Logic, Constraints, and Quantum Information

Phokion G. Kolaitis

UC Santa Cruz & IBM Research – Almaden

Joint work with

Albert Atserias, UPC and Simone Severini, UCL

Collaboration with Lauri Hella

- Hella, K …, Luosto: LICS 1994 & APAL 1997
  How to Define a Linear Order on Finite Models
- Dawar, Hella, K … : ICALP 1995
  Implicit Definability and Infinitary Logic in Finite Model Theory
- Hella, K …, Luosto: Bulletin of the ASL 1997
  Almost Everywhere Equivalence of Logics in Finite Model Theory
- Hella and K … : CSL 2016
  Dependence Logic vs. Constraint Satisfaction

Lauri Hella as I know him

- Brilliant researcher
- Principled scientist
- Wonderful human being
- True friend

Logic, Constraints, and Quantum Information

Phokion G. Kolaitis

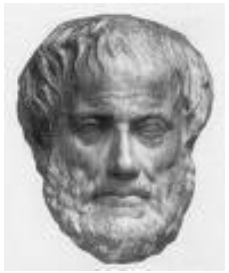
UC Santa Cruz & IBM Research – Almaden

Joint work with

Albert Atserias, UPC and Simone Severini, UCL

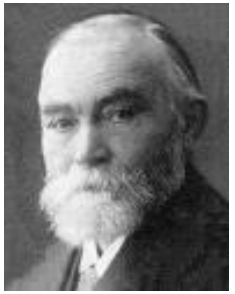
Three Milestones in the Development of Logic

- Aristotle, 384-322 BC
  Syllogistic Logic



- George Boole, 1815-1864
  Propositional Logic
  $(x \vee \neg y) \wedge (\neg x \vee z \vee \neg w)$

- Gottlob Frege, 1848-1925
  First-Order Logic
  $(\forall x)\,(\forall y)(E(x,y) \rightarrow \exists z\,(E(x,z) \wedge E(y,z))$



Computability and Undecidability

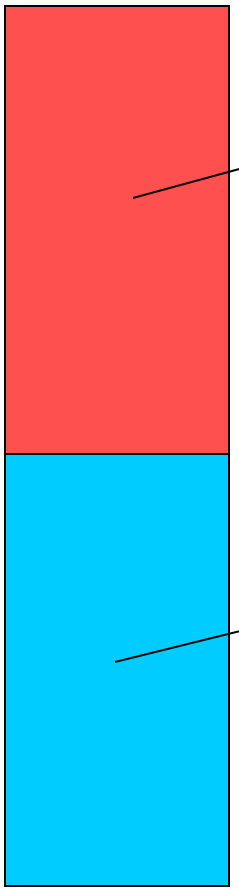
Kurt Gödel

Alan Turing

Alonzo Church

- Revolutionary research in mathematical logic and the foundations of mathematics in the 1930s.
- Formalization of the notion of computable function.
- Discovery of undecidable problems (no algorithm exists):
  Given a first-order formula φ, is φ true on (N, +, ·) ?

Computer Science and Computational Complexity



- Computer Science is the study of algorithms.
- Computational Complexity is the quantitative study of decidable problems.
- Decidable problems are organized in complexity classes according to the computational resources needed to solve them.

Complexity Classes

Definition:

- P = the class of all decision problems solvable by an algorithm in polynomial time
- NP = the class of all decision problems for which an alleged solution can be verified in polynomial time.

Main Open Question in Theoretical Computer Science:

Is P = NP?

Cook's Theorem (1971):

- NP contains complete problems (i.e., "hardest" in NP).
- 3SAT is NP-complete.

Boolean Satisfiability

- 3SAT: Given a 3CNF-formula φ, is it satisfiable?
  - 3CNF-formula: $c_1 \wedge \ldots \wedge c_m$, where each c_i is one of

$(x \vee y \vee z), (\neg x \vee y \vee z), (\neg x \vee \neg y \vee z), (\neg x \vee \neg y \vee \neg z)$

- 3SAT is in NP: Given a 3CNF-formula φ and an assignment s of values 0/1 to the variables of φ, we can verify in polynomial time whether or not s satisfies φ.

- Cook's Theorem: 3SAT is NP-complete, i.e., every problem in NP can be reduced to 3SAT in polynomial time. Hence,

P = NP if and only if 3SAT is in P.

