Consistency

II
Bounds Consistency

A primitive constraint $c$ is hyper-arc consistent with domain $D$ if for each variable $x \in \text{vars}(c)$ and domain assignment $d \in D(x)$, there is an assignment to the remaining variables in $c$ which is a solution to $c$.

Expensive to check.
• A CSP is *arithmetic* if each variable in the CSP ranges over a finite domain of integers and the primitive constraints are arithmetic constraints.

• A *range* \([l..u]\) represents the set of integers \(\{l, l + 1, \ldots, u\}\) if \(l \leq u\), otherwise it represents the empty set.
Bounds Consistency

A arithmetic primitive constraint $c$ is **bound consistent** with domain $D$ if for each $x \in \text{vars}(C)$, there is:

- an assignment $x \mapsto \min_D(x)$ and of real numbers to the remaining variables in $c$ such that

$$\min_D(x_j) \leq d_j \leq \max_D(x_j)$$

for each $d_j$ and which is a solution of $c$ and
Bounds Consistency Continued

- another assignment \( x \mapsto \max_D(x) \) and of real numbers to the remaining variables in \( c \) such that

\[
\min_D(x_j) \leq d_j \leq \max_D(x_j)
\]

for each \( d_j \) and which is a solution of \( c \)
Example

\[ X = 3Y + 5Z \]

with domain \( D \), where

\[ D(X) = [2..7], \; D(Y) = [0..2], \; D(Z) = [-1..2] \]

Applying bounds consistency yields:

\[ D(X) = [2..7], \; D(Y) = [0..2], \; D(Z) = [0..1] \]
Propagation rules for bounds consistency

Rewrite $X = Y + Z$ as

$$X = Y + Z, Y = X - Z, Z = X - Y$$

then we get

$$X \geq \min_D(Y) + \min_D(Z), X \leq \max_D(Y) + \max_D(Z)$$

$$Y \geq \min_D(X) - \max_D(Z), Y \leq \max_D(X) - \min_D(Z)$$

$$Z \geq \min_D(X) - \max_D(Y), Z \leq \max_D(X) - \min_D(Y)$$

Typical replacement rule

$$X_{\text{min}} := \max\{\min_D(X), \min_D(Y) + \min_D(Z)\}$$
bounds_consistent(C,D)
    let C be of the form $c_1 \land \ldots \land c_n$
    $C_0 := \{c_1, \ldots, c_n\}$
    while $C_0 \neq \emptyset$ do
        choose $c \in C_0$
        $C_0 := C_0 \setminus \{c\}$
        $D_1 := \text{bounds_consistent_primitive}(c, D)$
        if $D_1$ is a false domain
            then return $D_1$ endif
        for $i := 1$ to $n$ do
            if there exists $x \in \text{vars}(c_i)$
                such that $D_1(x) \neq D(x)$ then
                $C_0 := C_0 \cup \{c_i\}$
            endif
        endfor
        $D := D_1$
    endwhile
    return $D$
All different

alldifferent_consistent_primitive(c, D)
  let c be of the form alldifferent(c, D)
  while ∃v ∈ V with D(v) = {d} for some d
    V := V \ {v}
    for each v' ∈ V
      D(v') := D(v') \ {d}
  endfor
endwhile

nv := |V| | |
r := Ø
for each v ∈ V
  r := r ∪ D(v)
endfor
if nv > |r| then return false endif
return D
The cumulative Constraint

cumulative([S1, ..., Sm],
[D1, ..., Dm],
[R1, ..., Rm],
L)

$m$ tasks with start times $S1, ..., Sm$
of duration $D1, ..., Dm$ and
requiring $R1, ..., Rm$ units of a single resource
the maximum amount of the resource available
is $L$.

GnuProlog does not support this. (Exercise, write it!)
Some other global constraints

element(I, [V1, ..., Vm], X)

If $I = i$ then $X = Vi$.

In GnuProlog this is fd_element_var(I, List, X)

GnuProlog also supports:

fd_atmost(N, List, V)
fd_atleast(N, List, V)
fd_exactly(N, List, V)