Some Observations on Boolean Logic and Optimization

John Hooker
Carnegie Mellon University

January 2009

Outline

- Logic and cutting planes
- Logic of 0-1 inequalities
- Logic and linear programming
- Inference duality
- Constraint programming
- Good logic models

Logic and Cutting Planes

Logic and cutting planes

Theorem (Quine). The resolution method generates all **prime implicates** of a set of logical clauses.

Prime implicates = undominated implications.

Logic and cutting planes

Theorem (Quine). The resolution method generates all **prime implicates** of a set of logical clauses.

Prime implicates = undominated implications.

This means that resolution is a complete inference method for clauses.

Logic and cutting planes

Theorem (Quine). The resolution method generates all **prime implicates** of a set of logical clauses.

$$x_1 \vee \bar{x}_2$$
$$\bar{x}_1 \vee x_3$$
$$x_2 \vee x_3$$

Example

Logic and cutting planes

Theorem (Quine). The resolution method generates all **prime implicates** of a set of logical clauses.

$$x_1 \vee \bar{x}_2$$
$$\bar{x}_1 \vee x_3$$
$$x_2 \vee x_3$$

Resolve on x_1

Logic and cutting planes

Theorem (Quine). The resolution method generates all **prime implicates** of a set of logical clauses.

$$x_1 \vee \bar{x}_2$$
$$\bar{x}_1 \vee x_3$$
$$x_2 \vee x_3$$

$$x_1 \vee \bar{x}_2$$
$$\bar{x}_1 \vee x_3$$
$$x_2 \vee x_3$$
$$\bar{x}_2 \vee x_3$$

Resolve on x_1

Logic and cutting planes

Theorem (Quine). The resolution method generates all **prime implicates** of a set of logical clauses.

$$x_1 \vee \bar{x}_2$$
$$\bar{x}_1 \vee x_3$$
$$x_2 \vee x_3$$

$$x_1 \vee \bar{x}_2$$
$$\bar{x}_1 \vee x_3$$
$$x_2 \vee x_3$$
$$\bar{x}_2 \vee x_3$$

Resolve on x_2

Logic and cutting planes

Theorem (Quine). The resolution method generates all **prime implicates** of a set of logical clauses.

$$\begin{array}{l} x_1 \vee \bar{x}_2 \\ \bar{x}_1 \vee x_3 \\ x_2 \vee x_3 \end{array} \qquad \begin{array}{l} x_1 \vee \bar{x}_2 \\ \bar{x}_1 \vee x_3 \\ \boxed{\begin{array}{l} x_2 \vee x_3 \\ \bar{x}_2 \vee x_3 \end{array}} \end{array} \qquad \begin{array}{l} x_1 \vee \bar{x}_2 \\ \bar{x}_1 \vee x_3 \\ x_2 \vee x_3 \\ \bar{x}_2 \vee x_3 \\ x_3 \end{array}$$

Resolve on x_2

Logic and cutting planes

Theorem (Quine). The resolution method generates all **prime implicates** of a set of logical clauses.

$$
\begin{array}{lll}
x_1 \vee \bar{x}_2 & & \\
\bar{x}_1 & & \vee x_3 \\
 & x_2 & \vee x_3
\end{array}
\qquad
\begin{array}{lll}
x_1 \vee \bar{x}_2 & & \\
\bar{x}_1 & & \vee x_3 \\
 & x_2 & \vee x_3 \\
 & \bar{x}_2 & \vee x_3
\end{array}
\qquad
\begin{array}{lll}
x_1 \vee \bar{x}_2 & & \\
\bar{x}_1 & & \vee x_3 \\
 & x_2 & \vee x_3 \\
 & \bar{x}_2 & \vee x_3 \\
 & & x_3
\end{array}
$$

Drop redundant clauses

Logic and cutting planes

Theorem (Quine). The resolution method generates all **prime implicates** of a set of logical clauses.

$x_1 \vee \bar{x}_2$	$x_1 \vee \bar{x}_2$	$x_1 \vee \bar{x}_2$	$x_1 \vee \bar{x}_2$
$\bar{x}_1 \vee x_3$	$\bar{x}_1 \vee x_3$	$\bar{x}_1 \vee x_3$	
$x_2 \vee x_3$	$x_2 \vee x_3$	$x_2 \vee x_3$	
	$\bar{x}_2 \vee x_3$	$\bar{x}_2 \vee x_3$	
		x_3	x_3
		Drop redundant clauses	

Logic and cutting planes

Theorem (Quine). The resolution method generates all **prime implicates** of a set of logical clauses.

$$
\begin{aligned}
&x_1 \vee \bar{x}_2 \\
&\bar{x}_1 \vee x_3 \\
&x_2 \vee x_3
\end{aligned}
$$

$$
\begin{aligned}
&x_1 \vee \bar{x}_2 \\
&\bar{x}_1 \vee x_3 \\
&x_2 \vee x_3 \\
&\bar{x}_2 \vee x_3
\end{aligned}
$$

$$
\begin{aligned}
&x_1 \vee \bar{x}_2 \\
&\bar{x}_1 \vee x_3 \\
&x_2 \vee x_3 \\
&\bar{x}_2 \vee x_3 \\
&x_3
\end{aligned}
$$

$$
\begin{aligned}
&x_1 \vee \bar{x}_2 \\
&x_3
\end{aligned}
$$

Prime implicates remain

Logic and cutting planes

Theorem (Chvátal). Every cutting plane for a 0-1 system $Ax \geq b$ can be generated by repeatedly taking nonnegative linear combinations and rounding up.

This might be regarded as the fundamental theorem of cutting plane theory.

Logic and cutting planes

Theorem (Chvátal). Every cutting plane for a 0-1 system $Ax \geq b$ can be generated by repeatedly taking nonnegative linear combinations and rounding up.

This might be regarded as the fundamental theorem of cutting plane theory.

A key step of the proof uses the **resolution method**.

Logic and cutting planes

Theorem (Chvátal). Every cutting plane for a 0-1 system $Ax \geq b$ can be generated by repeatedly taking nonnegative linear combinations and rounding up.

This might be regarded as the fundamental theorem of cutting plane theory.

A key step of the proof uses the **resolution method**.

This suggests there are deep connections between resolution and cutting planes.

Logic and cutting planes

A resolution step generates a **rank 1 cut** (i.e., a cut generated by one step of Chvátal's method).

$$\begin{array}{lll} x_1 \vee \bar{x}_2 & & \\ \bar{x}_1 & \vee x_3 \end{array} \qquad \begin{array}{rrrl} x_1 & + (1-\bar{x}_2) & & \geq 1 \\ (1-x_1) & & + x_3 & \geq 1 \end{array}$$

Convert to 0-1 inequalities

Logic and cutting planes

A resolution step generates a **rank 1 cut** (i.e., a cut generated by one step of Chvátal's method).

$$x_1 \vee \bar{x}_2$$
$$\bar{x}_1 \vee x_3$$

$$\begin{array}{rl}
x_1 + (1-\bar{x}_2) \geq 1 & (1/2) \\
(1-x_1) + x_3 \geq 1 & (1/2) \\
(1-x_2) \geq 0 & (1/2) \\
x_3 \geq 0 & (1/2) \\
\hline
(1-\bar{x}_2) + x_3 \geq 1/2 &
\end{array}$$

Take linear combination

Logic and cutting planes

A resolution step generates a **rank 1 cut** (i.e., a cut generated by one step of Chvátal's method).