Constraint Satisfaction

Instance (V,D,C) of a Constraint Satisfaction Problem (CSP)

- Input:
  - Set V of variables
  - Set D for the values of the variables, called the domain
  - Set C of constraints of the form (t,R), where
    - t is a tuple $(x_1,\ldots,x_k)$ of variables
    - R is a k-ary relation on D (i.e., $R \subseteq D^k$)
- Question: Is there a solution?
  - Is there an assignment h of values to variables so that all constraints are satisfied?

    (i.e., $(h(x_1), \ldots, h(x_k)) \in R$, for each constraint (t,R) in C)

Logic and Constraint Satisfaction

- 3SAT: Given a 3CNF-formula φ, is it satisfiable?
  - 3CNF-formula: $c_1 \wedge \ldots \wedge c_m$, where each c_i is one of

$(x \vee y \vee z)$, $(\neg x \vee y \vee z)$, $(\neg x \vee \neg y \vee z)$, $(\neg x \vee \neg y \vee \neg z)$

- 3SAT as a Constraint Satisfaction Problem
  - V = set of variables occurring in φ
  - D = { 0,1 }
  - Constraints of the form (t,R_0), (t,R_1), (t,R_2), (t,R_3), where t = (x,y,z) is a triple of variables and $R_0 = \{ 0,1 \}^3 \setminus \{ (0,0,0) \}$, $R_1 = \{ 0,1 \}^3 \setminus \{ (1,0,0) \}$, $R_2 = \{ 0,1 \}^3 \setminus \{ (1,1,0) \}$, $R_3 = \{ 0,1 \}^3 \setminus \{ (1,1,1) \}$

Logic and Constraint Satisfaction

- 2SAT: Given a 2CNF-formula φ, is it satisfiable?
  - 2CNF-formula: $c_1 \wedge \ldots \wedge c_m$, where each c_i is one of $(x \vee y)$, $(\neg x \vee y)$, $(\neg x \vee \neg y)$

- 2SAT as a Constraint Satisfaction Problem
  - V = set of variables occurring in φ
  - D = { 0,1 }
  - Constraints of the form (t,P_0), (t,P_1), (t,P_2), where t = (x,y) is a pair of variables and $P_0 = \{ 0,1 \}^2 \setminus \{ (0,0) \}$, $P_1 = \{ 0,1 \}^2 \setminus \{ (1,0) \}$, $P_2 = \{ 0,1 \}^2 \setminus \{ (1,1) \}$

Generalized Satisfiability Problems

- A Boolean constraint language is a set Γ of Boolean relations, i.e., $\Gamma = \{ R_1, \ldots, R_i, \ldots, \}$ with each $R_i \subset \{ 0,1 \}^k$ for some k.

- CNF(Γ): Formulas of the form $c_1 \wedge \ldots \wedge c_m$, where each c_j is of the form $R_i(t)$ with t a tuple of k variables.

- SAT(Γ): Given a CNF(Γ)-formula φ, is it satisfiable?

- SAT(Γ) as a Constraint Satisfaction Problem
  - V = set of variables occurring in φ
  - D = { 0,1 }
  - Constraints of the form (t,R_i) with t a tuple of k variables.

Generalized Satisfiability Problems

- Example: 3SAT = SAT({ R_0, R_1, R_2, R_3 })
- Example: 2SAT = SAT({ P_0, P_1, P_2 })
- Example: POSITIVE-1-in-3-SAT

  Input: 3CNF-formula $c_1 \wedge \ldots \wedge c_m$, where each c_i is of the form $(x \vee y \vee z)$

  Question: Is there an assignment that makes true exactly one variable in each constraint?

  Fact: POSITIVE-1-in-3-SAT = SAT({ $R_{1/3}$ }), where

  $R_{1/3}$ = { (1,0,0), (0,1,0), (0,0,1) }

Computational Complexity of SAT(Γ)

Theorem:

- 2SAT is in P (Krom - 1967)
- 3SAT is NP-complete (Cook - 1971)
- POSITIVE-1-in-3-SAT is NP-complete (Schaefer – 1978).

