Proof Complexity

Olaf Beyersdorff

School of Computing
University of Leeds, UK

RiSE Winter School, Vienna, 2018

Outline of the Course

Proof Systems

Definition (Cook, Reckhow 79)

A proof system for a language L is a function f with $rng(f) = L$.
If $f(w) = x$, then w is called an f-proof of $x \in L$.

- correctness: $rng(f) \subseteq L$
- completeness: $L \subseteq rng(f)$
- efficiency: proofs should be easy to check,
  i.e. f should be easy to compute.

- Most research in proof complexity has studied propositional proof systems where $L = TAUT$.

A First Example: Truth Tables

A proof system for TAUT

$$TT(\alpha, \varphi) = \begin{cases} \varphi & \text{if } \alpha \text{ is a truth table for } \varphi \text{ with all entries } 1 \\ p \vee \neg p & \text{otherwise.} \end{cases}$$

Why is this not a good proof system?

- Most proofs are exponentially long in the size of the formula.

A First Example: Truth Tables

A proof system for TAUT

$$TT(\alpha, \varphi) = \begin{cases} \varphi & \text{if } \alpha \text{ is a truth table for } \varphi \text{ with all entries 1} \\ p \vee \neg p & \text{otherwise.} \end{cases}$$

Why is this not a good proof system?

- Most proofs are exponentially long in the size of the formula.
- We look for proof systems with shorter proofs.

The Most Studied Proof System: Resolution

- Introduced by Blake 1937, Davis & Putnam 1960, and Robinson 1965
- Resolution proofs operate with clauses.
- Resolution proofs are refutations.

The Most Studied Proof System: Resolution

- Introduced by Blake 1937, Davis & Putnam 1960, and Robinson 1965
- Resolution proofs operate with clauses.
- Resolution proofs are refutations.

Definition

Let C and D be clauses with $p \in C$ and $\neg p \in D$.
The Resolution rule applied to C and D yields the clause $(C \setminus \{p\}) \cup (D \setminus \{\neg p\})$.

Notation: $$\frac{C \qquad\qquad D}{(C \setminus \{p\}) \cup (D \setminus \{\neg p\})}$$

Resolution Derivations

Definition

Let Γ be a set of clauses. A Resolution derivation of a clause C from Γ is a sequence

$$C_1, \ldots, C_k = C$$

of clauses such that for all $i = 1, \ldots, k$:

1. $C_i \in \Gamma$ or
2. there exist $1 \leq j_1 \leq j_2 < i$ with

$$\frac{C_{j_1} \quad C_{j_2}}{C_i} \quad .$$

Resolution Refutations

Definition

A Resolution refutation of Γ is a Resolution derivation of the empty clause $\Box$ from Γ.

Resolution Refutations

Definition

A Resolution refutation of Γ is a Resolution derivation of the empty clause $\Box$ from Γ.

Example

$\Gamma = \{\{p, q\}, \{p, \neg q\}, \{\neg p, q\}, \{\neg p, \neg q\}\}$

A Resolution refutation of Γ is:

$$\dfrac{\dfrac{\{p, q\} \quad \{\neg p, q\}}{\{q\}} \qquad \dfrac{\{\neg p, \neg q\} \quad \{p, \neg q\}}{\{\neg q\}}}{\Box}$$

Resolution in the Cook-Reckhow Framework

Resolution is a proof system for tautologies in DNF

$$Res(C_1, \ldots, C_k, \varphi) = \begin{cases} \varphi & \text{if } C_1, \ldots, C_k = \Box \text{ is a Resolution} \\ & \text{refutation of the clause set for } \neg\varphi \\ p \vee \neg p & \text{otherwise.} \end{cases}$$

Resolution in the Cook-Reckhow Framework

Resolution is a proof system for tautologies in DNF

$$Res(C_1, \ldots, C_k, \varphi) = \begin{cases} \varphi & \text{if } C_1, \ldots, C_k = \Box \text{ is a Resolution} \\ & \text{refutation of the clause set for } \neg\varphi \\ p \vee \neg p & \text{otherwise.} \end{cases}$$

Resolution can be extended to a proof system for all tautologies by transforming formulas into DNF.

A Strong System: Frege

Axioms

$$p_1 \to (p_2 \to p_1)$$
$$(p_1 \to p_2) \to (p_1 \to (p_2 \to p_3)) \to (p_1 \to p_3)$$
$$p_1 \to p_1 \lor p_2$$
$$p_2 \to p_1 \lor p_2$$
$$(p_1 \to p_3) \to (p_2 \to p_3) \to (p_1 \lor p_2 \to p_3)$$
$$(p_1 \to p_2) \to (p_1 \to \neg p_2) \to \neg p_1$$
$$\neg\neg p_1 \to p_1$$
$$p_1 \land p_2 \to p_1$$
$$p_1 \land p_2 \to p_2$$
$$p_1 \to p_2 \to p_1 \land p_2$$

Modus Ponens

$$\frac{p_1 \qquad p_1 \to p_2}{p_2}$$

Frege Proofs

A Frege proof of a formula φ is a sequence

$$(\varphi_1, \ldots, \varphi_n = \varphi)$$

of propositional formulas such that for $i = 1, \ldots, n$:

- φ_i is a substitution instance of an axiom, or
- φ_i was derived by modus ponens from φ_j, φ_k with $j, k < i$.

Restrictions and Extensions of Frege Systems

Bounded-depth Frege

Allow only formulas of logical depth d in the proof for a given constant d.

Extended Frege *EF*

Abbreviations for complex formulas: $p \equiv \varphi$,
where p is a new propositional variable.

Frege systems with substitution *SF*

Substitution rule: $\frac{\varphi}{\sigma(\varphi)}$
for arbitrary substitutions σ

Extensions of *EF*

Let Φ be a polynomial-time computable set of tautologies.
$EF + \Phi$: Φ as axiom schemes

Reductions between Proof Systems

Definition (Cook, Reckhow 79, Krajíček, Pudlák 89)

Let f and g be proof systems for L.

- f simulates g, if for any g-proof w there is an f-proof w' of length $|w'| = |w|^{O(1)}$ s.t. $f(w') = g(w)$.
- If w' is computable from w in polynomial time, then f p-simulates g.
- f and g are (p-)equivalent if they (p-)simulate each other.

Reductions between Proof Systems

Definition (Cook, Reckhow 79, Krajíček, Pudlák 89)

Let f and g be proof systems for L.

- f simulates g, if for any g-proof w there is an f-proof w' of length $|w'| = |w|^{O(1)}$ s.t. $f(w') = g(w)$.
- If w' is computable from w in polynomial time, then f p-simulates g.
- f and g are (p-)equivalent if they (p-)simulate each other.

Definition (Krajíček, Pudlák 89)

A proof system f for L is (p)-optimal if f (p-)simulates every proof system for L.

Simulations Between Proof Systems

Theorem (Cook, Reckhow 79)

All Frege systems are polynomially equivalent.

Theorem (Krajíček, Pudlák 89)

Every proof system is simulated by a proof system of the form $EF + \Phi$.

Problem (Krajíček, Pudlák 89)

Do optimal proof systems exist?

The Propositional Sequent Calculus

- Historically one of the first and best analyzed proof systems [Gentzen 35]
- Widely used for propositional and first-order logic
- We describe the propositional sequent calculus LK.
- The basic objects of the sequent calculus are sequents

$$\varphi_1, \ldots, \varphi_m \vdash \psi_1, \ldots, \psi_k \ .$$

- Formally, these are ordered pairs of two sequences of propositional formulas separated by the symbol $\vdash$.
- The sequence $\varphi_1, \ldots, \varphi_m$ is called the antecedent and $\psi_1, \ldots, \psi_k$ is called the succedent.

Sequents

- An assignment α satisfies a sequent $\Gamma \vdash \Delta$ if

$$\alpha \models \bigvee_{\varphi \in \Gamma} \neg\varphi \vee \bigvee_{\psi \in \Delta} \psi \ .$$

- $\vdash \Delta$ abbreviates $\emptyset \vdash \Delta$.
- $\Gamma \vdash$ abbreviates $\Gamma \vdash \emptyset$.
- Sequents of the form

$$A \vdash A, \quad 0 \vdash, \quad \vdash 1$$

are called initial sequents.

Rules of *LK*

$$\frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A} \text{ (weakening)}$$

$$\frac{\Gamma_1, A, B, \Gamma_2 \vdash \Delta}{\Gamma_1, B, A, \Gamma_2 \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta_1, A, B, \Delta_2}{\Gamma \vdash \Delta_1, B, A, \Delta_2} \text{ (exchange)}$$

$$\frac{\Gamma_1, A, A, \Gamma_2 \vdash \Delta}{\Gamma_1, A, \Gamma_2 \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta_1, A, A, \Delta_2}{\Gamma \vdash \Delta_1, A, \Delta_2} \text{ (contraction)}$$

$$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta} \qquad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A} \text{ ($\neg$ introduction)}$$

Rules of *LK* (cont'd.)

$\wedge$ introduction rules:

$$\frac{A, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta} \qquad \frac{A, \Gamma \vdash \Delta}{B \wedge A, \Gamma \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta, A \qquad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}$$

$\vee$ introduction rules:

$$\frac{A, \Gamma \vdash \Delta \qquad B, \Gamma \vdash \Delta}{A \vee B, \Gamma \vdash \Delta} \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, A \vee B} \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash \Delta, B \vee A}$$

$$\frac{\Gamma \vdash \Delta, A \qquad A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta} \text{(cut rule)}$$

Derivations

Definition

As in Frege systems, an *LK*-proof of a propositional formula φ is a derivation of the sequent

$$\vdash \varphi$$

from initial sequents by the above rules.

