# Collaboration Opportunities for OR and Constraint Programming

John Hooker
Carnegie Mellon University

NSF CMMI Conference
July 2012

# Two Ways Fields Can Collaborate

- Combine complementary techniques
  - Can yield results.
- Unify the ideas.
  - More fun intellectually, also yields results.
- Let’s look at OR/CP collaboration from this perspective.

# Outline

- What is constraint programming?
- Integrating OR and CP
- A unifying principle – inference duality.
- Other areas of unification

Caveat: This is a high-level overview.
Don't worry about the technical
details.

# What Is Constraint Programming?

Brief intellectual history

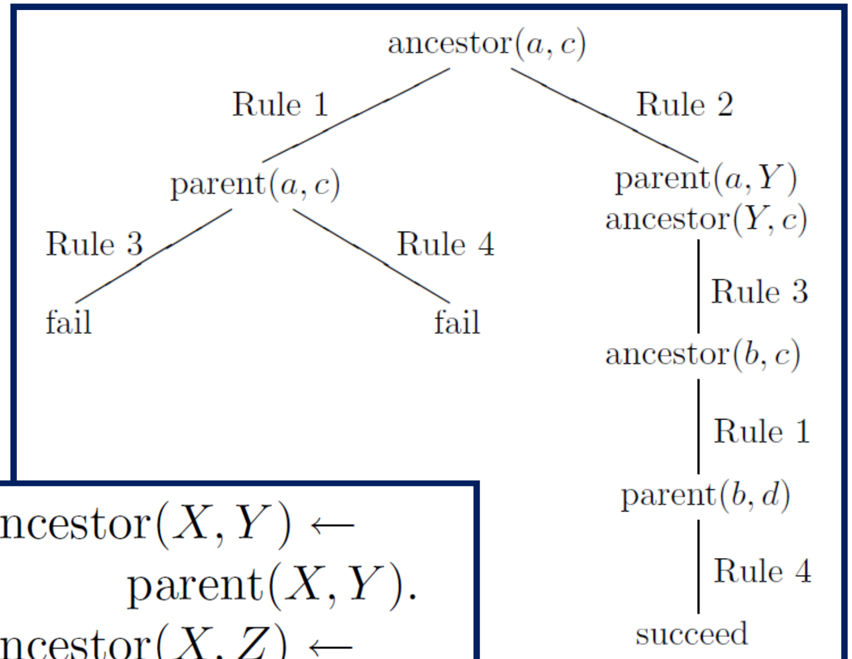
Modeling example

Applications

Strengths and Weaknesses

# First Step: Logic Programming

- Attempt to **unify** procedural and declarative modeling
  - **Procedural:** Write the algorithm (CS)
  - **Declarative:** Write the constraints (OR)
  - **Logic programming:** Propositions are also procedural goals

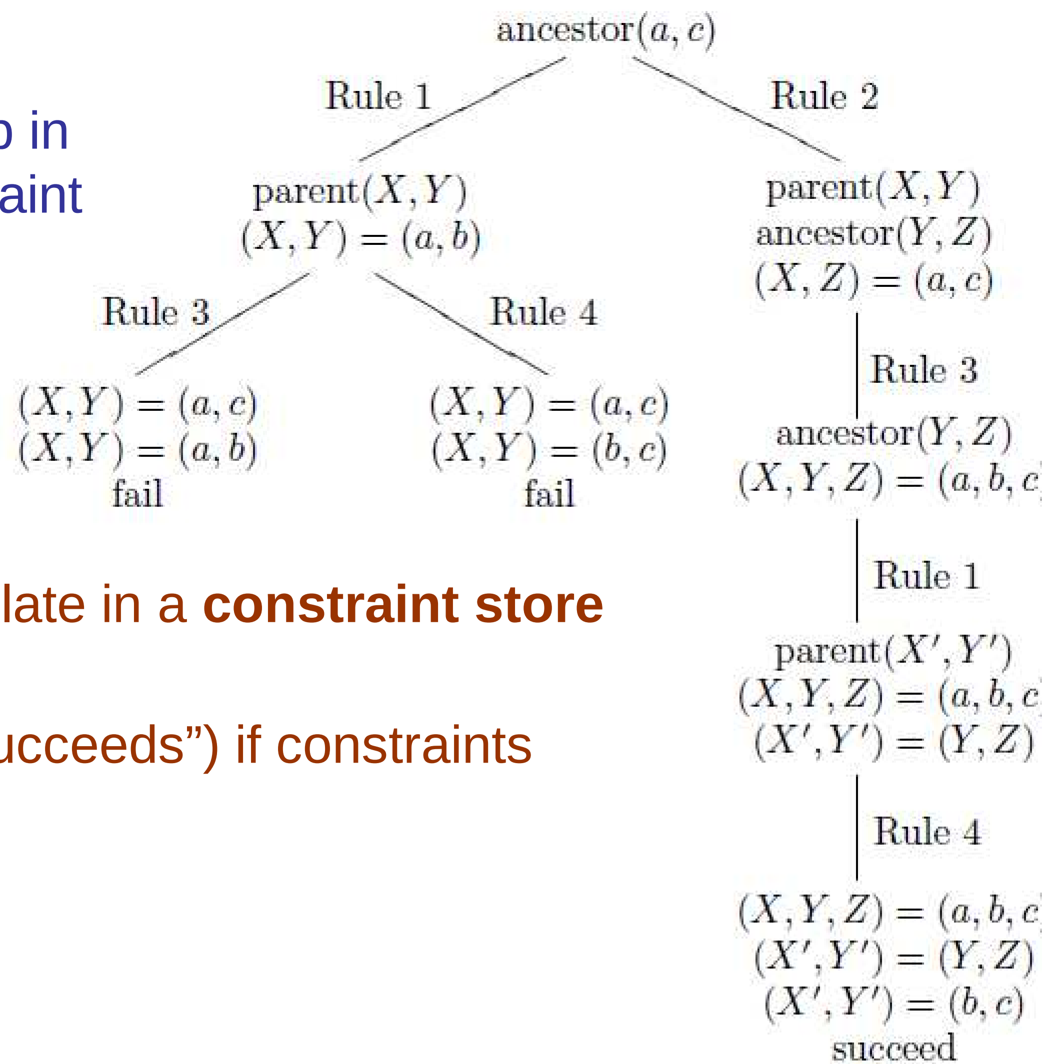



1. $\text{ancestor}(X, Y) \leftarrow \text{parent}(X, Y).$
2. $\text{ancestor}(X, Z) \leftarrow \text{parent}(X, Y), \text{ancestor}(Y, Z).$
3. $\text{parent}(a, b).$
4. $\text{parent}(b, c).$

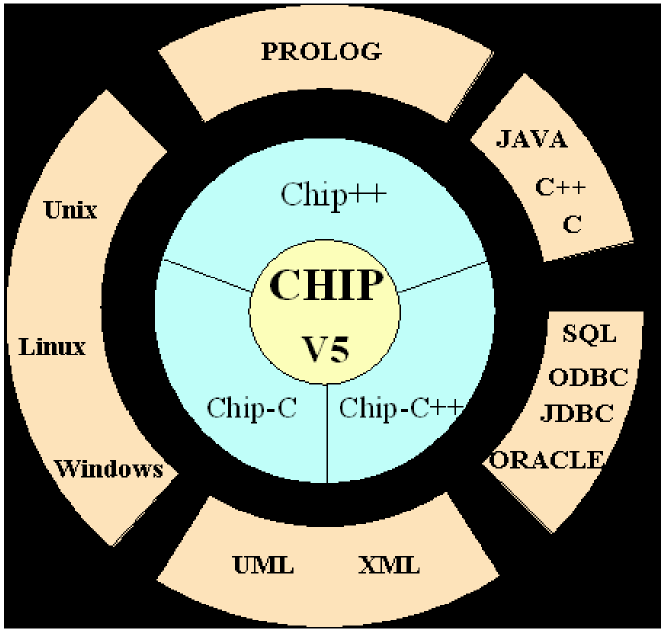
Example of Prolog

# Second Step: Constraint Logic Programming

- Interpret unification step in 1st order logic as constraint solving
  - Extend equality constraints in logic to more general constraints.
  - Constraints accumulate in a **constraint store** at each leaf node.
  - Node is feasible ("succeeds") if constraints have a solution.