Question:

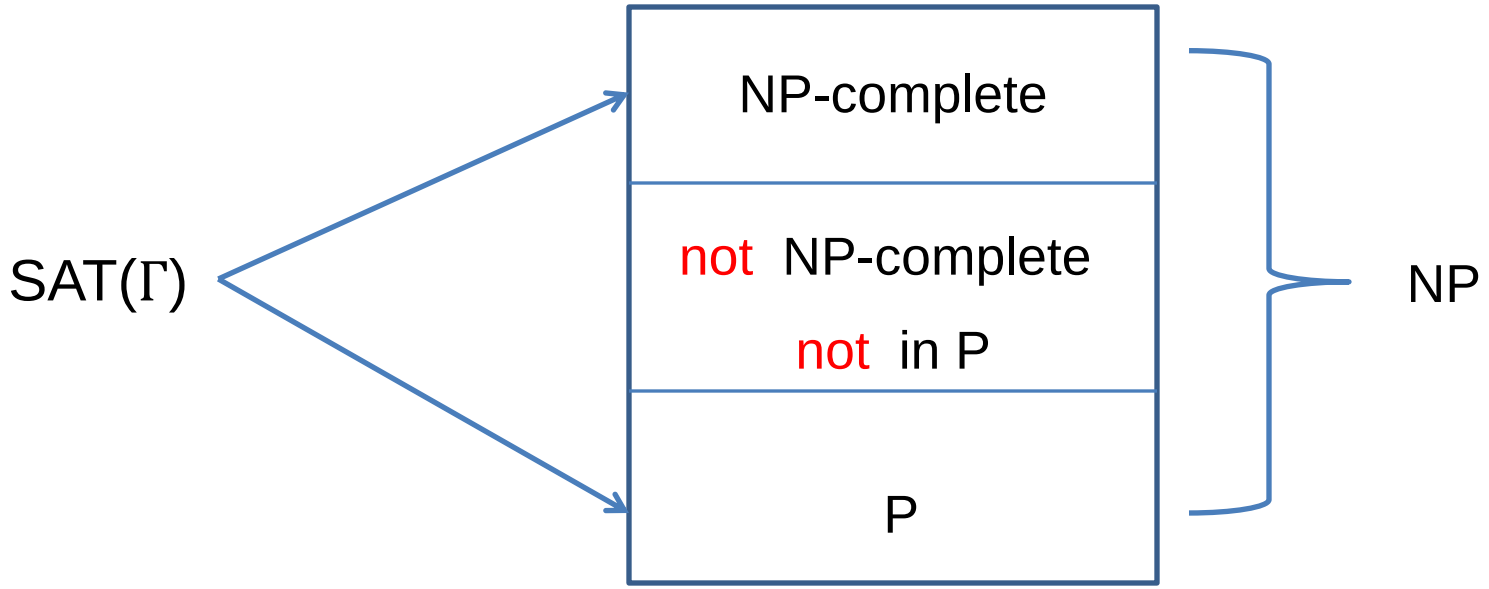
- Let Γ be a Boolean constraint language.
  What can we say about the complexity of SAT(Γ)?
- Is there a *general* result that *explains* the complexity of
  2SAT, 3SAT, and POSITIVE-1-in-3-SAT?

Computational Complexity of SAT(Γ)

Schaefer's Dichotomy Theorem (1978)

If Γ is a Boolean constraint language, then

either SAT(Γ) is in P or SAT(Γ) is NP-complete.



Six Special Types of Boolean Relations

Definition: Let $R \subseteq \{ 0,1 \}^k$ be a Boolean relation.

1. R is 0-valid if $(0,0,\ldots,0) \in R$.
2. R is 1-valid if $(1,1,\ldots,1) \in R$.
3. R is bijunctive if R is the set of satisfying assignments of a 2CNF-formula.
4. R is Horn if R is the set of satisfying assignments of a Horn formula, i.e., a CNF-formula each clause of which has at most one positive literal.
5. R is dual Horn if R is the set of satisfying assignments of a dual Horn formula, i.e., a CNF-formula each clause of which has at most one negative literal.
6. R is linear (affine) if R is the set of solutions of a system of linear equations over the 2-element field.

Computational Complexity of SAT(Γ)

Schaefer's Dichotomy Theorem – Revisited

Let Γ be a Boolean constraint language.