$$x_1 \vee \bar{x}_2$$
$$\bar{x}_1 \vee x_3$$

$$\begin{array}{rl} x_1 + (1-\bar{x}_2) \ge 1 & (1/2) \\ (1-x_1) + x_3 \ge 1 & (1/2) \\ (1-x_2) \ge 0 & (1/2) \\ x_3 \ge 0 & (1/2) \\ \hline (1-\bar{x}_2) + x_3 \ge 1/2 & \\ (1-\bar{x}_2) + x_3 \ge 1 & \text{Round up} \end{array}$$

$$\bar{x}_2 \vee x_3$$

Resolvent

Logic and cutting planes

Theorem (JNH). **Input resolution** generates precisely those clauses that are rank 1 cuts.

$$
\begin{array}{lcr}
x_1 \vee \bar{x}_2 & & \\
\bar{x}_1 & & \vee\, x_3 \\
 & x_2 & \vee\, x_3 \\
\hline
 & & x_3
\end{array}
$$

Result of input resolution

Input resolution = use at least one of the original clauses to obtain each resolvent

Logic and cutting planes

Theorem (JNH). **Input resolution** generates precisely those clauses that are rank 1 cuts.

$$
\begin{array}{l}
x_1 \vee \bar{x}_2 \\
\bar{x}_1 \vee x_3 \\
x_2 \vee x_3 \\
\hline
x_3
\end{array}
$$

Result of input resolution

$$
\begin{array}{lr}
x_1 + (1 - x_2) \geq 1 & (1/4) \\
(1 - x_1) + x_3 \geq 1 & (1/4) \\
x_2 + x_3 \geq 1 & (1/2) \\
(1 - x_2) \geq 0 & (1/4) \\
x_3 \geq 0 & (1/4) \\
\hline
x_3 \geq 1/4 &
\end{array}
$$

Take linear combination

Logic and cutting planes

Theorem (JNH). **Input resolution** generates precisely those clauses that are rank 1 cuts.

$$x_1 \vee \bar{x}_2$$
$$\bar{x}_1 \vee x_3$$
$$x_2 \vee x_3$$
$$\overline{}$$
$$x_3$$

Result of input resolution

$$
\begin{array}{lr}
x_1 + (1 - x_2) \ge 1 & (1/4) \\
(1 - x_1) + x_3 \ge 1 & (1/4) \\
x_2 + x_3 \ge 1 & (1/2) \\
(1 - x_2) \ge 0 & (1/4) \\
x_3 \ge 0 & (1/4) \\
\hline
x_3 \ge 1/4 & \\
x_3 \ge 1 & \text{Round up}
\end{array}
$$

Logic and cutting planes

By generating enough Chvátal cuts, we obtain the **convex hull** of the 0-1 solutions.

Logic and cutting planes

By generating enough Chvatal cuts, we obtain the **convex hull** of the 0-1 solutions.

Can we obtain the convex hull by generating **resolvents**?

Logic and cutting planes

By generating enough Chvatal cuts, we obtain the **convex hull** of the 0-1 solutions.

Can we obtain the convex hull by generating **resolvents**?

That is, do the **prime implicates** define the convex hull?

Logic and cutting planes

By generating enough Chvatal cuts, we obtain the **convex hull** of the 0-1 solutions.

Can we obtain the convex hull by generating **resolvents**?

That is, do the **prime implicates** define the convex hull?

Not in general.

They do, if and only if the underlying **set covering problems** define convex hulls.

Logic and cutting planes

Theorem (JNH). The prime implicates of a clause set define an integral polytope if and only if all maximal **monotone** subsets of the prime implicates define an integral polytope.

monotone = every variable has the same sign in all occurrences.

A monotone subset of clauses is a set covering problem (after complementing negated variables).

Logic and cutting planes

Theorem (JNH). The prime implicates of a clause set define an integral polytope if and only if all maximal **monotone** subsets of the prime implicates define an integral polytope.

$$x_1 \vee x_2$$
$$x_1 \vee x_3$$
$$\bar{x}_2 \vee x_3$$

Prime implicates

Logic and cutting planes

Theorem (JNH). The prime implicates of a clause set define an integral polytope if and only if all maximal **monotone** subsets of the prime implicates define an integral polytope.

$$x_1 \vee x_2$$
$$x_1 \vee x_3$$
$$\bar{x}_2 \vee x_3$$

Prime implicates

$$x_1 \vee x_2$$
$$x_1 \vee x_3$$

$$x_1 \vee x_3$$
$$\bar{x}_2 \vee x_3$$

Maximal monotone subsets

Logic and cutting planes

Theorem (JNH). The prime implicates of a clause set define an integral polytope if and only if all maximal **monotone** subsets of the prime implicates define an integral polytope.

$$x_1 \vee x_2$$
$$x_1 \vee x_3$$
$$\bar{x}_2 \vee x_3$$

Prime implicates

$$x_1 \vee x_2$$
$$x_1 \vee x_3$$

$$x_1 \vee x_3$$
$$\bar{x}_2 \vee x_3$$

Maximal monotone subsets

$$x_1 + x_2 \geq 1$$
$$x_1 + x_3 \geq 1$$

$$x_1 + x_3 \geq 1$$
$$(1 - x_2) + x_3 \geq 1$$

These systems define integral polytopes

Logic and cutting planes

Theorem (JNH). The prime implicates of a clause set define an integral polytope if and only if all maximal **monotone** subsets of the prime implicates define an integral polytope.

$$
\begin{aligned}
&x_1 \vee x_2 \\
&x_1 \vee x_3 \\
&\bar{x}_2 \vee x_3
\end{aligned}
$$

Prime implicates

$$
\begin{aligned}
&x_1 \vee x_2 \\
&x_1 \vee x_3
\end{aligned}
$$

$$
\begin{aligned}
&x_1 \vee x_3 \\
&\bar{x}_2 \vee x_3
\end{aligned}
$$

Maximal monotone subsets

$$
\begin{aligned}
x_1 + x_2 &\geq 1 \\
x_1 + x_3 &\geq 1
\end{aligned}
$$

$$
\begin{aligned}
x_1 + x_3 &\geq 1 \\
(1 - x_2) + x_3 &\geq 1
\end{aligned}
$$

These systems define integral polytopes

$$
\begin{aligned}
x_1 + x_2 &\geq 1 \\
x_1 + x_3 &\geq 1 \\
(1 - x_2) + x_3 &\geq 1
\end{aligned}
$$

Therefore this system defines an integral polytope

Logic and cutting planes

Theorem (JNH). The prime implicates of a clause set define an integral polytope if and only if all maximal **monotone** subsets of the prime implicates define an integral polytope.

Generalized by Guenin, and by Nobili & Sassano.

Logic of 0-1 Inequalities

Logic of 0-1 inequalities

0-1 inequalities can be viewed as **logical propositions**.

Logic of 0-1 inequalities

0-1 inequalities can be viewed as **logical propositions**.

Can the **resolution algorithm** be generalized to 0-1 inequalities?

Logic of 0-1 inequalities

0-1 inequalities can be viewed as **logical propositions**.

Can the **resolution algorithm** be generalized to 0-1 inequalities?

Yes. This results in a **logical analog** of Chvátal's theorem.

Logic of 0-1 inequalities

0-1 inequalities can be viewed as **logical propositions**.

Can the **resolution algorithm** be generalized to 0-1 inequalities?

Yes. This results in a **logical analog** of Chvátal's theorem.

Theorem (JNH). Classical resolution + **diagonal summation** generates all 0-1 prime implicates (up to logical equivalence).

