Constraint Satisfaction Problem \([\text{Var, Dom, Con}]\)

- a list \(\text{Var} = [X_1, \ldots, X_n]\) of variables \(X_i\)
- a list \(\text{Dom} = [D_1, \ldots, D_n]\) of finite sets \(D_i\) of size \(s_i\)
- 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 \(\prod_{i=1}^{n} s_i\))

E.g. SAT: \(D_i = \{0, 1\}\), \(s_i = 2\) for search space of size 2

Problem: satisfy all constraints in \(\text{Con}\), instantiating variables if necessary/convenient

\[
\begin{align*}
\text{?- X} = \text{Y, X=a, Y=b.} \\
\text{?- X=a, Y=b, X} = \text{Y.}
\end{align*}
\]

\(\text{no X = a, Y = b}\)
Constraint Satisfaction Problem [Var, Dom, Con]

- a list \( \text{Var} = [X_1, \ldots, X_n] \) of variables \( X_i \)
- a list \( \text{Dom} = [D_1, \ldots, D_n] \) of finite sets \( D_i \) of size \( s_i \)
- 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 \( \prod_{i=1}^{n} s_i \))

E.g. SAT: \( D_i = \{0, 1\} \), \( s_i = 2 \) for search space of size \( 2^n \)
Constraint Satisfaction Problem [Var, Dom, Con]

- a list Var = [X_1, \ldots, X_n] of variables X_i
- a list Dom = [D_1, \ldots, D_n] of finite sets D_i of size s_i
- a finite set 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)

E.g. SAT: \( D_i = \{0, 1\}, \ s_i = 2 \) for search space of size \( 2^n \)

Problem: satisfy all constraints in Con, instantiating variables if necessary/convenient
Constraint Satisfaction Problem [Var, Dom, Con]

- a list \( \text{Var} = [X_1, \ldots, X_n] \) of variables \( X_i \)
- a list \( \text{Dom} = [D_1, \ldots, D_n] \) of finite sets \( D_i \) of size \( s_i \)
- 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 \( \prod_{i=1}^{n} s_i \))

E.g. SAT: \( D_i = \{0, 1\}, \ s_i = 2 \) for search space of size \( 2^n \)

Problem: satisfy all constraints in \( \text{Con} \), instantiating variables if necessary/convenient

\[
\text{?- } X\neq Y, \ X=a, \ Y=b. \\
\text{no}
\]
Constraint Satisfaction Problem [Var, Dom, Con]

- a list \( \text{Var} = [X_1, \ldots, X_n] \) of variables \( X_i \)
- a list \( \text{Dom} = [D_1, \ldots, D_n] \) of finite sets \( D_i \) of size \( s_i \)
- 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 \( \prod_{i=1}^n s_i \))

E.g. SAT: \( D_i = \{0, 1\}, \ s_i = 2 \) for search space of size \( 2^n \)

Problem: satisfy all constraints in \( \text{Con} \), instantiating variables if necessary/convenient

\[
\begin{align*}
\text{no} & \quad \text{| } \neg X \leftarrow Y, \ X = a, \ Y = b. \\
X = a, \ Y = b & \quad \text{| } \neg X = a, \ Y = b, \ X \leftarrow Y.
\end{align*}
\]
Order-independent unification (Martelli-Montanari)

**Input:** set $\mathcal{E}$ of pairs $[t, t']$

**Output:** substitution $[[X_1, t_1], \ldots, [X_k, t_k]]$ unifying pairs in $\mathcal{E}$
Order-independent unification (Martelli-Montanari)

**Input:** set $\mathcal{E}$ of pairs $[t, t']$

**Output:** substitution $[[X_1, t_1], \ldots, [X_k, t_k]]$ unifying pairs in $\mathcal{E}$

Simplify $\mathcal{E}$ non-deterministically until no longer possible

1. $[f(s_1, \ldots, s_k), f(t_1, \ldots, t_k)]$ (allowing $k = 0$)
   \[\implies\text{replace by pairs } [s_1, t_1], \ldots, [s_k, t_k]\]

2. $[f(s_1, \ldots, s_k), g(t_1, \ldots, t_m)]$ where $f \neq g$ or $k \neq m$
   \[\implies\text{halt with failure}\]

3. $[X, X] \implies\text{delete}$

4. $[t, X]$ where $t$ is not a var \[\implies\text{replace by } [X, t]\]

5. $[X, t]$ where $X \notin \text{Var}(t)$ and $X$ occurs elsewhere
   \[\implies\text{apply } [X, t] \text{ to all other pairs}\]

6. $[X, t]$ where $X \in \text{Var}(t)$ and $X \neq t \implies\text{halt with failure}$