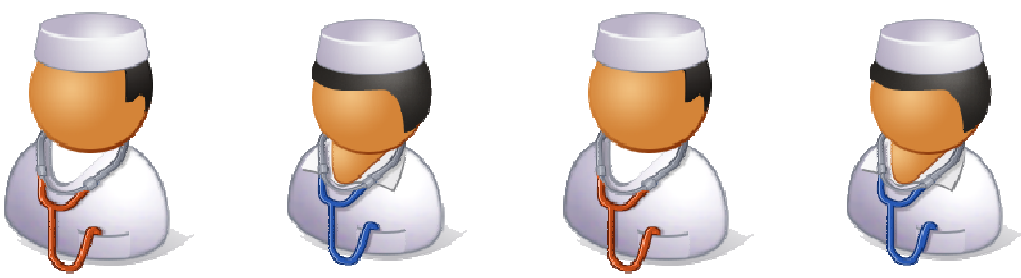
# Third Step: Constraint Programming




- Drop the logic programming framework.
- View each line of the program as specifying both a **constraint** and a **procedure**.
  - **Constraints** are high-level **global constraints**
  - The **procedure** removes infeasible values from variable domains (**filtering, domain consistency maintenance**)
  - Passes reduced domains to next constraint (**constraint propagation**).

# Modeling example: Employee scheduling

- Schedule four nurses in 8-hour shifts.
- A nurse works at most one shift a day, at least 5 days a week.
- Same schedule every week.
- No shift staffed by more than two different nurses in a week.
- A nurse cannot work different shifts on two consecutive days.
- A nurse who works shift 2 or 3 must do so at least two days in a row.



# Two ways to view the problem

## Assign nurses to shifts

| | Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|---|
| Shift 1 | A | B | A | A | A | A | A |
| Shift 2 | C | C | C | B | B | B | B |
| Shift 3 | D | D | D | D | C | C | D |

## Assign shifts to nurses

| | Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|---|
| Nurse A | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| Nurse B | 0 | 1 | 0 | 2 | 2 | 2 | 2 |
| Nurse C | 2 | 2 | 2 | 0 | 3 | 3 | 0 |
| Nurse D | 3 | 3 | 3 | 3 | 0 | 0 | 3 |

0 = day off

Use **both** formulations in the same model!

First, assign nurses to shifts.

Let $w_{sd}$ = nurse assigned to shift $s$ on day $d$

$$\text{alldiff}\left(w_{1d}, w_{2d}, w_{3d}\right), \quad \text{all } d$$

Schedule 3 different nurses on each day

Use **both** formulations in the same model!

First, assign nurses to shifts.

Let $w_{sd}$ = nurse assigned to shift $s$ on day $d$

$$\text{alldiff}\left(w_{1d}, w_{2d}, w_{3d}\right), \quad \text{all } d$$

$$\text{cardinality}\left(w \mid (A,B,C,D),(5,5,5,5),(6,6,6,6)\right)$$

Each nurse works at least 5 and at most 6 days a week

Use **both** formulations in the same model!

First, assign nurses to shifts.

Let $w_{sd}$ = nurse assigned to shift $s$ on day $d$

$$\text{alldiff}\left(w_{1d}, w_{2d}, w_{3d}\right), \quad \text{all } d$$

$$\text{cardinality}\left(w \mid (A,B,C,D),(5,5,5,5),(6,6,6,6)\right)$$

$$\text{nvalues}\left(w_{s,\text{Sun}},\ldots,w_{s,\text{Sat}} \mid 1,2\right), \quad \text{all } s$$

At least 1 and at most 2 nurses
work any given shift.

Remaining constraints are not easily expressed in this notation.

So, assign shifts to nurses.

Let $y_{id}$ = shift assigned to nurse $i$ on day $d$

$$\text{alldiff}\left(y_{1d}, y_{2d}, y_{3d}\right), \ \text{all } d$$

Assign a different nurse to each shift on each day.

Redundant, but speeds solution.

Remaining constraints are not easily expressed in this notation.

So, assign shifts to nurses.

Let $y_{id}$ = shift assigned to nurse *i* on day *d*

$$\text{alldiff}\left(y_{1d}, y_{2d}, y_{3d}\right), \text{ all } d$$

$$\text{stretch}\left(y_{i,\text{Sun}}, \ldots, y_{i,\text{Sat}} \mid (2,3),(2,2),(6,6),P\right), \text{ all } i$$

Shift 2 or 3 must be worked at least 2 days in a row.

Remaining constraints are not easily expressed in this notation.

So, assign shifts to nurses.

Let $y_{id}$ = shift assigned to nurse *i* on day *d*

$$\text{alldiff}\left(y_{1d}, y_{2d}, y_{3d}\right),\ \text{all } d$$
$$\text{stretch}\left(y_{i,\text{Sun}}, \ldots, y_{i,\text{Sat}} \mid (2,3),(2,2),(6,6),P\right),\ \text{all } i$$

Pattern constraint: $P = \{(s,0),(0,s) \mid s = 1,2,3\}$

Don't switch shifts without taking at least one day off.

Connect the $w_{sd}$ variables to the $y_{id}$ variables.

Use **channeling constraints**:

$$w_{y_{id}d} = i, \quad \text{all } i,d$$

$$y_{w_{sd}d} = s, \quad \text{all } s,d$$

Channeling constraints increase propagation and make the problem easier to solve.

The complete model is:

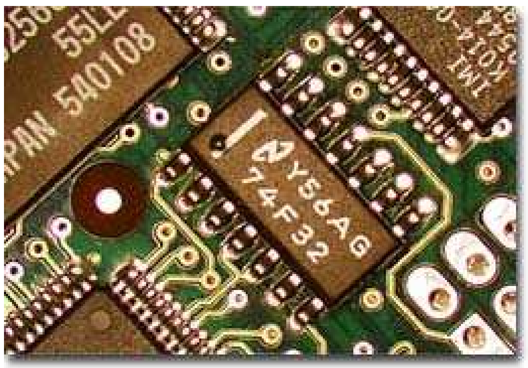
$$\text{alldiff}\left(w_{1d}, w_{2d}, w_{3d}\right), \quad \text{all } d$$
$$\text{cardinality}\left(w \mid (A, B, C, D), (5,5,5,5), (6,6,6,6)\right)$$
$$\text{nvalues}\left(w_{s,\text{Sun}}, \ldots, w_{s,\text{Sat}} \mid 1, 2\right), \quad \text{all } s$$

$$\text{alldiff}\left(y_{1d}, y_{2d}, y_{3d}\right), \quad \text{all } d$$
$$\text{stretch}\left(y_{i,\text{Sun}}, \ldots, y_{i,\text{Sat}} \mid (2,3), (2,2), (6,6), P\right), \quad \text{all } i$$

$$w_{y_{id}d} = i, \quad \text{all } i, d$$
$$y_{w_{sd}d} = s, \quad \text{all } s, d$$