Logic of 0-1 inequalities

Diagonal summation:

$$\begin{array}{r} x_1 + 5x_2 + 3x_3 + x_4 \geq 4 \\ 2x_1 + 4x_2 + 3x_3 + x_4 \geq 4 \\ 2x_1 + 5x_2 + 2x_3 + x_4 \geq 4 \\ 2x_1 + 5x_2 + 3x_3 \geq 4 \\ \hline 2x_1 + 5x_2 + 3x_3 + x_4 \geq 5 \end{array}$$

Each inequality is implied by an inequality in the set to which 0-1 resolution is implied.

Diagonal sum

Logic of 0-1 inequalities

Diagonal summation:

$$x_1 + 5x_2 + 3x_3 + x_4 \geq 4$$
$$2x_1 + 4x_2 + 3x_3 + x_4 \geq 4$$
$$2x_1 + 5x_2 + 2x_3 + x_4 \geq 4$$
$$2x_1 + 5x_2 + 3x_3 + 0x_4 \geq 4$$
$$\overline{2x_1 + 5x_2 + 3x_3 + x_4 \geq 5}$$

Each inequality is implied by an inequality in the set to which 0-1 resolution is implied.

Diagonal sum

Logic and Linear Programming

Logic and linear programming

Theorem: A **renamable Horn** set of clauses is satisfiable if and only if it has a **unit refutation**.

Horn = at most one positive literal per clause

Renamable Horn = Horn after complementing some variables.

Unit refutation = resolution proof of unsatisfiability in which at least one parent of each resolvent is a unit clause.

Logic and linear programming

Theorem: A **renamable Horn** set of clauses is satisfiable if and only if it has a **unit refutation**.

Horn = at most one positive literal per clause

Renamable Horn = Horn after complementing some variables.

Unit refutation = resolution proof of unsatisfiability in which at least one parent of each resolvent is a unit clause.

$$
\begin{array}{l}
x_1 \\
\bar{x}_1 \vee \bar{x}_2 \\
\bar{x}_1 \qquad\quad \vee\, x_3 \\
\bar{x}_1 \vee x_2 \vee \bar{x}_3
\end{array}
$$

Horn set

Logic and linear programming

Theorem: A **renamable Horn** set of clauses is satisfiable if and only if it has a **unit refutation**.

Horn = at most one positive literal per clause

Renamable Horn = Horn after complementing some variables.

Unit refutation = resolution proof of unsatisfiability in which at least one parent of each resolvent is a unit clause.

Horn set	Unit resolution
x_1	x_1
$\bar{x}_1 \vee \bar{x}_2$	$\bar{x}_1 \vee \bar{x}_2$
$\bar{x}_1 \vee x_3$	$\bar{x}_1 \vee x_3$
$\bar{x}_1 \vee x_2 \vee \bar{x}_3$	$\bar{x}_1 \vee x_2 \vee \bar{x}_3$

Logic and linear programming

Theorem: A **renamable Horn** set of clauses is satisfiable if and only if it has a **unit refutation**.

Horn = at most one positive literal per clause

Renamable Horn = Horn after complementing some variables.

Unit refutation = resolution proof of unsatisfiability in which at least one parent of each resolvent is a unit clause.

Horn set	Unit resolution	Unit resolution
x_1	x_1	x_1
$\bar{x}_1 \vee \bar{x}_2$	$\bar{x}_1 \vee \bar{x}_2$	$\bar{x}_1 \vee \bar{x}_2$
$\bar{x}_1 \vee x_3$	$\bar{x}_1 \vee x_3$	$\bar{x}_1 \vee x_3$
$\bar{x}_1 \vee x_2 \vee \bar{x}_3$	$\bar{x}_1 \vee x_2 \vee \bar{x}_3$	$\bar{x}_1 \vee x_2 \vee \bar{x}_3$

Logic and linear programming

We don't know a **necessary and sufficient condition** for solubility by unit refutation.

But we can identify sufficient conditions by **generalizing Horn sets.**

For example, to **extended Horn sets**, which rely on a rounding property of linear programming.

Logic and linear programming

Theorem: A satisfiable **Horn** set can be solved by rounding down a solution of the linear programming relaxation.

Logic and linear programming

Theorem: A satisfiable **Horn** set can be solved by rounding down a solution of the linear programming relaxation.

$$x_1$$
$$\bar{x}_1 \vee \bar{x}_2 \vee x_3$$
$$\bar{x}_2 \vee \bar{x}_3$$
$$\bar{x}_1 \vee x_2 \vee \bar{x}_3$$

Horn set

Logic and linear programming

Theorem: A satisfiable **Horn** set can be solved by rounding down a solution of the linear programming relaxation.

$$\begin{array}{l} x_1 \\ \bar{x}_1 \vee \bar{x}_2 \vee x_3 \\ \qquad \bar{x}_2 \vee \bar{x}_3 \\ \bar{x}_1 \vee x_2 \vee \bar{x}_3 \end{array}$$

Horn set

$$\begin{array}{l} x_1 \geq 1 \\ (1-x_1)+(1-x_2)+x_3 \geq 1 \\ (1-x_2)+(1-x_3) \geq 1 \\ (1-x_1)+x_2+(1-x_3) \geq 1 \\ 0 \leq x_j \leq 1 \end{array}$$

LP relaxation

Logic and linear programming

Theorem: A satisfiable **Horn** set can be solved by rounding down a solution of the linear programming relaxation.

$$x_1$$
$$\bar{x}_1 \vee \bar{x}_2 \vee x_3$$
$$\bar{x}_2 \vee \bar{x}_3$$
$$\bar{x}_1 \vee x_2 \vee \bar{x}_3$$

Horn set

$$\begin{aligned} x_1 &\geq 1 \\ (1-x_1)+(1-x_2)+x_3 &\geq 1 \\ (1-x_2)+(1-x_3) &\geq 1 \\ (1-x_1)+x_2+(1-x_3) &\geq 1 \\ 0 \leq x_j \leq 1 & \end{aligned}$$

LP relaxation

Solution: $(x_1,x_2,x_3) = (1,1/2,1/2)$

Logic and linear programming

Theorem: A satisfiable **Horn** set can be solved by rounding down a solution of the linear programming relaxation.

$$\begin{aligned} & x_1 \\ & \bar{x}_1 \vee \bar{x}_2 \vee x_3 \\ & \bar{x}_2 \vee \bar{x}_3 \\ & \bar{x}_1 \vee x_2 \vee \bar{x}_3 \end{aligned}$$

Horn set

$$\begin{aligned} x_1 &\geq 1 \\ (1-x_1) + (1-x_2) + x_3 &\geq 1 \\ (1-x_2) + (1-x_3) &\geq 1 \\ (1-x_1) + x_2 + (1-x_3) &\geq 1 \\ 0 \leq x_j &\leq 1 \end{aligned}$$

LP relaxation

Solution: $(x_1,x_2,x_3) = (1,1/2,1/2)$

Round down: $(x_1,x_2,x_3) = (1,0,0)$

Logic and linear programming

To generalize this, we use the following:

Theorem (Chandrasekaran): If $Ax \geq b$ has integral components and T is nonsingular such that:

