource allocation
- Design and configuration
- Graphics, visualization, interfaces
- Hardware verification and software eng.
- HCInteraction and decision support
- Molecular biology
- Robotics, machine vision and computational linguistics
- Transportation
- Qualitative and diagnostic reasoning

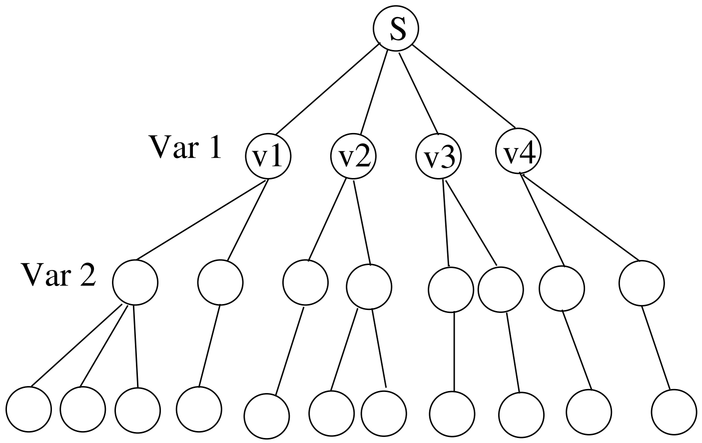
How to solve a CSP? Search $\begin{cases} \text{Constructive, systematic} \\ \text{Iterative repair} \end{cases}$

Systematic search:

$\rightarrow$ Starting from a root node

$\rightarrow$ Consider all values for a variable V_1

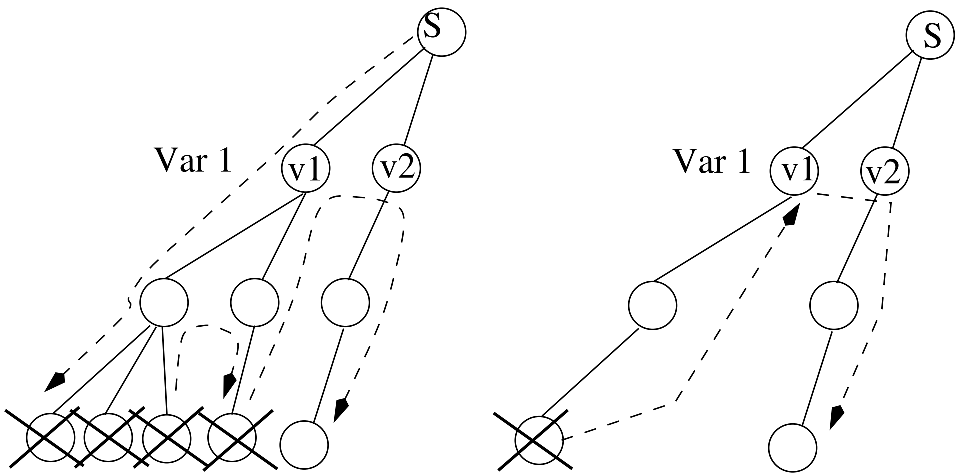
$\rightarrow$ For every value for V_1, consider all values for V_2, *etc.*



Systematic search (II) ***Intelligent backtracking***

- Intelligent pruning: back-checking, back-marking
- When dead-end, backtrack
- What if the reason for failure was higher up in the tree?

Backtrack to source of conflict!!

