Moving to a different formalism…

\[
\begin{align*}
\text{SEND} + \text{MORE} & \quad \text{-------} \\
& \quad \text{MONEY}
\end{align*}
\]

Consider state space for cryptarithmetic (e.g. DFS).

Is this (DFS) how humans tackle the problem?

**Human problem solving** appears more sophisticated!
For example, we derive new constraints on the fly.
→ little or no search!

---

**Constraint Satisfaction Problems (CSP)**

A powerful representation for (discrete) search problems

**A Constraint Satisfaction Problem (CSP)** is defined by:

- **X** is a set of n variables \(X_1, X_2, \ldots, X_n\) each defined by a finite domain \(D_1, D_2, \ldots, D_n\) of possible values.
- **C** is a set of constraints \(C_1, C_2, \ldots, C_m\). Each \(C_i\) involves a subset of the variables; specifies the allowable combinations of values for that subset.

A solution is an assignment of values to the variables that satisfies all constraints.

---

**Cryptarithmetic as a CSP**

**Problem:** \(TWO + TWO = FOUR\)

Variables:

- \(T \in \{0, \ldots, 9\}\)
- \(W \in \{0, \ldots, 9\}\)
- \(O \in \{0, \ldots, 9\}\)
- \(F \in \{0, \ldots, 9\}\)
- \(U \in \{0, \ldots, 9\}\)
- \(R \in \{0, \ldots, 9\}\)

**Constraints:**

- \(O + O = R + 10 \times X_1\)
- \(X_1 + W + W = U + 10 \times X_2\)
- \(X_1 + T + T = O + 10 \times X_3\)
- \(X_1 = F\)

Each letter has a different digit \((F \neq T, F \neq U, \text{etc.})\);
Types of Constraints

**Unary Constraints:**
Restriction on single variable

**Binary Constraints:**
Restriction on pairs of variables

**Higher-Order Constraints:**
Restriction on more than two variables

Constraint Satisfaction Problems (CSP)

For a given CSP the problem is one of the following:

1. find all solutions
2. find one solution
   - just a feasible solution, or
   - A “reasonably good” feasible solution, or
   - the optimal solution given an objective function
3. determine if a solution exists

How to View a CSP as a Search Problem?

**Initial State** - state in which all the variables are unassigned.

**Successor function** - assign a value to a variable from a set of possible values.

**Goal test** - check if all the variables are assigned and all the constraints are satisfied.

**Path cost** - assumes constant cost for each step

Branching Factor

**Approach 1** - any unassigned variable at a given state can be assigned a value by an operator: branching factor as high as sum of size of all domains.

**Approach 2** - since order of variable assignment not relevant, consider as the successors of a node just the different values of a single unassigned variable: max branching factor = max size of domain.

**Maximum Depth of Search Tree**

\[ n \text{ the number of variables } \rightarrow \text{ all solutions at depth } n. \]

Prefer DFS or BFS?

CSP – Goal Decomposed into Constraints

**Backtracking Search:** a DFS that
- chooses values for variables one at a time
- checks for consistency with the constraints.

**Decisions during search:**
- Which variable to choose next for assignment.
- Which value to choose next for the variable.

Forward Checking

**Idea:** Reduce domain of unassigned variables based on assigned variables.

- Each time variable is instantiated, delete from domains of the uninstantiated variables all of those values that conflict with current variable assignment.
- Identify dead ends without having to try them via backtracking.
General Purpose Heuristics

Variable and value ordering:

**Minimum remaining values (MRV):** choose the variable with the fewest possible values.

**Degree heuristic:** assign a value to the variable that is involved in the largest number of constraints on other unassigned variables.

**Least-constraining value heuristic:** choose a value that rules out the smallest number of values in variables connected to the current variable by constraints.

Comparison of CSP Algorithms

<table>
<thead>
<tr>
<th>Problem</th>
<th>BT</th>
<th>BT+MRV</th>
<th>BT+FC</th>
<th>BT+FC+MRV</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>(&gt;1,000K)</td>
<td>(&gt;1,000K)</td>
<td>2K</td>
<td>60</td>
</tr>
<tr>
<td>N-queens</td>
<td>(&gt;40,000K)</td>
<td>13,500K</td>
<td>(&gt;40,000K)</td>
<td>817K</td>
</tr>
</tbody>
</table>

Constraint Propagation (Arc Consistency)

**Arc Consistency** - state is arc-consistent, if every variable has some value that is consistent with each of its constraints (consider pairs of variables)

- **Init:** \( Q \) is queue with all (directed) arcs \((X_i, X_j)\) in CSP
- **While** \( Q \) is not empty
  - \((X_i, X_j) = \text{remove}\_first(Q)\)
  - FOR EACH \( x \in \text{dom}(X_i) \)
    - **If** no \( y \in \text{dom}(X_j) \) satisfies constraint \((X_i, X_j)\)
      - **Then** remove \( x \) from \( \text{dom}(X_i) \)
    - **If** \( \text{dom}(X_i) \) changed
      - **Then** add all arcs \((X_i, X_j) \in Q\) to \( Q \)

Constraint Propagation (K-Consistency)

- **K-Consistency** generalizes arc-consistency (2-consistency).
- Consistency of groups of K variables.

Local Search for CSPs

Remarks

- Infinite discrete domains and continuous domains
- Exploiting special problem structure
- Dramatic recent progress in Constraint Satisfaction. Methods can now handle problems with 10,000 to 100,000 variables, and up to 1,000,000 constraints.