Logic, Optimization, and Constraint Programming

A Fruitful Collaboration

John Hooker
Carnegie Mellon University

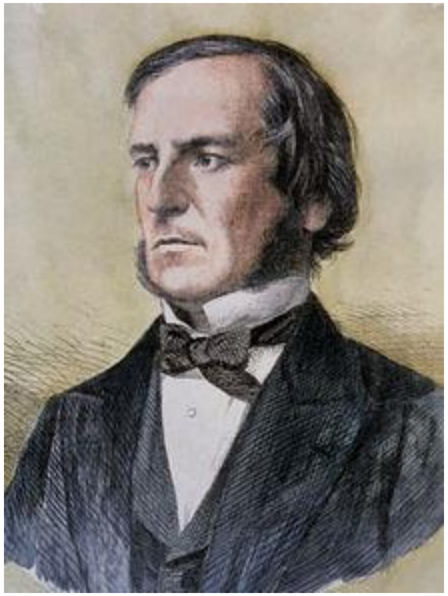
Simons Institute, UC Berkeley

April 2023

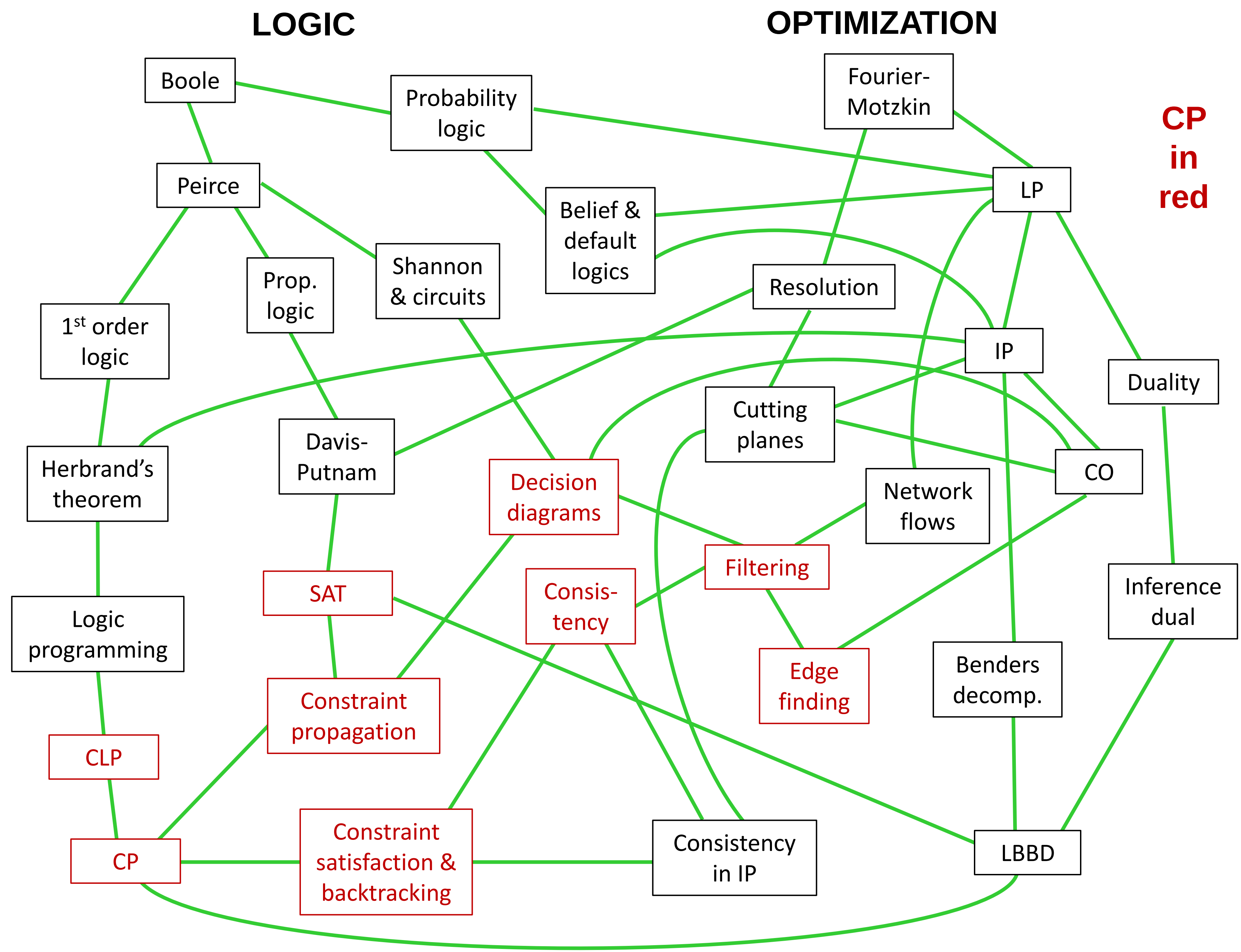
Logic, Optimization, and CP

There are **deep connections** between logic, optimization, and constraint programming (CP) – going back at least to George Boole.

This is a **broad overview** of these connections, as they developed over the 170-year period from Boole's work to today's research.

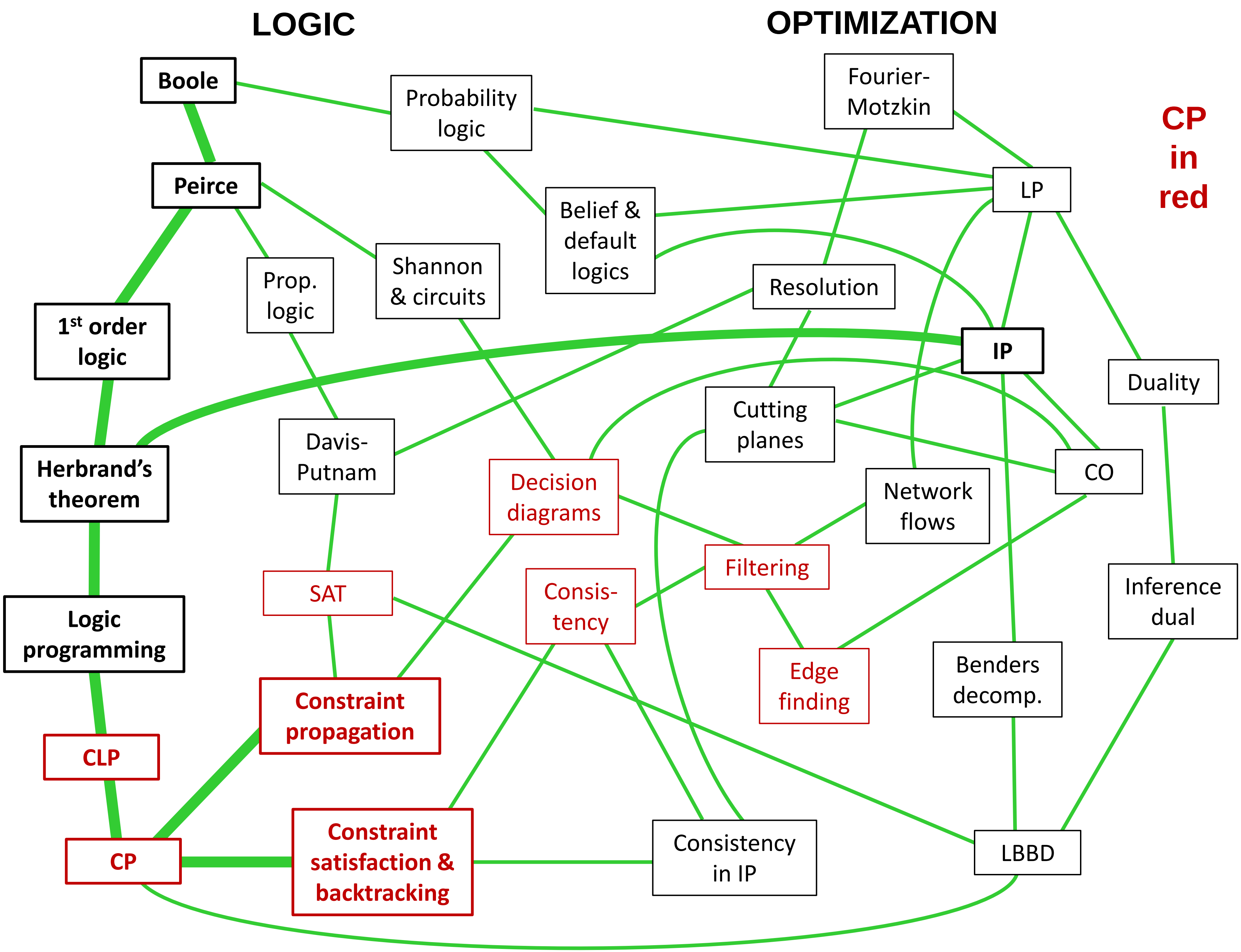


Collaboration among these fields could provide a fruitful trajectory for **future** research.



LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Fourier-Motzkin
Peirce
Belief & default logics
LP
Shannon & circuits
Prop. logic
Resolution
1st order logic
IP
Duality
Cutting planes
Herbrand's theorem
Davis-Putnam
Decision diagrams
CO
Network flows
Filtering
SAT
Consis-tency
Inference dual
Logic programming
Edge finding
Benders decomp.
Constraint propagation
CLP
Constraint satisfaction & backtracking
CP
Consistency in IP
LBBD

From Boole to Logic Programming & CP


LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Fourier-Motzkin
Peirce
Belief & default logics
LP
Prop. logic
Shannon & circuits
Resolution
1st order logic
IP
Duality
Cutting planes
Herbrand's theorem
Davis-Putnam
Decision diagrams
CO
Network flows
Filtering
SAT
Consis-tency
Inference dual
Logic programming
Edge finding
Benders decomp.
Constraint propagation
CLP
Constraint satisfaction & backtracking
Consistency in IP
LBBD
CP

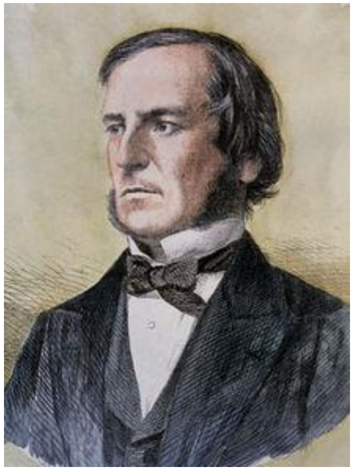
From Boole to Logic Programming & CP

Gottfried Wilhelm Leibniz
1646-1716

George Boole advanced a project begun by Leibniz, although Boole (largely self-taught) was initially unaware of Leibniz's work.

Leibniz believed that all of science can be formulated in a **logical language** (*characteristica universalis*) in which implications can be obtained by **calculation** (*calculus ratiocinator*), such as the calculus of infinitesimals.

Boole devised a language in which logical deductions can be **calculated**.



George Boole
1815-1864

From Boole to Logic Programming & CP

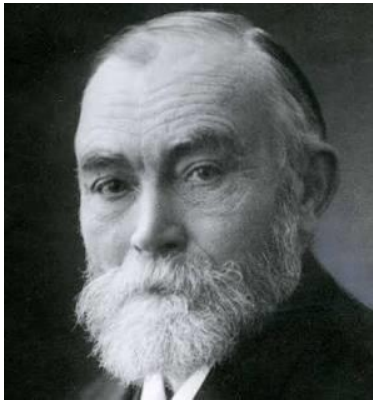
Boole's work was largely **forgotten** for a century.

But it was studied by philosopher **Charles Sanders Peirce** in the late 19^{th} century.

Boole introduced **multi-place predicates**, to which Pearce added **logical quantifiers** ("for all," "for some").

Gottlob Frege developed a fully formed **first-order logic** in the 1890s.

C. S. Peirce
1839-1914



Gottlob Frege
1848-1925

From Boole to Logic Programming & CP

Löwenheim, Skolem, Herbrand and others developed systematic **semantics** for first-order logic. They proved fundamental theorems, including Herbrand's **compactness theorem**.

There is an almost identical theorem in infinite-dimensional **integer programming**.

.

Leopold Löwenheim
1878-1957

Thoralf Skolem
1887-1963

Jacques Herbrand
1908-1931

From Boole to Logic Programming & CP

Herbrand's theorem (compactness)

A formula of first order logic is unsatisfiable if and only if some **finite set** of ground instances of the formula is unsatisfiable.

Compactness theorem for integer programming

An infinite set of linear inequalities with integer variables is unsatisfiable if and only of some **finite subset** is unsatisfiable.

Proof?

The 2 theorems are structurally almost identical and have almost exactly the same proof.

