Logic, Optimization, and CP

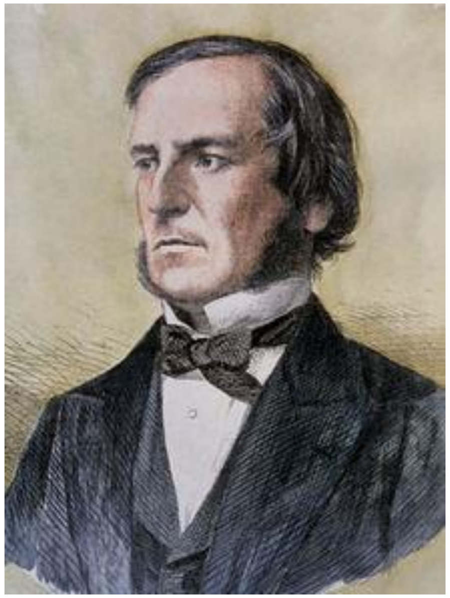
John Hooker
Carnegie Mellon University

CP 2021

Logic, Optimization, and CP

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

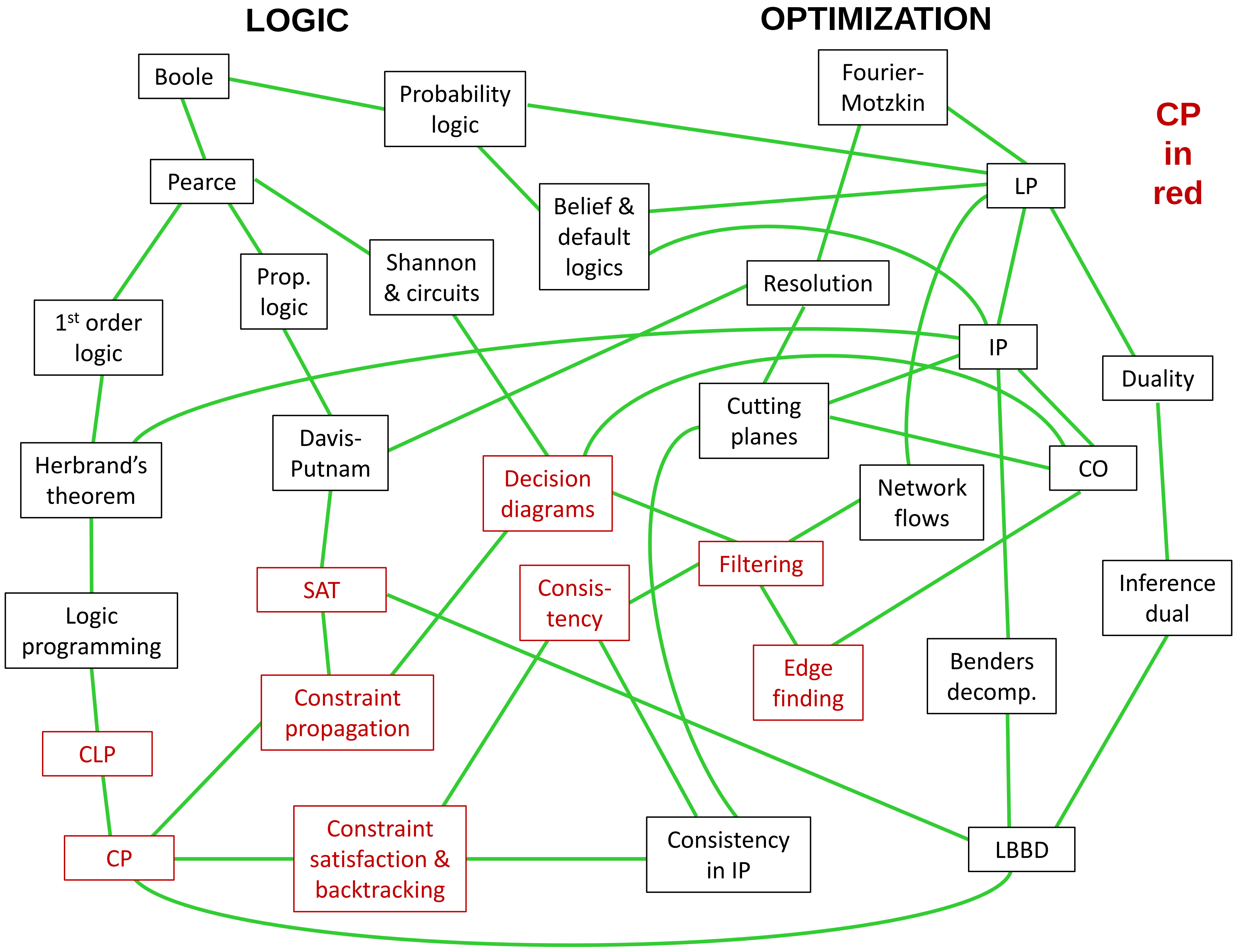
This is a brief retrospective that attempts to bring out some of these connections.