- If Γ satisfies at least one of the following six conditions, then SAT(Γ) is in P
  1. Γ is 0-valid (i.e., every relation in Γ is 0-valid);
  2. Γ is 1-valid (i.e., every relation in Γ is 1-valid);
  3. Γ is bijunctive (i.e., every relation in Γ is bijunctive);
  4. Γ is Horn (i.e., every relation in Γ is Horn);
  5. Γ is dual Horn (i.e., every relation in Γ is dual Horn);
  6. Γ is linear (i.e., every relation in Γ is linear).
- Otherwise, SAT(Γ) is NP-complete.

Computational Complexity of SAT(Γ)

Γ	Complexity of SAT(Γ)
0-valid	P
1-valid	P
Bijunctive	P
Horn	P
Dual Horn	P
Linear	P
None of the above	NP-complete

We always did feel the same
We just saw it from a different point
Of view
Tangled up in blue

Bob Dylan - 1975

A Change in Perspective

- Boolean Domain = $\{ 0,1 \}$
- Boolean relation $R \subseteq \{ 0,1 \}^k$
- Characteristic function $\chi_R : \{ 0,1 \}^k \rightarrow \{ 0,1 \}$

Consider the following translation:

$$0 \leftrightarrow +1, \quad 1 \leftrightarrow -1$$

- Boolean Domain = $\{ +1,-1 \}$
- Boolean relation $R \subseteq \{ +1,-1 \}^k$
- Characteristic function $\chi_R : \{ +1,-1 \}^k \rightarrow \{ +1,-1 \}$

A Change in Perspective

Fact: Let $R \subseteq \{ 0,1 \}^k$ be a Boolean relation. The characteristic function $\chi_R : \{ +1,-1 \}^k \rightarrow \{ +1,-1 \}$ of R can be uniquely represented by a multilinear polynomial.

Proof: It is the Fourier Transform.

Example 1: Let R be the relation defined by $(x \wedge y)$

- Then $\chi_R(x,y) = \frac{1}{2}(x+y-xy+1)$

Example 2: Let R be the relation defined by $(x \vee y)$

- Then $\chi_R(x,y) = \frac{1}{2}(x+y+xy-1)$

Example 3: Let R be the relation defined by $x+y+z = 1 \bmod(2)$.

- Then $\chi_R(x,y,z) = xyz$

Example 4: Let R be the relation defined by $x+y+z = 0 \bmod(2)$.

- Then $\chi_R(x,y,z) = -xyz$

Relaxations of Constraint Satisfaction

- Question: What is the benefit of the change in perspective?

- Answer:
  - The change in perspective allows for an expansion of the horizon.
  - By representing Boolean relations as multilinear polynomials, we can investigate relaxations of constraint satisfaction in which generalized assignments are allowed, i.e., the variables may take values in domain richer than the Boolean domain.

Mermin's Magic Square (1990)

- CSP instance given by the system of linear equations

$x_1 + x_2 + x_3 = 0 \bmod(2)$ $\quad$ $x_1 + x_4 + x_7 = 0 \bmod(2)$

$x_4 + x_5 + x_6 = 0 \bmod(2)$ $\quad$ $x_2 + x_5 + x_8 = 0 \bmod(2)$

$x_7 + x_8 + x_9 = 0 \bmod(2)$ $\quad$ $x_3 + x_6 + x_9 = 1 \bmod(2)$

- This system has no solutions in { 0,1 } because

$$0 = x_1 + x_2 + \ldots + x_9 = 1$$

X_1	X_2	X_3	0
X_4	X_5	X_6	0
X_7	X_8	X_9	0
0	0	1	

Mermin's Magic Square (1990)

- $x_1\, x_2\, x_3 = +1$ $\quad x_1\, x_4\, x_7 = +1$
  $x_4\, x_5\, x_6 = +1$ $\quad x_2\, x_5\, x_8 = +1$
  $x_7\, x_8\, x_9 = +1$ $\quad x_3\, x_6\, x_9 = -1$
- This system has no solutions in { +1,-1 }
- This system has a solution in 4×4 complex matrices

$(I \otimes Z)(Z \otimes I)(Z \otimes Z) = +I$ $\quad (I \otimes Z)(Z \otimes I)(X \otimes Z) = +I$

$(X \otimes I)(I \otimes X)(X \otimes X) = +I$ $\quad (Z \otimes I)(I \otimes X)(Z \otimes X) = +I$

$(X \otimes Z)(Z \otimes X)(Y \otimes Y) = +I$ $\quad (Z \otimes Z)(X \otimes X)(Y \otimes Y) = -I$

$$X = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Pauli matrices

Remarks on Mermin's Magic Square

Note:

The lack of solutions in { +1, -1 } depends on the pairwise commutativity of variables.