From Boole to Logic Programming & CP

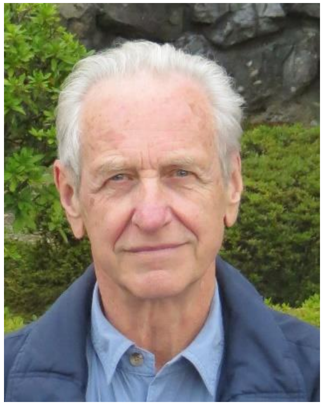
Logic programming arose from an effort to combine declarative and procedural modeling in quantified logic.

A **logic program** can be read as a **declarative** statement of the problem, as well as a **procedure** for obtaining the solution.

This later became a fundamental idea of **constraint programming**.

.

Alain Colmerauer
1941-2017



Robert Kowalski
1941-

From Boole to Logic Programming & CP

A key step in first order logic is **unification**, which finds variable substitutions that make two instantiations of a formula identical. This is essentially a **constraint solving** problem.

likes(Sue, X), likes(Y, Bob)
unified by setting X = Bob, Y = Sue

Logic programming was extended to **constraint logic programming** in **Prolog II**, which added disequations to the unification step. Other forms of constraint solving were added later.

Prolog

CLP(R)

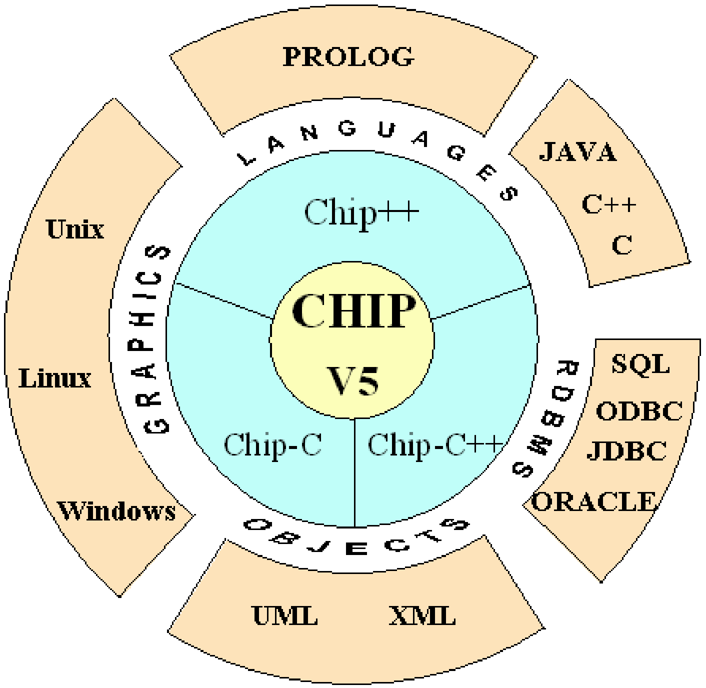
From Boole to Logic Programming & CP

Constraint programming "toolkits" retained constraint solving in a procedural/declarative framework, without requiring a strict logic programming formalism.

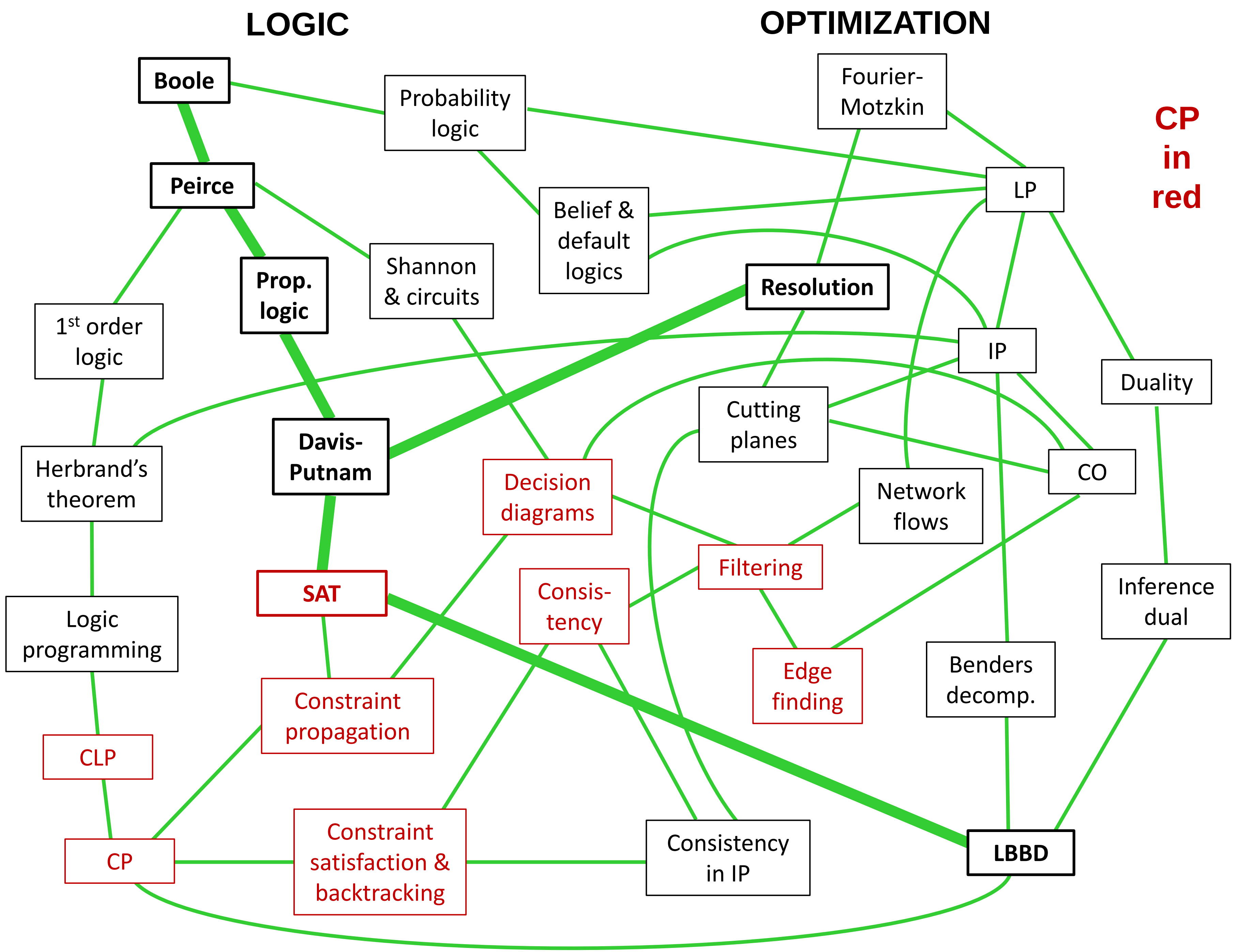
This led to **CP-style modeling** with **finite domains** and **global constraints**.

Constraint propagation allows efficient inference from **constraint sets**.

The **constraint satisfaction** literature studied **consistency** concepts and their connection with **backtracking** (more on this later).



From Boole to SAT


LOGIC
OPTIMIZATION
CP
in
red
Boole
Probability logic
Fourier-Motzkin
Peirce
LP
Belief & default logics
Shannon & circuits
Prop. logic
Resolution
1st order logic
IP
Duality
Cutting planes
Herbrand's theorem
Davis-Putnam
Decision diagrams
CO
Network flows
Filtering
SAT
Consis-tency
Inference dual
Logic programming
Edge finding
Benders decomp.
Constraint propagation
CLP
Constraint satisfaction & backtracking
Consistency in IP
CP
LBBD

From Boole to SAT

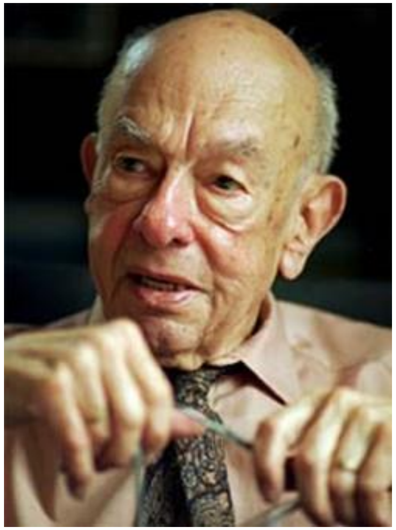
Much of Boole's and Pearce's work dealt with "Boolean algebra," which is essentially **propositional logic** ("ground level" propositions).

The philosopher W. V. Quine proposed (1950s) a **consensus** method for simplifying propositional formulas that is a complete inference method for propositional logic.

When applied to CNF rather than DNF, the method is **resolution**.

Resolution:

$$\frac{\begin{array}{l} x_1 \vee x_2 \vee x_4 \\ x_1 \vee \neg x_4 \end{array}}{x_1 \vee x_2}$$



W. V. Quine
1908-2000

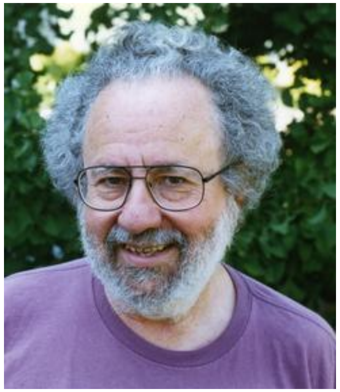
From Boole to SAT

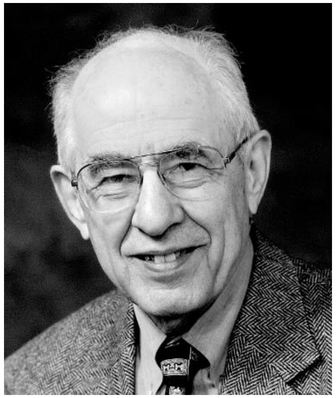
The **Davis-Putnam algorithm**, devised to check validity in first-order logic, applies resolution to instantiated (ground level) propositions.

Resolution was later replaced with more efficient methods for checking satisfiability of CNF formulas, such as **branching** in the David-Putnam-Loveland-Logemann (DPLL) method.

These led to today's highly efficient **SAT** methods, which use **watched literals**, **conflict clauses**, etc.

Martin Davis
1928-2023





Hilary Putnam
1926-2016

.

From Boole to SAT

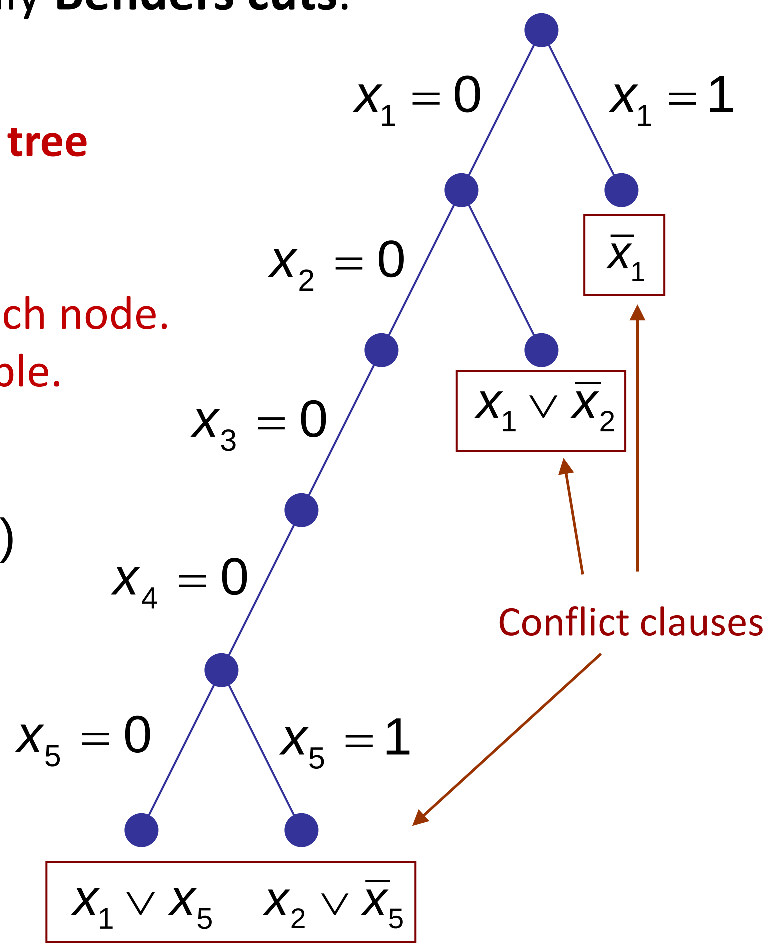
Conflict clauses lie at the heart of SAT algorithms.
We will see later that they are actually **Benders cuts**.