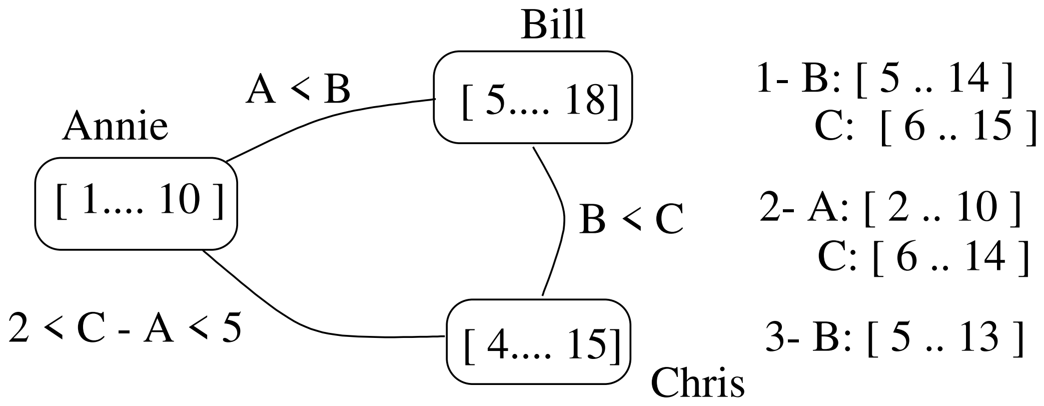


→ Backjumping, conflict-directed backtracking, etc.

Constraint checking (a.k.a. filtering, propagation)

Eliminate non-acceptable combinations of values prior to search

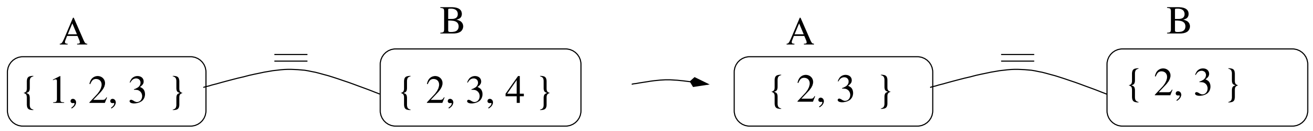
Arc-consistency:



→ reduces the size of the search space

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

Warning: arc-consistency does not solve the problem



(A=2) ∧ (B=3) still isn't a solution!

Non-systematic search

Iteration repair, local search

modifies a global but **inconsistent** solution

to decrease the number of violated constraints

Examples: Hill climbing, taboo search, simulated annealing,

GSAT, WalkSAT, Genetic Algorithms, etc.

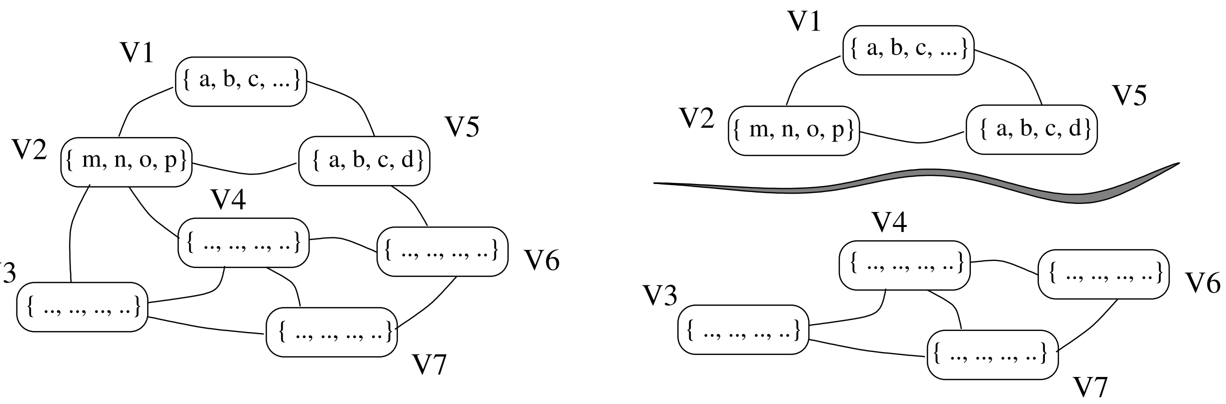
$\longrightarrow$ Incomplete & not sound