Derivations

Definition

As in Frege systems, an *LK*-proof of a propositional formula φ is a derivation of the sequent

$$\vdash \varphi$$

from initial sequents by the above rules.

Proposition (Cook, Reckhow 79)

Frege systems and the propositional sequent calculus LK are polynomially equivalent.

Polynomially Bounded Proof Systems

Polynomial Bounds on Proofs

A proof system f for L is polynomially bounded if there exists a polynomial p such that every $x \in L$ has an f-proof of size $\leq p(|x|)$.

Polynomially Bounded Proof Systems

Polynomial Bounds on Proofs

A proof system f for L is polynomially bounded if there exists a polynomial p such that every $x \in L$ has an f-proof of size $\leq p(|x|)$.

Examples

- The standard proof system for SAT is polynomially bounded:

$$sat(\alpha, \varphi) = \begin{cases} \varphi & \text{if } \alpha \text{ is a satisfying assignment for } \varphi \\ p & \text{otherwise.} \end{cases}$$

- The truth-table system is not a polynomially bounded proof system for TAUT.

The Cook-Reckhow Theorem

Question

Is there a polynomially bounded proof system for TAUT?

The Cook-Reckhow Theorem

Question

Is there a polynomially bounded proof system for TAUT?

Theorem (Cook, Reckhow 79)

A language L has a polynomially bounded proof system if and only if $L \in \mathrm{NP}$.

The Cook-Reckhow Theorem

Theorem (Cook, Reckhow 79)

A language L has a polynomially bounded proof system if and only if $L \in \mathrm{NP}$.

Proof. $\Rightarrow$

Let P be a polynomially bounded proof system with bounding polynomial p. Consider the following algorithm:

`Input: a string` x
`guess` $\pi \in \Sigma^{\leq p(|x|)}$
`IF` $P(\pi) = x$ `THEN accept ELSE reject`

The Cook-Reckhow Theorem

Theorem (Cook, Reckhow 79)

A language L has a polynomially bounded proof system if and only if $L \in \mathrm{NP}$.

Proof. $\Leftarrow$

Let $L \in \mathrm{NP}$ and let M be a nondeterministic polynomial time Turing machine M that accepts L. Let the polynomial p bound the running time of M. Then

$$P(\pi) = \begin{cases} x & \text{if } \pi \text{ codes an accepting computation of } M(x) \\ x_0 & \text{otherwise} \end{cases}$$

with fixed $x_0 \in L$ is a proof system for L which is polynomially bounded by p. □

The Cook-Reckhow Theorem

Question

Is there a polynomially bounded proof system for TAUT?

Theorem (Cook, Reckhow 79)

A language L has a polynomially bounded proof system if and only if $L \in \mathsf{NP}$.

The Cook-Reckhow Theorem

Question

Is there a polynomially bounded proof system for TAUT?

Theorem (Cook, Reckhow 79)

A language L has a polynomially bounded proof system if and only if $L \in \mathrm{NP}$.

For propositional proof systems

TAUT has a polynomially bounded proof system if and only if $\mathrm{NP} = \mathrm{coNP}$.

The Cook-Reckhow Programme

Separate NP from coNP (and hence P and NP) by showing super-polynomial lower bounds to the size of proofs in all propositional proof systems.

The Cook-Reckhow Programme

Separate NP from coNP (and hence P and NP) by showing super-polynomial lower bounds to the size of proofs in all propositional proof systems.

Showing lower bounds for a system P means

finding an infinite family θ_n of propositional tautologies s.t.

- $|\theta_n| = n^{O(1)}$;
- θ_n requires super-polynomial size proofs in P.

The Cook-Reckhow Programme

Separate NP from coNP (and hence P and NP) by showing super-polynomial lower bounds to the size of proofs in all propositional proof systems.

Showing lower bounds for a system P means

finding an infinite family θ_n of propositional tautologies s.t.

- $|\theta_n| = n^{O(1)}$;
- θ_n requires super-polynomial size proofs in P.
  - Better: ... exponential size proofs.

The Cook-Reckhow Programme

Separate NP from coNP (and hence P and NP) by showing super-polynomial lower bounds to the size of proofs in all propositional proof systems.

Showing lower bounds for a system P means

finding an infinite family θ_n of propositional tautologies s.t.

- $|\theta_n| = n^{O(1)}$;
- θ_n requires super-polynomial size proofs in P.
  - Better: ... exponential size proofs.

Even better

- Find a sequence of polynomially constructible formulas which require long proofs.
- This is usually the case: take θ_n as the propositional formalization of some combinatorial principle.
- Find a large set of formulas (e.g. random 3-CNF) which require long proofs.

The Cook-Reckhow Programme

Separate NP from coNP (and hence P and NP) by showing super-polynomial lower bounds to the size of proofs in all propositional proof systems.

The Cook-Reckhow Programme

Separate NP from coNP (and hence P and NP) by showing super-polynomial lower bounds to the size of proofs in all propositional proof systems.

Progress in this programme

- Haken (1985): exponential lower bound to the proof size in Resolution for the pigeonhole principle
- Ajtai (1988): Super-polynomial lower bound for bounded-depth Frege systems (Improved by Beame, Impagliazzo, Krajíček, Pitassi, Pudlák, Woods)
- Lower bounds for algebraic and geometric proof systems:
  - Cutting Planes
  - Polynomial Calculus
  - Nullstellensatz

Techniques and Barriers

Techniques for lower bounds

- feasible interpolation [Krajíček 97]
- size-width trade-offs [Ben-Sasson, Wigderson 01]
- game-theoretic techniques [Pudlák, Buss, Impagliazzo,...]
- proof complexity generators [Krajíček, Alekhnovich et al.]

The current barrier

Show lower bounds for Frege systems

Cutting Planes

- Cutting Planes uses the idea of linear programming.
- As in Resolution, CP is a refutation system that works with clauses.
- Clauses are translated into linear inequalities.

The Translation

- Clauses are translated into linear inequalities

$$a_1 p_1 + \cdots + a_n p_n \geq b \tag{1}$$

  with integer coefficients $a_1, \ldots, a_n$ and b.
- Propositional variables p are identically represented by integer variables p.
- $\neg p$ is translated to $1 - p$.
- A clause

$$C = \{l_1, \ldots, l_n\}$$

  with literals $l_i = p_i$ or $l_i = \neg p_i$ is translated into

$$f_1 + \cdots + f_n \geq 1$$

  with

$$f_i = \begin{cases} p_i & \text{if } l_i = p_i \\ 1 - p_i & \text{if } l_i = \neg p_i \end{cases}$$

  for $i = 1, \ldots, n$.
- To get an inequality of the form (1), constants are moved to the right hand side.

Axioms of CP

1. Let $\Gamma = \{C_1, \ldots, C_k\}$ be a set of clauses in variables $p_1, \ldots, p_n$.
2. As axioms in CP we use the translations of clauses $C_1, \ldots, C_k$ together with

$$p_i \geq 0, \quad -p_i \geq -1 \qquad i = 1, \ldots, n \ .$$

Rules of CP

1. **Addition:**

$$\frac{a_1 p_1 + \cdots + a_n p_n \geq b \qquad a'_1 p_1 + \cdots + a'_n p_n \geq b'}{(a_1 + a'_1) p_1 + \cdots + (a_n + a'_n) p_n \geq b + b'}$$

2. **Multiplication:**

$$\frac{a_1 p_1 + \cdots + a_n p_n \geq b}{c a_1 p_1 + \cdots + c a_n p_n \geq c b}$$

   with an arbitrary integer $c > 0$.

3. **Division:**

$$\frac{c a_1 p_1 + \cdots + c a_n p_n \geq b}{a_1 p_1 + \cdots + a_n p_n \geq \left\lceil \frac{b}{c} \right\rceil}$$

   with an arbitrary integer $c > 0$.

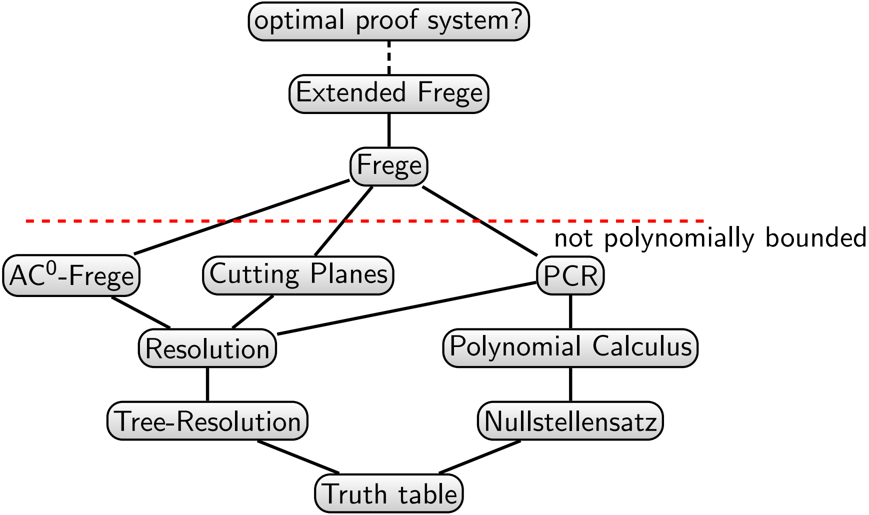
CP Refutations

- A CP refutation of a set of clauses Γ is a CP derivation of

$$0 \geq 1$$

  from the axioms corresponding to Γ.
- Easy to see: CP p-simulates Resolution.
- The converse is false.
- Frege systems p-simulate CP [Goerdt 91].

Simulations between important propositional proof systems



Summary

Proof Complexity

- is at the intersection of logic and complexity.
- uses concepts and intuition from algebra, geometry, ...