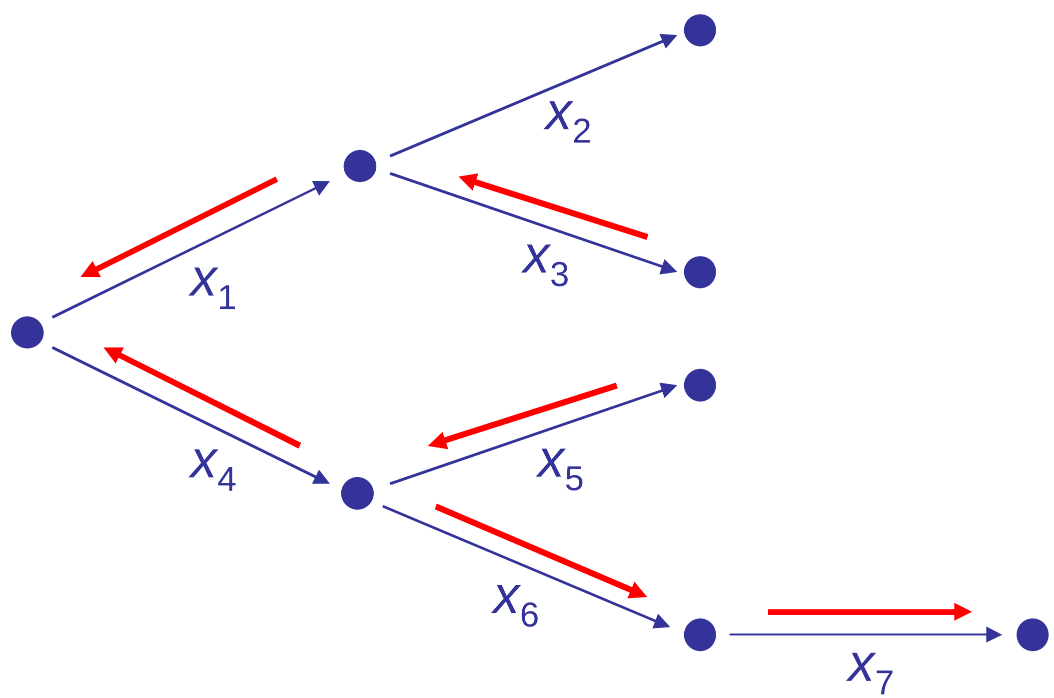
- T and T^{-1} are integral
- Each row of T^{-1} contains at most one negative entry, namely –1
- Each row of AT^{-1} contains at most one negative entry, namely –1

Then if x solves $Ax \geq b$, so does $T^{-1}\lceil Tx \rceil$

Logic and linear programming

A clause has the **extended star-chain property** if it corresponds to a set of edge-disjoint flows into the root of an arborescence and a flow on one additional chain.

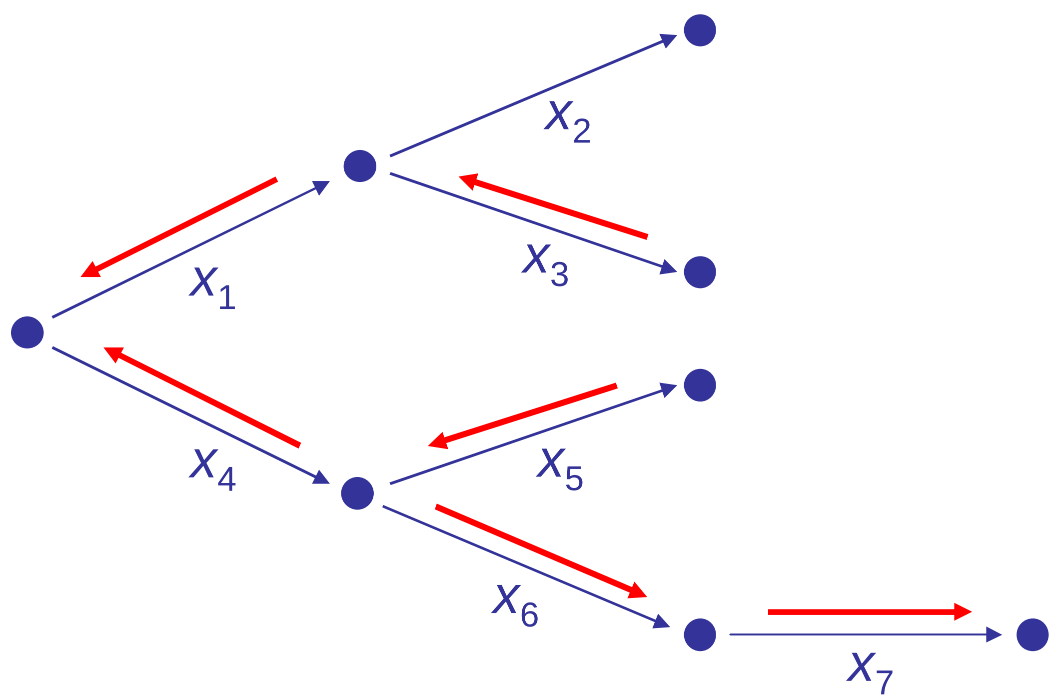
$$\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5 \vee x_6 \vee x_7$$



Logic and linear programming

A clause set is **extended Horn** if there is an arborescence for which every clause in the set has the extended star-chain property.

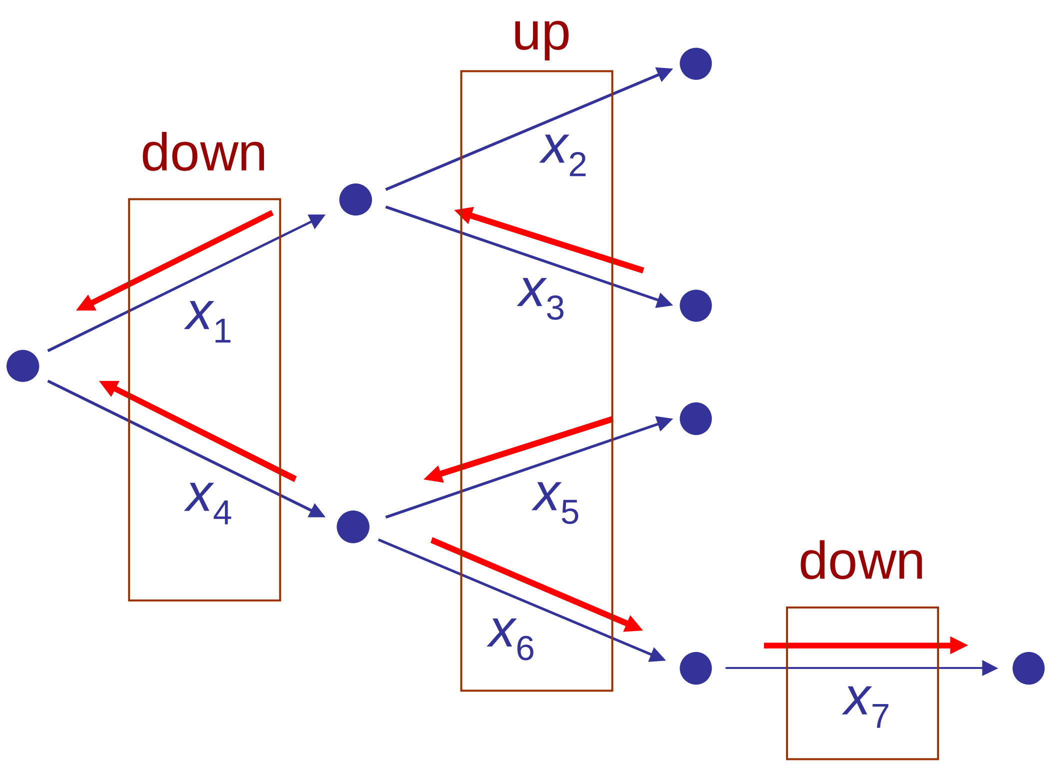
$$\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5 \vee x_6 \vee x_7$$



Logic and linear programming

Theorem (Chandru and JNH). A satisfiable extended Horn clause set can be solved by rounding a solution of the LP relaxation as shown:

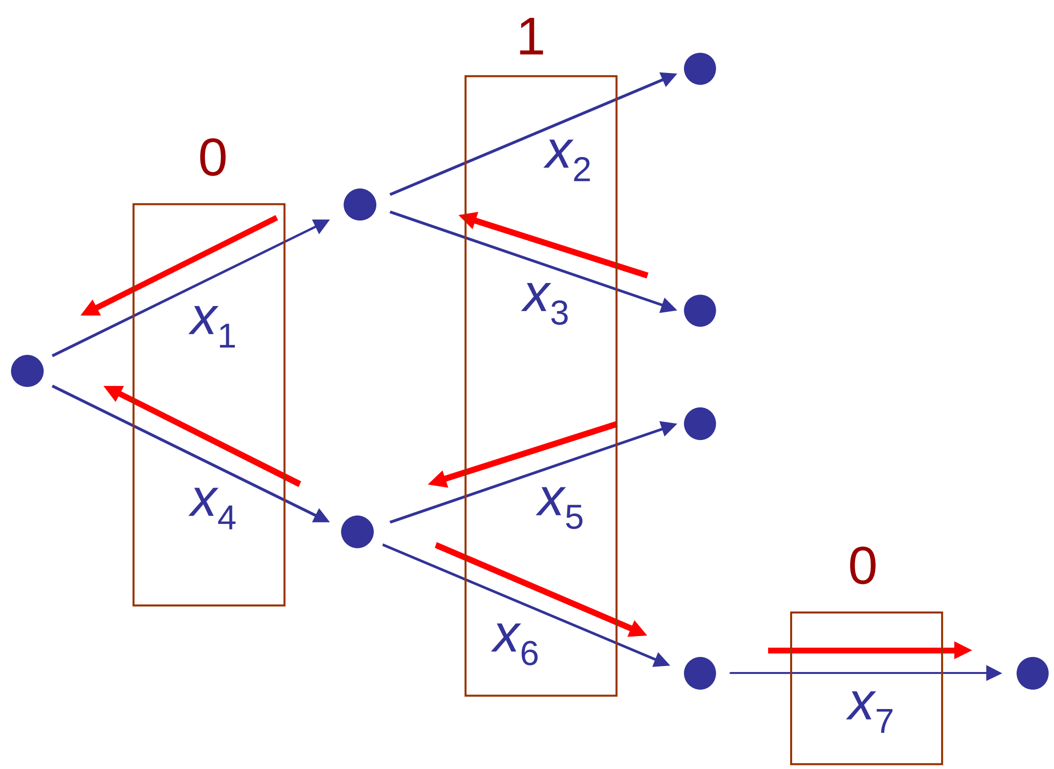
$$\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5 \vee x_6 \vee x_7$$



Logic and linear programming

Corollary. A satisfiable extended Horn clause set can be solved by assigning values as shown:

$$\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5 \vee x_6 \vee x_7$$



Logic and linear programming

Theorem (Chandru and JNH). A **renamable** extended Horn clause is satisfiable if and only if it has no unit refutation.

Logic and linear programming

Theorem (Chandru and JNH). A **renamable** extended Horn clause is satisfiable if and only if it has no unit refutation.

Theorem (Schlipf, Annexstein, Franco & Swaminathan). These results hold when then incoming chains are not edge disjoint.

.

Logic and linear programming

Theorem (Chandru and JNH). A **renamable** extended Horn clause is satisfiable if and only if it has no unit refutation.

Theorem (Schlipf, Annexstein, Franco & Swaminathan). These results hold when then incoming chains are not edge disjoint.

Corollary (Schlipf, Annexstein, Franco & Swaminathan). A one-step lookahead algorithm solves a satisfiable extended Horn problem without knowledge of the arborescence.

Inference duality

Inference duality

Consider an optimization problem:

$$\min f(x)$$
$$S$$
$$x \in D$$

S ← Constraint set

$x \in D$ ← Variable domain

Inference duality

Consider an optimization problem:

$$\begin{aligned} &\min f(x) \\ &\mathcal{S} \\ &x \in D \end{aligned}$$

$\mathcal{S}$ ← Constraint set

$x \in D$ ← Variable domain

An inference dual is:

$$\begin{aligned} &\max v \\ &\mathcal{S} \overset{P}{\Rightarrow} \left(f(x) \geq v \right) \\ &v \in \mathbb{R}, \quad P \in \mathscr{P} \end{aligned}$$

There is a proof P of $f(x) \geq v$ from premises in $\mathcal{S}$

Family of admissible proofs

Inference duality

Linear programming:

$$\begin{aligned} &\min cx \\ &Ax \geq b \\ &x \geq 0 \end{aligned}$$

Let $Ax \geq b \Rightarrow cx \geq v$ when $uAx \geq ub$ **dominates** $cx \geq v$ for some $u \geq 0$.

dominates = $uA \leq c$ and $ub \geq v$

Inference dual is:

$$\begin{aligned} &\max v \\ &(Ax \geq b) \overset{P}{\Rightarrow} (cx \geq v) \\ &v \in \mathbb{R}, \quad P \in \mathscr{P} \end{aligned}$$

Inference duality

Linear programming:

$$\min cx$$
$$Ax \geq b$$
$$x \geq 0$$

Let $Ax \geq b \Rightarrow cx \geq v$ when $uAx \geq ub$ **dominates** $cx \geq v$ for some $u \geq 0$.

dominates = $uA \leq c$ and $ub \geq v$

Inference dual is:

$$\max v$$
$$(Ax \geq b) \overset{P}{\Rightarrow} (cx \geq v)$$
$$v \in \mathbb{R},\quad P \in \mathscr{P}$$

This becomes the **classical LP dual.**

Inference duality

Linear programming:

$$\min cx$$
$$Ax \geq b$$
$$x \geq 0$$

Let $Ax \geq b \Rightarrow cx \geq v$ when $uAx \geq ub$ **dominates** $cx \geq v$ for some $u \geq 0$.

dominates = $uA \leq c$ and $ub \geq v$

Inference dual is:

$$\max v$$
$$(Ax \geq b) \overset{P}{\Rightarrow} (cx \geq v)$$
$$v \in \mathbb{R},\quad P \in \mathscr{P}$$

This becomes the **classical LP dual.**

This is a **strong dual** because the inference method is **complete** (Farkas Lemma).

Inference duality

General inequality constraints:

$$\begin{aligned}&\min f(x)\\&g(x) \geq 0\\&x \in S\end{aligned}$$

Let $g(x) \geq 0 \Rightarrow f(x) \geq v$ when $ug(x) \geq 0$ **implies** $f(x) \geq v$ for some $u \geq 0$.

implies = all $x \in S$ satisfying $ug(x) \geq 0$ satisfy $f(x) \geq v$.

Inference dual is:

$$\begin{aligned}&\max v\\&\left(g(x) \geq 0\right) \stackrel{P}{\Rightarrow} \left(f(x) \geq v\right)\\&v \in \mathbb{R},\quad P \in \mathscr{P}\end{aligned}$$

Inference duality

General inequality constraints:

$$\min f(x)$$
$$g(x) \geq 0$$
$$x \in S$$

Let $g(x) \geq 0 \Rightarrow f(x) \geq v$ when $ug(x) \geq 0$ **implies** $f(x) \geq v$ for some $u \geq 0$.

implies = all $x \in S$ satisfying $ug(x) \geq 0$ satisfy $f(x) \geq v$.