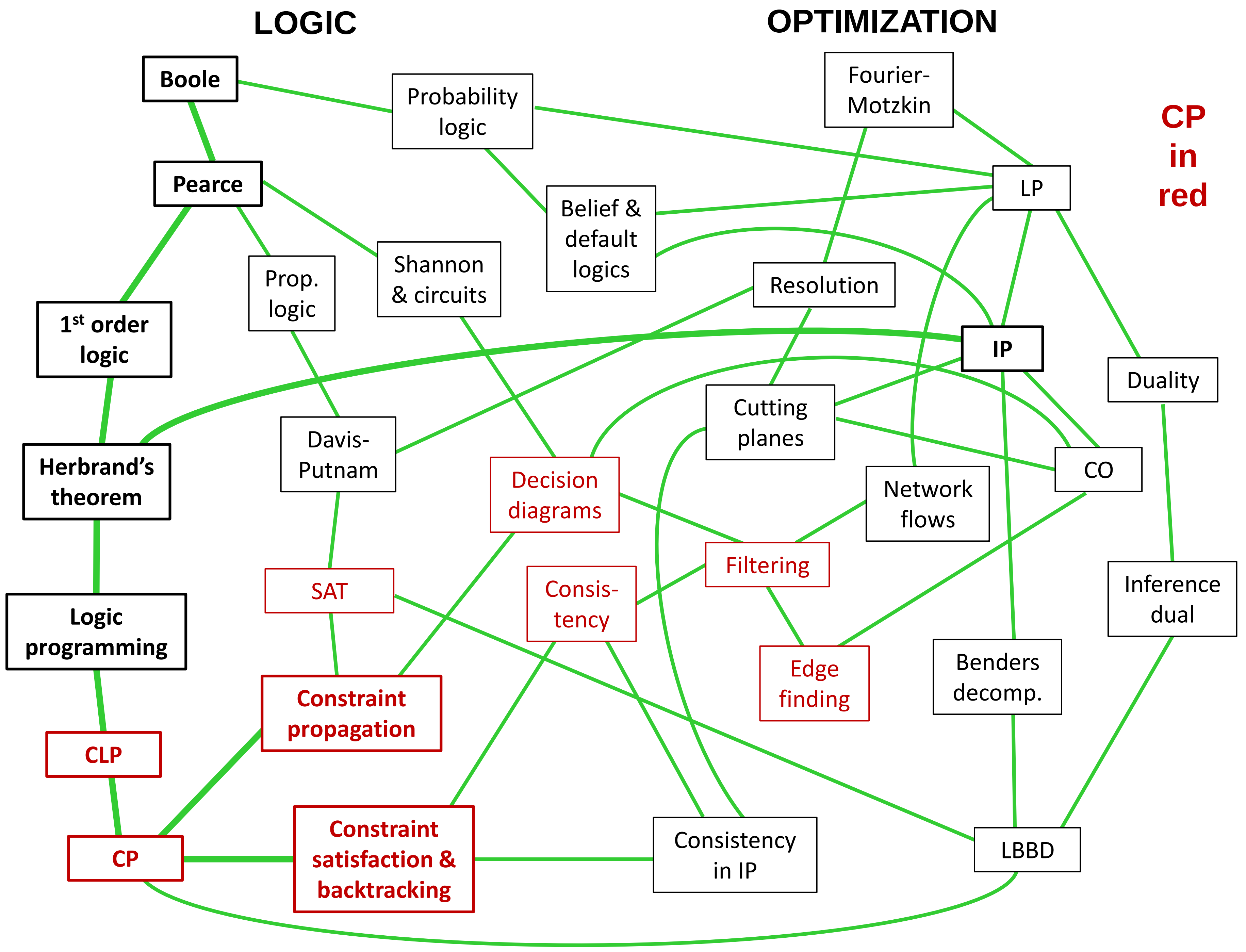
Logic, Optimization, and CP

I apologize in advance for my failure to mention most of the important contributors to these fields.

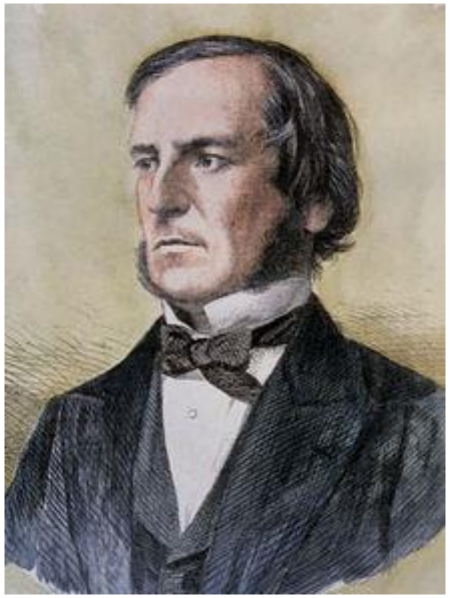
My rule has been to cite by name only those who are no longer living.


LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Pearce
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
Consistency in IP
Fourier-Motzkin
LP
Resolution
IP
Cutting planes
Duality
CO
Network flows
Filtering
Edge finding
Inference dual
Benders decomp.
LBBD

From Boole to Logic Programming & CP


LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Fourier-Motzkin
Pearce
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
Logic programming
SAT
Consis-tency
Inference dual
Edge finding
Benders decomp.
Constraint propagation
CLP
Constraint satisfaction & backtracking
CP
Consistency in IP
LBBD

From Boole to Logic Programming & CP


George Boole
1815-1864

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 likewise devised a language in which logical deductions can be calculated.

From Boole to Logic Programming & CP

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

But it was studied by philosopher **Charles Sanders Pearce** in the 1880s-90s.

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. Pearce
1842-1902

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

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

From Boole to Logic Programming & CP

A key step in first order logic is **unification**, which in essence solves equations.

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.

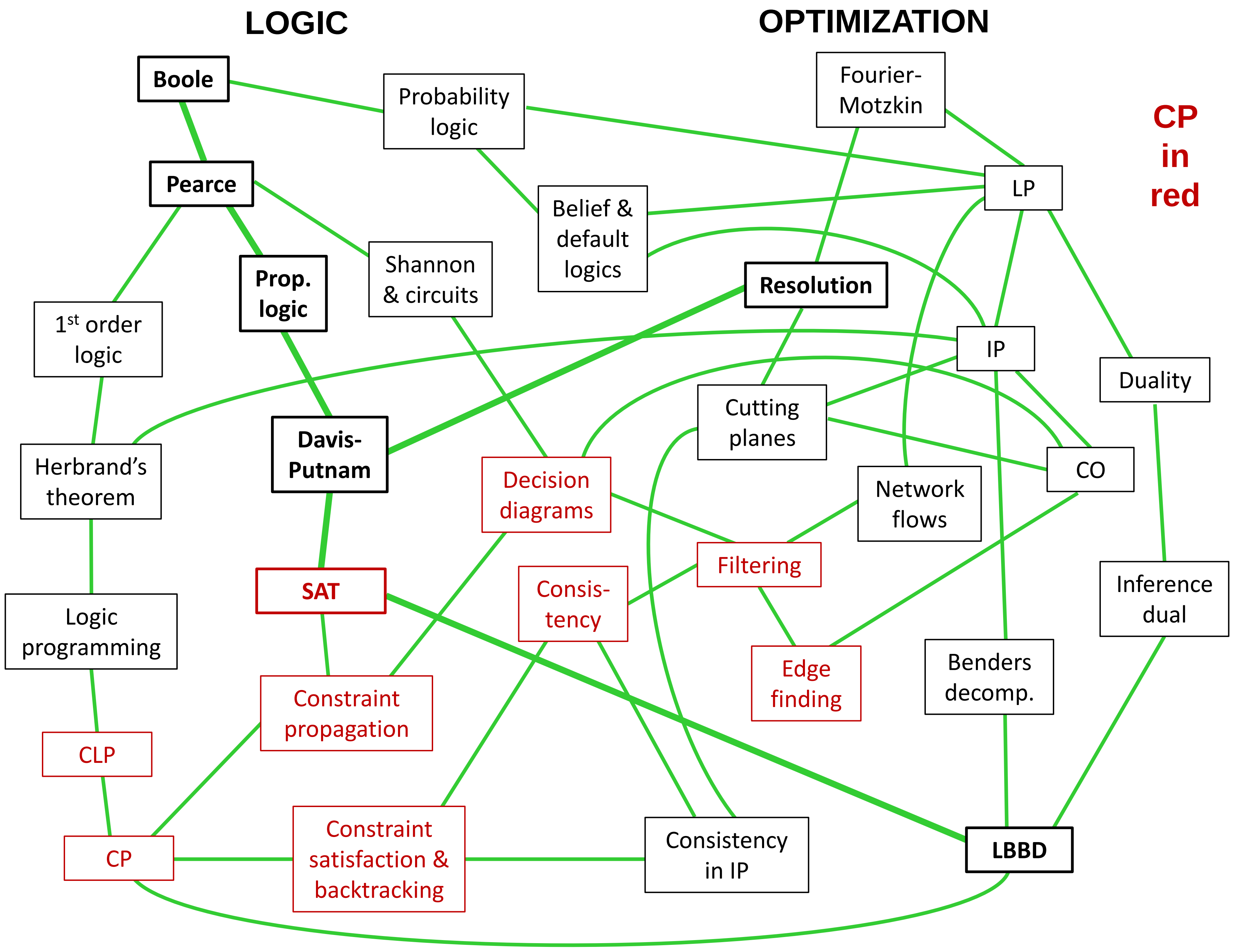
Constraint programming "toolkits" retained constraint solving in a procedural/declarative framework, without requiring a strict logic programming formalism.

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

Constraint propagation was developed (particularly for or "binary" constraints) to allow efficient inference from **constraint sets**.

The constraint satisfaction literature studied **consistency** concepts and their connection with **backtracking**.

From Boole to SAT


LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Fourier-Motzkin
Pearce
Belief & default logics
LP
Shannon & circuits
Resolution
1st order logic
Prop. logic
IP
Duality
Cutting planes
Herbrand's theorem
Davis-Putnam
Decision diagrams
CO
Network flows
SAT
Filtering
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).