Main Objective

study lengths of proofs

Connections to other areas

- Separation of complexity classes
- Analysis of SAT algorithms
- Proof search – Automatizability
- First-Order Logic – Bounded Arithmetic
- Proving lower bounds is hard!

Tree-like Resolution

Tree-like Resolution

Refutational system for unsatisfiable CNF

- Resolution rule $\dfrac{C \cup \{x\} \quad D \cup \{\neg x\}}{C \cup D}$
- tree-like refutations: each derived clause is used at most once

$$\dfrac{\dfrac{\{x_1, x_2\} \quad \{\neg x_1, x_2\}}{\{x_2\}} \qquad \dfrac{\{\neg x_1, \neg x_2\} \quad \{x_1, \neg x_2\}}{\{\neg x_2\}}}{\square}$$

Proof size

- Number of clauses in the proof, i.e. nodes in the trees
- DPLL algorithms on unsatisfiable CNF produce tree-like Resolution refutations.
- Tree-like Resolution is not polynomially bounded.

Tree-like Resolution

- A Resolution refutation of F can be depicted as a directed graph were vertices are labeled with the clauses of the refutation and a Resolution step

$$\frac{C \quad D}{E}$$

yields edges (C, E) and (D, E).

- As this graph is acyclic, we also refer to the general Resolution system as dag-like Resolution.
- If the graph is a tree we call the refutation tree like. When we allow only tree-like refutations we get the tree-like Resolution system.
- In tree-like Resolution, each derived clause can be used at most once as a prerequisite of the Resolution rule.

An Equivalent Model: Boolean Decision Trees

Definition

- A boolean decision tree for F is a binary tree where inner nodes are labeled with variables from F and leafs are labeled with clauses from F.
- Each path in the tree corresponds to a partial assignment where a variable x gets value 0 or 1 according to whether the path branches left or right at the node labeled with x.
- In the tree, each path α must lead to a clause which is falsified by the assignment corresponding to α.

Boolean Decision Trees and the Search Problem

- A boolean decision tree solves the search problem for F:
  - given an assignment α,
  - find a clause from F falsified by α.
- Each tree-like Resolution refutation of F yields a boolean decision tree for F and vice versa, where the size of the Resolution proof equals the number of nodes in the decision tree.

DPLL Algorithms

The DPLL algorithm was developed by Davis, Logemann and Loveland using an earlier algorithm of Davis and Putnam.

Notation

- Let F be a formula and α a partial assignment.
- By $F|_{\alpha}$ we denote the simplified formula which results from substituting constants $0/1$ for variables in the domain of α.

Idea of the DPLL Algorithm

- Input: Formula F as a set of clauses
- Check if F is trivially satisfiable (F is the empty clause set) or trivially unsatisfiable (F contains the empty clause)
- Choose a variable x
- Consider $F|_{x=0}$ and $F|_{x=1}$
- If F is satisfiable, then at least one of the formulas $F|_{x=0}$ or $F|_{x=1}$ is satisfiable.
- Alternatively: F is unsatisfiable if both formulas $F|_{x=0}$ and $F|_{x=1}$ are unsatisfiable.

The DPLL Algorithm

DPLL(F, α)
IF $F|_\alpha = 0$ THEN Return unsatisfiable
IF $F|_\alpha = 1$ THEN Return α
choose a variable x in $F|_\alpha$ and $a \in \{0, 1\}$
$\beta :=$ DPLL($F, \alpha \cup [x := a]$)
IF $\beta \neq$ "unsatisfiable" THEN Return β
ELSE Return DPLL($F, \alpha \cup [x := (1 - a)]$)

Improvements of the DPLL algorithm

Unit propagation

If a clause is a unit clause, i.e. it contains only a single unassigned literal, this clause can only be satisfied by assigning the necessary value to make this literal true. Thus, no choice is necessary.
In practice, this often leads to deterministic cascades of units, thus avoiding a large part of the naive search space.

Example

- $p \vee q, \neg p \vee r, \neg r \vee \neg s, p$
- set $p = 1$ and obtain $r, \neg r \vee \neg s$
- set $r = 1$ and obtain $\neg s$
- set $s = 0$ and obtain the empty clause set which is trivially satisfiable

Improvements of the DPLL algorithm

Pure literal elimination

If a propositional variable occurs with only one polarity in the formula, it is called pure. Pure literals can always be assigned in a way that makes all clauses containing them true. Thus, these clauses do not constrain the search anymore and can be deleted.

Example

- $p \vee q, \neg p \vee r, \neg r \vee \neg s, p$
- q occurs only positively, set $q = 1$ and obtain $\neg p \vee r, \neg r \vee \neg s, p$
- s occurs only negatively, set $s = 0$ and obtain $\neg p \vee r, p$
- r occurs only positively, set $r = 1$ and obtain p
- p occurs only positively, set $p = 1$ and obtain the empty clause set which is trivially satisfiable

The DPLL Algorithm

```
DPLL(F)
IF F is empty THEN Return satisfiable
IF F contains the empty clause THEN Return unsatisfiable
for every unit clause l in F
        F := unit-propagate(l, F)
for every literal l that occurs pure in F
        F:= pure-literal-assign(l, F)
choose a variable x in F and a ∈ {0, 1}
IF DPLL(F|x:=a) = "satisfiable" THEN Return satisfiable
ELSE Return DPLL(F|x:=(1−a))
```

Modern SAT solvers

build on DPLL and enhance it by further features

- backjumping: non-chronological backtracking
- clause learning: adding new clauses from conflicts
- restarts
- different heuristics for choosing the branching literals and for learning clauses
- implementation tuning

Active community

yearly SAT competitions, affiliated with the SAT conference

SAT Solvers

- DPLL algorithms (combined with further techniques and heuristics) are the basis for most modern SAT solvers.

SAT Solvers

- DPLL algorithms (combined with further techniques and heuristics) are the basis for most modern SAT solvers.
- What is the running time of these algorithms?

SAT Solvers

- DPLL algorithms (combined with further techniques and heuristics) are the basis for most modern SAT solvers.
- What is the running time of these algorithms?
- The worst-case running time of DPLL algorithms is exponential in the length of the formula.

SAT Solvers

- DPLL algorithms (combined with further techniques and heuristics) are the basis for most modern SAT solvers.
- What is the running time of these algorithms?
- The worst-case running time of DPLL algorithms is exponential in the length of the formula.
- Why?

SAT Solvers

- DPLL algorithms (combined with further techniques and heuristics) are the basis for most modern SAT solvers.
- What is the running time of these algorithms?
- The worst-case running time of DPLL algorithms is exponential in the length of the formula.
- Why?
- On unsatisfiable formulas, the DPLL algorithm produces a Boolean decision tree (e.g. a tree-like Resolution refutation) of the formula.

SAT Solvers

- DPLL algorithms (combined with further techniques and heuristics) are the basis for most modern SAT solvers.
- What is the running time of these algorithms?
- The worst-case running time of DPLL algorithms is exponential in the length of the formula.
- Why?
- On unsatisfiable formulas, the DPLL algorithm produces a Boolean decision tree (e.g. a tree-like Resolution refutation) of the formula.
- We show a lower bound for tree-like Resolution.

A Game for Tree-like Resolution

Prover-Delayer games [Pudlák & Impagliazzo 00]

- Let F be a set of clauses in n variables $x_1, \ldots, x_n$.
- Prover and Delayer build a (partial) assignment to $x_1, \ldots, x_n$.
- The game is over as soon as the partial assignment falsifies a clause from F.
- In each round, Prover suggests a variable x_i, and Delayer either chooses a value $0/1$ for x_i or leaves the choice to Prover.
- If Prover sets the value, then Delayer gets 1 point.
- Prover can always win the game on unsatisfiable formulas, but how many points can Delayer earn?

Scores and Lengths of Proofs

Idea

Good strategies for Delayer for a unsatisfiable CNF F yield lower bounds for tree-like Resolution refutations of F.

Theorem (Pudlák & Impagliazzo 00)

Let F be an unsatisfiable formula in CNF.
If F has a tree-like Resolution refutation of size at most S, then Delayer gets at most $\log S$ points in each Prover-Delayer game played on F.

Corollary

If Delayer scores p points during a game on F, then tree-like Resolution refutations of F are of size $2^{\Omega(p)}$.

The Proof

- Let F be an unsatisfiable CNF in variables $x_1, \ldots, x_n$ and let Π be a tree-like Resolution refutation of F.
- Prover and Delayer play a game on F where they successively construct an assignment α.
- Let α_i be the partial assignment constructed after i rounds of the game.
- By p_i we denote the number of Delayer's points after i rounds.
- Let Π_{α_i} be the sub-tree of Π which has as its root the node reached in Π along the path specified by α_i.

Invariant during the game

$$|\Pi_{\alpha_i}| \leq \frac{|\Pi|}{2^{p_i}} \text{ for any round } i.$$

Invariant during the game

Invariant

$$|\Pi_{\alpha_i}| \leq \frac{|\Pi|}{2^{p_i}} \text{ for any round } i.$$

The invariant yields the theorem

- At the end of the game a contradiction has been reached and the size of Π_α is 1.
- By the invariant

  $$1 \leq \frac{|\Pi|}{2^{p_\alpha}} \quad ,$$

  yielding $p_\alpha \leq \log |\Pi|$.

Invariant during the game

Invariant

$$|\Pi_{\alpha_i}| \leq \frac{|\Pi|}{2^{p_i}} \text{ for any round } i.$$

Beginning

In the beginning of the game, Π_{α_0} is the full tree and the Delayer has 0 points. Therefore the invariant holds.