Inference dual is:

$$\max v$$
$$\left(g(x) \geq 0\right) \overset{P}{\Rightarrow} \left(f(x) \geq v\right)$$
$$v \in \mathbb{R}, \quad P \in \mathscr{P}$$

This becomes the **surrogate dual**.

Inference duality

General inequality constraints:

$$\min f(x)$$
$$g(x) \geq 0$$
$$x \in S$$

Let $g(x) \geq 0 \Rightarrow f(x) \geq v$ when $ug(x) \geq 0$ **dominates** $f(x) \geq v$ for some $u \geq 0$.

Inference dual is:

$$\max v$$
$$\left(g(x) \geq 0\right) \overset{P}{\Rightarrow} \left(f(x) \geq v\right)$$
$$v \in \mathbb{R}, \quad P \in \mathscr{P}$$

Inference duality

General inequality constraints:

$$\min f(x)$$
$$g(x) \geq 0$$
$$x \in S$$

Let $g(x) \geq 0 \Rightarrow f(x) \geq v$ when $ug(x) \geq 0$ **dominates** $f(x) \geq v$ for some $u \geq 0$.

Inference dual is:

$$\max v$$
$$\left(g(x) \geq 0\right) \overset{P}{\Rightarrow} \left(f(x) \geq v\right)$$
$$v \in \mathbb{R}, \quad P \in \mathscr{P}$$

This becomes the **Lagrangean dual**

Inference duality

Integer linear programming:

$$\min cx$$
$$Ax \geq b$$
$$x \in S$$

Let $Ax \geq$ b $\Rightarrow cx \geq v$ when $h(Ax) \geq h(b)$ dominates $cx \geq v$ for some **subadditive** and homogeneous function *h*.

Inference dual is:

$$\max v$$
$$(Ax \geq b) \overset{P}{\Rightarrow} (cx \geq v)$$
$$v \in \mathbb{R}, \quad P \in \mathscr{P}$$

Inference duality

Integer linear programming:

$$\min cx$$
$$Ax \ge b$$
$$x \in S$$

Let $Ax \ge b \Rightarrow cx \ge v$ when $h(Ax) \ge h(b)$ dominates $cx \ge v$ for some **subadditive** and homogeneous function h.

Inference dual is:

$$\max v$$
$$(Ax \ge b) \overset{P}{\Rightarrow} (cx \ge v)$$
$$v \in \mathbb{R},\quad P \in \mathscr{P}$$

This becomes the **subadditive dual**.

Inference duality

Integer linear programming:

$$\min cx$$
$$Ax \geq b$$
$$x \in S$$

Let $Ax \geq$ b $\Rightarrow$ $cx \geq v$ when $h(Ax) \geq h(b)$ dominates $cx \geq v$ for some **subadditive** and homogeneous function h.

Inference dual is:

$$\max v$$
$$(Ax \geq b) \overset{P}{\Rightarrow} (cx \geq v)$$
$$v \in \mathbb{R},\quad P \in \mathscr{P}$$

This becomes the **subadditive dual**.

This is a **strong dual** because the inference method is complete, due to Chvátal's theorem.

Appropriate Chvátal function is subadditive and can found by Gomory's cutting plane method.

Inference duality

Inference duality permits a generalization of **Benders decomposition.**

Inference duality

Inference duality permits a generalization of **Benders decomposition.**

In classical Benders, a **Benders cut** is a linear combination of the subproblem constraints using dual multipliers.

Inference duality

Inference duality permits a generalization of **Benders decomposition.**

In classical Benders, a **Benders cut** is a linear combination of the subproblem constraints using dual multipliers.

The Benders cut rules out solutions of the master problem for which the proof of optimality in the subproblem is **still valid**.

Inference duality

Inference duality permits a generalization of **Benders decomposition.**

In classical Benders, a **Benders cut** is a linear combination of the subproblem constraints using dual multipliers.

The Benders cut rules out solutions of the master problem for which the proof of optimality in the subproblem is **still valid**.

For general optimization, a Benders cut does the same, but the proof of optimality is a solution of the general **inference dual**.

Inference duality

Inference duality permits a generalization of **Benders decomposition.**

In classical Benders, a **Benders cut** is a linear combination of the subproblem constraints using dual multipliers.

The Benders cut rules out solutions of the master problem for which the proof of optimality in the subproblem is **still valid**.

For general optimization, a Benders cut does the same, but the proof of optimality is a solution of the general **inference dual**.

This has led to orders-of-magnitude speedups in solution of scheduling and other problems by **logic-based Benders decomposition**.

Constraint Programming

Constraint programming

Constraint programming uses **logical inference** to reduce backtracking.

Constraint programming

Constraint programming uses **logical inference** to reduce backtracking.

Inference takes the form of **consistency maintenance**.

Constraint programming

Constraint programming uses **logical inference** to reduce backtracking.

Inference takes the form of **consistency maintenance**.

A constraint set S containing variables $x_1, \ldots, x_n$ is ***k*-consistent** if
- for any subject of variables $x_1, \ldots, x_j, x_{j+1}$
- and any partial assignment $(x_1, \ldots, x_j) = (v_1, \ldots, v_j)$ that violates no constraint in S,

there is a v_{j+1} such that $(x_1, \ldots, x_{j+1}) = (v_1, \ldots, v_{j+1})$ violates no constraint in S.

Constraint programming

Constraint programming uses **logical inference** to reduce backtracking.

Inference takes the form of **consistency maintenance**.

A constraint set S containing variables $x_1, \ldots, x_n$ is ***k*-consistent** if
- for any subject of variables $x_1, \ldots, x_j, x_{j+1}$
- and any partial assignment $(x_1, \ldots, x_j) = (v_1, \ldots, v_j)$ that violates no constraint in S,

there is a v_{j+1} such that $(x_1, \ldots, x_{j+1}) = (v_1, \ldots, v_{j+1})$ violates no constraint in S.

S is **strongly *k*-consistent** if it is j-consistent for $j = 1, \ldots, k$.

Constraint programming

Theorem (Freuder). If constraint set S is strongly k-consistent, and its **dependency graph** has width less than k (with respect to the branching order), then S can be solved without backtracking.

Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

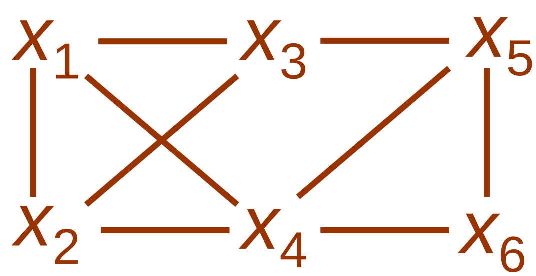
$$\bar{x}_1 \vee x_2 \vee \bar{x}_3$$

$$x_1 \vee \bar{x}_2 \vee x_4$$

$$x_3 \vee x_5$$

$$x_4 \vee \bar{x}_5 \vee x_6$$

Dependency graph

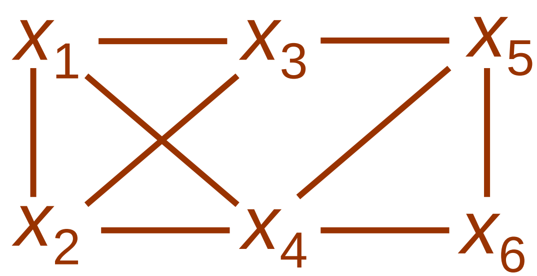


Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

$$\bar{x}_1 \vee x_2 \vee \bar{x}_3$$
$$x_1 \vee \bar{x}_2 \vee x_4$$
$$x_3 \vee x_5$$
$$x_4 \vee \bar{x}_5 \vee x_6$$

Dependency graph



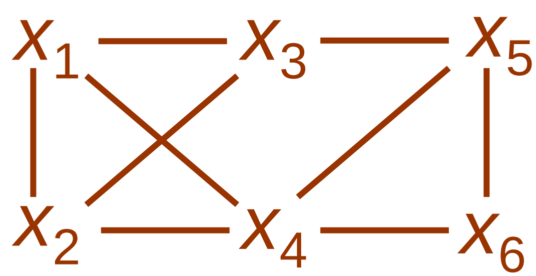
Width = max in-degree = 2

Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

$$\bar{x}_1 \vee x_2 \vee \bar{x}_3$$
$$x_1 \vee \bar{x}_2 \vee x_4$$
$$x_3 \vee x_5$$
$$x_4 \vee \bar{x}_5 \vee x_6$$

Dependency graph



Width = max in-degree = 2

We will show that this is strongly 3-consistent.