# Early commercial successes

- Circuit design (Siemens)




- Container port scheduling (Hong Kong and Singapore)

- Real-time control (Siemens, Xerox)

# Applications

- Job shop scheduling
- Assembly line smoothing and balancing
- Cellular frequency assignment
- Nurse scheduling
- Shift planning
- Maintenance planning


- Airline crew rostering and scheduling
- Airport gate allocation and stand planning

# Applications

- Production scheduling
    - chemicals
    - aviation
    - oil refining
    - steel
    - lumber
    - photographic plates
    - tires
- Transport scheduling (food, nuclear fuel)
- Warehouse management
- Course timetabling

# CP and Mathematical Programming

## Comparison

| CP | MP |
| --- | --- |
| Logic processing | Numerical calculation |
| Inference (filtering, constraint propagation) | Relaxation |
| High-level modeling (global constraints) | Atomistic modeling (linear inequalities) |
| Branching | Branching |
| Constraint-based processing | “Independence” of model and algorithm |

# Integrating OR and CP

Complementary strengths

How to integrate

Computational advantages

Applications & Software

# Complementary Strengths

- CP:
  - Inference methods
  - Modeling
  - Exploits local structure
  - Good at scheduling
- OR:
  - Relaxation methods
  - Duality theory
  - More robust
  - Good with continuous variables

Let's bring them together!

# How to Integrate

- Constraint propagation + relaxation
  - Propagation reduces search space.
  - Relaxation bounds prune the search
- CP-based column generation
  - In branch-and-price methods
  - CP accommodates complex constraints on columns
- Decomposition methods
  - Distinguish master problem and subproblem
  - MILP solves one, CP the other.
- Use CP-style modeling

# Computational Advantage of Integrating CP and OR

Using CP + relaxation from OR

| *Problem* | *Speedup* | |
|---|---|---|
| Lesson timetabling<br>(LP + red. cost var. fixing) | 2 to 50 times faster than CP | Focacci, Lodi, Milano (1999) |
| Piecewise linear costs<br>(LP relaxation) | 2 to 120 times faster than MILP | Refalo (1999), Yunes, Aron & Hooker (2010) |
| Flow shop scheduling, etc.<br>(LP relaxation + cuts) | 4 to 150 times faster than MILP. | Hooker & Osorio (1999) |

# Computational Advantage of Integrating CP and OR

## Using CP + relaxation from OR

| *Problem* | *Speedup* | |
|---|---|---|
| Product configuration (LP relaxation + cuts) | 30 to 40 times faster than CP, MILP | Thorsteinsson & Ottosson (2001) |
| Automatic recording (Lagrangean relaxation) | 1 to 10 times faster than CP, MILP | Sellmann & Fahle (2001) |
| Stable set problem (semidefinite relaxation) | Better than CP in less time | Van Hoeve (2001) |

# Computational Advantage of Integrating CP and OR

## Using CP + relaxation from OR

| *Problem* | *Speedup* | |
|---|---|---|
| Structural design (nonlinear) (LP quasi-relaxation + logic cuts) | Up to 600 times faster than MILP. | Bollapragada, Ghattas & Hooker (2001) |
| Radiation therapy planning (Lagrangean relaxation) | 10 times faster than CP, MILP | Cambazard, O'Mahony and O'Sullivan (2010) |

# Computational Advantage of Integrating CP and OR

## Using CP-based Branch and Price

| *Problem* | *Speedup* | |
|---|---|---|
| Urban transit crew scheduling | Schedules 210 trips, vs. 120 for traditional branch and price | Yunes, Moura & de Souza (1999) |
| Airline crew rostering | Incorporates complicated work rules | Fahle et al. (2002) |
| Traveling tournament scheduling | First to solve 8-team instance | Easton, Nemhauser & Trick (2002) |

# Computational Advantage of Integrating CP and OR

## Using CP/MILP Benders methods

| *Problem* | *Speedup* | |
|---|---|---|
| Min-cost planning & scheduling | 20 to 1000 times faster than CP, MILP | Jain & Grossmann (2001), Thorsteinsson (2001) |
| Min-cost planning & scheduling | Solved some MIP-intractable instances in <1 sec | Yunes, Aron & Hooker (2010) |
| Polypropylene batch scheduling at BASF | Solved previously insoluble problem in 10 min | Timpe (2002) |

# Computational Advantage of Integrating CP and OR

## Using CP/MILP Benders methods

| *Problem* | *Speedup* | |
|---|---|---|
| Call center scheduling | Solved twice as many instances as traditional Benders | Benoist, Gaudin, Rottembourg (2002) |
| Min-cost, min-makespan planning & cumulative scheduling | 100-1000 times faster than CP, MILP | Hooker (2004) |
| Min tardiness planning & cumulative scheduling | 10-1000 times faster than CP, MILP | Hooker (2005) |

# Computational Advantage of Integrating CP and OR

## Using CP/MILP Benders methods

| *Problem* | *Speedup* | |
|---|---|---|
| Sports scheduling | Several orders of magnitude speedup vs MILP | Rasmussen & Trick (2007) |
| Single-facility scheduling | Schedules several times as many jobs as MILP. Faster and/or more robust than CP. | Coban & Hooker (2012) |

# Applications of Integrated CP and OR

## Using CP + relaxation from OR

| *Application* | |
|---|---|
| Orthogonal Latin squares | Appa, Magos & Mourtos (2002) |
| Truss structure design | Bollapragada, Gattas & Hooker (2001) |
| Chemical processing network design | Grossmann et al. (1994), Hooker & Osorio (1999) |
| Vehicle routing | Sadykov (2004) |
| Resource-constrained scheduling | Demassy, Artiques & Michelon (2002), Berthold et al. (2010) |
| Multiple machine scheduling | Bockmayr & Pisaruk (2003) |
| Shuttle transit routing | Quadrifoglio, Dessouky & Ordóñez (2008) |
| Boat party scheduling | Hooker & Osorio (1999) |

# Applications of Integrated CP and OR

## Using CP + relaxation from OR

| *Application* | |
|---|---|
| Multidimensional knapsack problem | Osorio & Glover (2001) |
| Factory retrofit planning | Sawaya & Grossmann (2005) |
| Strip packing | Sawaya & Grossmann (2005) |
| Chemical process engineering | Lee & Grossmann (2003), Vecchietti et al (2001), Swaya & Grossmann (2007) |
| Transport & production problems with piecewise linear costs | Refalo (1999), Ottosson et al. (1999) |
| TSP with time windows | Milano & van Hoeve (2002) |
| Product configuration | Milano & van Hoeve (2002) |

# Applications of Integrated CP and OR

## Using CP + relaxation from OR

| *Application* | |
|---|---|
| Network design | Cronholm & Ajili (2004) |
| Automatic digital recording | Sellmann & Fahle (2001) |
| Traveling tournament problem | Benoist, Laburthe & Rottembourg (2001) |
| Resource-constrained shortest path problem | Gellermann, Sellmann & Wright (2005) |
| Radiation therapy planning | Cambazard, O'Mahony & O'Sullivan (2010) |
| CP domain filtering | Khemmoudj, Bennaceur & Nagih (2005) |