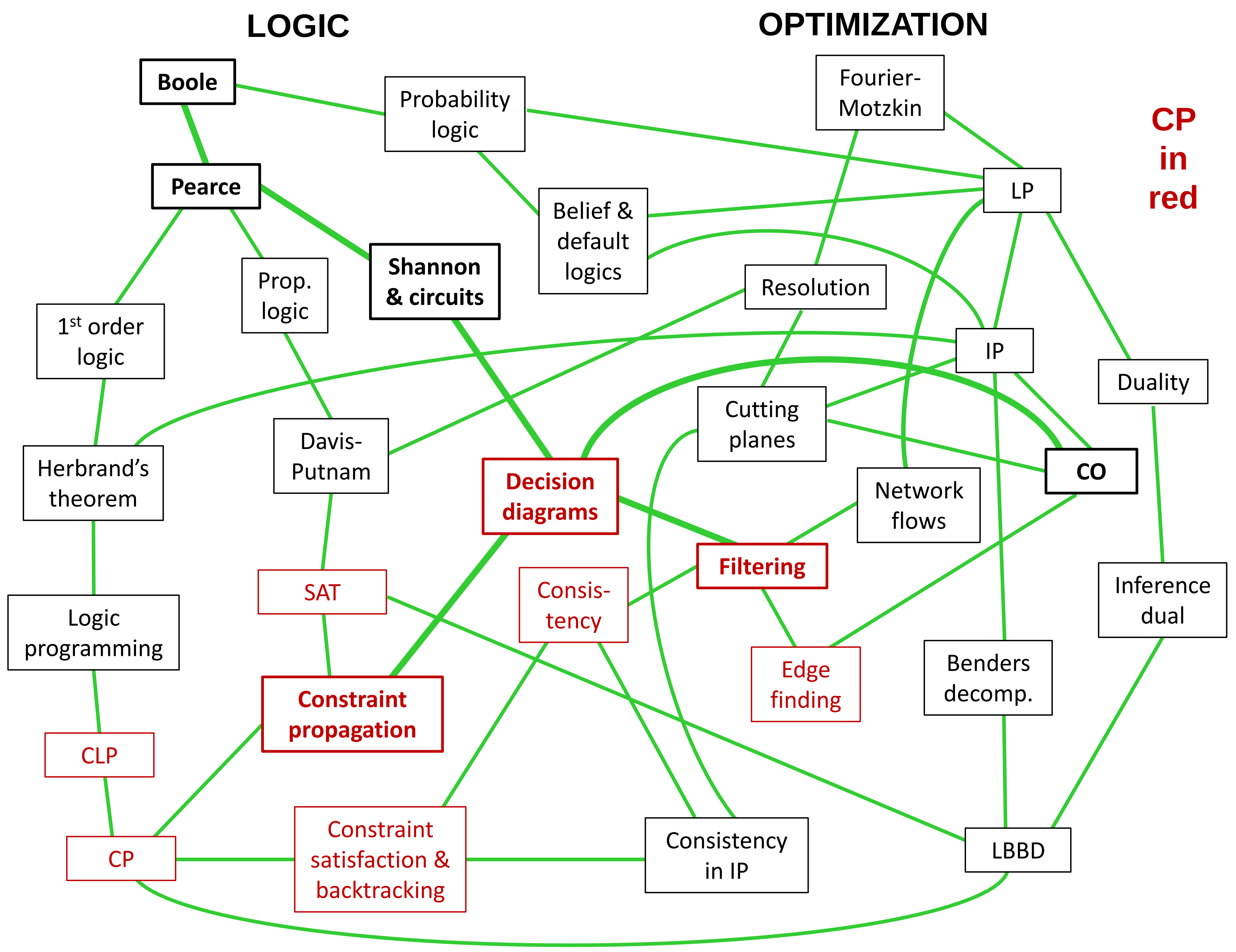
The philosopher W. V. Quine proposed (1950s) a consensus method for simplifying propositional formulas that is a complete inference method for propositional logic. When applied to CNF rather than DNF, the method is **resolution**.

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.

This led to today’s **SAT** methods, which we will see are related to **Benders decomposition** in optimization.

From Boole to Decision Diagrams


LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Fourier-Motzkin
Pearce
Shannon & circuits
Belief & default logics
LP
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
CP
Constraint satisfaction & backtracking
Consistency in IP
LBBD

From Boole to Decision Diagrams

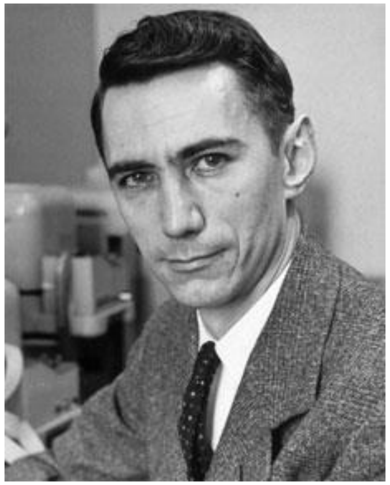
Pearce also applied Boolean methods to **switching circuits**.

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

Claude Shannon was required to take an undergraduate philosophy course at the University of Michigan, which exposed him to Pearce's work.