N.B. Prolog omits $X \in \text{Var}(t)$ in 5, 6 for speed-up.
Order-independent unification (Martelli-Montanari)

**Input:** set $\mathcal{E}$ of pairs $[t, t']$

**Output:** substitution $[[X_1, t_1], \ldots, [X_k, t_k]]$ unifying pairs in $\mathcal{E}$

Simplify $\mathcal{E}$ non-deterministically until no longer possible

1. $[f(s_1, \ldots, s_k), f(t_1, \ldots, t_k)]$ (allowing $k = 0$)
   $\implies$ replace by pairs $[s_1, t_1], \ldots, [s_k, t_k]$

2. $[f(s_1, \ldots, s_k), g(t_1, \ldots, t_m)]$ where $f \neq g$ or $k \neq m$
   $\implies$ halt with failure

3. $[X, X] \implies$ delete

4. $[t, X]$ where $t$ is not a var $\implies$ replace by $[X, t]$

5. $[X, t]$ where $X \not\in \text{Var}(t)$ and $X$ occurs elsewhere
   $\implies$ apply $[X, t]$ to all other pairs

6. $[X, t]$ where $X \in \text{Var}(t)$ and $X \neq t$ $\implies$ halt with failure

**N.B.** Prolog omits *occurs check* $X \in \text{Var}(t)$ in 5, 6 for speed-up
Instantiate before negating (as failure)

% \+p :- (p,!,fail); true.

\begin{verbatim}
\begin{verbatim}
p(X) :- \+q(X), r(X).
q(a).  q(b).
r(a).  r(c).
\end{verbatim}
\end{verbatim}

\begin{verbatim}
\begin{verbatim}
\end{verbatim}
\end{verbatim}
\end{verbatim}

\begin{verbatim}
\begin{verbatim}
l ?- p(X).  % contra ?- p(c).
\end{verbatim}
\end{verbatim}
Instantiate before negating (as failure)

\% \(+p \ :- \ (p,!,fail); \ true.\)

\p(X) \ :- \ \(+q(X), \ r(X).\)
\q(a). \ q(b).
\ r(a). \ r(c).
-------------------
| \ ?- \ p(X). \ \% contra \ ?- \ p(c). |

\p(X) \ :- \ \(+q(X).\)
\q(X) \ :- \ \(+r(X).\)
\ r(c).
-------------------
| \ ?- \ p(X). \ \% contra \ ?- \ p(a). |
Generate-and-test

brute force: instantiate all variables before testing constraints

\[
\text{genTest}(D_1\ldots D_n) :- \text{node}(X_1\ldots X_n, D_1\ldots D_n), \\
\text{constraint}(X_1\ldots X_n).
\]

\[
\text{node}(X_1\ldots X_n, D_1\ldots D_n) :- \text{member}(X_1, D_1), \ldots, \\
\text{member}(X_n, D_n).
\]
Generate-and-test

brute force: instantiate all variables before testing constraints

generate(D1...Dn) :- node(X1...Xn,D1...Dn),
                    constraint(X1...Xn).

node(X1...Xn,D1...Dn) :- member(X1,D1),...,
                      member(Xn,Dn).

For each of the $\prod_{i=1}^{n} s_i$-choices of X1...Xn such that

node(X1...Xn,D1...Dn)

(with Di of size si), assume

constraint(X1...Xn)

can be checked within a polynomial of X1...Xn.
**Generate-and-test**

brute force: instantiate all variables before testing constraints

\[
\text{genTest}(D_1\ldots D_n) :- \text{node}(X_1\ldots X_n, D_1\ldots D_n), \\
\quad \text{constraint}(X_1\ldots X_n).
\]

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

For each of the $\prod_{i=1}^{n} s_i$-choices of $X_1\ldots X_n$ such that

\[
\text{node}(X_1\ldots X_n, D_1\ldots D_n)
\]

(with $D_i$ of size $s_i$), assume

\[
\text{constraint}(X_1\ldots X_n)
\]

can be checked within a polynomial of $X_1\ldots X_n$.

Nodes are generated in lexicographic order without regard to constraints.
Inferring changes

Horn-SAT by minimal changes to 00 · · · 0 (all variables 0/false)

<table>
<thead>
<tr>
<th>CSAT</th>
<th>definite clause</th>
<th>list encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \overline{u} \lor x \lor \overline{z} )</td>
<td>( x :\neq u, z ).</td>
<td>([x, u, z] )</td>
</tr>
<tr>
<td>( \overline{u} \lor \overline{v} )</td>
<td>( \text{false} :\neq u, v ).</td>
<td>([\text{false}, u, v] )</td>
</tr>
</tbody>
</table>
Inferring changes

Horn-SAT by minimal changes to 00⋯0 (all variables 0/false)