Fact: The solution in 4×4 complex matrices has the following properties:

- The values of variables occurring in the same equation pairwise commute.
- Each value A is Hermitian (self-adjoint), i.e., A = A*.
- Each value A is such that $A^2 = +I$ (hence, A is unitary)

$(I \otimes Z)^2 = (X \otimes X)^2 = (Z \otimes Z)^2 = (X \otimes Z)^2 = (Y \otimes Y)^2 = \ldots = +I$

Satisfiability via Operator Assignments

Definition: Cleve and Mittal - 2015

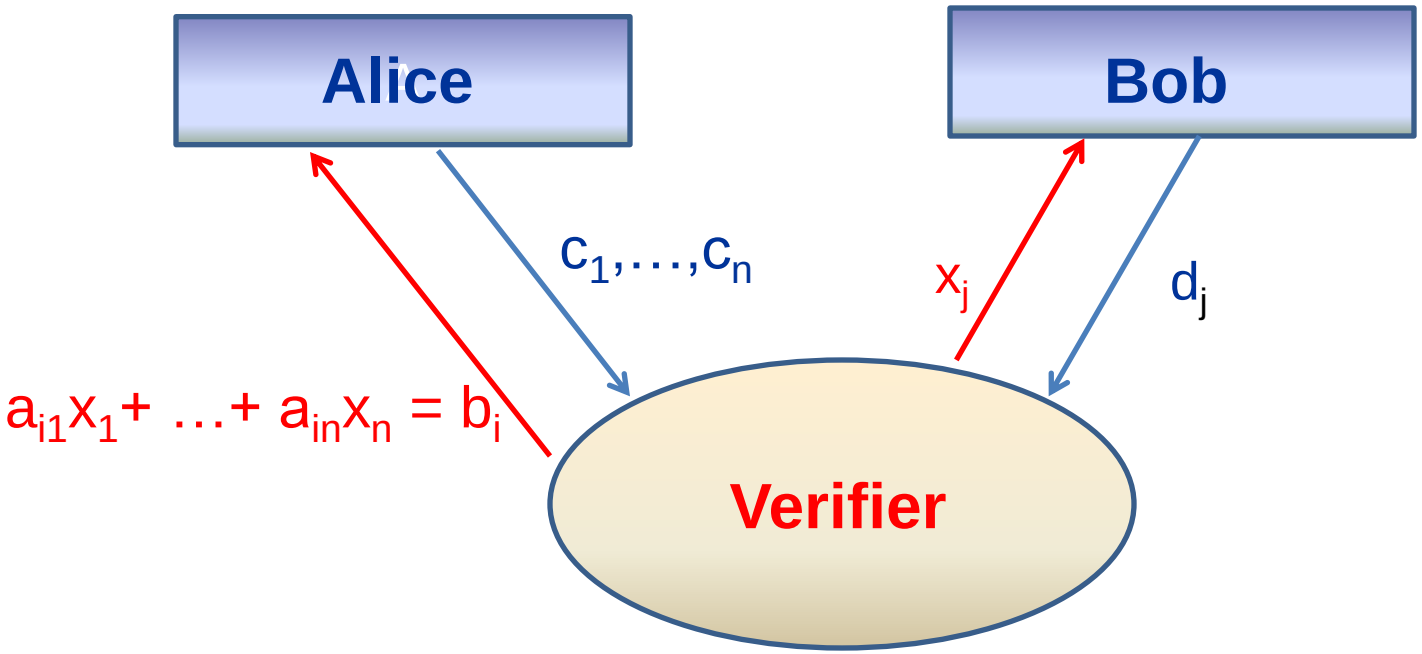
Let Γ be a Boolean constraint language and let $\varphi \equiv c_1 \wedge \ldots \wedge c_m$ be a CNF(Γ)-formula with variables $x_1,\ldots,x_n$.