# Applications of Integrated CP and OR

## Using CP-based branch and price

| *Application* | |
|---|---|
| Airline crew rostering | Junker et al. (1999), Kohl (2000), Chabrier (2000), Fahle et al. (2002), Sellmann et al. (2002) |
| Aircraft scheduling | Grönqvist (2003) |
| Bus crew scheduling | Yunes, Moura & de Souza (2005) |
| Vehicle routing | Rousseau, Gendreau & Pesant (2002) |
| Network design | Chabrier (2003) |
| Employee timetabling | Demassey, Pesant & Rousseau (2005) |
| Physician scheduling | Gendron, Lebbah & Pesant (2005) |
| Radiation therapy planning | Cambazard, O'Mahony & O'Sullivan (2010) |
| Traveling tournament problem | Easton, Nemhauser & Trick (2002), Trick & Yildiz (2007) |

# Applications of Integrated CP and OR

## Using CP/MILP Benders methods

| *Application* | |
|---|---|
| Logic circuit verification | Hooker & Yan (1995) |
| Propositional satisfiability | Hooker (2000), Hooker & Ottosson (2003) |
| Planning & scheduling | Jain & Grossmann (2001), Hooker (2000,2004,2005), Harjunkoski & Grossmann (2002), Chu & Xia (2005) |
| Dispatch of automated vehicles | Corréa, Langevin & Rousseau (2004) |
| Steel production scheduling | Harjunkoski & Grossmann (2001) |
| Chemical batch scheduling | Timpe (2002), Maravelias & Grossmann (2004) |
| Computer processor scheduling | Cambazard et al. (2004), Benini et al. (2005,2008), |

# Applications of Integrated CP and OR

## Using CP/MILP Benders methods

| *Application* | |
|---|---|
| Location-allocation | Fazel-Zarandi & Beck (2009) |
| Traffic diversion | Xia, Eremin & Wallace (2004) |
| Transport network design | Peterson & Trick (2009) |
| Queuing design & control | Terekhov, Beck & Brown (2007) |
| Sports scheduling | Rasmussen & Trick (2007), Cheung (2009) |

# Software for Integrated Methods

- ECLiPSe
  - Exchanges information between ECLiPSEe solver, Xpress-MP
- OPL Studio (IBM)
  - Combines CPLEX and ILOG CP Optimizer with script language
- Mosel (FICO)
  - Combines Xpress-MP, Xpress-Kalis with low-level modeling
- SIMPL (CMU)
  - Full integration with high-level modeling (prototype)
- SCIP (ZIB)
  - Combines MILP and CP-based propagation
- G12 (NICTA)
  - Converts generic model to one that invokes cooperating solvers
- BARON (global optimization)
  - Combines convexification and interval propagation

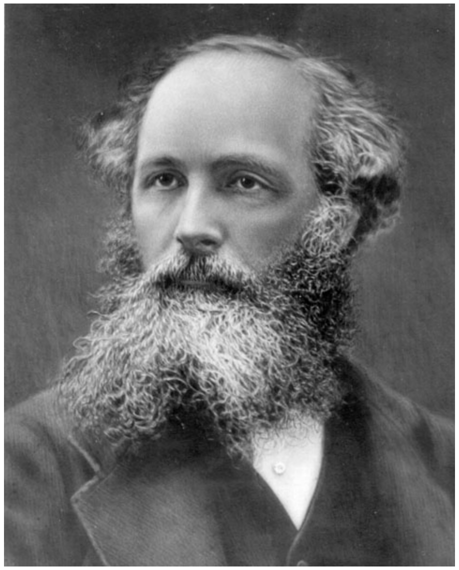
# A Unifying Principle – Inference Duality

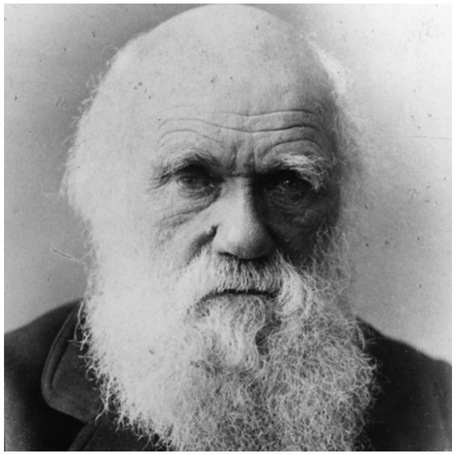
Optimization duals
Constraint-directed search
Example: Conflict clauses in SAT
Application to new problems

# Unifying Theories

- **Synthesis** – 19$^{th}$ century
  - Maxwell's equations
  - Evolution by natural selection
- **Analysis** – 20$^{th}$ century
  - Except in reductive schemes (DNA, TOE in physics)
  - Abstraction is **not** synthesis (Bourbaki school)
- **Return** to synthesis in 21$^{st}$ century?

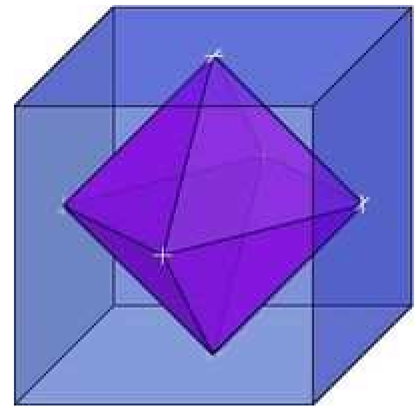
- **Duality** – a unifying principle for OR/CP/AI.

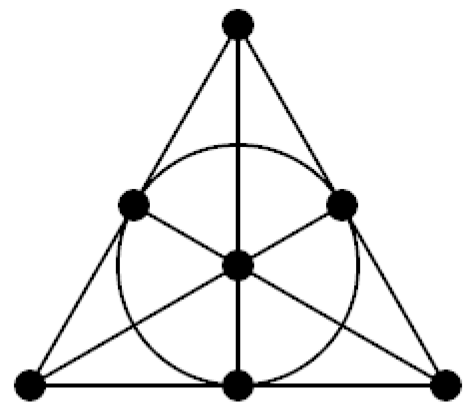




# Duality in Mathematics

- Most mathematical duals are symmetric
  - Dual of dual = original
- Polar duality of polytopes
  - Facets / vertices
- Duality in projective geometry
  - Points / lines
- Duality of vector spaces
  - Row rank / column rank





**Fano plane**

# Optimization Duals

- Generally not symmetric:
  - Surrogate dual
  - Lagrangean dual
  - Superadditive dual
  - LP dual is an exception
- All are **inference duals** and **relaxation duals**.

## Inference Dual

- Primal problem:
  - Find a feasible solution
- Dual problem:
  - Find a **proof** of optimality
- Dual is defined by **logical inference method**
  - Nonnegative linear combination + domination → **surrogate dual**
  - Nonnegative linear combination + different type of domination → **Lagrangean dual**
  - **LP dual** is special case of both

# Linear Programming Duality

An LP can be viewed as an inference problem…

$$\begin{aligned} \min\ & cx \\ & Ax \geq b \\ & x \geq 0 \end{aligned} \quad = \quad \boxed{\begin{array}{l} \max\ v \\ Ax \geq b \overset{x \geq 0}{\Longrightarrow} cx \geq v \end{array}}$$

implies

**Dual** problem: Find the tightest lower bound on the objective function that is **implied** by the constraints.

# Linear Programming Duality

An LP can be viewed as an inference problem…

$$\begin{array}{lcl} \min \; cx & = & \max \; v \\ Ax \geq b & & Ax \geq b \overset{x \geq 0}{\Rightarrow} cx \geq v \\ x \geq 0 & & \end{array}$$