Refutation using DPLL tree and conflict clauses:

$$x_1 \vee x_5 \vee x_6$$
$$\bar{x}_6 \vee x_1 \vee x_5$$
$$\bar{x}_5 \vee x_2 \vee x_6$$
$$\bar{x}_5 \vee \bar{x}_6 \vee x_2$$
$$(\bar{x}_1 . \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee x_3 \vee x_4)$$
$$(\bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_3 \vee \bar{x}_2) \wedge (\bar{x}_4 \vee \bar{x}_1) \wedge (\bar{x}_4 \vee \bar{x}_2)$$

Apply unit resolution at each node.
Backtrack when unsatisfiable.

Confict clauses enable **backjumping** and reduce search.



From Boole to SAT

Conflict clauses lie at the heart of SAT algorithms.
We will see later that they are actually **Benders cuts**.

Refutation using DPLL tree and conflict clauses:

$x_1 \vee x_5 \vee x_6$

$\bar{x}_6 \vee x_1 \vee x_5$

$\bar{x}_5 \vee x_2 \vee x_6$

$\bar{x}_5 \vee \bar{x}_6 \vee x_2$

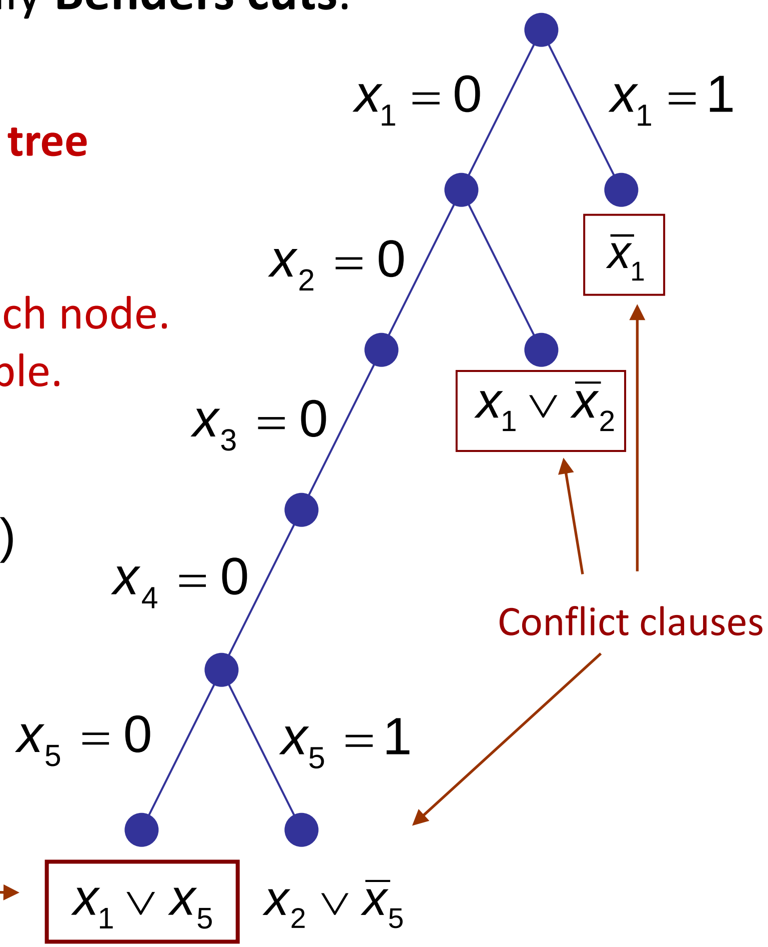
$(\bar{x}_1.\vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee x_3 \vee x_4)$

$(\bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_3 \vee \bar{x}_2) \wedge (\bar{x}_4 \vee \bar{x}_1) \wedge (\bar{x}_4 \vee \bar{x}_2)$

Apply unit resolution at each node.
Backtrack when unsatisfiable.

Confict clauses enable **backjumping** and reduce search.

We will **derive** this conflict clause

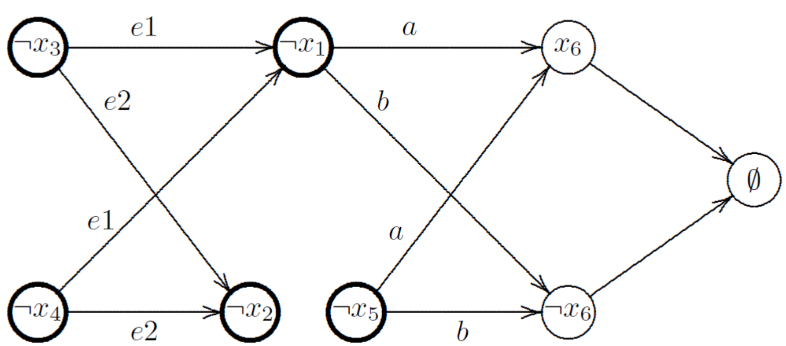


From Boole to SAT

Conflict clause $x_1 \vee x_5$ is obtained from unit refutation by analyzing the implication graph at that node.

Implication graph

Darker circles indicate branching literals

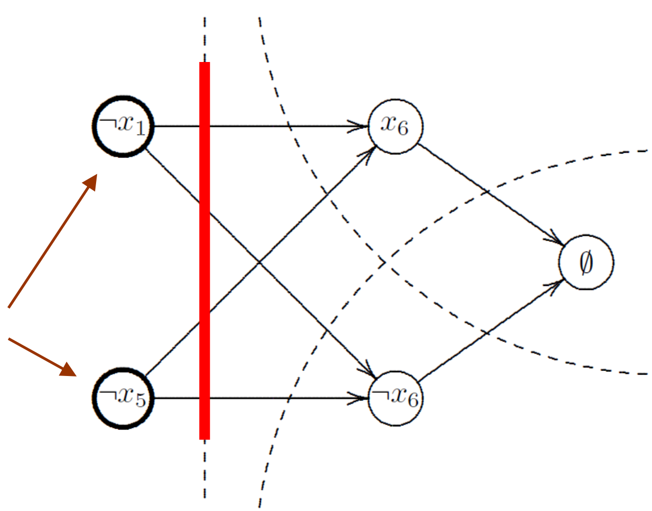


$x_1 \vee x_5 \vee x_6$	(a)
$x_1 \vee x_5 \vee \bar{x}_6$	(b)
$x_2 \vee \bar{x}_5 \vee x_6$	(c)
$x_2 \vee \bar{x}_5 \vee \bar{x}_6$	(d)
$\bar{x}_1 \vee x_3 \vee x_4$	$(e1)$
$\bar{x}_2 \vee x_3 \vee x_4$	$(e2)$
$\bar{x}_3 \vee \bar{x}$	$(f1)$
$\bar{x}_3 \vee \bar{x}_2$	$(f2)$
$\bar{x}_4 \vee \bar{x}_1$	$(f3)$
$\bar{x}_4 \vee \bar{x}_2$	$(f4)$

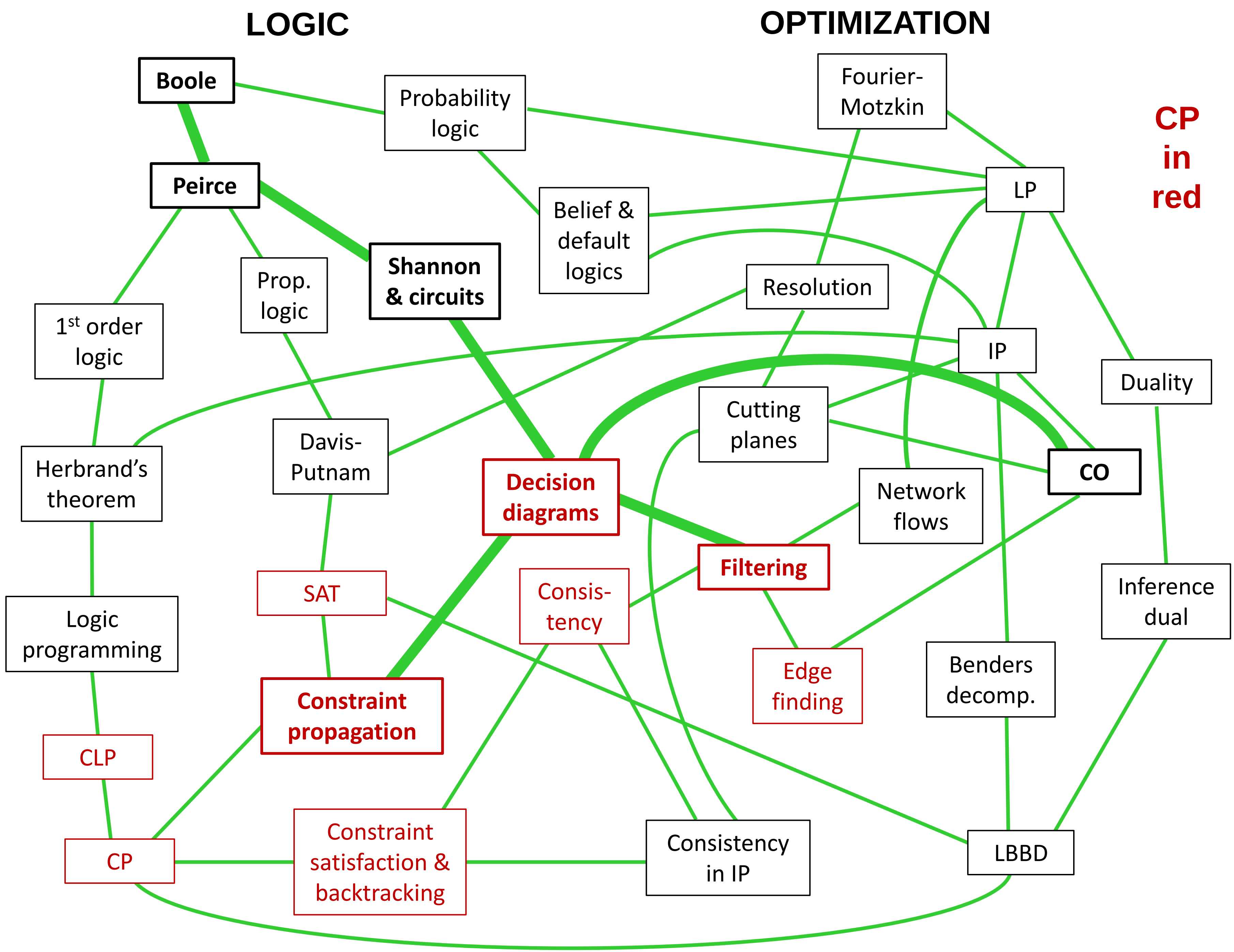
Conflict graph
from implication graph

Literals on "reason side" indicate conflict clause

$x_1 \vee x_5$



From Boole to Decision Diagrams



LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Fourier-Motzkin
Peirce
Belief & default logics
LP
Shannon & circuits
Prop. logic
Resolution
1st order logic
IP
Duality
Cutting planes
Herbrand's theorem
Davis-Putnam
Decision diagrams
CO
Network flows
Filtering
SAT
Consis-tency
Inference dual
Logic programming
Edge finding
Benders decomp.
Constraint propagation
CLP
Constraint satisfaction & backtracking
Consistency in IP
LBBD
CP

From Boole to Decision Diagrams

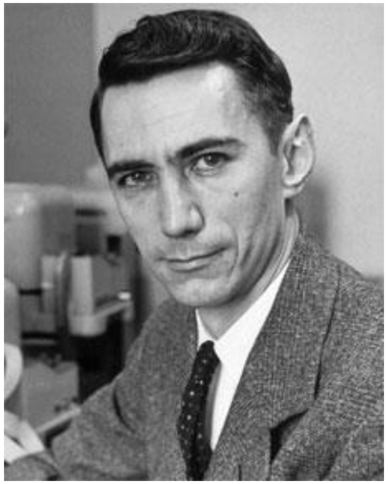
C. S. Peirce applied Boolean methods to **electrical switching circuits … in 1886**!

This work was again **forgotten** for decades.

Claude Shannon was required to take a philosophy course at the University of Michigan in the 1930s, which exposed him to Peirce's work.

This gave him the idea for his famous master's thesis at MIT (1937), in which he applied Boolean logic to **electronic switching circuits**.

This gave rise to the **computer age**.



Claude Shannon
1916-2001

From Boole to Decision Diagrams

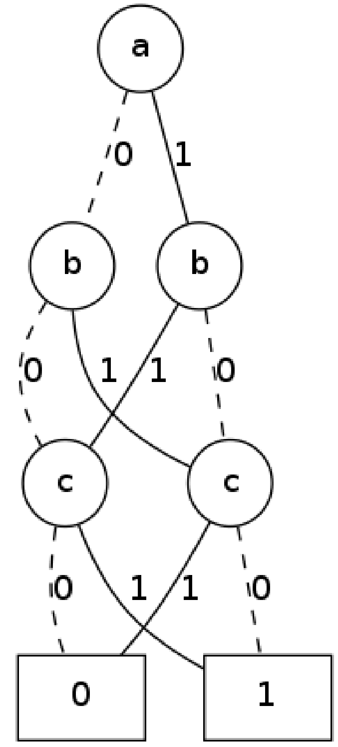
Meanwhile, C. Y. Lee (1959) proposed **binary-decision programs** as a means of calculating the output of switching circuits.

