Constraint satisfaction via generate-and-test

Constraints $\varphi_1, \varphi_2, \ldots, \varphi_m$ on variables $V_1, V_2, \ldots, V_n$

$$\text{goal}([V_1, V_2, \ldots, V_n]) :\sim \varphi_1, \varphi_2, \ldots, \varphi_m.$$ 

Caution: variable instantiation bugs (neg-as-fail), errors ($V_1 > V_2$)

Typically, a variable $V_i$ is required to take values from a list $L_i$

$$\text{node}([V_1, \ldots, V_n]) :\sim \text{member}(V_1, L_1), \ldots, \text{member}(V_n, L_n).$$

A simple algorithm

$$\text{generate-and-test} :\sim \text{node}(\text{Node}), \text{goal}(\text{Node}).$$

Two Exercises

1. Modify generate-and-test to return a goal node.

2. Formulate generate-and-test as a depth-first search

$$\text{search}([h_1, \ldots, h_n])$$

where $h_i$ is the head of $L_i$ (for $1 \leq i \leq n$), and

$$\text{search}(\text{Node}) :\sim \text{goal}(\text{Node}).$$

$$\text{search}(\text{Node}) :\sim \text{arc}(\text{Node}, \text{Next}), \text{search}(\text{Next}).$$
CSP heuristically (e.g. A-star)?

Try enumerating nodes in order of increasing $f$-value.
- Reorder $Li$’s in order of most promising values.
- Let

$$i(V_1, \ldots, V_n) := \text{the number of constraints satisfied by } [V_1, \ldots, V_n]$$

What’s wrong with defining

$$h([V_1, \ldots, V_n]) := m - i(V_1, \ldots, V_n)$$

where $m$ is the number of constraints to satisfy?

**Moral.** Mix generate node(Node) in with test goal(Node).

---

Node sets and un-instantiated values

A node is essentially a member of

$$\prod_{i=1}^{n} L_i \text{ with cardinality } \prod_{i=1}^{n} |L_i| \quad \text{— call it } P.$$ 

There are $2^P$ many sets of nodes.
Adjoin an “undefined” value ? to each list $L_i$, and redefine a node to allow variables to be undefined (i.e. instantiated to ?) — e.g.

$$\text{node-set}([1, ?, 3, ?]) = \{[1, i, 3, j] \mid i \in L_2 \text{ and } j \in L_4\}$$

Adjoining ? takes the number of nodes to

$$\prod_{i=1}^{n}(|L_i| + 1) \ll 2^P \text{ but still } \gg P.$$ 

**Idea.** Define arcs to incrementally replace ? satisfying some constraints.