From Farkas Lemma,

$$Ax \geq b \overset{x \geq 0}{\Rightarrow} cx \geq v \quad \text{iff} \quad \lambda Ax \geq \lambda b \; \boxed{\text{dominates}} \; cx \geq v$$

$$\text{for some } \lambda \geq 0$$

$\lambda A \leq c$ and $\lambda b \geq v$

# Linear Programming Duality

An LP can be viewed as an inference problem…

$$\begin{array}{l} \min\ cx \\ Ax \geq b \\ x \geq 0 \end{array} \quad = \quad \begin{array}{l} \max\ v \\ Ax \geq b \overset{x \geq 0}{\Rightarrow} cx \geq v \end{array} \quad = \quad \begin{array}{l} \max\ \lambda b \\ \lambda A \leq c \\ \lambda \geq 0 \end{array}$$

We get **classical LP dual**

From Farkas Lemma:

$$Ax \geq b \overset{x \geq 0}{\Rightarrow} cx \geq v \quad \text{iff} \quad \lambda Ax \geq \lambda b \ \boxed{\text{dominates}}\ cx \geq v \ \text{ for some } \lambda \geq 0$$

$\lambda A \leq c$ and $\lambda b \geq v$

# Constraint-based Search

- **Inference dual** is the basis for a wide variety of **constraint-based search** methods
  - Branching and dynamic backtracking.
  - Benders decompositions and its generalizations.
  - Satisfiability solvers.

# Constraint-based Search

- Most combinatorial methods search over **partial solutions**.
  - Nodes of a branching tree.
  - Solutions of Benders master problem.
  - Partial solution defines a **subproblem**.
- Dual solution (proof) of subproblem defines a **nogood constraint**.
  - A form of **learning**.
  - What can deduced from the **same proof schema** if the premises change?
  - That is, for different partial solutions?

# Constraint-based Search

- Nogood constraints:
  - Rule out infeasible partial solutions.
  - Or bound value of future partial solutions.
- Example: Benders cuts
  - Classical cuts from LP dual of subproblem.
- Example: Conflict clauses in SAT
  - Based on unit resolution proof at leaf node.

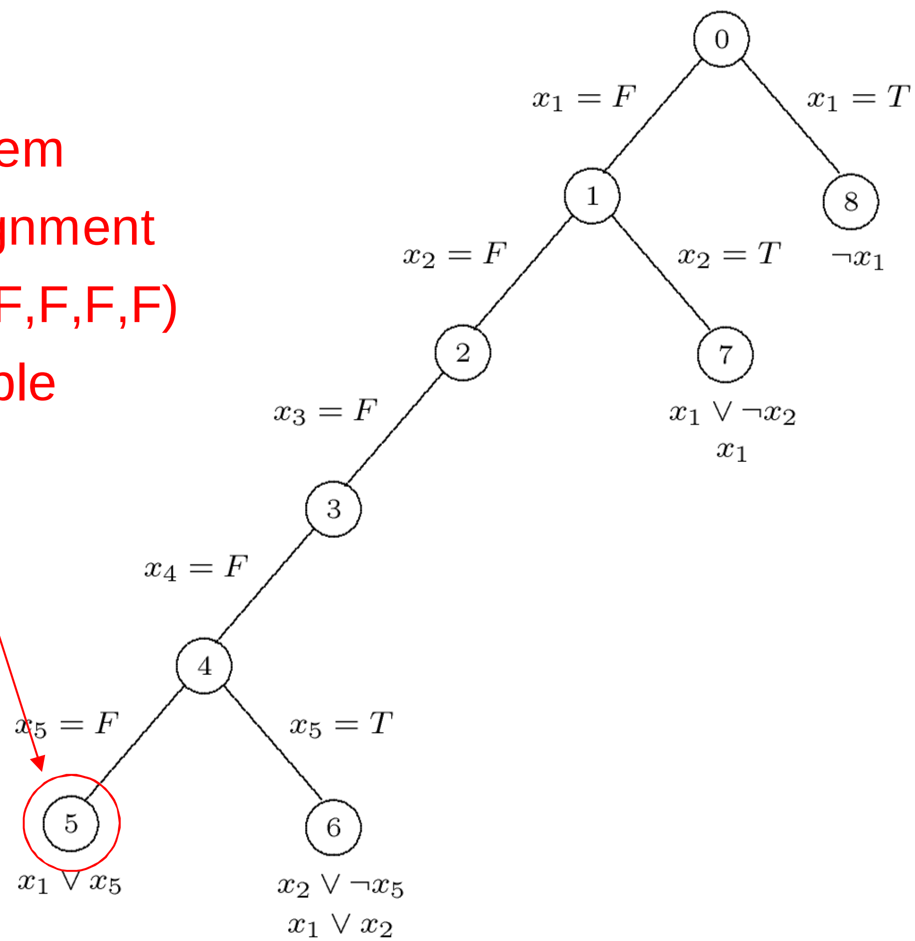
## Conflict clauses in SAT

Example:

$$
\begin{array}{rrrrrrrrrrr}
x_1 & & & & & & \vee & x_5 & \vee & x_6 \\
x_1 & & & & & & \vee & x_5 & \vee & \neg x_6 \\
 & x_2 & & & & & \vee & \neg x_5 & \vee & x_6 \\
 & x_2 & & & & & \vee & \neg x_5 & \vee & \neg x_6 \\
\neg x_1 & & \vee & x_3 & \vee & x_4 & & & & \\
 & \neg x_2 & \vee & x_3 & \vee & x_4 & & & & \\
\neg x_1 & & \vee & \neg x_3 & & & & & & \\
\neg x_1 & & & & \vee & \neg x_4 & & & & \\
 & \neg x_2 & \vee & \neg x_3 & & & & & & \\
 & \neg x_2 & & & \vee & \neg x_4 & & & &
\end{array}
$$