S. B. Akers (1978) later represented these as **binary decision diagrams**.

Randy Bryant (1986) showed that **ordered BDDs** provide a unique minimal representation of a Boolean function.

This led to applications in **logic circuits** and **product configuration**.

Decision diagrams are now used for **filtering and propagation in CP**…



From Boole to Decision Diagrams

Propagation through domains. Let x_1, x_2, x_3 have domain $\{1,2,3\}$

$x_1 + 2x_2 + 3x_3 \leq 10$ ⟵ **filters** domains to $x_1, x_2 \in \{1,2,3\}$, $x_3 \in \{1,2\}$

$\text{all-different}(x_1, x_2, x_3)$ ⟵ no more filtering possible for **propagated** domains

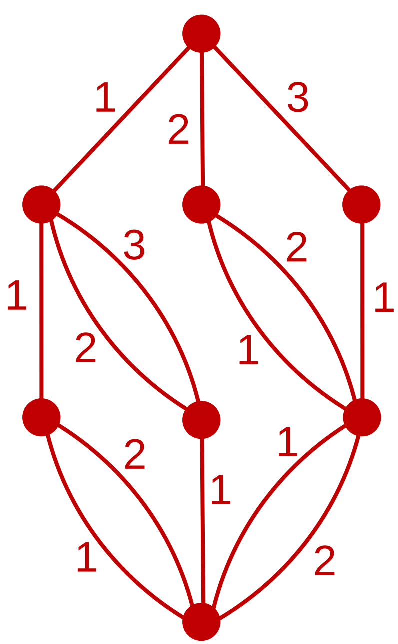
From Boole to Decision Diagrams

Propagation through domains. Let x_1, x_2, x_3 have domain $\{1, 2, 3\}$

$x_1 + 2x_2 + 3x_3 \leq 10$ ← **filters** domains to $x_1, x_2 \in \{1, 2, 3\}$, $x_3 \in \{1, 2\}$

$\text{all-different}(x_1, x_2, x_3)$ ← no more filtering possible for **propagated** domains

Propagation through a relaxed decision diagram.

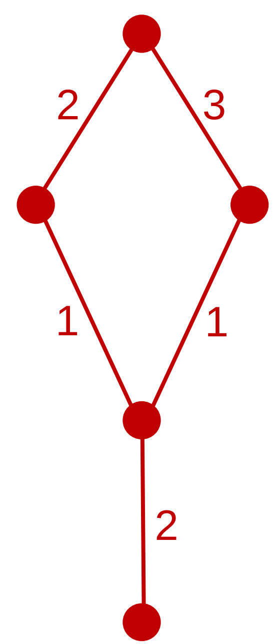


$x_1 \in \{1, 2, 3\}$

$x_2 \in \{1, 2, 3\}$

$x_3 \in \{1, 2\}$

Relaxed diagram for
$x_2 + 2x_2 + 3x_3 \leq 10$



$x_1 \in \{2, 3\}$

$x_2 \in \{1\}$

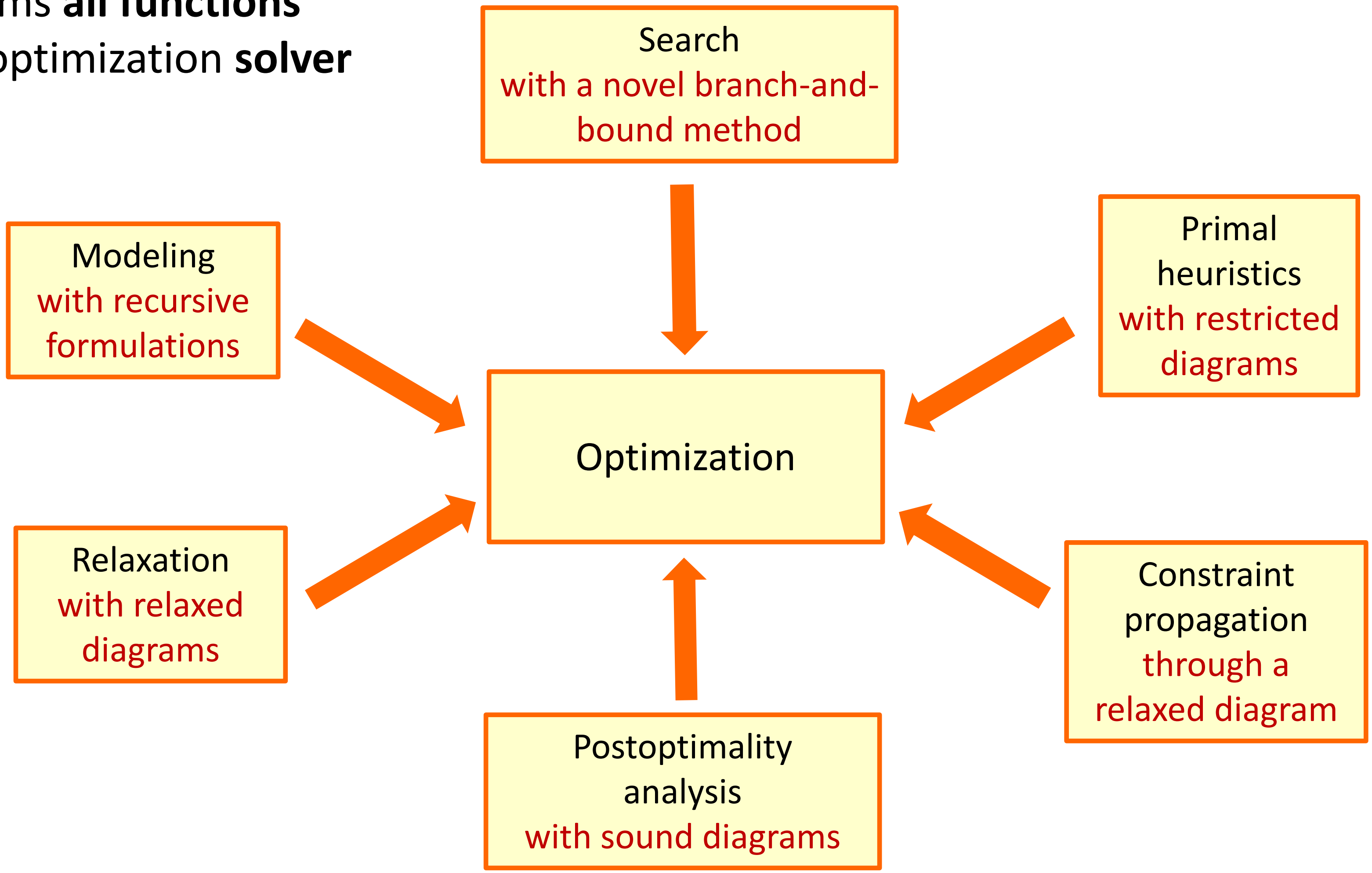
smaller domains

$x_3 \in \{2\}$

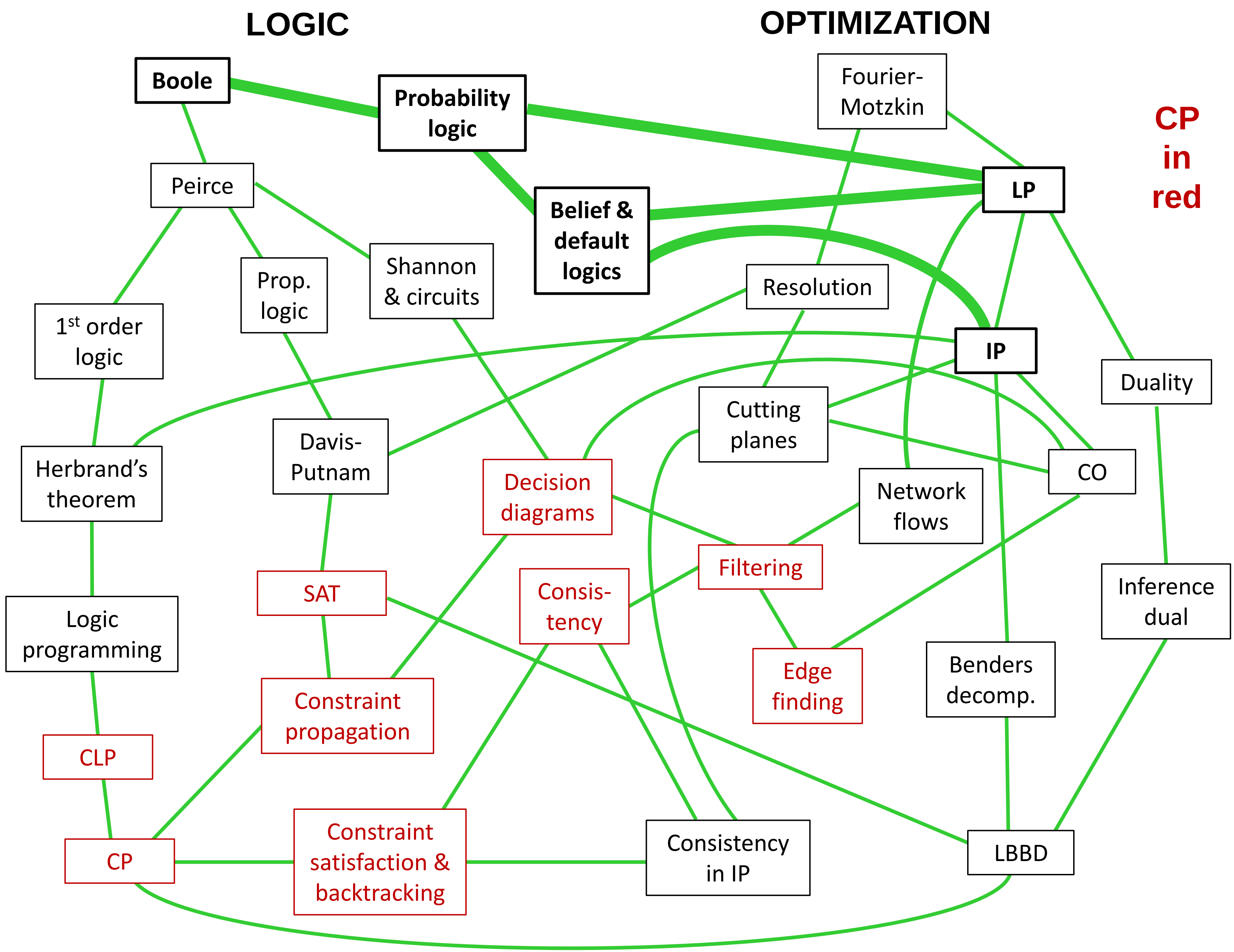
Relaxed diagram after propagating
$\text{all-different}(x_1, x_2, x_3)$

From Boole to Decision Diagrams

Decision diagrams can performs **all functions** of an optimization **solver**



From Boole to Probability and Belief Logics


LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Belief & default logics
Peirce
Prop. logic
Shannon & circuits
1st order logic
Herbrand's theorem
Logic programming
CLP
CP
Davis-Putnam
SAT
Constraint propagation
Constraint satisfaction & backtracking
Decision diagrams
Consis-tency
Filtering
Edge finding
Consistency in IP
Fourier-Motzkin
Resolution
Cutting planes
LP
IP
Network flows
CO
Duality
Inference dual
Benders decomp.
LBBD

From Boole to Probability and Belief Logics

Boole considered **probability logic** to be his most important contribution. His major work was *An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and **Probabilities*** (1854).

Theodore Hailperin (1976) showed that Boole's probability logic poses a **linear programming** problem.

Nils Nilsson (1986) proposed a very similar model for probability logic **in AI**.

This model is naturally solved by **column generation**, a widely used method in OR that generalizes Dantzig-Wolfe decomposition.

Theodore Hailperin
1915-2014

Nils Nilsson
1933-2019

From Boole to Probability and Belief Logics

Example: What are the possible probabilities of statement C, given the following?