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

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



Claude Shannon
1916-2001

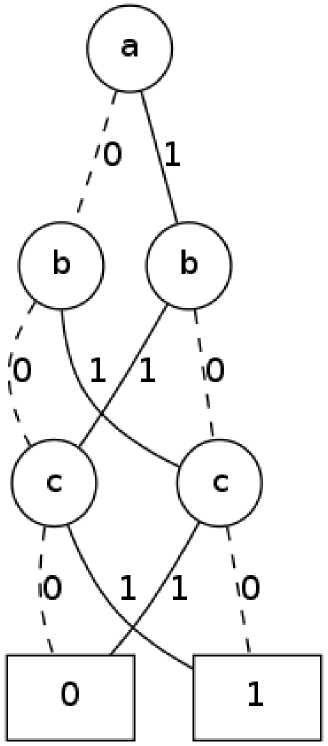
From Boole to Decision Diagrams

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

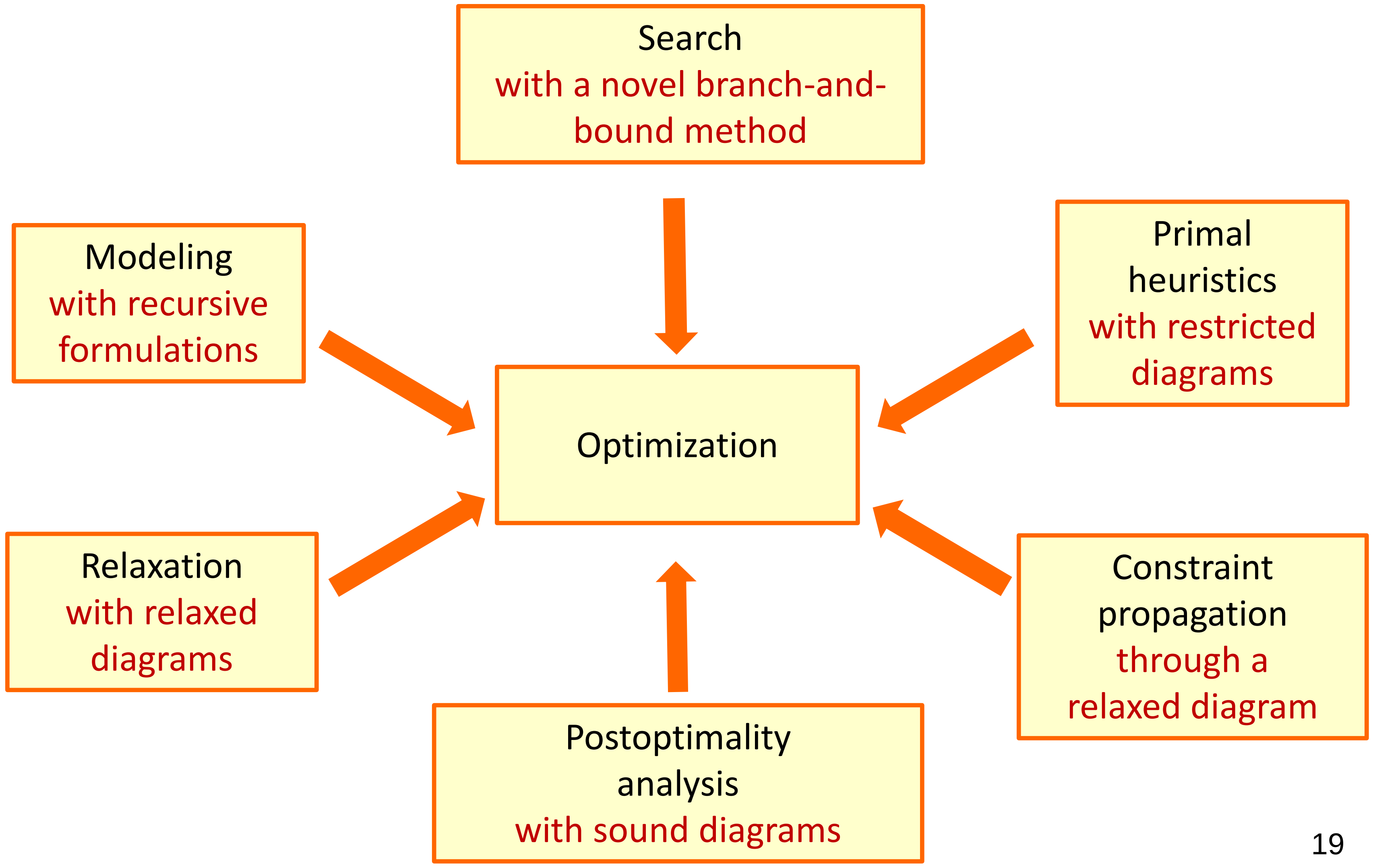
These were later represented as **binary decision diagrams** (1978). **Ordered BDDs** provide a unique minimal representation of a Boolean function (1986).

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

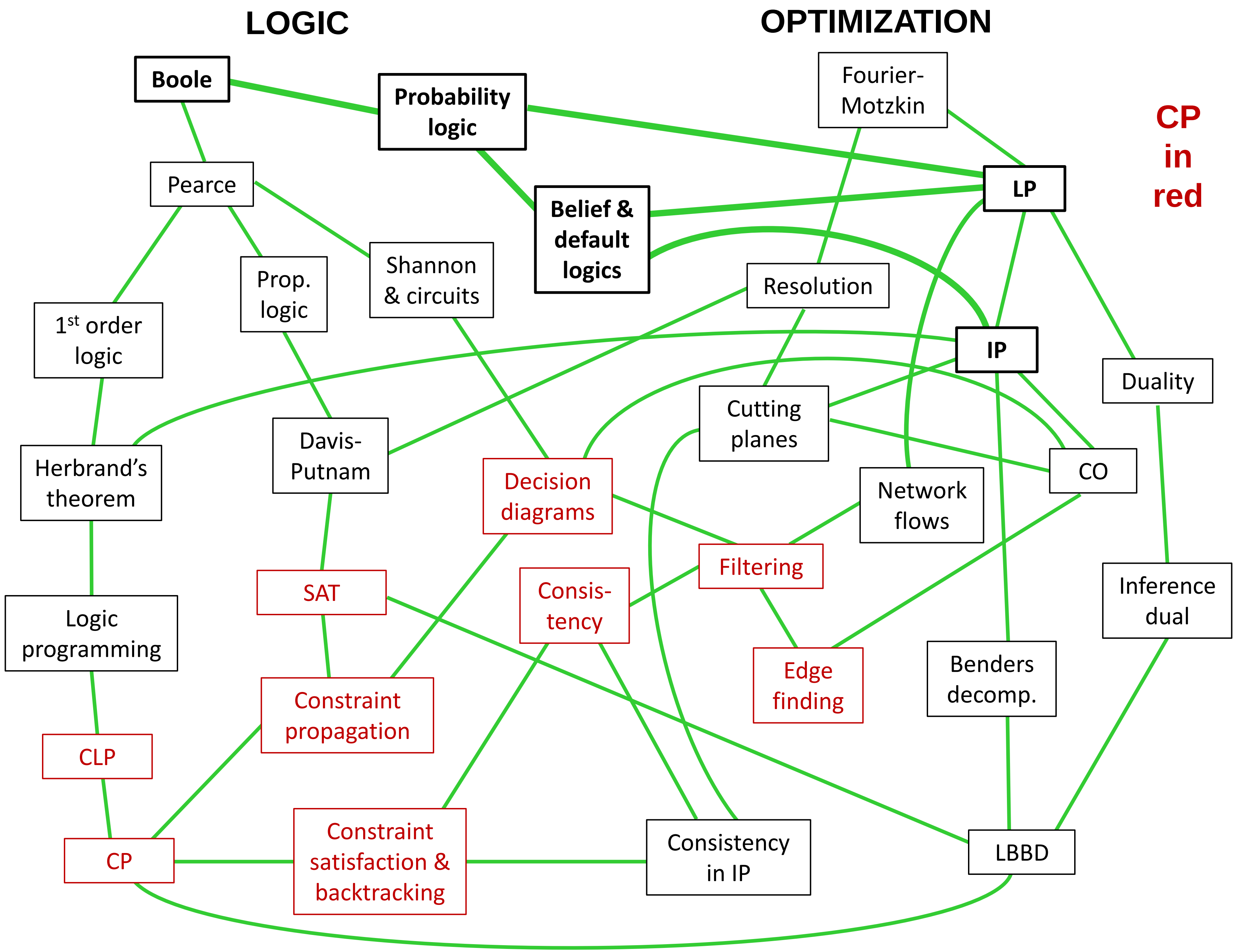
They also provide a **comprehensive framework** for optimization and constraint solving…



From Boole to Decision Diagrams



From Boole to Probability Logic


LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Belief & default logics
Pearce
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
Network flows
LP
IP
CO
Duality
Inference dual
Benders decomp.
LBBD

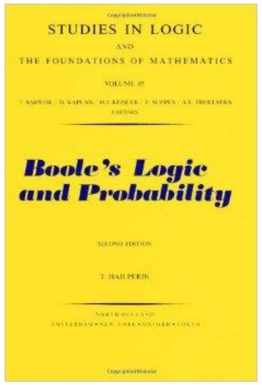
From Boole to Probability Logic

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.