- φ is satisfiable via operators if there are linear operators $A_1, \ldots, A_n$ on some Hilbert space **H** such that
  - A_i is self-adjoint (i.e., $A_i = A^*_i$) and $A_i^2 = +I$ for each $i \leq n$.
  - $A_iA_j = A_jA_i$, for all i and j such that both x_i and x_j appear in some constraint c_k of φ.
  - $A_1, \ldots, A_n$ satisfy every constraint c_k of φ, where c_k is viewed as a multilinear polynomial.
- φ is satisfiable via finite-dimensional operators (fd-operators) if φ is satisfiable via operators in some Hilbert space of finite dimension (i.e., in $\mathbf{C}^d$, for some $d \geq 1$).

Non-Local Games

- Two players, Alice and Bob, play against a Verifier using a system A**x** = **b** of linear equations mod(2) as a board.
- Alice and Bob know the system and can communicate before the game starts, but not during the game (non-local).
- In a play of the game, the Verifier
  - sends Alice one of the equations $a_{i1}x_1 + \ldots + a_{in}x_n = b_i$
  - sends Bob one of the variables x_j so that $a_{ij} \neq 0$.
- Alice assigns values $c_1,\ldots,c_n \in \{ 0,1 \}$ to the variables $x_1, \ldots, x_n$ so that the equation $a_{i1}c_1 + \ldots + a_{in}c_n = b_i$ is satisfied.
- Bob assigns a value $d_j \in \{ 0,1 \}$ to x_j.
- Alice and Bob win if $c_j = d_j$.

Non-Local Games



Winning Condition

$$a_{i1}c_1 + \ldots + a_{in}c_n = b_i$$

$$c_j = d_j$$

Entangled Non-Local Games

Fact: Alice and Bob have a winning strategy

if and only if

the system $A\mathbf{x} = \mathbf{b}$ is satisfiable in { 0,1 }.

Theorem (Cleve-Mittal 2015 and Cleve-Liu-Slofstra 2016)

- Alice and Bob have a winning strategy that uses an entangled state in the tensor-product model

  if and only if

  the system $A\mathbf{x} = \mathbf{b}$ is satisfiable via fd-operators.
- Alice and Bob have a winning strategy that uses an entangled state in the commuting-operator model

  if and only if

  the system $A\mathbf{x} = \mathbf{b}$ is satisfiable via operators.

Three Variants of Satisfiability

Definition: Let Γ be a Boolean constraint language.

- SAT(Γ): (classical satisfiability)

  Given a CNF(Γ)-formula φ, is φ satisfiable in { +1, -1 }?

- SAT*(Γ): (satisfiability via fd-operators)

  Given a CNF(Γ)-formula φ, is φ satisfiable via fd-operators?

- SAT**(Γ): (satisfiability via operators)

  Given a CNF(Γ)-formula φ, is φ satisfiable via operators?

Note: classical sat. $\Rightarrow$ sat. via fd-operators $\Rightarrow$ sat via operators

Gaps in Satisfiability

Definition: Let Γ be a Boolean constraint language.

- A CNF(Γ)-formula φ has

- a gap of the first kind if
  φ is "**yes**" for SAT*(Γ) and "**no**" for SAT(Γ);
- a gap of the second kind if
  φ is "**yes**" for SAT**(Γ) and "**no**" for SAT(Γ);
- a gap of the third kind if
  φ is "**yes**" for SAT**(Γ) and "**no**" for SAT*(Γ).

- Γ has a gap of the i-th kind if there is a CNF(Γ)-formula that has a gap of the i-th kind, i = 1, 2, 3.

Mnemonic: Add the stars to determine the kind of the gap.

Gaps in Satisfiability

Theorem: Let LIN be the Boolean constraint language that consists of all linear Boolean relations.

- (Mermin – 1990) LIN has a gap of the first kind.
- (Slofstra – 2016) LIN has a gap of the third kind

Hence, LIN has gaps of every kind.

Proof:

- Mermin's Magic Square yields a gap of the first kind for LIN.
- Slofstra showed that there is a system of linear equations that is satisfiable via operators in some infinite-dimensional Hilbert space, but it is not satisfiable via fd-operators.

  The proof uses deep results about finitely-presentable groups.