Inductive step

If the Delayer chooses the value, then $p_{i+1} = p_i$ and hence

$$|\Pi_{\alpha_{i+1}}| \leq |\Pi_{\alpha_i}| \leq \frac{|\Pi|}{2^{p_i}} = \frac{|\Pi|}{2^{p_{i+1}}} \quad .$$

Invariant during the game

Inductive step

- If Delayer defers the choice to Prover, then Prover chooses the value x which leads to the smaller subtree, i. e. Prover sets $x = 0$ if
$$|\Pi_{\alpha_i \cup \{x=0\}}| \leq \frac{|\Pi_{\alpha_i}|}{2},$$
otherwise he sets $x = 1$.
- Thus, if Prover's choice is $x = j$ with $j \in \{0, 1\}$, then
$$|\Pi_{\alpha_{i+1}}| = |\Pi_{\alpha_i \cup \{x=j\}}| \leq \frac{|\Pi_{\alpha_i}|}{2} \leq \frac{|\Pi|}{2 \cdot 2^{p_i}} = \frac{|\Pi|}{2^{p_i+1}} = \frac{|\Pi|}{2^{p_{i+1}}} \quad .$$

The Pigeonhole Principle

- PHP^m_n with $m > n$ uses variables $x_{i,j}$ with $i \in [m]$ and $j \in [n]$,
- $x_{i,j}$ indicates that pigeon i goes into hole j.
- PHP^m_n consists of the clauses

$$\bigvee_{j \in [n]} x_{i,j} \quad \text{for all pigeons } i \in [m]$$

and

$$\neg x_{i_1,j} \vee \neg x_{i_2,j}$$

for all choices of distinct pigeons $i_1, i_2 \in [m]$ and holes $j \in [n]$.

Tree-like Resolution Lower Bounds for PHP

- We prove that PHP_n^{n+1} is hard for tree-like Resolution.
- Showing the lower bound by the Prover-Delayer game requires a suitable Delayer strategy.

Tree-like Resolution Lower Bounds for PHP

- We prove that PHP_n^{n+1} is hard for tree-like Resolution.
- Showing the lower bound by the Prover-Delayer game requires a suitable Delayer strategy.

Theorem

Any tree-like Resolution refutation of PHP_n^m for $m > n$ has size $2^{\Omega(n)}$.

Delayer's Strategy

Let us say that a hole j is occupied if there exists $i \in [m]$ such that $x_{i,j}$ was assigned to 1 in the game.

Delayer's Strategy

Let us say that a hole j is occupied if there exists $i \in [m]$ such that $x_{i,j}$ was assigned to 1 in the game.

Delayer's strategy

If Prover asks variable $x_{i,j}$, then Delayer answers 0 if hole j is already occupied, otherwise she leaves the decision to Prover.

Delayer's Strategy

Let us say that a hole j is occupied if there exists $i \in [m]$ such that $x_{i,j}$ was assigned to 1 in the game.

Delayer's strategy

If Prover asks variable $x_{i,j}$, then Delayer answers 0 if hole j is already occupied, otherwise she leaves the decision to Prover.

Observation

- The game never ends by falsifying a clause $\neg x_{i_1,j} \vee \neg x_{i_2,j}$.
- Therefore the game stops at one of the big clauses $\bigvee_{j\in[n]} x_{i,j}$, i. e., for some $i \in [m]$ all variables $x_{i,j}$ with $j \in [n]$ have been assigned to 0 by either Prover or Delayer.

Number of Points for Delayer

- For some $i \in [m]$, all variables $x_{i,j}$ with $j \in [n]$ have been assigned to 0 by either Prover or Delayer.
- We claim that Delayer earns at least n points in the game.
- If $x_{i,j}$ was set to 0 by Prover, then Delayer earns 1 point.
- If $x_{i,j}$ was set to 0 by Delayer, then according to Delayer's strategy, there was some other pigeon $i' \neq i$ sitting in hole j, i. e., $x_{i',j}$ was assigned to 1. This decision was made by Prover, as Delayer never sets a variable to 1.
- In total Delayer earns a point for each variable $x_{i,j}$ with $j \in [n]$.
- The lower bound follows by the previous theorem.

The Complexity of the Pigeonhole Principle

Theorem

Any tree-like Resolution refutation of PHP^m_n for $m > n$ has size $2^{\Omega(n)}$.

The Complexity of the Pigeonhole Principle

Theorem
Any tree-like Resolution refutation of PHP_n^m for $m > n$ has size $2^{\Omega(n)}$.

This is not the optimal lower bound.

- Showing lower bounds by the PD-game only works if (the graph of) every tree-like Resolution refutation contains a balanced sub-tree as a minor.
- The height of that sub-tree gives the size lower bound.

Theorem (Iwama & Miyazaki 99)
Any tree-like Resolution refutation of PHP_n^m has size $2^{\Omega(n \log n)}$.

Asymmetric Prover-Delayer Games

For a partial assignment α and a variable x, let $c_0(x, \alpha)$ and $c_1(x, \alpha)$ be functions such that

$$\frac{1}{c_0(x, \alpha)} + \frac{1}{c_1(x, \alpha)} = 1$$

The asymmetric (c_0, c_1)-game

Assume α is the partial assignment built so far in the game and Prover queries x. Then Delayer gets

0 points	if Delayer chooses the value
$\log c_0(x, \alpha)$ points	if Prover sets x to 0
$\log c_1(x, \alpha)$ points	if Prover sets x to 1.

A Generalization

- The same lower bound holds for the functional pigeonhole principle.
- In addition to the clauses from PHP_n^m we also include

$$\neg x_{i,j_1} \vee \neg x_{i,j_2}$$

for all pigeons $i \in [m]$ and distinct holes $j_1, j_2 \in [n]$.

Tree-like vs. DAG-like Resolution

Tree-like Resolution

- A Resolution refutation of F can be depicted as a directed graph were vertices are labeled with the clauses of the refutation and a Resolution step

$$\frac{C \quad D}{E}$$

yields edges (C, E) and (D, E).

- As this graph is acyclic, we also refer to the general Resolution system as dag-like Resolution.
- If the graph is a tree we call the refutation tree like. When we allow only tree-like refutations we get the tree-like Resolution system.
- In tree-like Resolution, each derived clause can be used at most once as a prerequisite of the Resolution rule.

Tree-like vs. DAG-like Proof Systems

A general question

Are dag-like proof systems more powerful than tree-like systems?
Is the dag-like proof system simulated by the corresponding tree-like proof system?

The answer depends on the proof system.

- For Resolution: Dag-like systems are more powerful (exponential separation).
- For Frege systems: dag-like and tree-like versions are equivalent.

Tree-like vs. DAG-like Proof Systems

Theorem (Krajíček 95)

Tree-like Frege systems p-simulate (dag-like) Frege.

Proof.

- Let $A_1, \ldots, A_m$ be a proof in (dag-like) Frege.
- Let
$$B_i = A_1 \wedge \cdots \wedge A_i$$
for $i = 1, \ldots, m$.
- We get linear-size tree-like Frege proofs of
$$B_i \to B_{i+1}$$
for $i = 1, \ldots, m-1$.
- $m - 1$ applications of Modus Ponens give A_m.
- The proof is tree-like.

□

Tree-like vs. DAG-like Proof Systems

The result

There is a family of unsatisfiable CNF that have polynomial-size dag-like Resolution refutations, but require exponential-size tree-like Resolution refutations.

History

- Goerdt 92: first separation: example with poly-size dag-like refutations, but only quasi-polynomial tree-like refutations (modification of PHP).
- Bonet, Galesi, Esteban, Johannsen 98: first exponential separation
- Ben-Sasson, Impagliazzo, Wigderson 04: simplified and improved separation by using games

Separation of Tree-like and DAG-like Resolution

- separating formulas: pebbling formulas
- derived from a pebbling game
- proof method: Prover-Delayer games
- we follow Ben-Sasson, Impagliazzo, Wigderson 04

Pebbling Games

- pebbling games are played on DAGs
- source nodes: in-degree 0
- target nodes: out-degree 0
- game: place pebbles on nodes according to rules
- aim: place a pebble at some target node

Rules

1. Source nodes can be pebbled freely.
2. All other nodes can be pebbled if all their parents are pebbled.
3. Pebbles can be removed at any time.

Pebbling Number

Complexity measure

Maximal number of pebbles placed simultaneously on the graph.

Pebbling number of a strategy to pebble a graph

- Let S be a strategy to pebble the dag G.
- $P(G, S) =$ max # of pebbles placed simultaneously on G while following strategy S

Pebbling number of G

$P(G) = \min\{\, P(G, S) \mid S \text{ is a strategy to pebble } G \,\}$

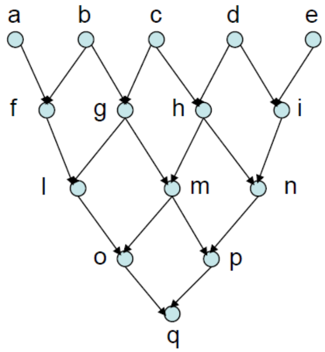
Graphs with High Pebbling Numbers

Theorem (Celoni, Paul, Tarjan 77)

There exist graphs G with n vertices such that

$$P(G) = \Omega\left(\frac{n}{\log n}\right).$$

- The proof is constructive.
- Example: pyramidal graphs



Pebbling Formulas

DAG $G = (V, E)$

Propositional variables

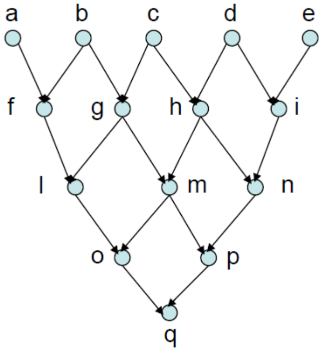
- x_v for all $v \in V$
- Meaning: $x_v = 1$ if v has been pebbled

Clauses in $Peb^0(G)$

$$
\begin{array}{cl}
x_v & \text{for any source node } v \\
\left(\bigwedge_{u \in N^-(v)} x_u\right) \to x_v & \text{for all nodes } v \\
 & \text{where } N^-(v) \text{ are the parents of } v \\
\neg x_v & \text{for any target node } v
\end{array}
$$

Complexity of Peb^0

- $Peb^0(G)$ is unsatisfiable.
- But: They have polynomial-size tree-like Resolution refutations.
- Idea: Start from the bottom and explore the graph in a breadth-first fashion.