STUDIES IN LOGIC AND THE FOUNDATIONS OF MATHEMATICS

Boole's Logic and Probability

SECOND EDITION

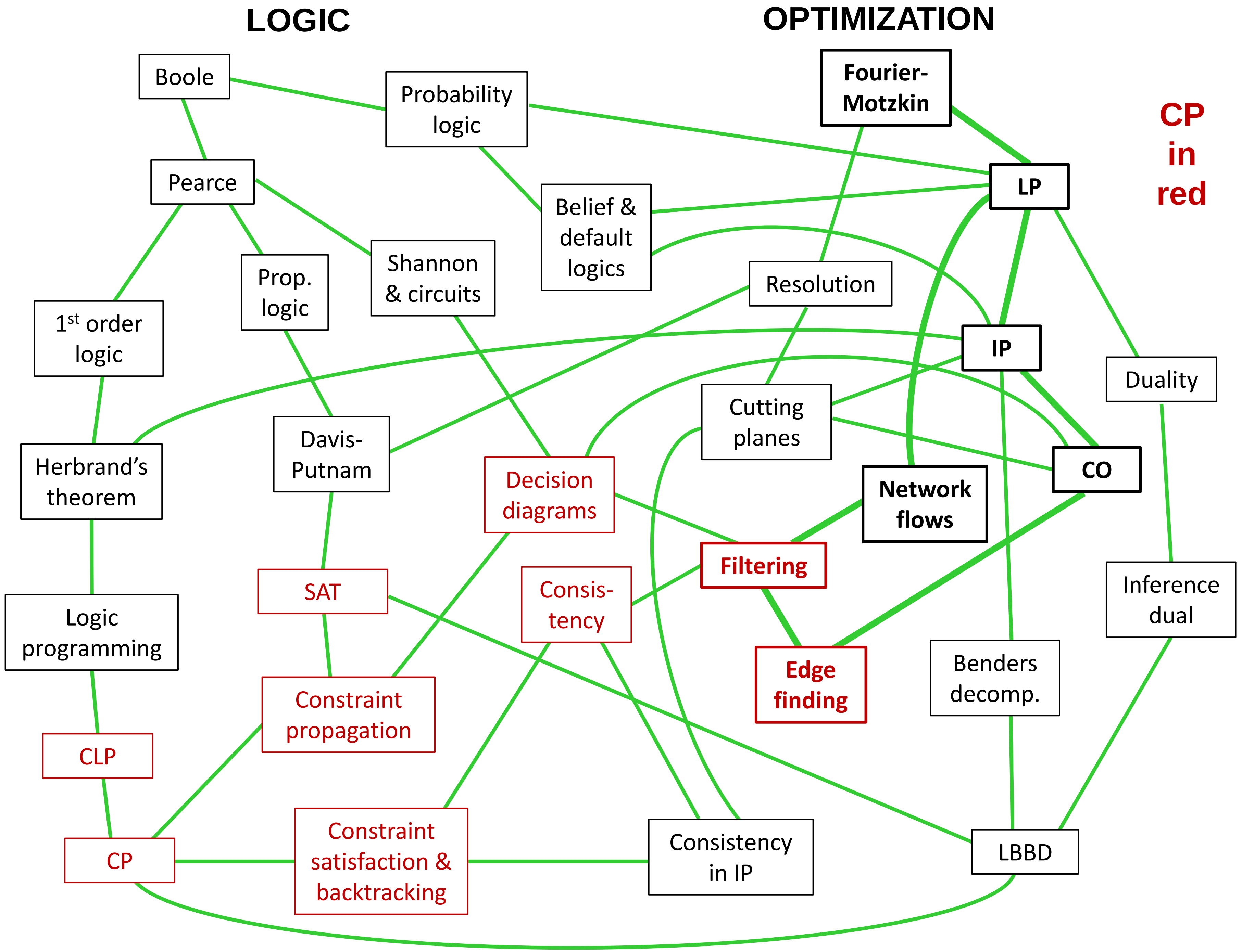
T. HAILPERIN

NORTH-HOLLAND

Theodore Hailperin
1915-2014

Nils Nilsson
1933-2019

From Fourier to Filtering



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

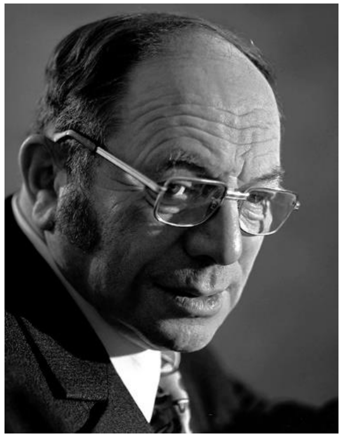
From Fourier to Filtering

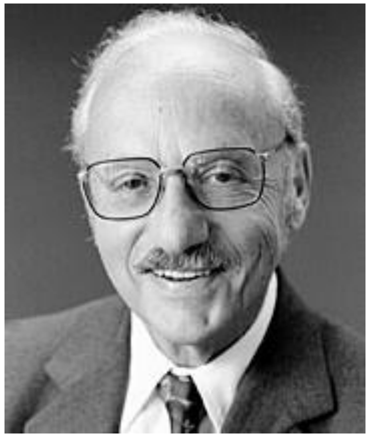
Fourier (1820s) developed a theory of **linear inequalities** and a method of solving them, later rediscovered by Motzkin (1936). The method is now called Fourier-Motzkin elimination.

Kantorovich (1939) formulated a linear optimization problem subject to inequality constraints. Dantzig (1940s) independently proposed a similar model, which we now call **linear programming** (LP).

Joseph Fourier
1768-1830

Theodore Motzkin
1908-1970


Leonid Kantorovich
1912-1986


George Dantzig
1914-2005

From Fourier to Filtering

LP problems can be solved by Fourier-Motzkin elimination, 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.

We have seen connections between IP and various logics. There is more to come…

From Fourier to Filtering

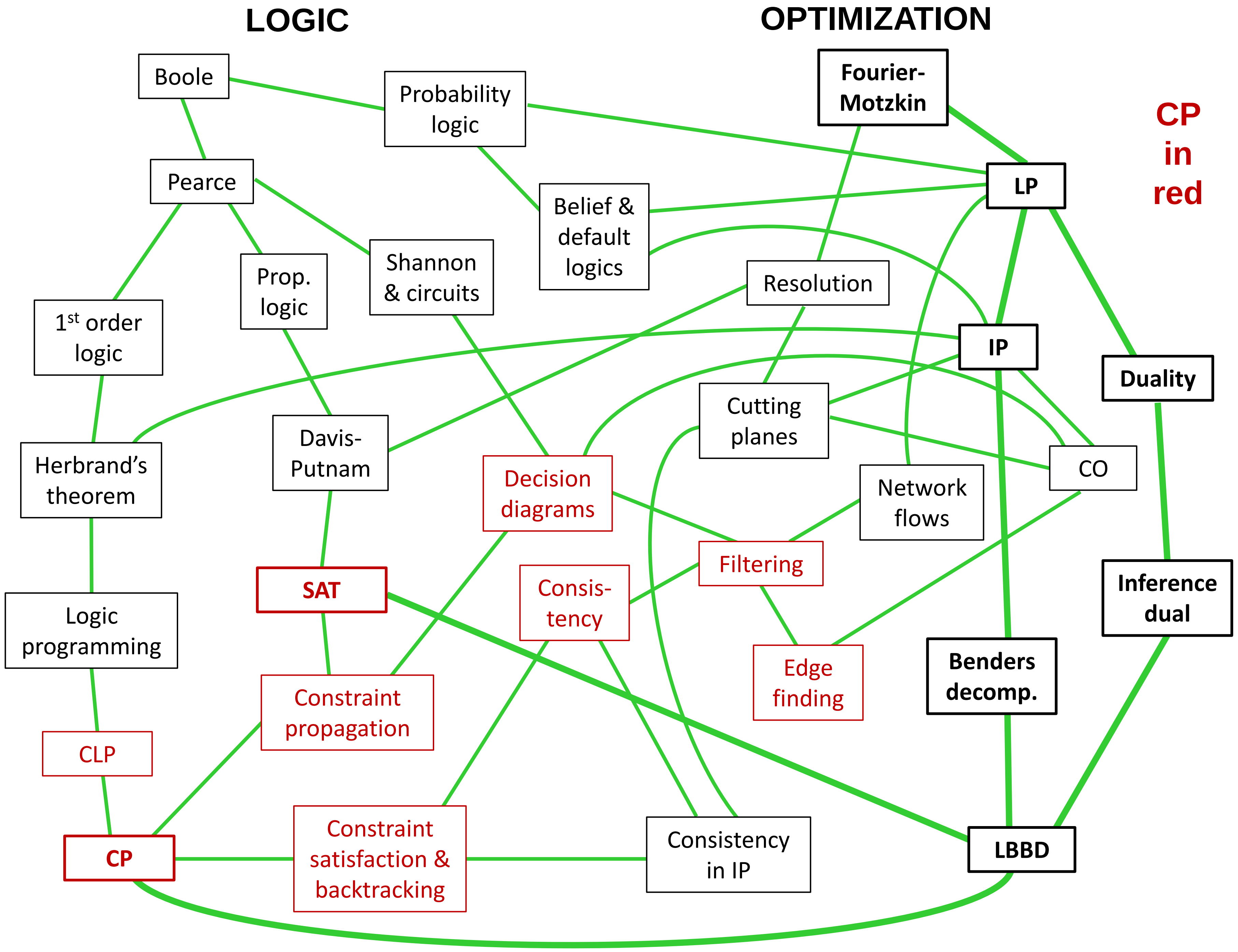
Another outgrowth of LP is **network flow theory**, developed more or less independently in electrical engineering.