Statement	Probability
A	0.9
A → B	0.8
B →C	0.4

Solve the linear programming problems:

$$\textit{max/min}\ \ p_{001} + p_{011} + p_{101} + p_{111}$$

subject to

$$p_{100} + p_{101} + p_{110} + p_{111} = 0.9$$
$$p_{000} + p_{001} + p_{010} + p_{011} + p_{110} + p_{111} = 0.8$$
$$p_{000} + p_{001} + p_{011} + p_{100} + p_{101} + p_{111} = 0.4$$
$$p_{000} + \ldots + p_{111} = 1,\ p_{ijk} \geq 0$$

There are exponentially many variables, but LP column generation deals with this.

There are 8 possible outcomes:

A	B	C	Prob.
false	false	false	p_{000}
false	false	true	p_{001}
false	true	false	p_{010}
false	true	true	p_{011}
true	false	false	p_{100}
true	false	true	p_{101}
true	true	false	p_{110}
true	true	true	p_{111}

The result is a **range** of probabilities for C:

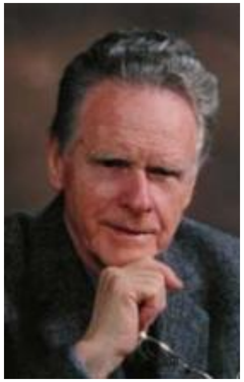
0.1 to 0.4

From Boole to Probability and Belief Logics

Dempster-Shafer theory (belief logic) has a **linear programming** model similar to the one for Boole's probability logic.

Nonmonotonic logic has a succinct **integer programming** model that arguably makes the concept clearer than a logical formulation.

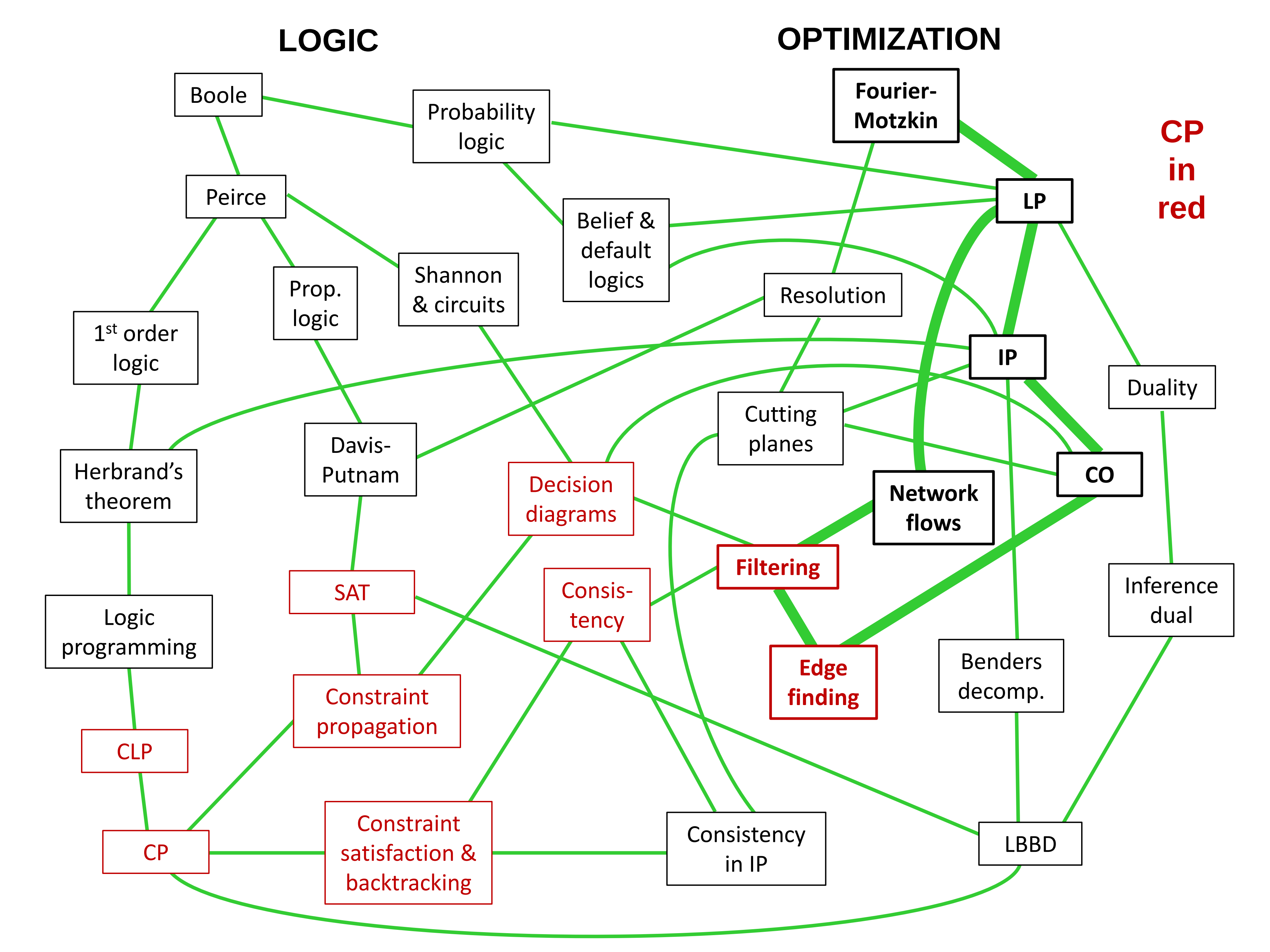
Confidence factors in **rule-based systems** have a mixed integer/linear programming model.



A. P. Dempster
1929-

Glenn Shafer
1946-

From Fourier to Filtering


LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Fourier-Motzkin
Peirce
Belief & default logics
LP
Shannon & circuits
Prop. logic
Resolution
1st order logic
IP
Duality
Cutting planes
Davis-Putnam
Herbrand's theorem
Decision diagrams
CO
Network flows
Filtering
SAT
Consis-tency
Inference dual
Logic programming
Edge finding
Benders decomp.
Constraint propagation
CLP
Constraint satisfaction & backtracking
Consistency in IP
LBBD
CP

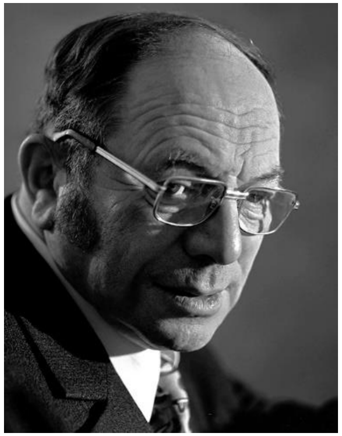
From Fourier to Filtering

Fourier (1820s) developed a theory of **linear inequalities** and a method of solving them, later rediscovered by Motzkin (1936). The method is now called Fourier-Motzkin elimination.

Kantorovich (1939) formulated a linear optimization problem subject to inequality constraints – i.e., **linear programming**.
Dantzig (1940s) independently proposed and solved the same model.

Joseph Fourier
1768-1830

Theodore Motzkin
1908-1970



Leonid Kantorovich
1912-1986

George Dantzig
1914-2005

From Fourier to Filtering

Fourier-Motzkin elimination can solve LP problems, but Dantzig's **simplex method** is far more efficient and remains the method of choice for most applications today.

LP with integer variables, or **integer programming**, followed shortly thereafter…

…along with the study of **combinatorial optimization** in general, beginning with the traveling salesman problem.

From Fourier to Filtering

Two **major success stories** for collaboration between CP and optimization:

1. **Network flow theory**, a special case of LP, has been widely applied to **filtering methods in CP**, beginning with the all-different constraint.

LP duality plays a key role in this work.

2. **Edge-finding**, an algorithm for combinatorial scheduling, led to powerful **domain reduction methods** for scheduling problems in CP.

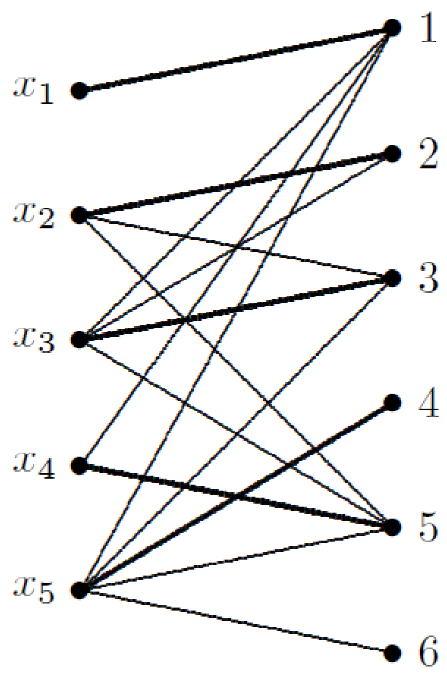
Edge finding was originally published in the **OR** journal *Management Science* (Carlier and Pinson 1989), with most subsequent papers in the **CP** literature.

From Fourier to Filtering

Example of **network flows and filtering**.

An **all-different** constraint has a solution if and only if there is a perfect matching:

$$\text{alldiff}\left(x_1, x_2, x_3, x_4, x_5\right)$$
$$x_1 \in \{1\}$$
$$x_2 \in \{2,3,5\}$$
$$x_1 \in \{1,2,3,5\}$$
$$x_1 \in \{1,5\}$$
$$x_1 \in \{1,3,4,5,6\}$$

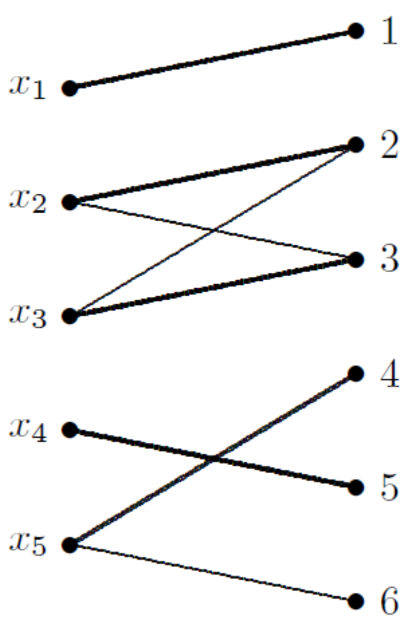


Solution shown:
$(x_1,x_2,x_3,x_4,x_5) = (1,2,3,5,4)$

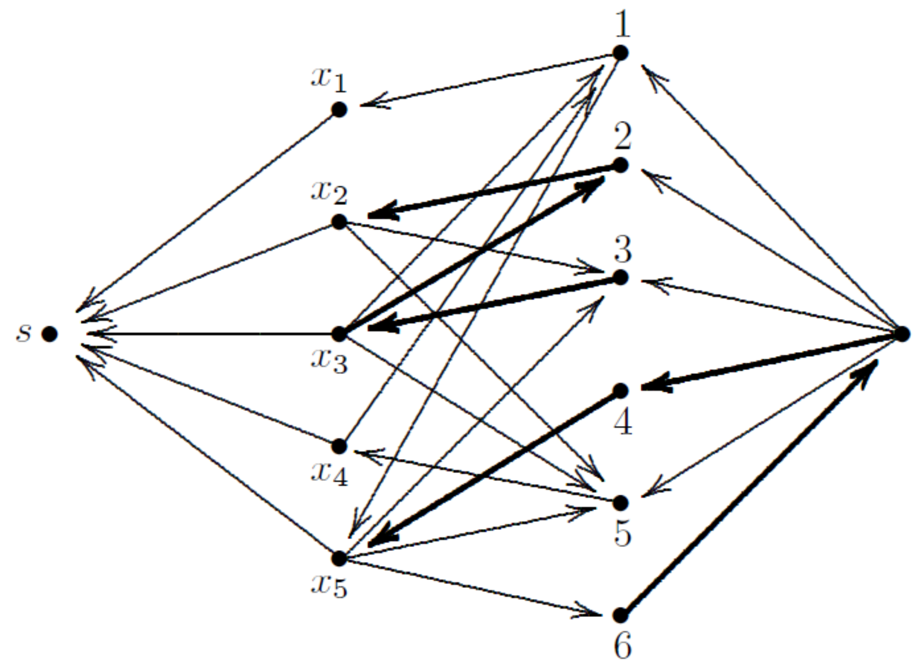
From Fourier to Filtering

The matching problem can be viewed as a **maximum flow** problem on a network, which is a **linear programming** problem.

