Constraint Satisfaction and Instantiation

A Constraint Satisfaction Problem is a triple consisting of

- a finite list $X_1, \ldots, X_n$ of variables
- finite sets $D_i$ of size $s_i$ for $1 \leq i \leq n$
- a finite set $\text{Con}$ of constraints that may or may not be satisfied by a node instantiating $X_i$ with a value in $D_i$ (search space size $\prod_{i=1}^{n} s_i$)

Goal satisfy all constraints in $\text{Con}$, instantiating variables if necessary/convenient

0. **Generate-and-test (brute force)** instantiate all variables before testing constraints

$$\text{genTest}(D_1, \ldots, D_n) :- \text{node}(X_1, \ldots, X_n, D_1, \ldots, D_n), \text{con}(X_1, \ldots, X_n).$$

$$\text{node}(X_1, \ldots, X_n, D_1, \ldots, D_n) :- \text{member}(X_1, D_1), \ldots, \text{member}(X_n, D_n).$$

Nodes are generated in lexicographic order (see zeb.pl) with no reason that the next node should fare any better than the previous.

1. **Incremental instantiation** (interleaving generation with testing)

node may map $X_i$ to $\?$, increasing search space size to $\prod_{i=1}^{n} (s_i + 1)$

Pay-off: many instantiations may be ruled out/in based on a single node

Given a node $N$ and a set $V$ of variables, let

- $\text{ivar}(N)$ be the set of variables that $N$ instantiates

$$\text{ivar}(N) := \{X_i \mid 1 \leq i \leq n \text{ and } N(X) \neq \?\}$$

- $\text{Con}_N$ be the set of constraints mentioning only variables in $\text{ivar}(N)$

$$\text{Con}_N := \{\gamma \in \text{Con} \mid \text{var}(\gamma) \subseteq \text{ivar}(N)\}.$$

where $\text{var}(\gamma)$ is the set of variables occurring in $\gamma$

- the restriction $N|V$ of $N$ to $V$ be the part of $N$ with variables in $V$

$$N|V := \{(X, N(X)) \mid X \in V\}.$$

Incremental instantiation is based on the assumption that when $\gamma \in \text{Con}_N$, $N$ satisfies $\gamma$ precisely if its restriction to $\text{var}(\gamma)$ does

$$N \models \gamma \iff N|\text{var}(\gamma) \models \gamma.$$
A node $N$ is **Con-live** if $N$ satisfies every constraint in $\text{Con}_N$.

**Idea:** consider only **Con-live** nodes, with start node $\emptyset$ (= $\text{ivar}(\emptyset)$), goal

$$\text{goal}(N) \iff \text{Con}_N = \text{Con} \text{ and } N \text{ is Con-live}$$

and arcs such that

$$\text{arc}(N, N', \text{Con}) \implies N \text{ is Con-live and } N' \neq N \text{ and } N =_{\text{ivar}(N)} N'$$

where for any set $V$ of variables,

$$N =_V N' \iff N \upharpoonright V = N' \upharpoonright V.$$

Assuming an arc adds exactly one instantiated variable, the heuristic

$$h(N) := \text{number of variables } N \text{ maps to } ?$$

is an under-estimate, with best-first yielding depth-first.

**N.B.** search tree with depth $\leq n$ and branching factor $\max\{s_1, \ldots, s_n\}$
(compared to generate-and-test: non-branching, depth $= \prod_{i=1}^n s_i$)

**Next:** we may test a constraint $\gamma$ against $N$ even if $\gamma \notin \text{Con}_N$; consider
$$\varphi \supset \psi \text{ or } \varphi \land \psi \text{ where } N \text{ satisfies } \neg \varphi. \text{ (Recall BDDs skip variables that don't matter, and simplify boolean expression when branching.)}$$

### 2. Refine notion of node

- keep track of remaining constraints-to-satisfy, simplifying Con (“constraint propagation”)
- possible values for variables via a predicate $\text{forbidden(Node,Var,Val)}$ (“forward checking”)

Sharpen $\text{arc}$ above with 2 heuristics for new variable-value pair $(X, a)$

- instantiate variable $X$ with Minimum Remaining Values (MRV)
  - fail-first (pessimistic) – minimize branching - see zebra notes
- choose $a$ to be the Least Constraining Value (LCV) via an updated binary constraint satisfaction graph – satisfy-first (optimistic)