Network flow theory (and duality) have been widely applied to **filtering methods in CP**, beginning with the alldiff constraint.

The study of optimization led to the analysis of **combinatorial optimization problems**, beginning in the 1950s with the traveling salesman problem.

An **edge-finding** algorithm for a combinatorial scheduling problem, published in the OR journal *Management Science* (1989), led to powerful **domain reduction methods** for scheduling problems in CP.

From Fourier to Inference Duality


LOGIC
OPTIMIZATION
CP in red
Boole
Probability logic
Fourier-Motzkin
Pearce
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
CP
LBBD

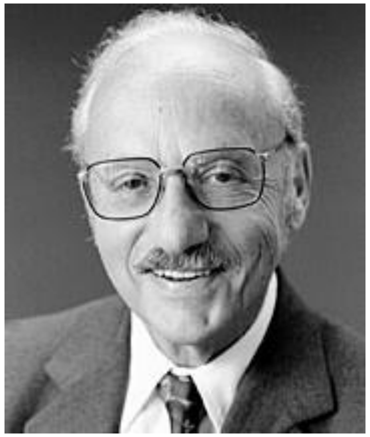
From Fourier to Inference Duality

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

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

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

Dual problem:
Inference

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

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 originally designed for mixed integer programming problems that become LPs after some variables are fixed.

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

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

Benders cuts are obtained by an analysis of the **optimality or infeasibility proof** when the subproblem is solved.

Jacques Benders
1924-2017

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.

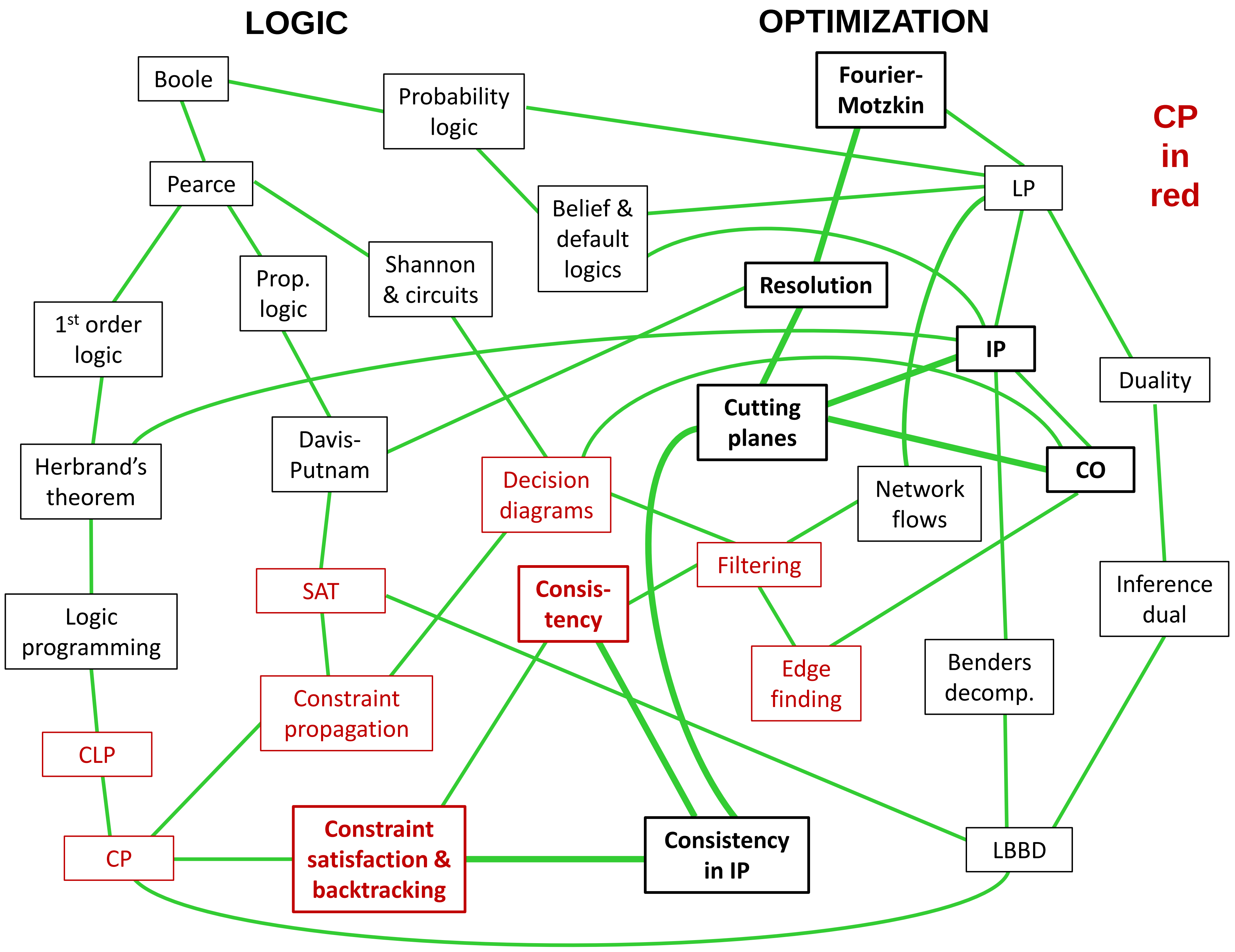
It is a useful tool for **combining optimization and CP**.

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.

In SAT, the inference method that defines the inference dual is **unit resolution**. Benders cuts are obtained by analyzing a **conflict graph**.