The **dual solution** of the problem indicates how to **filter domains**:



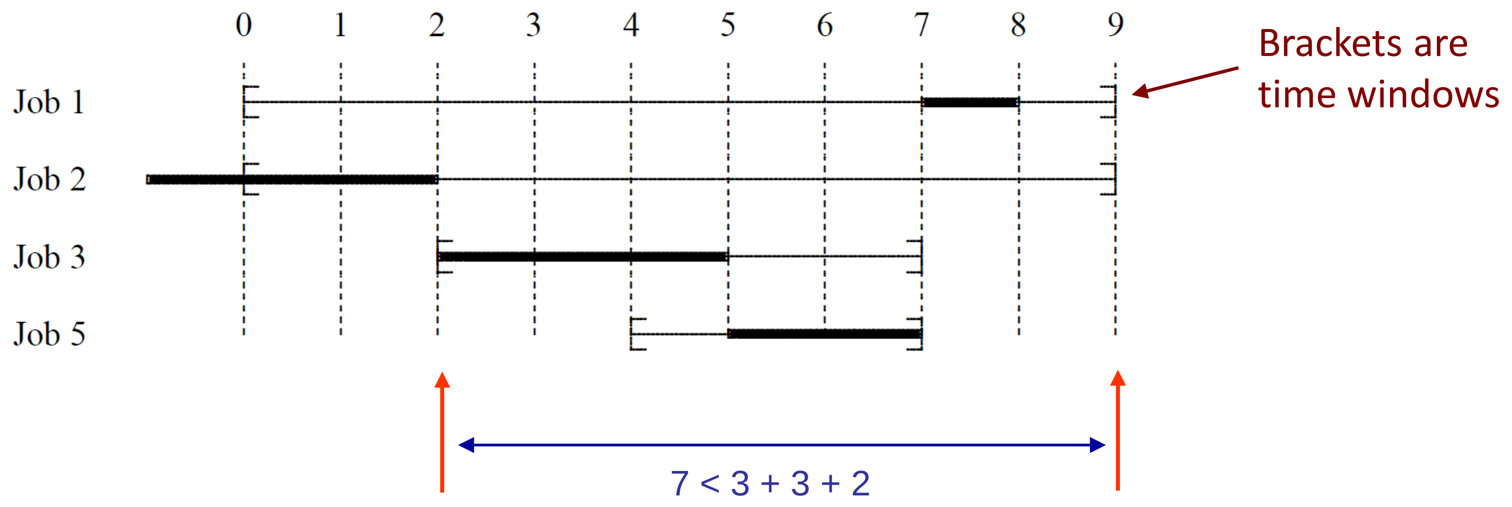
$$x_1 \in \{1\}$$
$$x_2 \in \{2,3\}$$
$$x_3 \in \{2,3\}$$
$$x_4 \in \{5\}$$
$$x_5 \in \{4,6\}$$



From Fourier to Filtering

Example of edge-finding and filtering.

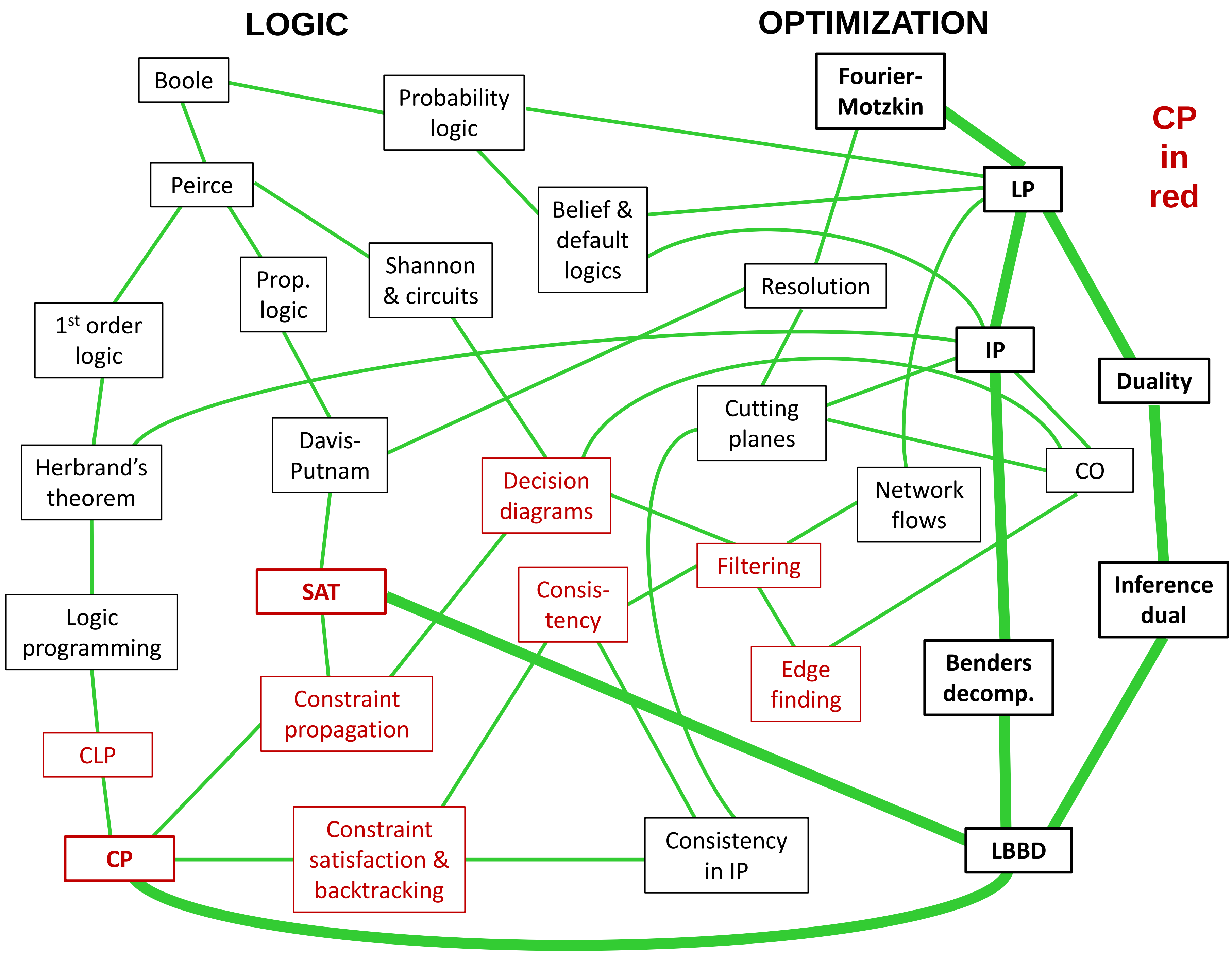
Edge-finding argument that job 2 must precede jobs 3 and 5:



If job 2 is **not before 3 and 5**, then there is not enough time in their time windows (7 hours) to run all 3 jobs (requiring 8 hours).
So, time window for job 2 must be **reduced** to [0,2], which in infeasible.

Edge-finding check can run in **polynomial time**, and can be **generalized** to other scheduling problems .

From Fourier to Inference Duality



LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Peirce
Belief & default logics
Shannon & circuits
Prop. logic
1st order logic
Herbrand's theorem
Logic programming
CLP
CP
Davis-Putnam
SAT
Constraint propagation
Constraint satisfaction & backtracking
Decision diagrams
Consis-tency
Filtering
Edge finding
Consistency in IP
Fourier-Motzkin
LP
Resolution
IP
Cutting planes
Duality
CO
Network flows
Inference dual
Benders decomp.
LBBD

From Fourier to Inference Duality

After a chance meeting on a rail platform near Princeton University, Dantzig and von Neumann combined ideas from LP and game theory to arrive at **LP duality**.

Duality has become a powerful idea in optimization, e.g. Lagrangian duality, **Dantzig-Wolfe decomposition** (column generation), and **Benders decomposition** (row generation).

Joseph-Louis Lagrange
1736-1813

John von Neumann
1903-1957

George Dantzig
1914-2005

From Fourier to Inference Duality

All optimization duals are special cases of **inference duality**

Primal problem:
Optimization

$$\min \; f(\boldsymbol{x})$$
$$\boldsymbol{x} \in S$$

Find **best** feasible solution by searching over **values of *x***.

Dual problem:
Inference

$$\max \; v$$
$$\boldsymbol{x} \in S \overset{P}{\Rightarrow} f(\boldsymbol{x}) \geq v$$
$$P \in \mathcal{P}$$

Find a **proof** of optimal value by searching over proofs *P*.

The **type** of dual depends on the **inference method** used.
In **classical LP**, the proof is a tuple of **dual multipliers**.
A **complete** inference method yields a **strong dual** (no duality gap)

From Fourier to Inference Duality

Type of Dual	Inference Method	Strong?
Linear programming	Nonnegative linear combination + material implication	Yes*
Lagrangian	Nonnegative linear combination + domination	No
Surrogate	Nonnegative linear combination + material implication	No
Subadditive	Cutting planes	Yes**

*Due to Farkas Lemma

**Due to Chvátal's theorem

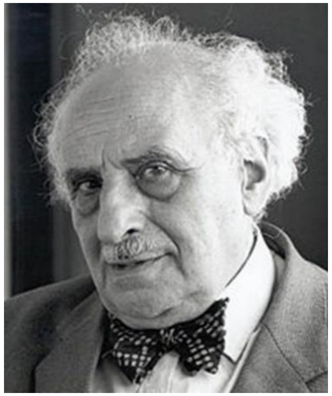
From Fourier to Inference Duality

Benders decomposition was designed for problems that become LPs after some variables are fixed.

The **dual** of the LP subproblem provides a **Benders cut** that excludes undesirable solutions.

Generalization to logic-based Benders cuts:

Using the **inference dual**, the subproblem can in principle be **any** optimization or constraint satisfaction problem.



Jacques Benders
1924-2017

So, a **logical perspective** leads to a substantial generalization with many new applications.

From Fourier to Inference Duality

Classical Benders decomposition

Solve the problem
$$\begin{aligned} &\min\ f(\boldsymbol{x}) + \boldsymbol{c}\boldsymbol{y} \\ &g(\boldsymbol{x}) + A\boldsymbol{y} \geq \boldsymbol{b} \end{aligned}$$

- Subproblem must be an LP.
- Benders cuts are based on classical duality.

Master problem

$$\begin{aligned} &\min\ z \\ &z \geq f(\boldsymbol{x}) + \boldsymbol{u}_k\big(\boldsymbol{b} - \boldsymbol{g}_k(\boldsymbol{x})\big), \\ &\qquad\qquad \text{all } k \ \text{(Benders cuts)} \end{aligned}$$

Minimize cost *z* subject to bounds given by Benders cuts, obtained from values of ***x*** attempted in previous iterations *k*.

Trial value $\bar{x}$ that solves master

Benders cut

Subproblem

$$\begin{aligned} &\min\ f(\bar{\boldsymbol{x}}) + \boldsymbol{c}\boldsymbol{y} \\ &A\boldsymbol{y} \geq \boldsymbol{b} - \boldsymbol{g}(\bar{\boldsymbol{x}}) \end{aligned}$$

Obtain proof of optimality (solution ***u*** of LP dual).
Use dual solution to obtain a Benders cut.

Repeat until the master problem and subproblem have the same optimal value.

From Fourier to Inference Duality

Logic-based Benders decomposition (LBBD)

Solve the problem
$$\min\ f(\boldsymbol{x},\boldsymbol{y})$$
$$(\boldsymbol{x},\boldsymbol{y}) \in S$$
$$\boldsymbol{x} \in D$$

- Subproblem can be **any** optimization problem.
- View the subproblem dual as a **logical inference** problem.

Master problem

$$\min\ z$$
$$z \geq v_k(\boldsymbol{x}),\ \text{all } k \ (\text{Benders cuts})$$

Minimize cost *z* subject to bounds given by Benders cuts, obtained from values of ***x*** attempted in previous iterations *k*.

Trial value $\bar{x}$ that solves master

Benders cut

Subproblem

$$\min\ f(\bar{\boldsymbol{x}},\boldsymbol{y})$$
$$(\bar{\boldsymbol{x}},\boldsymbol{y}) \in S$$

Obtain proof of optimality (solution of inference dual). Use **same proof** to deduce cost bounds for other values of ***x***, yielding a Benders cut

Repeat until the master problem and subproblem have the same optimal value.

From Fourier to Inference Duality

LBBD has been applied to a wide range of problems that simplify (perhaps by decoupling) when some variables are fixed.

It is a useful tool for **combining optimization and CP**.

Typically, an optimization method (such as MILP) solves the master problem and **CP solves the subproblem** (often a scheduling problem).

The **conflict clauses** that are central to **SAT solvers** are a special case of logic-based Benders cuts.

SAT-modulo-theories are also solved as a special case of LBBD.