Adding Complexity to $Peb^0(G)$

Idea

- Use pebbles of two different colors: black and white
- Consider a node pebbled if it has a black or white pebble on it

The new principle

- Source nodes can always be pebbled black or white.
- For an internal node v, if all its parents are pebbled black or white, then v can be pebbled either black or white.
- No target node is pebbled black or white.

The New Pebbling Formulas

DAG $G = (V, E)$ with in-degree ≤ 2

Propositional variables

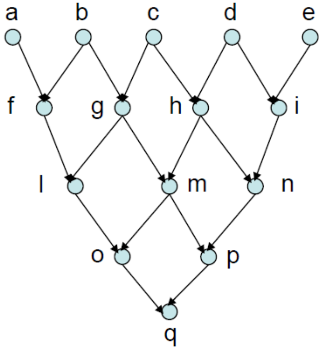
- $x_{v,c}$ for all $v \in V$ and $c \in \{B, W\}$
- Meaning: $x_{v,B} = 1$ if v has been pebbled black
  $x_{v,W} = 1$ if v has been pebbled white

Clauses in $Peb(G)$

$x_{v,B} \vee x_{v,W}$	for any source node v
$x_{u,a} \wedge x_{w,b} \rightarrow x_{v,B} \vee x_{v,W}$	for all nodes $v \in V$, $a, b \in \{B, W\}$ where u and w are the parents of v
$\neg x_{v,B}, \neg x_{v,W}$	for any target node v

Complexity of $Peb(G)$

- $Peb(G)$ is unsatisfiable.
- Proof strategy as for Peb^0 does not work anymore.
- But: They have polynomial-size dag-like Resolution refutations.
- Our aim: Show a lower bound for tree-like Resolution



The Pebbling Formulas in Tree-like Resolution

Main Theorem

Let G be a DAG with in-degree ≤ 2. Then Delayer has a strategy to win $P(G) - 3$ points in any PD-game played on $Peb(G)$.

Theorem (Celoni, Paul, Tarjan 77)

There exist graphs G with n vertices such that

$$P(G) = \Omega\left(\frac{n}{\log n}\right).$$

Corollary

There exist graphs G with n vertices for which $Peb(G)$ requires tree-like Resolution refutations of size $2^{\Omega\left(\frac{n}{\log n}\right)}$.

Proof of Main Theorem

Let G be the DAG with source nodes S and target nodes T.

Strategy of Delayer

- Keep two sets S' and T'.
- In the beginning, set $S' = S$ and $T' = T$.
- Denote by $P(G, S', T')$ the pebbling number of G with source nodes S' and target nodes T'.
- If Prover asks variable $x_{v,\cdot}$ belonging to node v, then Delayer reacts as follows
  1. If $v \in S'$, then answer 1.
  2. If $v \in T'$, then answer 0.
  3. If $v \notin S' \cup T'$ and $P(G, S', T' \cup \{v\}) = P(G, S', T')$, then answer 0 and set $T' = T' \cup \{v\}$.
  4. If $v \notin S' \cup T'$ and $P(G, S', T' \cup \{v\}) < P(G, S', T')$, then leave decision to Prover and set $S' = S' \cup \{v\}$.

Intuition for the Strategy

If Prover asks variable $x_{v,\cdot}$ belonging to node v, then Delayer reacts as follows

1. If $v \in S'$, then answer 1.
   Source nodes are always pebbled.
2. If $v \in T'$, then answer 0.
   Target nodes are never pebbled.
3. If $v \notin S' \cup T'$ and $P(G, S', T' \cup \{v\}) = P(G, S', T')$, then answer 0 and set $T' = T' \cup \{v\}$.
   If pebbling number remains the same, v is added to T' and is not pebbled.
4. If $v \notin S' \cup T'$ and $P(G, S', T' \cup \{v\}) < P(G, S', T')$, then leave decision to Prover and set $S' = S' \cup \{v\}$.
   If pebbling number decreases, v is added to S' and Prover has to pay. But he can only choose the color of v.

How many points does Delayer earn?

Intuition

- Whenever the pebbling number decreases, Delayer gets a point.
- Hence Delayer scores according to the pebbling number of G.

Lemma

When the game terminates, $P(G, S', T') \leq 3$.

Lemma

For any node v *and sets* S, T

$$P(G, S, T) \leq \max\{P(G, S, T \cup \{v\}), P(G, S \cup \{v\}, T) + 1\}.$$

How many points does Delayer earn?

Lemma

When the game terminates, $P(G, S', T') \leq 3$.

Lemma

For any node v *and sets* S, T

$$P(G, S, T) \leq \max\{P(G, S, T \cup \{v\}), P(G, S \cup \{v\}, T) + 1\}.$$

Lemma

After any round, if Delayer has earned p *points, then* $P(G, S', T') \geq P(G, S, T) - p$.

Corollary

Delayer scores at least $P(G, S, T) - 3$ *points.*

The Result

Theorem

There exists an infinite family of explicitly constructible formulas θ_n s.t.

1. $|\theta_n| = O(n)$;
2. θ_n *require tree-like Resolution refutations of size* $2^{\Omega\left(\frac{n}{\log n}\right)}$;
3. θ_n *have Resolution refutations of size* $O(n)$.

Linear Resolution Refutations of Pebbling Formulas

- Fix a topological sort of G.
- In order of this sort we inductively derive $x_{v,B} \vee x_{v,W}$.
- If v has no predecessors, then $v \in S$ and $x_{v,B} \vee x_{v,W}$ is an axiom.
- If v has 2 predecessors u, w, then we have inductively derived $x_{u,B} \vee x_{u,W}$ and $x_{w,B} \vee x_{w,W}$.
- Together with the four pebbling axioms for v, these formulas imply $x_{v,B} \vee x_{v,W}$.
- By completeness of Resolution, we have a Resolution derivation of $x_{v,B} \vee x_{v,W}$ from these clauses.
- The derivation is of constant size as only it only contains 6 variables.
- Thus we derive $x_{t,B} \vee x_{t,W}$ for some target $t \in T$ in linear size.
- Using the target axioms, we get a contradiction.

DAG-like Resolution

Boolean Circuits

Definition

A Boolean circuit is a directed acyclic graph where

- nodes with in-degree 0 are labeled with variables $x_1, x_2, \ldots$ or constants $0/1$;
- nodes with in-degree ≥ 1 are gates labeled with $\neg$, $\wedge$, or $\vee$;
- nodes with out-degree 0 are called output gates.

Non-uniform Complexity Classes

Functions computed by Boolean circuits

- Let C_n be a Boolean circuit in n input variables $x_1, \ldots, x_n$ and one output gate.
- Then C_n computes a Boolean function $\{0,1\}^n \mapsto \{0,1\}$.
- The family $(C_n)_{n \geq 1}$ computes a function $\{0,1\}^* \mapsto \{0,1\}$.
- Non-uniformity: for each input length we use a different algorithm.

Definition

The class P/poly contains all languages L for which the characteristic function is computable by a family of Boolean circuits.

Lower Bounds

Lower bounds are hard

We do not know any specific function which cannot be computed by linear size Boolean circuits.

Lower Bounds

Lower bounds are hard

We do not know any specific function which cannot be computed by linear size Boolean circuits.

A restricted model

- A monotone Boolean circuit is a circuit without $\neg$ gates.

Lower Bounds

Lower bounds are hard

We do not know any specific function which cannot be computed by linear size Boolean circuits.

A restricted model

- A monotone Boolean circuit is a circuit without $\neg$ gates.
- In this model we know exponential lower bounds [Alon & Boppana 87].

Clique-Colour Formulas

Clique-Colour Formulas

- Idea: a graph with a $k+1$-clique is not k-colourable.
- Let $Clique_n^{k+1}(\bar{p}, \bar{r})$ be a propositional formula expressing that the graph of size n encoded in the variables $\bar{p}$ contains a clique of size $k+1$.
- Similarly, $Colour_n^{k}(\bar{p}, \bar{s})$ expresses that the graph specified by $\bar{p}$ is k-colourable.
- $Clique_n^{k+1}(\bar{p}, \bar{r}) \rightarrow \neg Colour_n^{k}(\bar{p}, \bar{s})$ are propositional tautologies.

A Lower Bound for Monotone Circuits

Definition

A Boolean circuit $C(\bar{p})$ interpolates the Clique-Colour formulas if

- the graph $\bar{p}$ contains a $k+1$-clique $\Rightarrow C(\bar{p}) = 1$;
- the graph $\bar{p}$ is k-colourable $\Rightarrow C(\bar{p}) = 0$.

A Lower Bound for Monotone Circuits

Definition

A Boolean circuit $C(\bar{p})$ interpolates the Clique-Colour formulas if

- the graph $\bar{p}$ contains a $k+1$-clique $\Rightarrow C(\bar{p}) = 1$;
- the graph $\bar{p}$ is k-colourable $\Rightarrow C(\bar{p}) = 0$.

Theorem (Alon, Boppana 87)

For $k = \sqrt{n}$, the Clique-Colour formulas require monotone interpolating circuits of size $2^{\Omega(n^{\frac{1}{4}})}$.