From Fourier to Cutting Planes


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

From Fourier to Cutting Planes

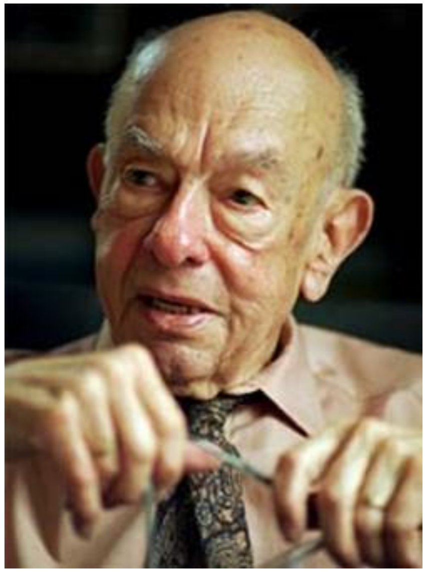
Quine's **resolution method** is very similar to Fourier-Motzkin elimination.

Resolution:

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

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

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


W. V. Quine
1908-2000

From Fourier to Cutting Planes

Resolution lies at the heart of **integer programming**. A resolvent is a **rank 1 Chvátal-Gomory cut.**

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

A **fundamental theorem** of IP states that any valid cutting plane can be obtained from repeated generation of Chvátal-Gomory cuts.

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

Cutting planes play a fundamental role in the solution of IP and other combinatorial problems, beginning with the traveling salesman problem.

From Fourier to Cutting Planes

Cutting planes can also achieve **consistency in integer programming!**

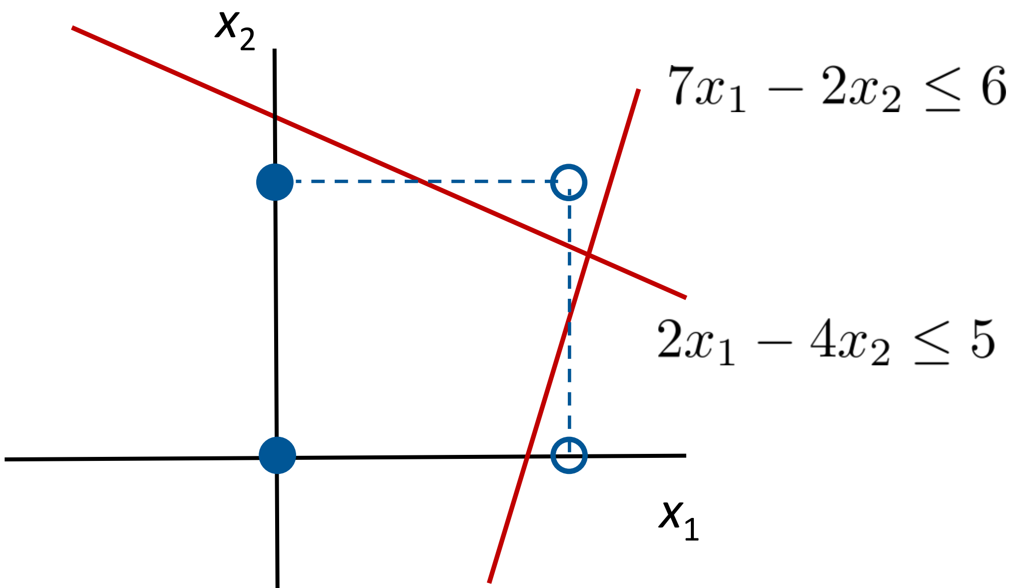
We can view a consistent constraint set as one in which any infeasible **partial assignment** violates a constraint. This avoids backtracking.

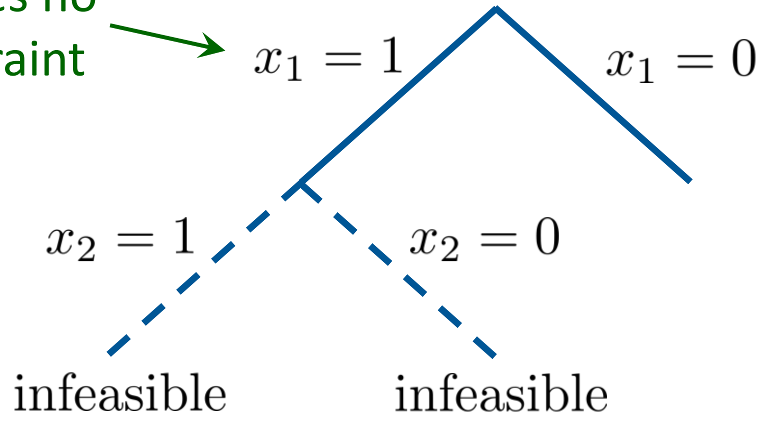
From Fourier to Cutting Planes

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.

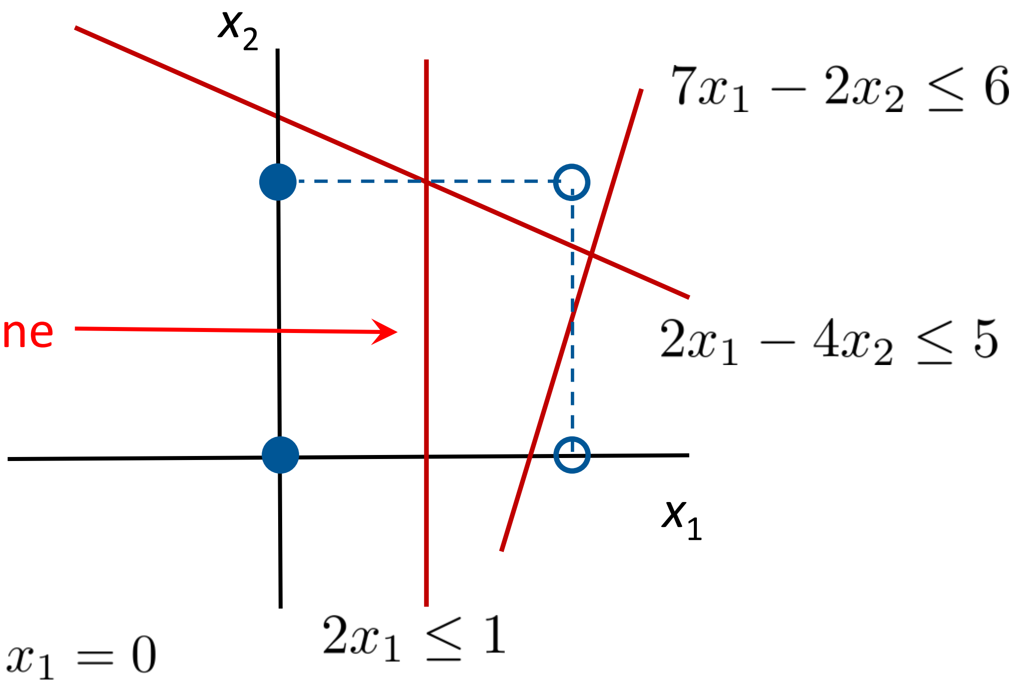
From Fourier to Cutting Planes

The constraint set

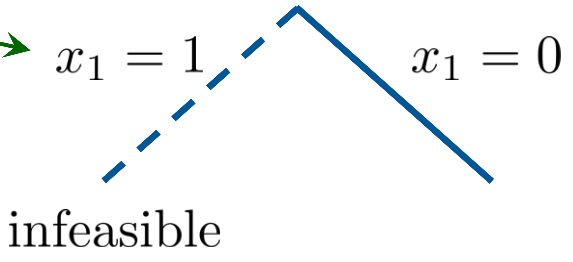
$$2x_1 + 4x_2 \le 5$$
$$7x_1 - 2x_2 \le 6$$
$$\boxed{2x_1 \le 1}$$
$$x_1, x_2 \in \{0, 1\}$$

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

Cutting planes can also achieve **consistency in integer programming!**

We can view a consistent constraint set as one in which any infeasible **partial assignment** violates a constraint. This avoids backtracking.