From Fourier to Inference Duality

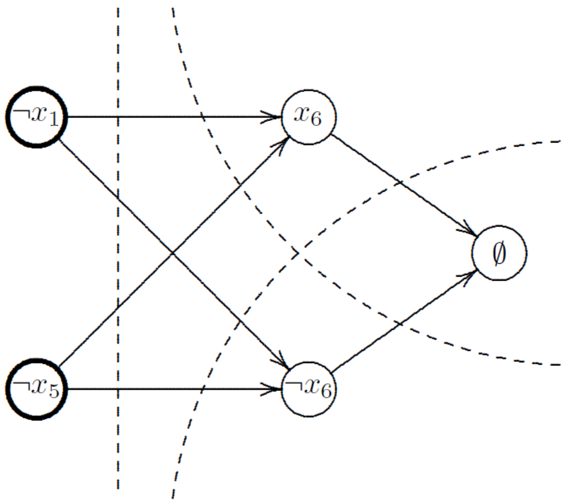
Conflict clauses as logic-based Benders cuts

- The **subproblem** is the problem at a node of the DPLL search tree.
- The **inference dual** is defined by **unit resolution**.
- The **dual solution** is a unit refutation, encoded in a **conflict graph**.

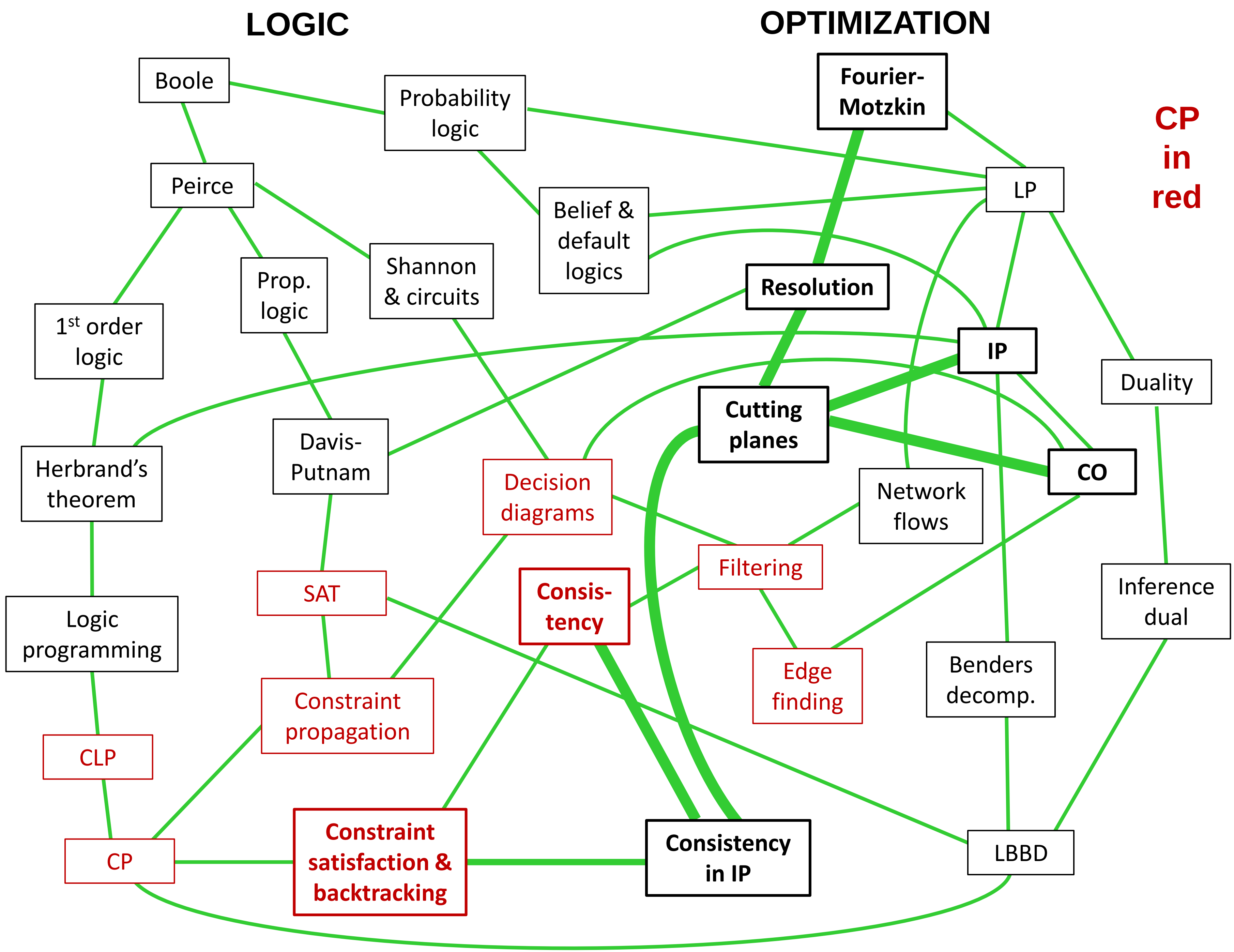
The conflict graph is a solution of the **inference dual**

The resulting **conflict clause** is a **Benders cut**

$x_1 \vee x_5$



From Fourier to Cutting Planes

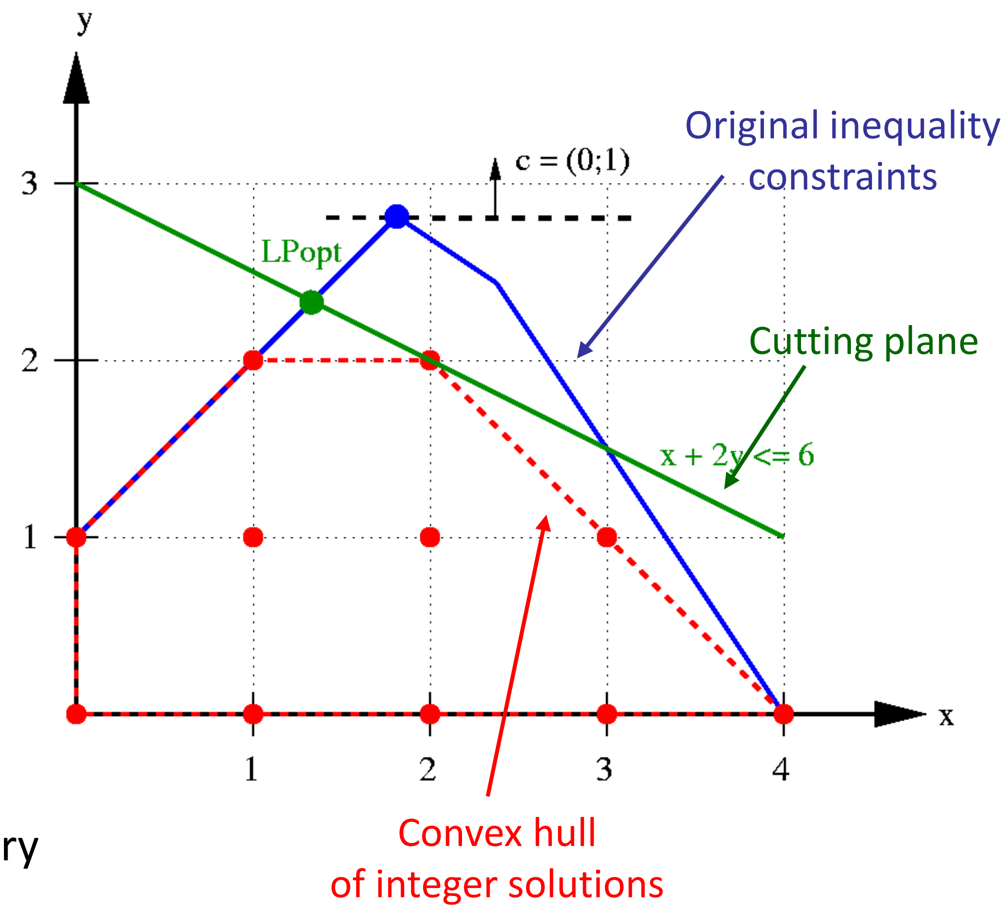

LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Peirce
Belief & default logics
Fourier-Motzkin
LP
Prop. logic
Shannon & circuits
Resolution
1st order logic
IP
Duality
Cutting planes
Davis-Putnam
Herbrand's theorem
Decision diagrams
CO
Network flows
Filtering
SAT
Consis-tency
Inference dual
Logic programming
Edge finding
Benders decomp.
Constraint propagation
CLP
Constraint satisfaction & backtracking
Consistency in IP
CP
LBBD

From Fourier to Cutting Planes

Cutting planes, studied for over 60 years, are an essential component of **integer programming** solvers.

They are closely related to **resolution** and **Fourier-Motzkin elimination**.

They approximate the convex hull of integer solutions, so that the LP relaxation gives a tighter bound.



Ralph Gomory
1929-

From Fourier to Cutting Planes

Quine's **resolution method** is very similar to Fourier-Motzkin elimination.

Resolution:

$$\begin{array}{l} x_1 \vee x_2 \vee x_4 \\ x_1 \vee \neg x_4 \\ \hline x_1 \vee x_2 \end{array}$$

A **projection** method for logical clauses

When the logical clauses are written as inequalities (as suggested by Dantzig), resolution is Fourier-Motzkin elimination combined with **rounding** of fractions.

$$\begin{array}{rcrcrl} x_1 & + & x_2 & + & x_4 & \geq 1 \qquad (1/2) \\ x_1 & & & - & x_4 & \geq 0 \qquad (1/2) \\ & & x_2 & & & \geq 0 \qquad (1/2) \\ \hline x_1 & + & x_2 & & & \geq \lceil \frac{1}{2} \rceil \end{array}$$

A **projection** method for linear inequalities

From Fourier to Cutting Planes

This means that a resolvent is a **rank 1 Chvátal-Gomory cut.**

$$\begin{array}{rcrcrl} x_1 & + & x_2 & + & x_4 & \geq 1 \qquad (1/2) \\ x_1 & & & - & x_4 & \geq 0 \qquad (1/2) \\ & & x_2 & & & \geq 0 \qquad (1/2) \\ \hline x_1 & + & x_2 & & & \geq \lceil \frac{1}{2} \rceil \end{array}$$

The fundamental theorem of cutting planes (due to Chvátal) states that any valid cutting plane can be obtained from repeated generation of rank 1 Chvátal-Gomory cuts.

The **proof** of this theorem is based on the **resolution algorithm**!

Cutting planes lie at the heart of integer programming, and logic lies at the heart of cutting planes

Vašek Chvátal
1946-

From Fourier to Cutting Planes

Consistency is a fundamental concept in CP.

It is not satisfiability or feasibility.

We can view a consistent constraint set as one in which **any infeasible partial assignment is inconsistent with some constraint**.

This avoids backtracking, because **each node** corresponds to a **partial assignment** defined by branches so far. We can detect whether deeper branching can find a feasible solution.

CP solvers try to achieve various kinds of **partial** consistency
(e.g, domain consistency) to reduce backtracking.

From Fourier to Cutting Planes

Cutting planes are normally viewed as tightening an LP relaxation to obtain better bounds.

Separating cuts exclude fractional solutions.

But cutting planes also **achieve** (partial) **consistency!**

They exclude **inconsistent partial assignments**.

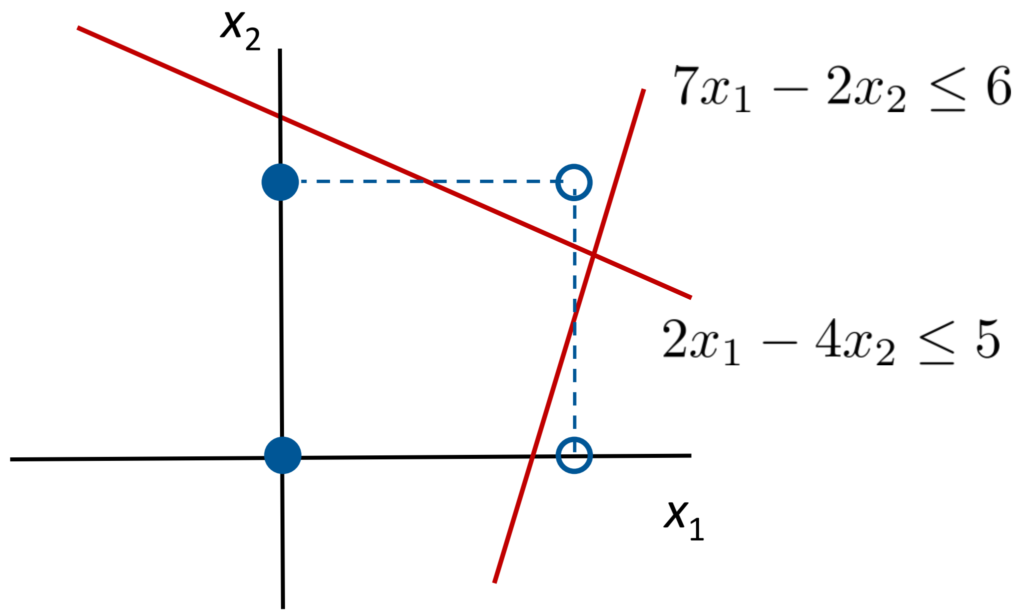
This helps to explain why they can reduce backtracking.