# Conflict clauses in SAT

SAT problem
+ partial assignment
$(x_1,\ldots,x_5) = (F,F,F,F,F)$
is infeasible



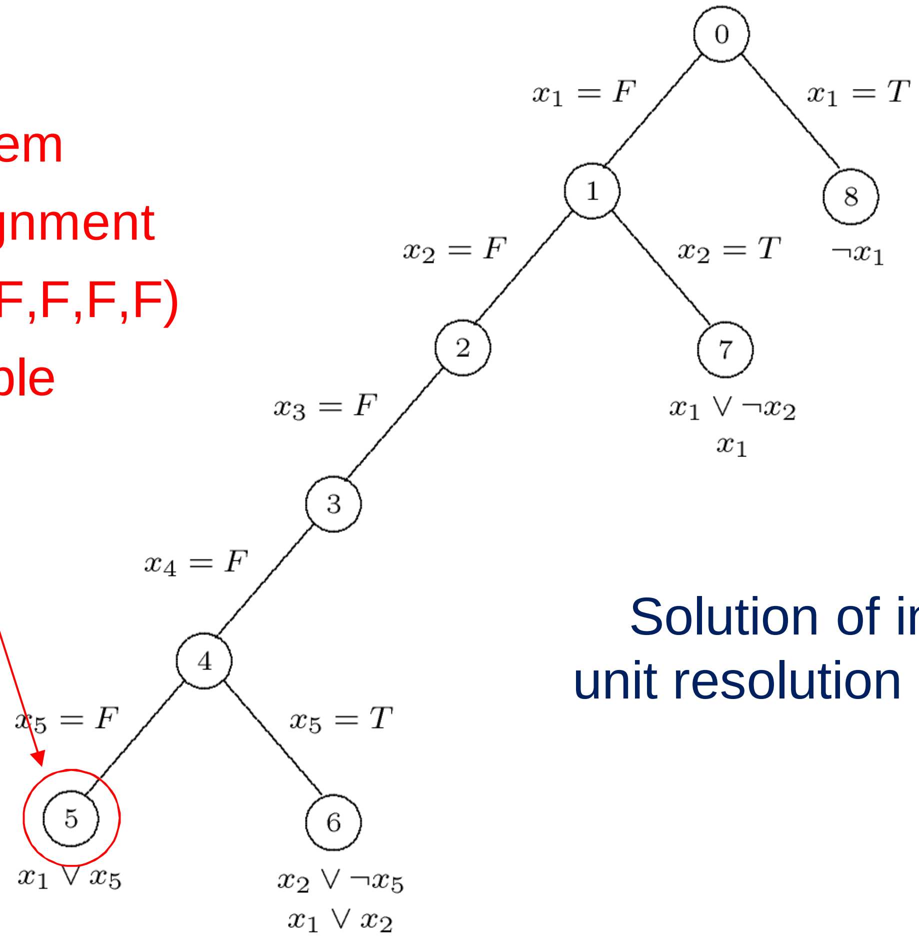
# Conflict clauses in SAT

SAT problem

+ partial assignment

$(x_1,\ldots,x_5) = (F,F,F,F,F)$

is infeasible




Solution of inference dual is a unit resolution proof of infeasibility.

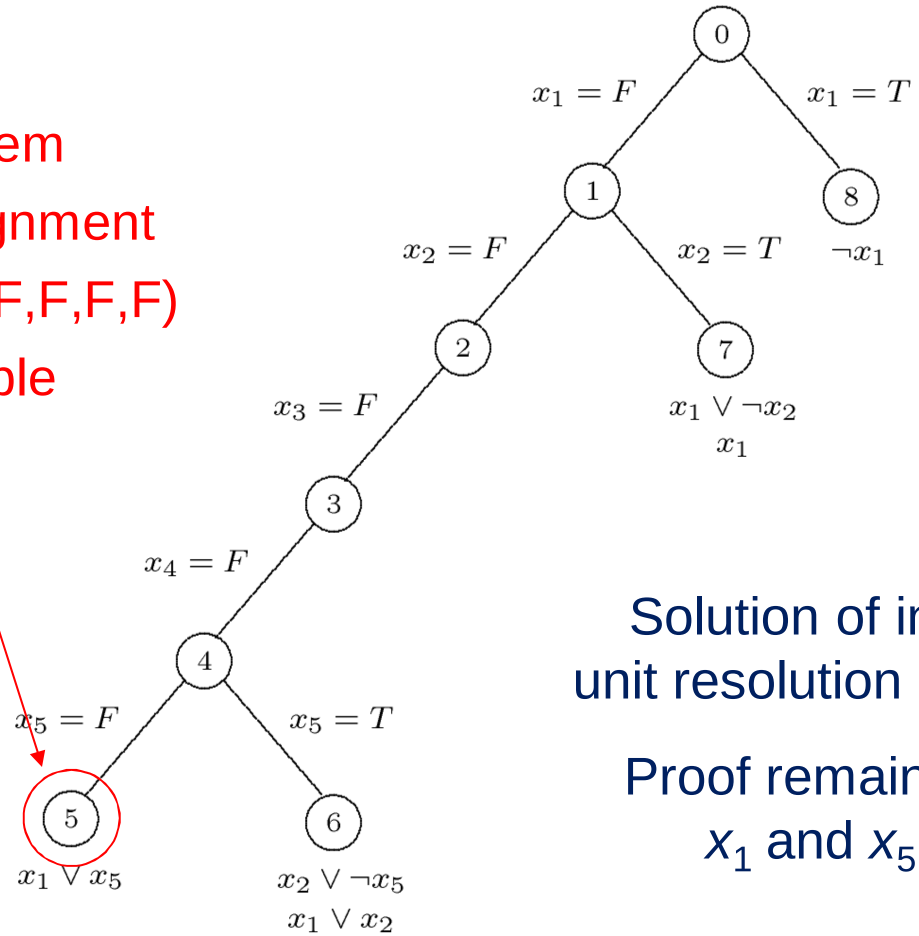
# Conflict clauses in SAT

SAT problem

+ partial assignment

$(x_1,\ldots,x_5) = (F,F,F,F,F)$

is infeasible

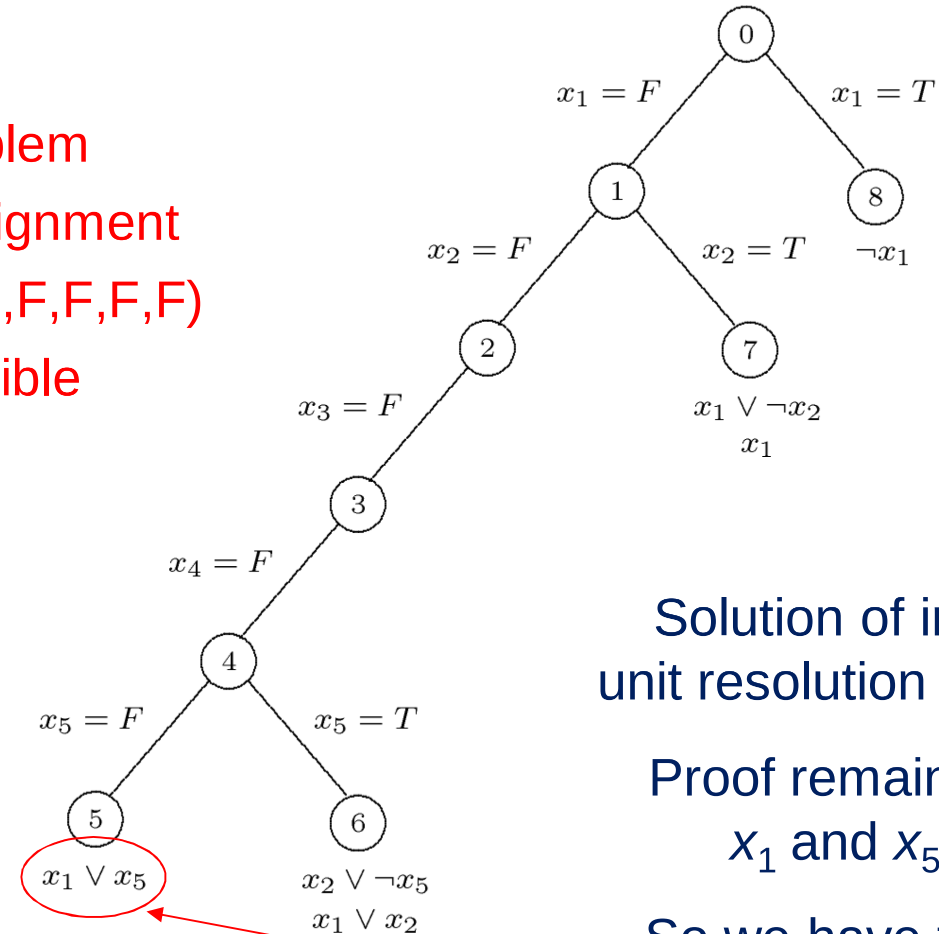



Solution of inference dual is a unit resolution proof of infeasibility.

Proof remains valid when only $x_1$ and $x_5$ are fixed to F.