A concept of **LP-consistency** is more useful in IP:

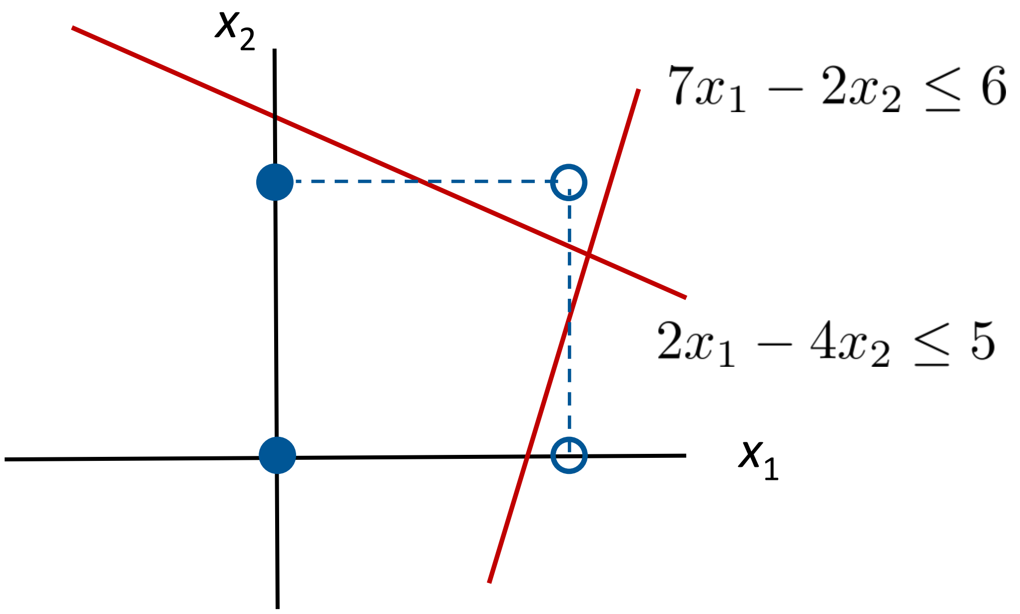
An LP-consistent constraint set is one in which any infeasible partial assignment is infeasible in the **LP relaxation**.

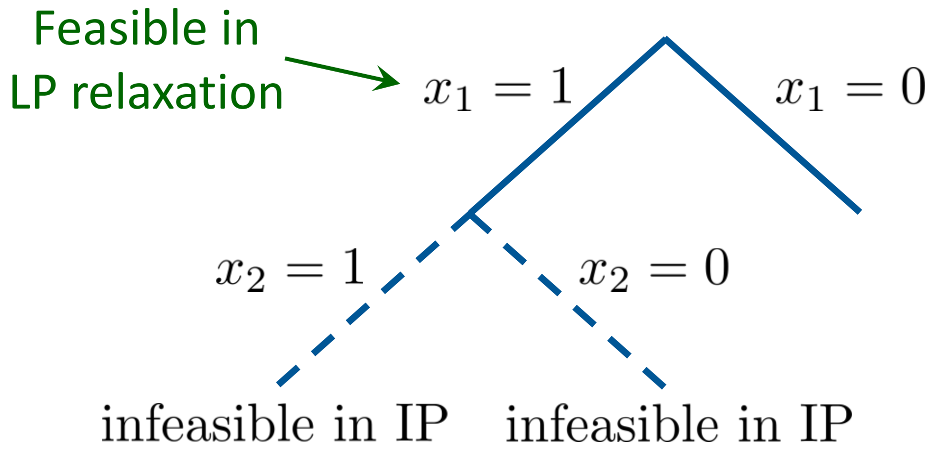
From Fourier to Cutting Planes

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**





Backtracking can result even with forward checking.

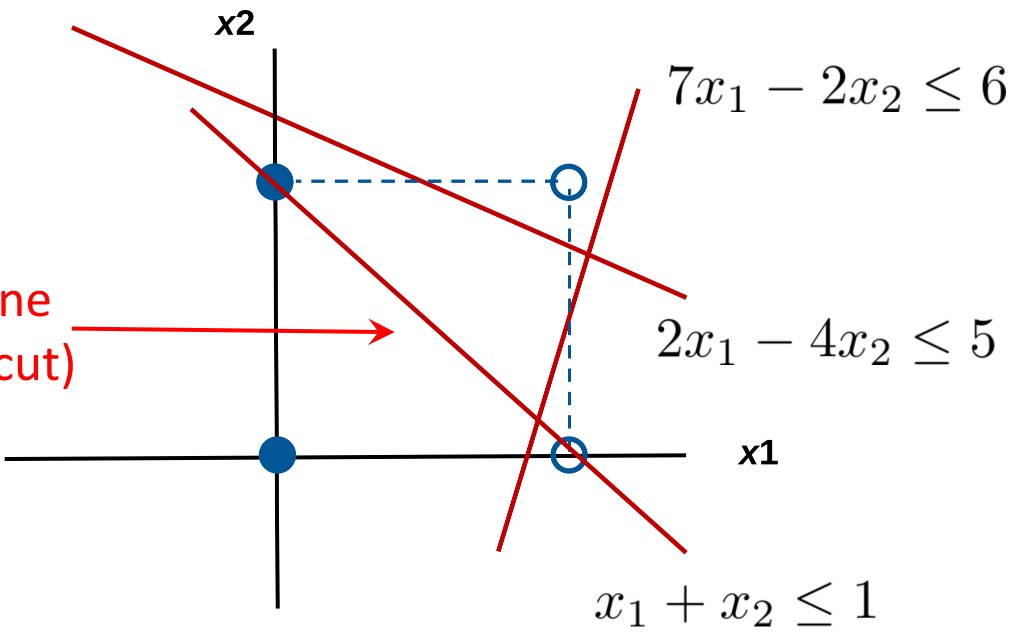
From Fourier to Cutting Planes

The constraint set

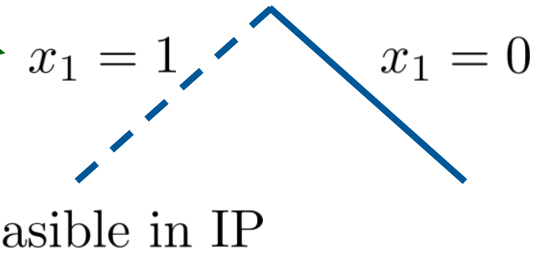
$$2x_1 + 4x_2 \le 5$$
$$7x_1 - 2x_2 \le 6$$
$$\boxed{x_1 + x_2 \le 1}$$
$$x_1, x_2 \in \{0, 1\}$$

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

An IP constraint set is LP-consistent **if and only if** all implied **clausal** inequalities are rank 1 Chvátal-Gomory cuts.

This provides a **new perspective** on cutting planes, which are traditionally used as **separating cuts** (i.e., they cut off fractional solutions).

They can also cut off infeasible partial assignments (**consistency cuts**).

From Fourier to Cutting Planes

An IP constraint set is LP-consistent **if and only if** all implied **clausal** inequalities are rank 1 Chvátal-Gomory cuts.

This provides a **new perspective** on cutting planes, which are traditionally used as **separating cuts** (i.e., they cut off fractional solutions).

They can also cut off infeasible partial assignments (**consistency cuts**).

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

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

Research program: Explore **consistency properties** of other cuts.

Thanks for your attention!