No Gaps in Satisfiability

Theorem: (Ji 2014)

- 2SAT has no gaps of the first kind.
- Horn SAT has no gaps of the first kind.

Proof Idea:

The polynomial-time algorithms for 2SAT and for Horn SAT can be used to show that if a 2CNF-formula or a Horn formula is not satisfiable in the Boolean domain, then it is not satisfiable via fd-operators.

Gaps in Satisfiability

Summary:

- LIN has gaps of every kind
- 2SAT has no gaps of the first kind.
- Horn SAT has no gaps of the first kind.

Question: Let Γ be an arbitrary Boolean constraint language.

- Does Γ have any kind of gaps?
- If so, what kinds of gaps does Γ have?

Classification of Gaps in Satisfiability

Theorem: (Atserias, K …, Severini – 2017)

If Γ is a Boolean constraint language, then

either Γ has gaps of every kind or Γ has gaps of no kind.

Moreover, Γ has gaps of no kind precisely when satisfies at least one of the following five conditions:

1. Γ is 0-valid;
2. Γ is 1-valid;
3. Γ is bijunctive;
4. Γ is Horn;
5. Γ is dual Horn.

Complexity of SAT(Γ) vs. Gaps for Γ

Γ	Complexity of SAT(Γ)	Gaps for Γ
0-valid	P	No kind
1-valid	P	No kind
Bijunctive	P	No kind
Horn	P	No kind
Dual Horn	P	No kind
Linear	P	Every kind
None of the above	NP-complete	Every kind

Classification of Gaps in Satisfiability

Theorem: (Atserias, K …, Severini – 2017)

If Γ is a Boolean constraint language, then

either Γ has gaps of every kind or Γ has gaps of no kind.

Proof: Main ingredients:

- pp-definability and gap-preserving reductions;
- Post's Lattice;
- Mermin's Magic Square
- Slofstra's Theorem about gaps for LIN.

Primitive Positive Definability

Definition: Let Γ be a Boolean constraint language.

A Boolean relation R is pp-definable from Γ if

$R(x_1,\ldots,x_k) \equiv \exists z_1 \ldots \exists z_s (B_1 \wedge \ldots \wedge B_m)$, where each B_i is a relation in Γ with variables from $x_1,\ldots,x_k, z_1,\ldots,z_s$.

Example: Not-All-Equal Relation NAE

Consider NAE = $\{ 0,1 \}^3 \setminus \{ (0,0,0), (1,1,1) \}$

- $NAE(x,y,z) \equiv (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$
- $NAE(x,y,z) \equiv R_0(x,y,z) \wedge R_3(x,y,z)$

Thus, NAE is pp-definable from $\{ R_0, R_3 \}$

Gap-Preserving Reductions

Note: Extensive study of pp-definability in logic, constraint satisfaction, and database theory.

Lemma: Let Γ and Δ be two Boolean constraint languages. If every relation in Δ is pp-definable from Γ, then gaps for Δ imply gaps of the same kind for Γ.

Proof: Uses the Spectral Theorem.

Note: The preceding lemma provides a tool for establishing gaps for a constraint language using pp-definability and known gaps for some other constraint language.