Advantage: anytime algorithm

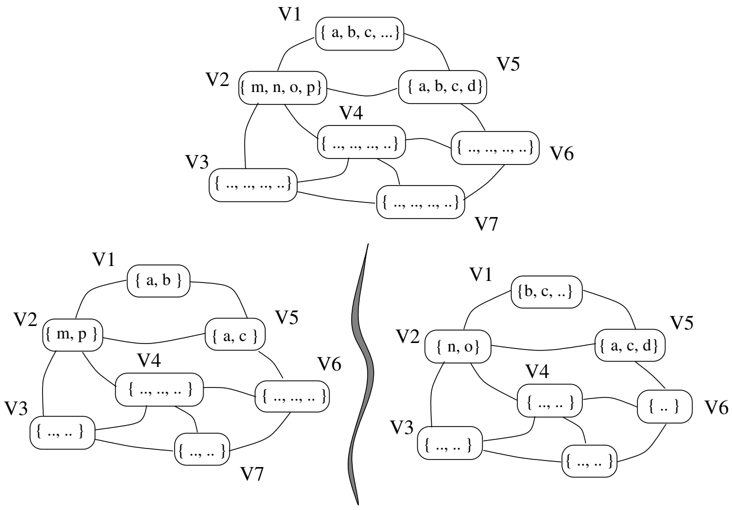
Shortcoming: Cannot tell that no solution exist

Decomposition
- conjunctive
- disjunctive

Conjunctive



Disjunctive



Deep analysis

Uncover particular properties, *e.g.*

- bounds the required level of consistency
- predicts ease/difficulty of solving a given instance

- **C-graph topology**
  tree, DAGs, chordal, etc.
- **Constraints type**
  subsets of Allen's relations,
  all-diffs, functional, monotonic, row-convex, etc.
- **Order parameter** (phase transition)

Multi agents

- **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:**
  cooperating vs. competing
  transparent vs. secretive
  negotiation + alliance formation

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 vs. empirical studies
random problems?

Cross-fertilization:

SAT, DB, mathematical programming,
interval mathematics, planning, etc.

Modeling & Reformulation

Multi agents:

Distribution and negotiation
$\rightarrow$ decomposition & alliance formation

CSP in a nutshell (I)

Solution technique: Search $\begin{cases} \text{constructive} \\ \text{iterative repair} \end{cases}$

Enhancing search: $\begin{cases} \text{intelligent backtrack} \\ \text{variable/value ordering} \\ \text{consistency checking} \\ \text{hybrid search} \\ \heartsuit \ \text{symmetries} \\ \heartsuit \ \text{decomposition} \end{cases}$

CSP in a nutshell (II)

Deep analysis: exploit problem structure
- ♡ graph topology
- ♡ constraint semantics
- phase transition

Research:
- k-ary constraints, soft constraints
- continuous vs. finite domains
- evaluation of algorithms (empirical)
- cross-fertilization (mathematical program.)
- ♡ reformulation and approximation
- ♡ architectures (multi-agent, negotiation)

Constraint Logic Programming (CLP)

A merger of

$\surd$ Constraint solving

$\longrightarrow$ Logic Programming, mostly Horn clauses (*e.g.*, Prolog)

Building blocks

- Constraint: primitives *but also user-defined*
  - cumulative/capacity (linear ineq), MUTEX, cycle, *etc.*
  - domain: Booleans, natural/rational/real numbers, finite
- Rules (declarative): a statement is a conjunction of constraints and is tested for satisfiability before execution proceeds further
- Mechanisms: satisfiability, entailment, delaying constraints

Your future: jobs

Commercial companies: I2 Technologies, Trilogy, Ilog, PeopleSoft/Red Pepper, Carmen Systems (Sweden), J.D. Edwards/Numetrix, COSYTEC, *etc.*

Prestigious research centers: NASA Ames, JPL, BT Labs (UK), IBM Watson+Alamaden, *etc.*

Start your own: Blue Pumpkin, Parc Technologies Ltd, Calico, Firepond, *etc.*

Academic: constraint languages, modeling, representation, automated reasoning, *etc.*

Constraint Processing Techniques are the basis of new languages:

Were you to ask me which programming paradigm is likely to gain most in commercial significance over the next 5 years I'd have to pick Constraint Logic Programming (CLP), even though it's perhaps currently one of the least known and understood. That's because CLP has the power to tackle those difficult combinatorial problems encountered for instance in job scheduling, timetabling, and routing which stretch conventional programming techniques beyond their breaking point. Though CLP is still the subject of intensive research, it's already being used by large corporations such as manufacturers Michelin and Dassault, the French railway authority SNCF, airlines Swissair, SAS and Cathay Pacific, and Hong Kong International Terminals, the world's largest privately-owned container terminal.

Byte, Dick Pountain