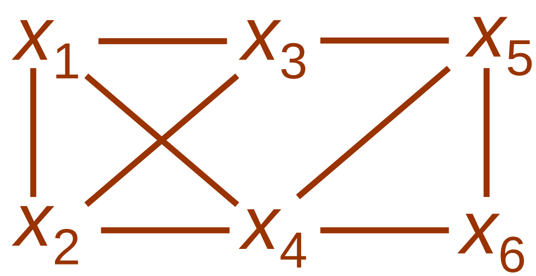
We can therefore solve it without backtracking

Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

$$\bar{x}_1 \vee x_2 \vee \bar{x}_3$$
$$x_1 \vee \bar{x}_2 \vee x_4$$
$$x_3 \vee x_5$$
$$x_4 \vee \bar{x}_5 \vee x_6$$

Dependency graph



Width = max in-degree = 2

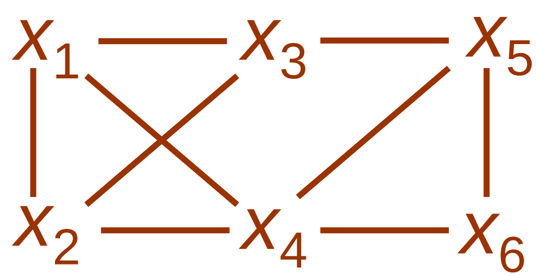
x_1	x_2	x_3	x_4	x_5	x_6
0					

Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

$$
\begin{aligned}
&\bar{x}_2 \vee x_4 \\
&x_3 \vee x_5 \\
&x_4 \vee \bar{x}_5 \vee x_6
\end{aligned}
$$

Dependency graph



Width = max in-degree = 2

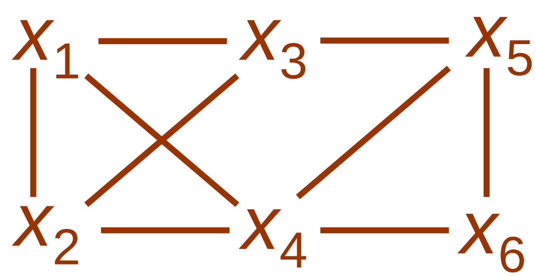
x_1	x_2	x_3	x_4	x_5	x_6
0					

Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

$$\bar{x}_2 \vee x_4$$
$$x_3 \vee x_5$$
$$x_4 \vee \bar{x}_5 \vee x_6$$

Dependency graph



Width = max in-degree = 2

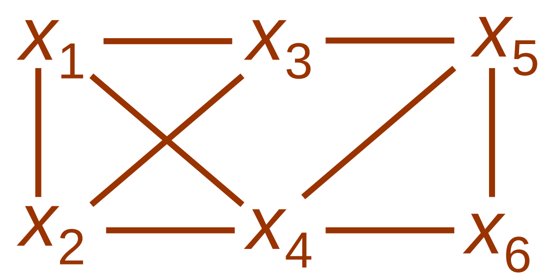
x_1	x_2	x_3	x_4	x_5	x_6
0	0				

Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

$$x_3 \qquad \vee x_5$$
$$x_4 \vee \bar{x}_5 \vee x_6$$

Dependency graph



Width = max in-degree = 2

x_1	x_2	x_3	x_4	x_5	x_6
0	0				

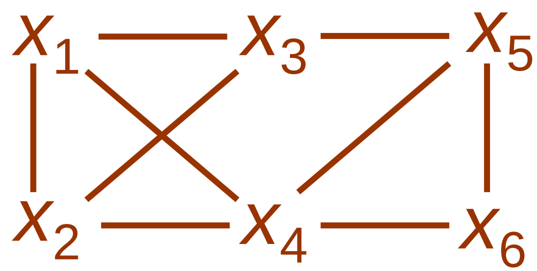
Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

$$x_3 \qquad \vee x_5$$

$$x_4 \vee \bar{x}_5 \vee x_6$$

Dependency graph



Width = max in-degree = 2

x_1	x_2	x_3	x_4	x_5	x_6
0	0	0			

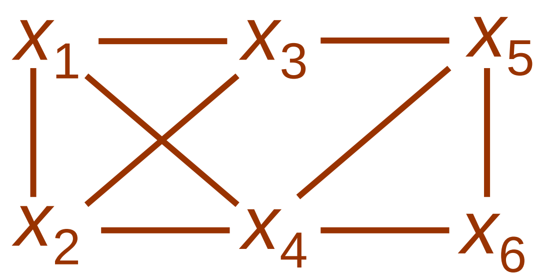
Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

x_5

$x_4 \vee \bar{x}_5 \vee x_6$

Dependency graph



Width = max in-degree = 2

x_1	x_2	x_3	x_4	x_5	x_6
0	0	0			

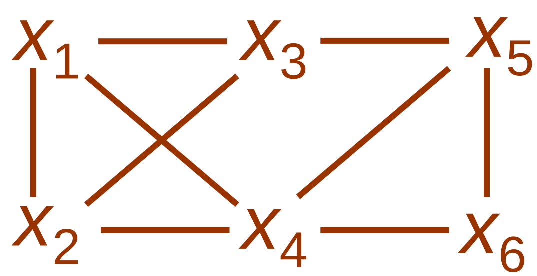
Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

x_5

$x_4 \vee \bar{x}_5 \vee x_6$

Dependency graph



Width = max in-degree = 2

x_1	x_2	x_3	x_4	x_5	x_6
0	0	0	0		

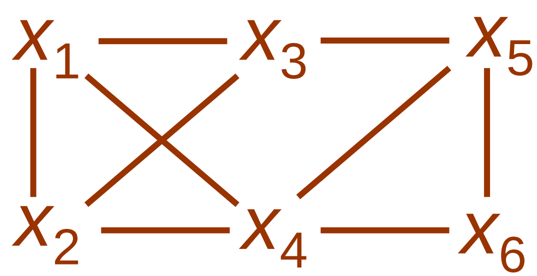
Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

x_5

$\bar{x}_5 \vee x_6$

Dependency graph



Width = max in-degree = 2

x_1	x_2	x_3	x_4	x_5	x_6
0	0	0	0		

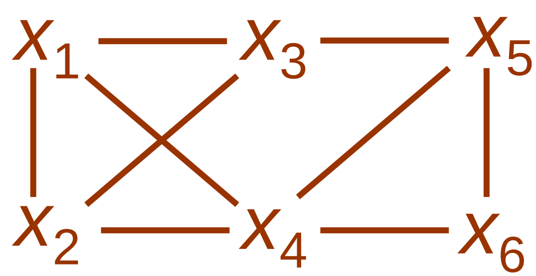
Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