Craig's Interpolation Theorem

Theorem (Craig's Interpolation Theorem)

Let $\varphi(\bar{x}, \bar{y})$ *and* $\psi(\bar{x}, \bar{z})$ *be propositional formulas with all variables displayed. Let* $\bar{y}$ *and* $\bar{z}$ *be distinct tuples of variables such that* $\bar{x}$ *are the common variables of* φ *and* ψ*. If*

$$\varphi(\bar{x}, \bar{y}) \to \psi(\bar{x}, \bar{z})$$

is a tautology, then there exists a propositional formula $\theta(\bar{x})$ *using only the common variables of* φ *and* ψ *such that*

$$\varphi(\bar{x}, \bar{y}) \to \theta(\bar{x}) \quad \text{and} \quad \theta(\bar{x}) \to \psi(\bar{x}, \bar{z})$$

are tautologies.

A Key Technique – Feasible Interpolation

Definition (Krajíček 97)

A proof system P has feasible interpolation if there exists a polynomial time procedure that takes as input an implication $\varphi(\bar{x}, \bar{y}) \rightarrow \psi(\bar{x}, \bar{z})$ and a P-proof π of $\varphi(\bar{x}, \bar{y}) \rightarrow \psi(\bar{x}, \bar{z})$ and outputs a Boolean circuit $C(\bar{x})$ such that C computes an interpolant of φ and ψ.

Conditional Lower Bounds

Theorem

Let P be a proof system with feasible interpolation.
If $\mathsf{NP} \cap \mathsf{coNP} \not\subseteq \mathsf{P/poly}$, *then P is not polynomially bounded.*

Proof idea

- Suppose we know that a sequence of formulas $\varphi_0^n \vee \varphi_1^n$ cannot be interpolated by polynomial-size circuits as above.
- Then $\varphi_0^n \vee \varphi_1^n$ do not have polynomial-size proofs in any proof system which has feasible interpolation.
- Such formulas $\varphi_0^n \vee \varphi_1^n$ are easy to construct under suitable assumptions.
- For instance, the formulas could express that factoring integers is not possible in polynomial time (which implies $\mathsf{NP} \cap \mathsf{coNP} \not\subseteq \mathsf{P/poly}$).

Unconditional Lower Bounds

Theorem (Krajíček 97)

Resolution has the monotone feasible interpolation property, i.e. there exist monotone interpolating circuits.

Theorem (Alon, Boppana 87)

For $k = \sqrt{n}$, the Clique-Colour formulas require monotone interpolating circuits of size $2^{\Omega(n^{\frac{1}{4}})}$.

Theorem

For $k = \sqrt{n}$, the clause sets expressing the negation of the Clique-Colour formulas require Resolution refutations of size $2^{\Omega(n^{\frac{1}{4}})}$.

Lower Bounds for Cutting Planes

Theorem (Pudlák 97)

Cutting Planes has the monotone feasible interpolation property.

Corollary

For $k = \sqrt{n}$*, the clause sets expressing the negation of the Clique-Colour formulas require Cutting Planes refutations of size* $2^{\Omega(n^{\frac{1}{4}})}$.

Feasible Interpolation for Stronger Systems?

Theorem (Krajíček & Pudlák 98)

Extended Frege systems do not have feasible interpolation unless RSA is insecure.

Theorem (Bonet, Pitassi, Raz 00)

Frege systems do not have feasible interpolation unless Blum integers can be factored in polynomial time
(a Blum integer is the product of two primes which are both congruent 3 modulo 4).

Theorem (Bonet, Domingo, Gavaldà, Maciel, Pitassi 04)

Bounded-depth Frege systems do not have feasible interpolation under cryptographic assumptions.

Proof complexity of modal and intuitionistic logics

Proof Complexity of Non-classical Logics

In the last decade

Intense research on complexity of proofs in non-classical logics

Why non-classical logics?

- Non-classical logics such as modal logics, tree logics, or non-monotonic logics have numerous applications, e. g. verification, model checking, expert systems, or modeling common sense reasoning.
- Yields better understanding of propositional proofs – we see new phenomena which do not appear in classical logic.
- Separation of complexity classes.

Separation of Complexity Classes

- Non-classical logics are often more expressive than propositional logic.
- They are associated with large complexity classes.
- Satisfiability of the modal logic K is PSPACE-complete [Ladner 77].
- As in the Cook-Reckhow programme, proving lower bounds to the lengths of proofs in non-classical logics aims to separate NP from PSPACE.
- Intuitively, lower bounds to the lengths of proofs in non-classical logic should be easier to obtain ($\mathrm{NP} \neq \mathrm{coNP} \quad \Longrightarrow \quad \mathrm{NP} \neq \mathrm{PSPACE}$)
- In contrast to classical logic, we have exponential lower bounds for modal and intuitionistic Frege systems [Hrubeš 07, Jeřábek 09]

A Classical Frege System

Axioms

$$p_1 \to (p_2 \to p_1)$$
$$(p_1 \to p_2) \to (p_1 \to (p_2 \to p_3)) \to (p_1 \to p_3)$$
$$p_1 \to p_1 \lor p_2$$
$$p_2 \to p_1 \lor p_2$$
$$(p_1 \to p_3) \to (p_2 \to p_3) \to (p_1 \lor p_2 \to p_3)$$
$$(p_1 \to p_2) \to (p_1 \to \neg p_2) \to \neg p_1$$
$$\neg\neg p_1 \to p_1$$
$$p_1 \land p_2 \to p_1$$
$$p_1 \land p_2 \to p_2$$
$$p_1 \to p_2 \to p_1 \land p_2$$

Modus Ponens

$$\frac{p_1 \qquad p_1 \to p_2}{p_2}$$

Frege Systems for Modal Logics

Modal language

In addition to the propositional connectives the modal language contains the unary connective $\Box$.

New axioms and rules

- Axiom of distributivity $\Box(p \to q) \to (\Box p \to \Box q)$
- Rule of necessitation $\dfrac{p}{\Box p}$

Modal logics

The modal logic K is defined as the set of all modal formulas derivable in this Frege system.

Further Modal Logics

Other modal logics can be obtained by adding further axioms:

modal logic	axioms		
$K4$	K	$+$	$\Box p \to \Box\Box p$
KB	K	$+$	$p \to \Box\neg\Box\neg p$
GL	K	$+$	$\Box(\Box p \to p) \to \Box p$
$S4$	$K4$	$+$	$\Box p \to p$
$S4Grz$	$S4$	$+$	$\Box(\Box(p \to \Box p) \to p) \to \Box p$

Frege Systems for Intuitionistic Logic

While modal logics extend the classical propositional calculus, intuitionistic logics are restrictions thereof.

Axioms

$$p_1 \to (p_2 \to p_1)$$
$$(p_1 \to p_2) \to (p_1 \to (p_2 \to p_3)) \to (p_1 \to p_3)$$
$$p_1 \to p_1 \vee p_2$$
$$p_2 \to p_1 \vee p_2$$
$$(p_1 \to p_3) \to (p_2 \to p_3) \to (p_1 \vee p_2 \to p_3)$$
$$\bot \to p_1$$
$$p_1 \wedge p_2 \to p_1$$
$$p_1 \wedge p_2 \to p_2$$
$$p_1 \to p_2 \to p_1 \wedge p_2$$

Modus Ponens

$$\frac{p_1 \qquad p_1 \to p_2}{p_2}$$

Lower Bounds for Clique-Colour Tautologies

- In order to prove lower bounds for the Clique-Colour tautologies we need a monotone feasible interpolation theorem where the interpolating circuits are monotone.
- Such a result is known for Resolution and Cutting Planes, but does not hold for Frege systems under reasonable assumptions (factoring integers is not possible in polynomial time) [Krajíček, Pudlák 98, Beame, Pitassi, Raz 00]
- Therefore we cannot expect a full version of monotone feasible interpolation for modal extensions of classical Frege.

The Idea of the Lower Bound for K-Frege

Hrubeš modified the Clique-Colouring formulas in a clever way by introducing $\Box$ in appropriate places:

$$Clique_n^{k+1}(\Box\bar{p}, \bar{r}) \to \Box(\neg Colour_n^k(\bar{p}, \bar{s})) \tag{2}$$

with $k = \sqrt{n}$. Hrubeš showed

- the formulas (2) are modal tautologies;
- if the formulas (2) are provable in K with $m(n)$ distributivity axioms, then the original Clique-Colour formulas can be interpolated by monotone circuits of size $O(m(n)^2)$.

Theorem (Hrubeš 09)

The formulas (2) *are* K*-tautologies. Every* K*-Frege proof of the formulas* (2) *uses* $2^{n^{\Omega(1)}}$ *steps.*

A Version of Monotone Interpolation for K

Theorem

Let π be a proof of the formula

$$\varphi \to \Box\psi$$

in the Frege system for K which uses n modal rules.
Let $\Box A_1, \ldots, \Box A_k$ be the immediate modal subformulas of φ.
Then there exists a monotone circuit C of size $O(n^2)$ in k variables such that

- $\varphi(\Box A_1, \ldots, \Box A_k, \bar{s}) \to C(\Box A_1, \ldots, \Box A_k)$ *and*
- $C(\Box A_1, \ldots, \Box A_k) \to \Box\psi$

are K-tautologies.

The Lower Bound for K

Theorem (Hrubeš 09)

Every K-Frege proof of the formulas

$$Clique_n^{\sqrt{n}+1}(\Box\bar{p}, \bar{r}) \to \Box(\neg Colour_n^{\sqrt{n}}(\bar{p}, \bar{s}))$$

uses $2^{n^{\Omega(1)}}$ steps.

Lower Bounds for Intuitionistic Logic

Along the same lines, Hrubeš proved lower bounds for intuitionistic Frege systems. For this he modified the Clique-Colour formulas to the intuitionistic version

$$\bigwedge_{i=1}^{n}(p_i \vee q_i) \rightarrow \Big(\neg Colour_n^k(\bar{p},\bar{s}) \vee \neg Clique_n^{k+1}(\neg\bar{q},\bar{r})\Big) \qquad (3)$$

where again $k = \sqrt{n}$.

