Simplification

Optimization
Constraint Simplification

- Constraint *implication* in terms of solution sets.

- *Redundant* constraints

- *Simplification* removes redundant constraints
Projection

- *Extension* of valuations

- *Partial solution*, a valuation that can be extended to a solution.

- *Projection* of a constraint.
Examples of projection

The constraints:

\[ X + Y \leq 1 \land X - Y \leq 1 \]

\[-X + Y \leq 1 \land -X - Y \leq 1 \]

projects on \( X \) to the constraint

\[-1 \leq X \leq +1 \]
Fourier’s Algorithm

fourier_eliminate($C, y$)

let $C^0$ be the inequalities of $C$ not involving $y$

let $C^+$ be those inequalities that can be written $t \leq y$

let $C^-$ be those inequalities that can be written $y \leq t$
for each $t_1 \leq y \in C^+$
  
  for each $y \leq t_2 \in C^-$
  
  $C^0 := C^0 \cup \{t_1 \leq t_2\}$

endfor

endfor

return $C^0$
Constraint Simplifiers

• *Equivalence* of constraints, defined in terms of partial solutions.

• A simplifier for a constraint domain \( \mathcal{D} \) and set of variables \( V \) takes a constraint \( C_1 \) and returns a constraint \( C_2 \), such that \( C_1 \) and \( C_2 \) are equivalent in \( \mathcal{D} \) with respect to \( V \).
A tree constraint simplifier

tree_simplify(C, V)

\[
C_1 := \text{unify}(C)
\]

\[
\text{if } C_1 = false \text{ then return false}
\]

\[
\text{endif}
\]

\[
S := true
\]
while $C_1$ is of the form
\[ x = t \land C_1 \]
\[ C_1 := C_2 \]
if $x \in V$ then
  if $t$ is a variable and $t \notin X$ then
    substitute $x$ for $t$ in $C_1$
    and $S$
  else
    $S := S \land x = t$
  endif
endif
endif
endwhile
return $S$
Properties of simplifiers

• Projecting

\[ \text{vars}(\text{simpl}(C, V)) \subseteq V \]

• redundancy-free
Properties of simplifiers II

• Weakly projecting

\[ | \text{vars}(\text{simpl}(C_1, V)) - V | \leq | \text{vars}(C_2) - V | \]

for all constraints \( C_2 \) such that \( C_1 \) and \( C_2 \) are equivalent with respect to \( V \).