x_5

$\bar{x}_5 \vee x_6$

Dependency graph



Width = max in-degree = 2

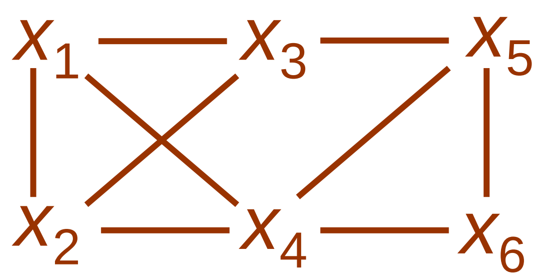
x_1	x_2	x_3	x_4	x_5	x_6
0	0	0	0	1	

Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

x_6

Dependency graph



Width = max in-degree = 2

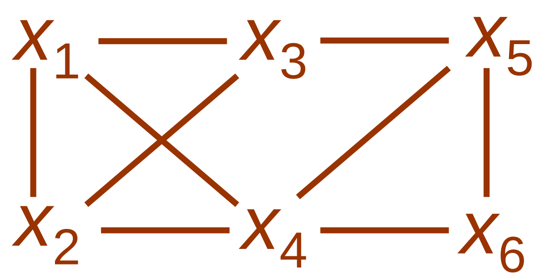
x_1	x_2	x_3	x_4	x_5	x_6
0	0	0	0	1	

Constraint programming

Theorem (Freuder). If constraint set *S* is strongly *k*-consistent, and its **dependency graph** has width less than *k* (with respect to the branching order), then S can be solved without backtracking.

x_6

Dependency graph



Width = max in-degree = 2

x_1	x_2	x_3	x_4	x_5	x_6
0	0	0	0	1	1

Constraint programming

Theorem. Application of ***k*-resolution** makes a clause set strongly *k*-consistent.

k-resolution = generate only resolventss with fewer than k literals.

$$\bar{x}_1 \vee x_2 \vee \bar{x}_3$$
$$x_1 \vee \bar{x}_2 \vee x_4$$
$$x_3 \vee x_5$$
$$x_4 \vee \bar{x}_5 \vee x_6$$

All resolvents have 3 or more literals.

Clause set is therefore strongly 3-consistent, as claimed.

Constraint programming

Constraint programmers are primarily concerned with **domain consistency**.

Constraint programming

Constraint programmers are primarily concerned with **domain consistency**.

A constraint set S is **domain consistent** if for any given variable x_j and any value v_j in its domain, $x_j = v_j$ in some solution of S.

Constraint programming

Constraint programmers are primarily concerned with **domain consistency**.

A constraint set S is **domain consistent** if for any given variable x_j and any value v_j in its domain, $x_j = v_j$ in some solution of S.

Domain consistency = generalized arc consistency = hyperarc consistency.

Constraint programming

Constraint programmers are primarily concerned with **domain consistency**.

A constraint set S is **domain consistent** if for any given variable x_j and any value v_j in its domain, $x_j = v_j$ in some solution of S.

Domain consistency = generalized arc consistency = hyperarc consistency.

Filtering algorithms that achieve or approximate domain consistency have been devised for a wide variety of constraints.

Constraint programming

Constraint programmers are primarily concerned with **domain consistency**.

A constraint set S is **domain consistent** if for any given variable x_j and any value v_j in its domain, $x_j = v_j$ in some solution of S.

Domain consistency = generalized arc consistency = hyperarc consistency.

Filtering algorithms that achieve or approximate domain consistency have been devised for a wide variety of constraints.

The **resolution algorithm** achieves domain consistency for clause sets.

Constraint programming

Constraint programmers are primarily concerned with **domain consistency**.

A constraint set S is **domain consistent** if for any given variable x_j and any value v_j in its domain, $x_j = v_j$ in some solution of S.

Domain consistency = generalized arc consistency = hyperarc consistency.

Filtering algorithms that achieve or approximate domain consistency have been devised for a wide variety of constraints.

The **resolution algorithm** achieves domain consistency for clause sets.

Filtering (**= logical inference**) is the workhorse of constraint programming, as solving relaxations is the workhorse of integer programming.

Good Logic Models

Good logic models

Boolean models should be **reformulated** before solution to achieve two goals:

Good logic models

Boolean models should be **reformulated** before solution to achieve two goals:

- A high degree of **consistency** (in the constraint programming sense)
  - We talked about **resolution** as a means of achieving consistency.

Good logic models

Boolean models should be **reformulated** before solution to achieve two goals:

- A high degree of **consistency** (in the constraint programming sense)
  - We talked about **resolution** as a means of achieving consistency for boolean models.
- A tight **linear relaxation**.
  - We talked about logic and **cutting planes**.
  - Logic constraints can also be given **convex hull formulations**…

Good logic models

Example: **cardinality rules**

We have 3 possible sites for factories and 3 possible products.

Rule 1: If at least 2 plants are built, then at least 2 products should be made.

Rule 2. Only 1 product should be made, unless plants are built at both sites 1 and 2.

$$(x_1 + x_2 + x_3 \geq 2) \Rightarrow (y_1 + y_2 + y_3 \geq 2)$$

$$(y_1 + y_2 + y_3 \geq 2) \Rightarrow (x_1 + x_2 \geq 2)$$

Good logic models

$$(x_1 + x_2 + x_3 \geq 2) \Rightarrow (y_1 + y_2 + y_3 \geq 2)$$

Inequality form:

$$-2(x_1 + x_2 + x_3) + 2(y_1 + y_2 + y_3) \geq -2$$
$$-2(x_1 + x_2) + y_1 + y_2 + y_3 \geq -2$$
$$-2(x_1 + x_3) + y_1 + y_2 + y_3 \geq -2$$
$$-2(x_2 + x_3) + y_1 + y_2 + y_3 \geq -2$$
$$-x_1 - x_2 - x_3 + 2(y_1 + y_2) \geq -1$$
$$-x_1 - x_2 - x_3 + 2(y_1 + y_3) \geq -1$$
$$-x_1 - x_2 - x_3 + 2(y_2 + y_3) \geq -1$$

$$-x_1 - x_2 + y_1 + y_2 \geq -1$$
$$-x_1 - x_2 + y_1 + y_3 \geq -1$$
$$-x_1 - x_2 + y_2 + y_3 \geq -1$$
$$-x_1 - x_3 + y_1 + y_2 \geq -1$$
$$-x_1 - x_3 + y_1 + y_3 \geq -1$$
$$-x_1 - x_3 + y_2 + y_3 \geq -1$$
$$-x_2 - x_3 + y_1 + y_2 \geq -1$$
$$-x_2 - x_3 + y_1 + y_3 \geq -1$$
$$-x_2 - x_3 + y_2 + y_3 \geq -1$$

Good logic models

$$(y_1 + y_2 + y_3 \geq 2) \Rightarrow (x_1 + x_2 \geq 2)$$

Inequality form:

$$-2(y_1 + y_2 + y_3) + x_1 \geq -3$$
$$-2(y_1 + y_2) + x_1 \geq -1$$
$$-2(y_1 + y_3) + x_1 \geq -1$$
$$-2(y_2 + y_3) + x_1 \geq -1$$
$$-2(y_1 + y_2 + y_3) + x_2 \geq -3$$
$$-2(y_1 + y_2) + x_1 \geq -1$$
$$-2(y_1 + y_3) + x_1 \geq -1$$
$$-2(y_2 + y_3) + x_1 \geq -1$$

Good logic models

Theorem (Yan and JNH): These describe the convex hull of the feasible set.

Generalized by Balas, Bockmayr, Pisaruk & Wolsey.