From Fourier to Cutting Planes

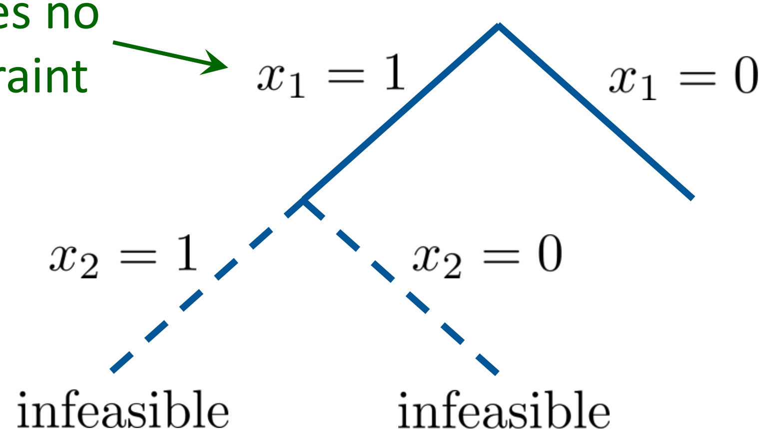
The constraint set

$$2x_1 + 4x_2 \leq 5$$
$$7x_1 - 2x_2 \leq 6$$
$$x_1, x_2 \in \{0, 1\}$$

is **not consistent**



Violates no constraint



Backtracking can result even with forward checking.

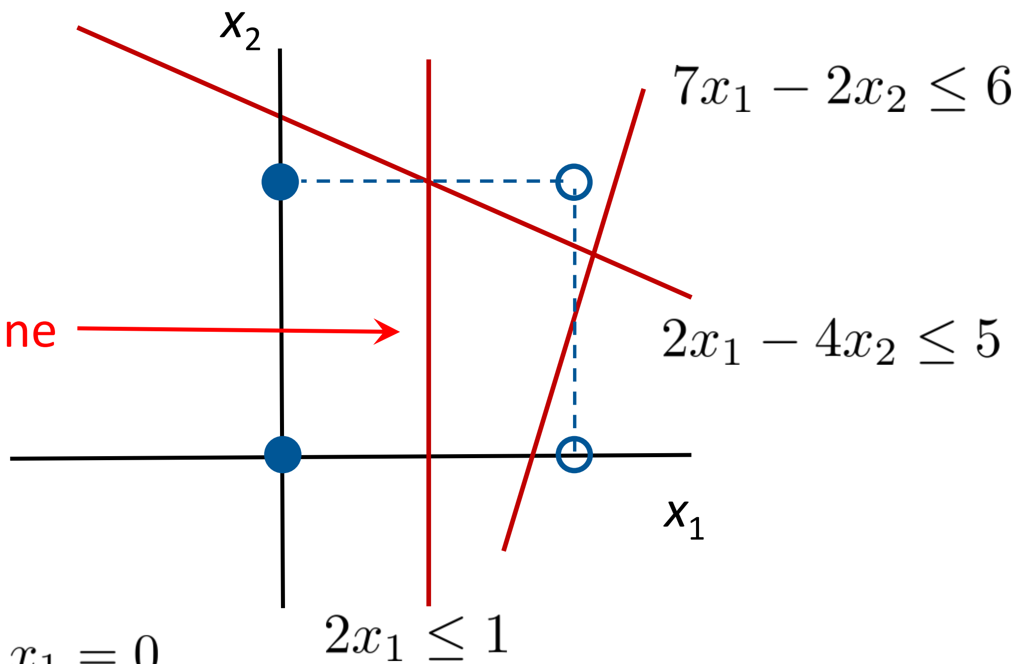
From Fourier to Cutting Planes

The constraint set

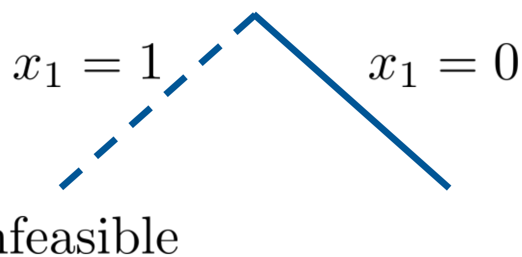
$$2x_1 + 4x_2 \le 5$$
$$7x_1 - 2x_2 \le 6$$
$$\boxed{2x_1 \le 1}$$
$$x_1, x_2 \in \{0, 1\}$$

is **consistent**

Cutting plane



Violates a constraint



No backtracking
with forward checking

Don't take the $x_1 = 1$ branch

From Fourier to Cutting Planes

LP-consistency is a type of consistency that is relevant to IP:

An LP-consistent constraint set is one in which **any infeasible partial assignment is infeasible in the LP relaxation**.

This allows us to recognize inconsistent partial assignments by solving an LP.

Cutting planes can achieve **partial** LP consistency and thereby reduce backtracking.

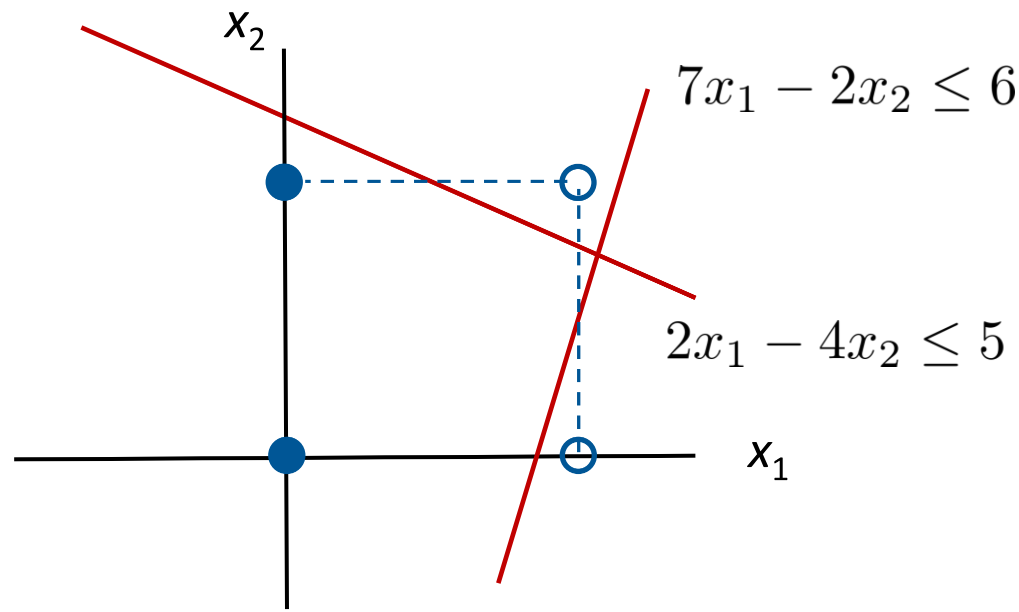
Bounding and fractional solutions need not play a role.

From Fourier to Cutting Planes

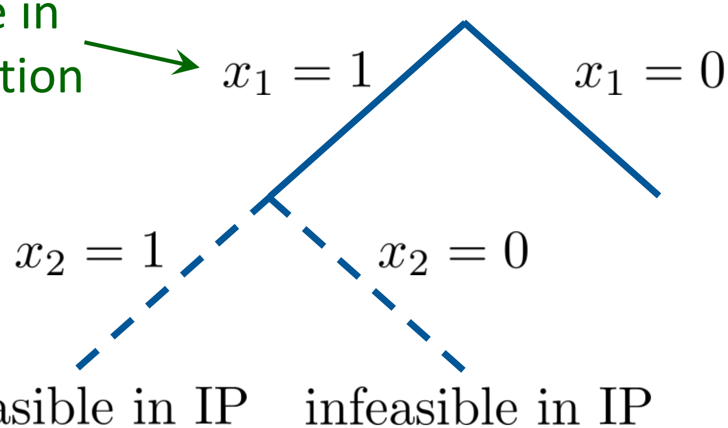
The constraint set

$$2x_1 + 4x_2 \leq 5$$
$$7x_1 - 2x_2 \leq 6$$
$$x_1, x_2 \in \{0, 1\}$$

is **not LP-consistent**



Feasible in LP relaxation



Backtracking can result even with forward checking.

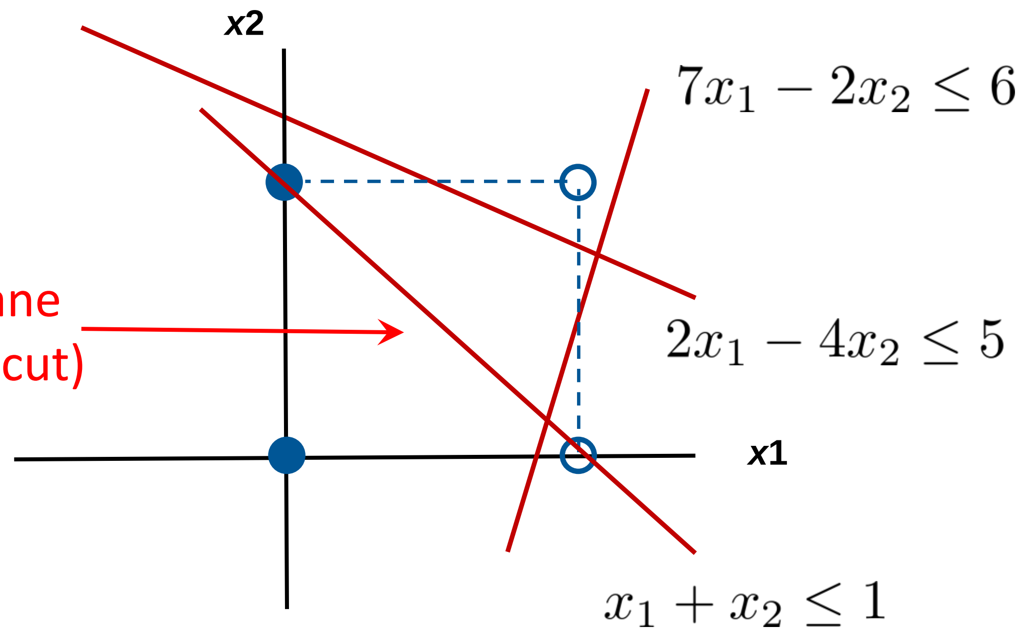
From Fourier to Cutting Planes

The constraint set

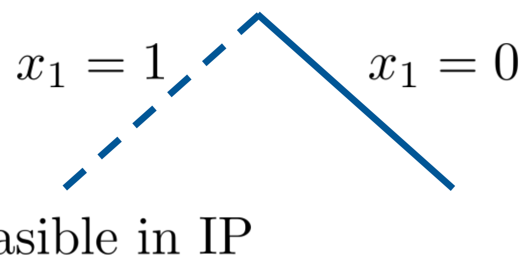
$$2x_1 + 4x_2 \leq 5$$
$$7x_1 - 2x_2 \leq 6$$
$$\boxed{x_1 + x_2 \leq 1}$$
$$x_1, x_2 \in \{0, 1\}$$

is LP-consistent

Cutting plane
(rank 1 C-G cut)



Infeasible in
LP relaxation



No backtracking
with forward checking

Don't take the x_1 = 1 branch

From Fourier to Cutting Planes

Theorem: An IP constraint set is LP-consistent **if and only if** all implied **logical clauses** (written as inequalities) are **rank 1 Chvátal-Gomory cuts**.

This again links **logic** and **cutting planes**.

From Fourier to Cutting Planes

Theorem: An IP constraint set is LP-consistent **if and only if** all implied **logical clauses** (written as inequalities) are **rank 1 Chvátal-Gomory cuts**.

This again links **logic** and **cutting planes**.

Let **consistency cuts** be cutting planes that cut off inconsistent partial assignments**.**

We can achieve partial LP consistency with a restricted form of **RLT** (reformulation and linearization technique).

RLT-based consistency cuts can **reduce the search tree** substantially more than traditional separating RLT cuts, also with time savings.

Summing Up

Advances in the logic-optimization-CP interface

- Constraint logic programming, leading to CP
- Fundamental theorem of cutting planes in IP
- Conflict-directed clause generation in SAT
- Logic-based Benders decomposition
- Combinatorial optimization with decision diagrams
- IP models for first-order logic, nonmonotonic logic
- LP model with column generation for probability logic
- LP models for belief logics
- Flow-based filtering methods in CP
- CP-based solution of scheduling problems
- Reinterpretation of cutting planes as consistency maintenance
- More to come?

Thanks for your attention!