# Conflict clauses in SAT

SAT problem
+ partial assignment
$(x_1,\ldots,x_5) = (F,F,F,F,F)$
is infeasible



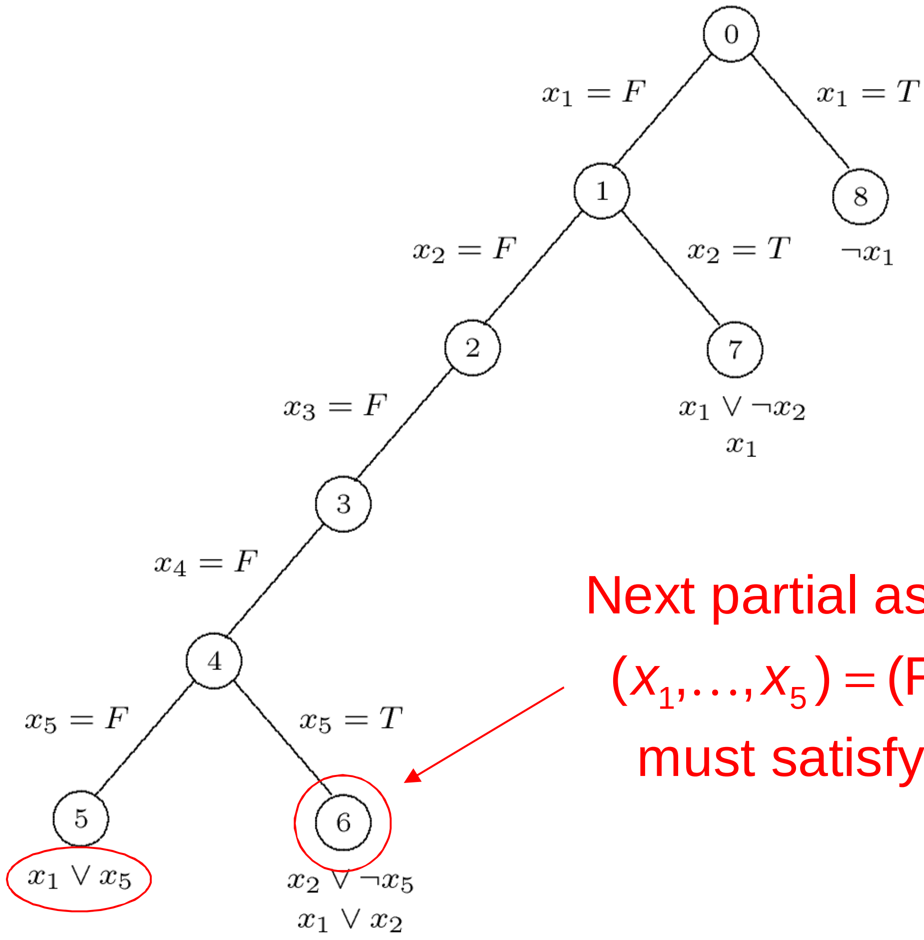

Solution of inference dual is a unit resolution proof of infeasibility.

Proof remains valid when only $x_1$ and $x_5$ are fixed to F.