Theorem (Hrubeš 09)

The formulas (3) *are intuitionistic tautologies and require intuitionistic Frege proofs with* $2^{n^{\Omega(1)}}$ *steps.*

Lower Bounds for Intuitionistic Logic

Along the same lines, Hrubeš proved lower bounds for intuitionistic Frege systems. For this he modified the Clique-Colour formulas to the intuitionistic version

$$\bigwedge_{i=1}^{n}(p_i \vee q_i) \rightarrow \Big(\neg Colour_n^k(\bar{p},\bar{s}) \vee \neg Clique_n^{k+1}(\neg\bar{q},\bar{r})\Big) \qquad (3)$$

where again $k = \sqrt{n}$.

Theorem (Hrubeš 09)

The formulas (3) *are intuitionistic tautologies and require intuitionistic Frege proofs with* $2^{n^{\Omega(1)}}$ *steps.*

The lower bounds were extended by Jeřábek (2009) to all modal and superintuitionistic logics with infinite branching.

QBF proof complexity

Quantified Boolean Formulas (QBF)

- QBFs are propositional formulas with boolean quantifiers ranging over 0,1.

Quantified Boolean Formulas (QBF)

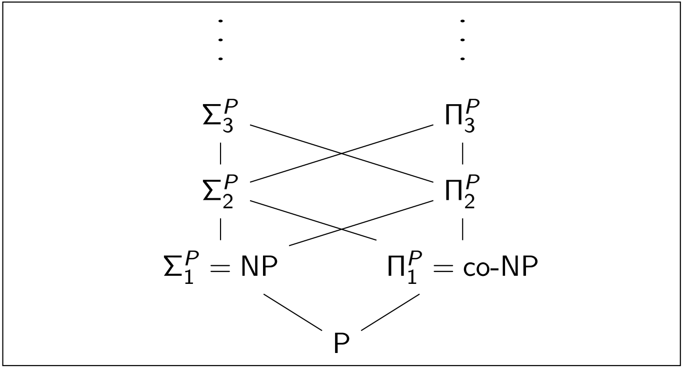
- QBFs are propositional formulas with boolean quantifiers ranging over 0,1.
- Deciding QBF is PSPACE complete.

Quantified Boolean Formulas (QBF)

- QBFs are propositional formulas with boolean quantifiers ranging over 0,1.
- Deciding QBF is PSPACE complete.

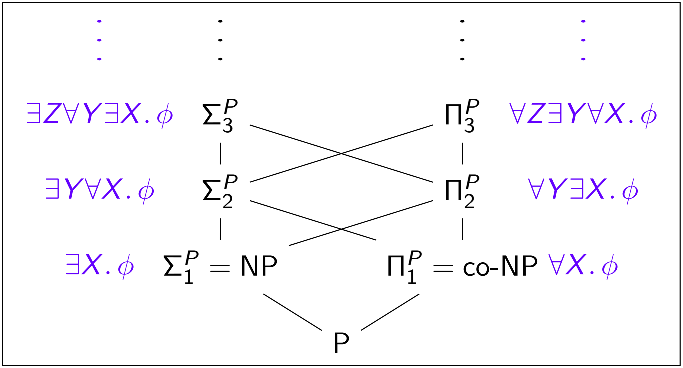
Quantified Boolean Formulas (QBF)

- QBFs are propositional formulas with boolean quantifiers ranging over 0,1.
- Deciding QBF is PSPACE complete.



Quantified Boolean Formulas (QBF)

- QBFs are propositional formulas with boolean quantifiers ranging over 0,1.
- Deciding QBF is PSPACE complete.



Quantified Boolean Formulas (QBF)

What's different in QBF from propositional proof complexity?

Quantified Boolean Formulas (QBF)

What's different in QBF from propositional proof complexity?

- Quantification!

Quantified Boolean Formulas (QBF)

What's different in QBF from propositional proof complexity?

- Quantification!
- Boolean quantifiers ranging over 0/1

Quantified Boolean Formulas (QBF)

What's different in QBF from propositional proof complexity?

- Quantification!
- Boolean quantifiers ranging over 0/1

Why QBF proof complexity?

- driven by QBF solving
- shows different effects from propositional proof complexity
- connects to circuit complexity, bounded arithmetic, ...

QBF proof complexity vs solving

Impact for proof complexity

different resolution systems defined that capture ideas in solving:

- CDCL
- expansion of universal variables
- dependency schemes

Impact for solving

- proves soundness of new algorithmic approaches
- upper/lower bounds suggest new directions in solving

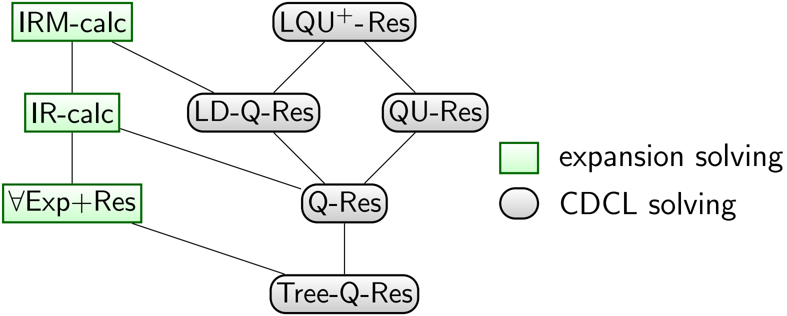
Interesting test case for algorithmic progress

SAT revolution

SAT	NP	main breakthrough late 90s
QBF	PSPACE	reaching industrial applicability now
DQBF	EXPTIME	very early stage

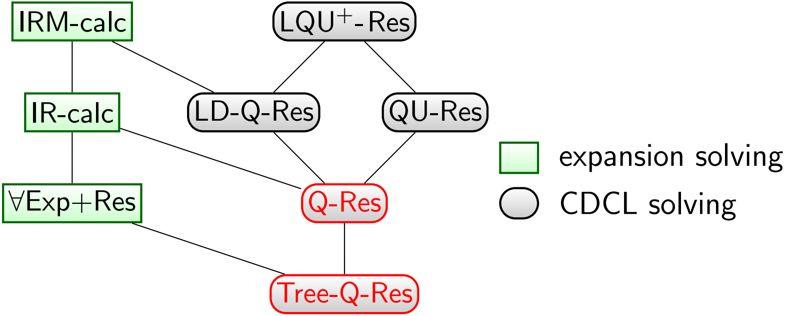
QBF proof systems

- There are two main paradigms in QBF solving: Expansion based solving and CDCL solving.
- Various QBF proof systems model these different solvers.



- Various sequent calculi exist as well.
  [Krajíček & Pudlák 90], [Cook & Morioka 05], [Egly 12]

QBF proof systems at a glance



Q-Resolution (Q-Res)

- QBF analogue of Resolution (?)
- introduced by [Kleine Büning, Karpinski, Flögel 95]
- Tree-Q-Res: tree-like version

Q-resolution

Q-resolution $=$ resolution rule $+ \forall$-reduction

Resolution

$$\frac{l \vee C_1 \qquad \neg l \vee C_2}{C_1 \vee C_2} \quad (l \text{ existentially quantified})$$

Tautologous resolvents are generally unsound and not allowed.

$\forall$-reduction

$$\frac{C \vee k}{C} \quad (k \in C \text{ is universal with innermost quant. level in } C)$$

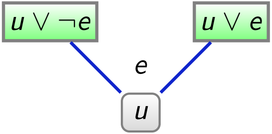
Q-resolution Example

$$\forall \mathbf{u} \exists \mathbf{e}. (\mathbf{u} \vee \neg \mathbf{e}) \wedge (\mathbf{u} \vee \mathbf{e})$$

$u \vee \neg e$ $\qquad$ $u \vee e$

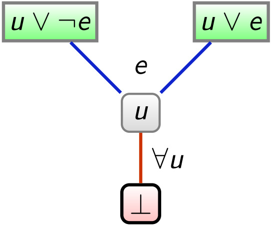
Q-resolution Example

$$\forall \mathbf{u} \exists \mathbf{e}.\ (\mathbf{u} \vee \neg \mathbf{e}) \wedge (\mathbf{u} \vee \mathbf{e})$$

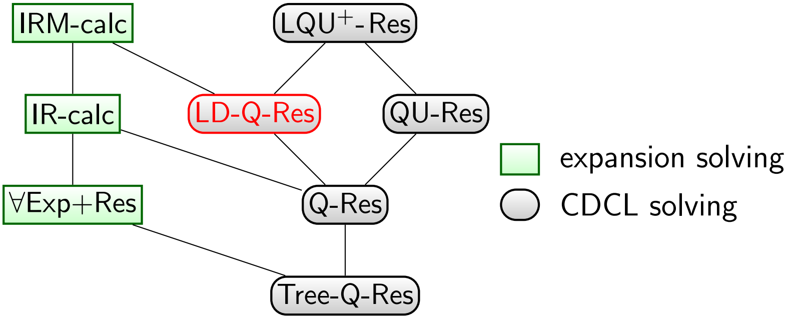


Q-resolution Example

$$\forall \mathbf{u} \exists \mathbf{e}.\, (\mathbf{u} \vee \neg \mathbf{e}) \wedge (\mathbf{u} \vee \mathbf{e})$$