<table>
<thead>
<tr>
<th>CSAT</th>
<th>definite clause</th>
<th>list encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\overline{u} \lor x \lor \overline{z}$</td>
<td>$x :- u, z$.</td>
<td>$[x, u, z]$</td>
</tr>
<tr>
<td>$\overline{u} \lor \overline{v}$</td>
<td>$false :- u, v$.</td>
<td>$[false, u, v]$</td>
</tr>
</tbody>
</table>

For each stage $i$, collect the variables set at stage $i$ to 1/true in $A_i$

- $A_0 := \emptyset$ (all variables false)
- $A_{i+1} := \{x | \text{member([}x|T], KB) \text{ and all(}T, A_i)\}$
- $x :- t_1 \ldots t_k$ in $KB$ \quad $\{t_1 \ldots t_k\} \subseteq A_i$

check: $false \not\in A_n$
Inferring changes

Horn-SAT by minimal changes to 00 \cdots 0 (all variables 0/false)

<table>
<thead>
<tr>
<th>CSAT</th>
<th>definite clause</th>
<th>list encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\overline{u} \lor x \lor \overline{z})</td>
<td>(x :\overline{u}, z).</td>
<td>([x, u, z])</td>
</tr>
<tr>
<td>(\overline{u} \lor \overline{v})</td>
<td>(\text{false} :\overline{u}, v).</td>
<td>([\text{false}, u, v])</td>
</tr>
</tbody>
</table>

For each stage \(i\), collect the variables set at stage \(i\) to 1/true in \(A_i\)

\[
A_0 := \emptyset \quad \text{(all variables false)}
\]

\[
A_{i+1} := \{x \mid \text{member(}[x|T], KB) \text{ and all(}T, A_i)\}
\]

\[
x : t_1 \ldots t_k \text{ in } KB \quad \{t_1 \ldots t_k\} \subseteq A_i
\]

check: \(\text{false} \not\in A_n\)

No minimal set for non-Horn \(x \lor y\) (or xor).
Instantiate one variable at a time

allow node to map $X_i$ to $\text{?}$, raising search space size from

$$\prod_{i=1}^{n} s_i \text{ to } \prod_{i=1}^{n} (s_i + 1) \text{ from adding } \text{?} \text{ to } D_i$$

Pay-off: search tree of depth $n$ and branching factor $\max_i s_i$ with start node instantiating no variable, and an arc instantiating least uninstantiated variable
Instantiate one variable at a time

allow node to map $X_i$ to $?$, raising search space size from

$$\prod_{i=1}^{n} s_i \text{ to } \prod_{i=1}^{n} (s_i + 1)$$

from adding $?$ to $D_i$

**Pay-off:** search tree of depth $n$ and branching factor $\max_i s_i$

with start node instantiating no variable, and

an arc instantiating least uninstantiated variable

E.g. $n = 2$, $D_1 = D_2 = \{a, b\}$

$$X_1 = ?, X_2 = ?$$

$$X_1 = a, X_2 = ?$$

$$X_1 = b, X_2 = ?$$

$$X_1 = a, X_2 = a$$

$$X_1 = a, X_2 = b$$

$$X_1 = b, X_2 = a$$

$$X_1 = b, X_2 = b$$
Interleave generation with testing, backtracking asap whenever \(\text{arc}(N_0, N_1)\),

- \(N_1\) instantiates one more variable than \(N_0\), and
- \(N_1\) satisfies every constraint on instantiated variables
Interleave generation with testing, backtracking asap whenever $arc(N0, N1)$,

$N1$ instantiates one more variable than $N0$, and $N1$ satisfies every constraint on instantiated variables.

Further optimizations (illustrated by 3-Color)

- MRV: instantiate variable with minimum remaining values (to minimize branching/cases)
- LCV: assign value that is least constraining (for greatest chance of success)
Interleave generation with testing, backtracking asap whenever \( \text{arc}(N0, N1) \),

\( N1 \) instantiates one more variable than \( N0 \), and
\( N1 \) satisfies every constraint on instantiated variables

Further optimizations (illustrated by 3-Color)
- MRV: instantiate variable with minimum remaining values (to minimize branching/cases)
- LCV: assign value that is least constraining (for greatest chance of success)
- reduce domains of un-instantiated variables via constraints

**Constraint Graph:** node = variable

\[
\text{arc}(X_i, X_j) \iff \text{Con}[X_i, X_j] \neq \emptyset
\]

*arc consistency:* for \( \text{arc}(X_i, X_j) \) and \( i < j \),

\[
(\forall d \in D(X_i))(\exists d' \in D(X_j)) \ d, d' \models \text{Con}[X_i, X_j]
\]
removing \( d \) from \( D(X_i) \) when no such \( d' \) exists