So we have the conflict clause

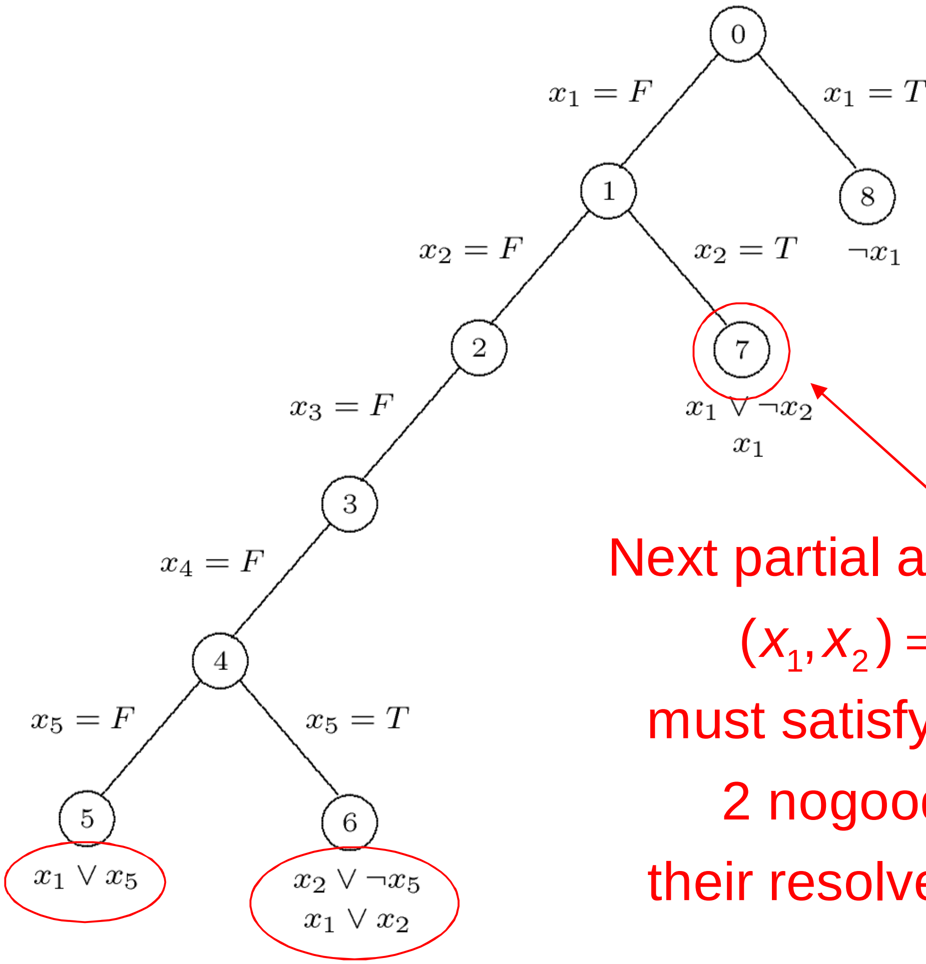
$x_1 \vee x_5$

# Conflict clauses in SAT




Next partial assignment
$(x_1,\ldots,x_5) = (F,F,F,F,T)$
must satisfy nogood

# Conflict clauses in SAT




Next partial assignment

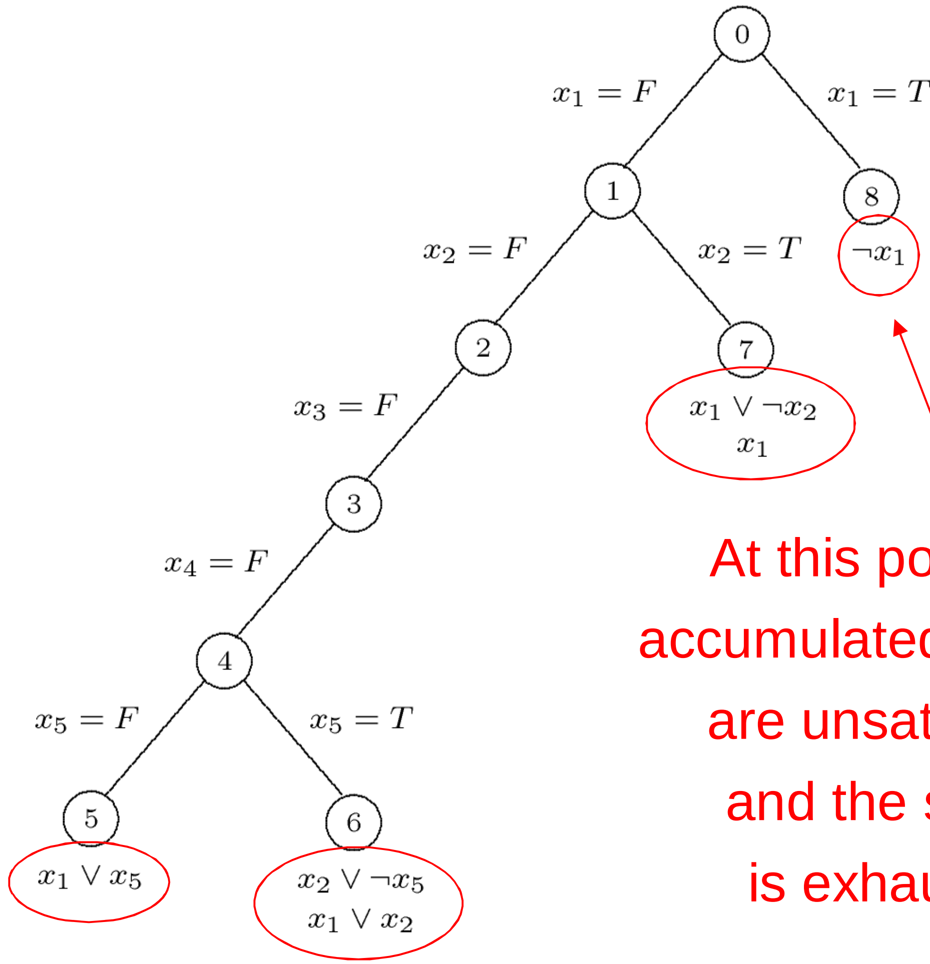
$(x_1, x_2) = (F,T)$

must satisfy previous

2 nogoods and

their resolvent $x_1 \vee x_2$

# Conflict clauses in SAT




At this point the accumulated nogoods are unsatisfiable and the search is exhaustive

# Conflict Clauses in SAT

- In practice…
  - Conflict clauses are obtained from **implication graph**.
- A case of constraint-directed branching
  - Long used in AI (backjumping, dynamic backtracking)
  - Recently used in **MIP solvers**

# Application to New Problems

- Algorithmic payoff
  - In **any** optimization subproblem, inference dual can give rise to nogood constraints
  - Leading to a **new constraint-based search method**.
- Example: Logic-based Benders
  - Applications and speedups mentioned earlier.

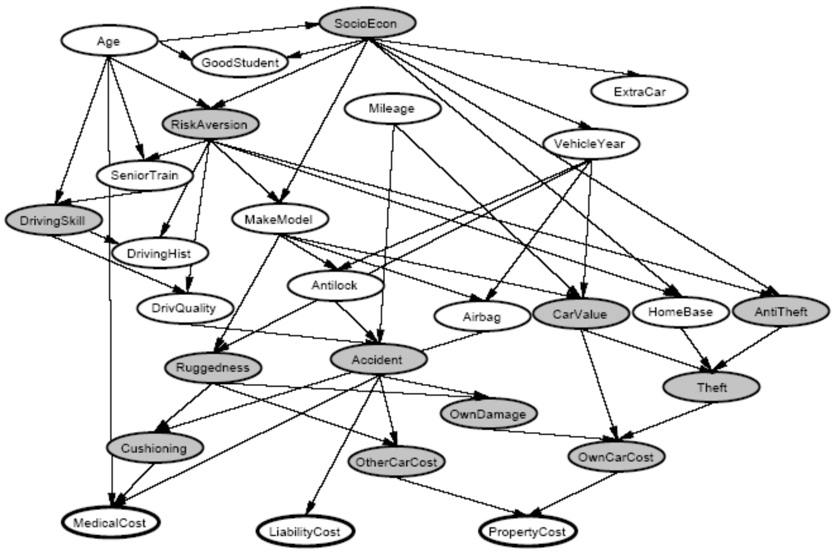
# Other Possibilities for Unification

# Cutting Planes as Logical Inference

- Previous work
  - Resolution as a cutting plane method.
  - Cutting plane proof theory
- Cutting planes and CP
  - How do cutting planes reducing backtracking…
    by helping to achieve domain consistency or *k*-consistency?
  - Almost completely unexplored

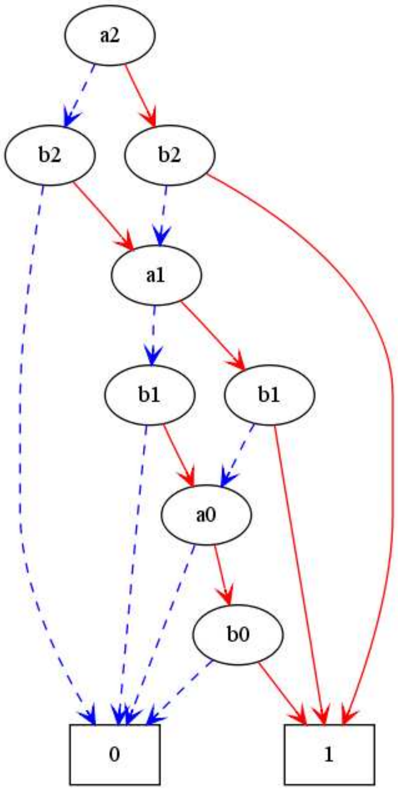
# Dependency Graphs and Induced Width

- Unites ideas from several areas
  - Nonserial dynamic programming, Markov trees, join trees, belief logics, Bayesian networks, database theory, *k*-trees, pseudoboolean optimization, bucket elimination, location theory
- Provides:
  - Measure of complexity
  - Recursive solution algorithms
  - Relaxation and relaxation duals.



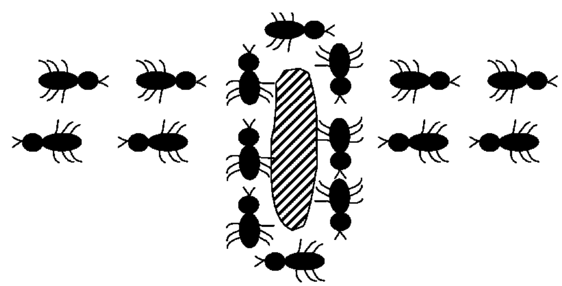
# Graphical Representations of Feasible Sets

- And/or graphs, binary decision diagrams.
- Provide tools for restriction, relaxation
  - Restriction provides a primal heuristic
  - Relaxation provides bounds, etc.
- Structural similarity to continuous relaxation
  - Bounds, pseudocosts, separating cuts
  - Example: MDD relaxations

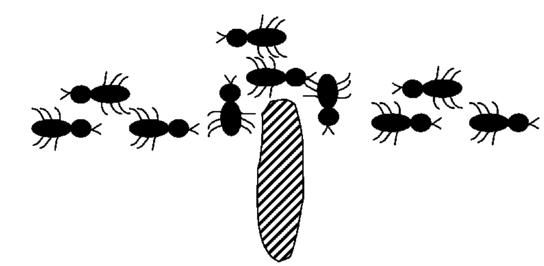


# Exhaustive and Heuristic Search

- Goal: Move gracefully from one to the other
- Unifiying idea:
  - Special cases of the same **restrict-infer-relax** paradigm
  - Example: tabu search as constraint-directed search
- Incorporate **inference** and **relaxation** mechanisms into metaheuristics
  - Simulated annealing, evolutionary algorithms, ant colony optimization, particle swarm optimization.



(c)



(d)

# Stochastic Optimization, DP, and Learning

- Next talk…