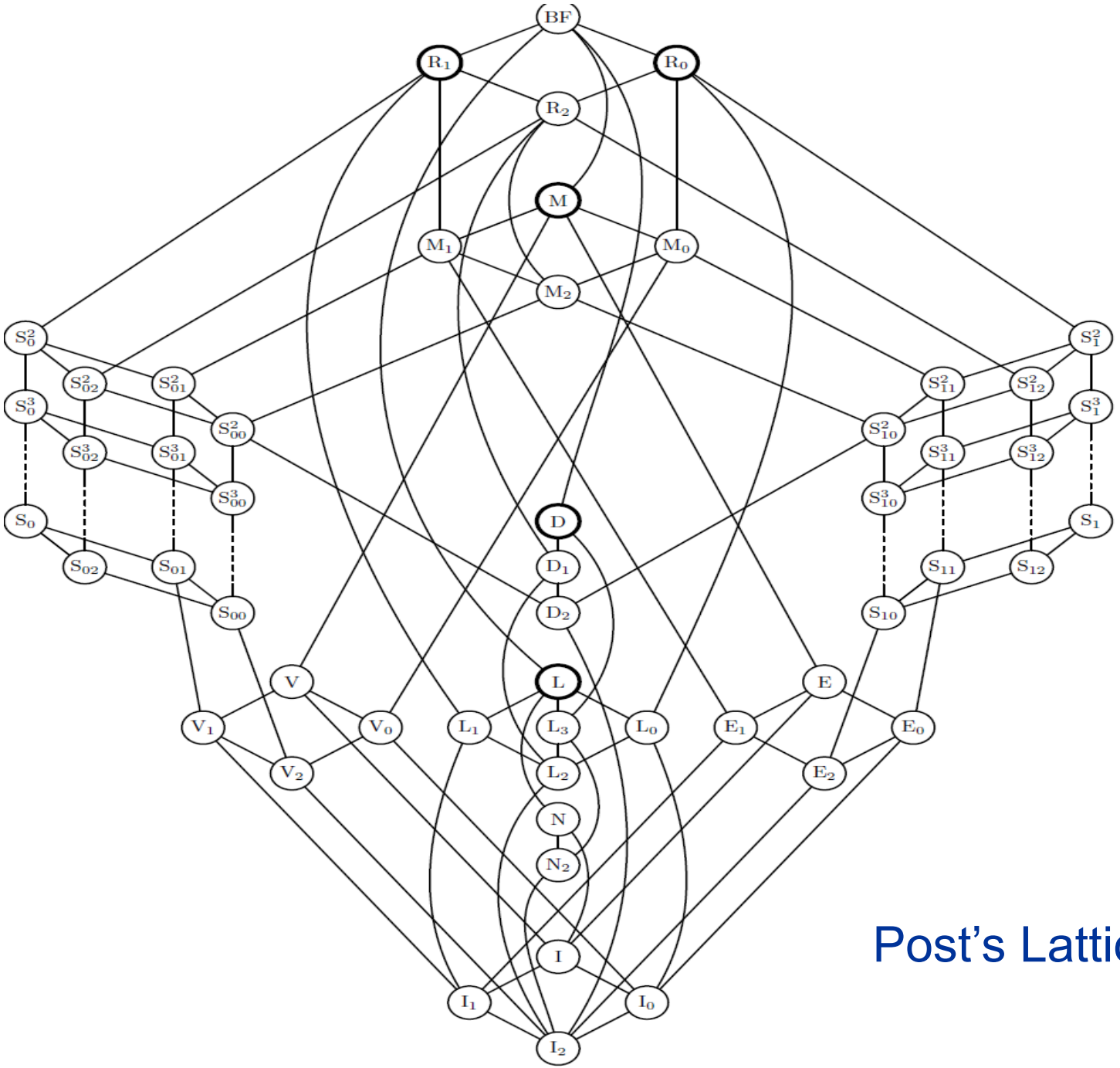
Primitive Positive Definability

Definition: Let Γ be a Boolean constraint language.

We write $[\Gamma]$ to denote the collection of all Boolean relations that are pp-definable from Γ.

Theorem: (Post – 1941)

- There are countably many collections of the form $[\Gamma]$, where Γ varies over all Boolean constraint languages (there are uncountably many constraint languages Γ).
- Explicit description of the lattice of all such collections $[\Gamma]$ with respect to set-theoretic containment $\subseteq$.



Post's Lattice

Classification of Gaps in Satisfiability

Theorem: (Atserias, K …, Severini – 2017)

If Γ is a Boolean constraint language, then

either Γ has gaps of every kind or Γ has gaps of no kind.

Moreover, the following statements are equivalent:

- Γ has gaps of every kind.
- LIN is pp-definable from Γ.
- Γ is not 0-valid, 1-valid, bijunctive, Horn, dual Horn.

Algorithmic Aspects

Fact: SAT(LIN) is solvable in polynomial time
(e.g., using Gaussian elimination).

Theorem: (Slofstra – 2016)
SAT**(LIN) is undecidable
Proof: Uses the undecidability of the word problem for groups.

Open Problem:

- Is SAT*(LIN) decidable?
- If so, what is the exact complexity of SAT*(LIN)?

Φύσις κρύπτεσθαι φιλεῖ

Nature likes to hide

Heraclitus, Fragment B123 DK