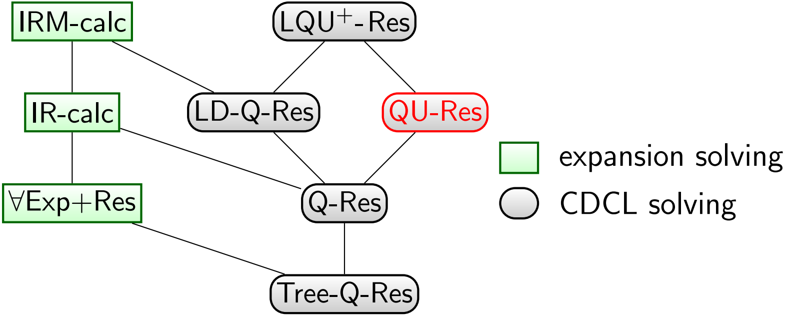
Further systems at a glance



Long-distance resolution (LD-Q-Res)

- allows certain resolution steps forbidden in Q-Res
- merges universal literals u and $\neg u$ in a clause to u^*
- introduced by [Zhang & Malik 02] [Balabanov & Jiang 12]

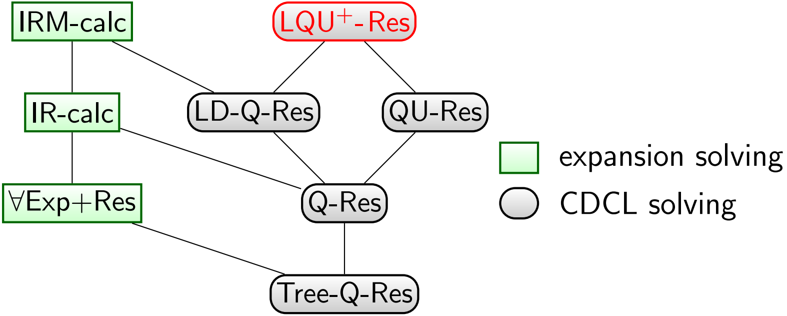
QBF proof systems at a glance



Universal resolution (QU-Res)

- allows resolution over universal pivots
- introduced by [Van Gelder 12]

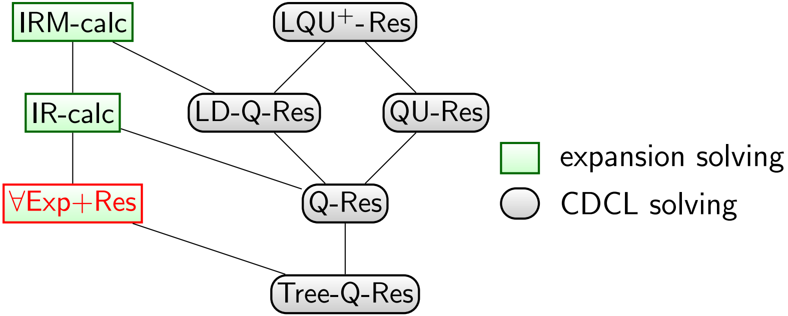
QBF proof systems at a glance



LQU$^+$-Res

- combines long-distance and universal resolution
- introduced by [Balabanov, Widl, Jiang 14]

Expansion based calculi



∀Exp+Res

- expands universal variables (for one or both values 0/1)
- introduced by [Janota & Marques-Silva 13]

$\forall$Exp+Res

Annotated literals

couple together existential and universal literals: l^{α}, where

- l is an existential literal.
- α is a partial assignment to universal literals.

Rules of $\forall$Exp+Res

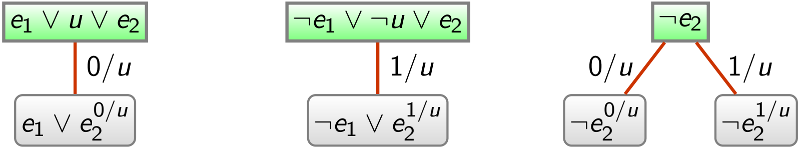
$$\frac{C \text{ in matrix}}{\left\{ l^{[\tau]} \mid l \in C, l \text{ is existential} \right\}} \text{(Axiom)}$$

- τ is a complete assignment to universal variables
  s.t. there is *no* universal literal $u \in C$ with $\tau(u) = 1$.
- $[\tau]$ takes only the part of τ that is $< l$.

$$\frac{x^{\tau} \vee C_1 \qquad \neg x^{\tau} \vee C_2}{C_1 \cup C_2} \text{(Resolution)}$$

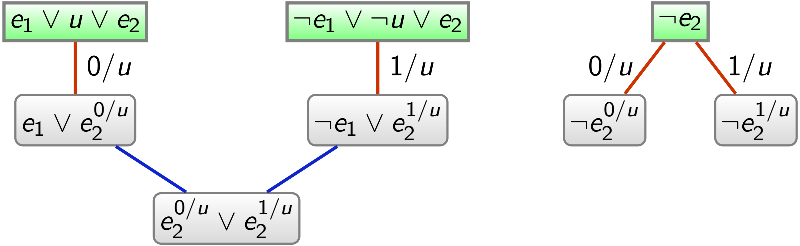
Example proof in ∀Exp+Res

$\exists \mathbf{e_1} \forall \mathbf{u} \exists \mathbf{e_2}$



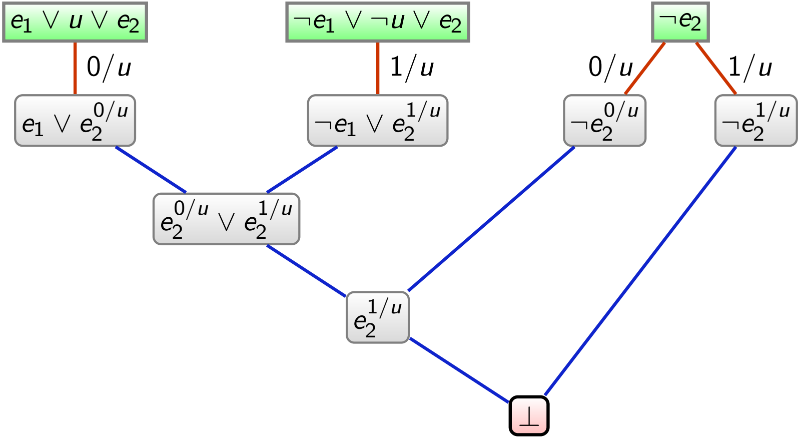
Example proof in ∀Exp+Res

$\exists \mathbf{e_1} \forall \mathbf{u} \exists \mathbf{e_2}$

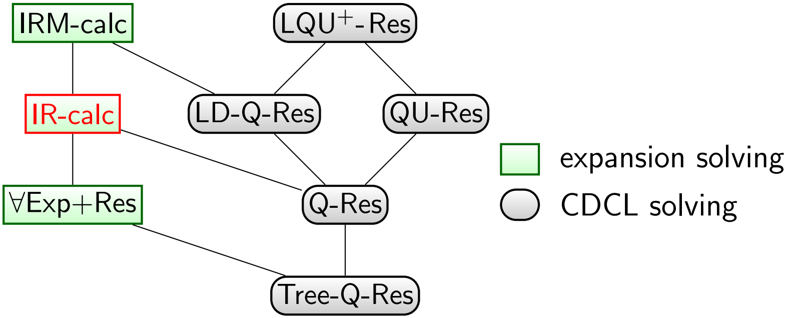


Example proof in ∀Exp+Res

$\exists \mathbf{e_1} \forall \mathbf{u} \exists \mathbf{e_2}$



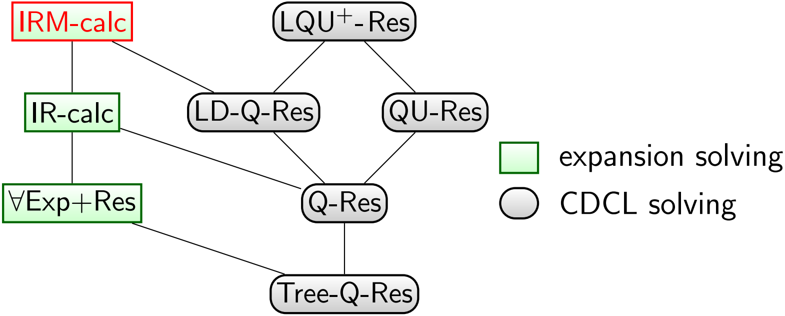
Further expansion-based systems at a glance



IR-calc

- Instantiation + Resolution
- 'delayed' expansion
- introduced by [B., Chew, Janota 14]

Further expansion-based systems at a glance



IRM-calc

- Instantiation + Resolution + Merging
- allows merged universal literals u^*
- introduced by [B., Chew, Janota 14]

From propositional proof systems to QBF

A general ∀red rule

- Fix a prenex QBF Φ.
- Let $F(\bar{x}, u)$ be a propositional line in a refutation of Φ, where u is universal with innermost quant. level in F

$$\frac{F(\bar{x}, u)}{F(\bar{x}, 0)} \qquad \frac{F(\bar{x}, u)}{F(\bar{x}, 1)}$$

New QBF proof systems

For any 'natural' line-based propositional proof system P define the QBF proof system $P + \forall\text{red}$ by adding ∀red to the rules of P.

Proposition (B., Bonacina & Chew 16)

$P + \forall\text{red}$ *is sound and complete for QBF.*

Genuine QBF lower bounds

Propositional hardness transfers to QBF

- If $\phi_n(\vec{x})$ is hard for P, then $\exists \vec{x}\, \phi_n(\vec{x})$ is hard for $P + \forall$red.
- propositional hardness: not the phenomenon we want to study.

Genuine QBF hardness

- in $P + \forall$red: just count the number of $\forall$red steps
- can be modelled precisely by allowing NP oracles in QBF proofs [Chen 16; B., Hinde & Pich 17]

QBF systems with only genuine lower bounds

A relaxation of a quantifier prefix

- can turn $\forall$ into $\exists$
- move $\forall$ to the left

The QBF system $P + \forall\text{red}^{\Sigma_k^p}$ has the rules:

- of the propositional system P
- $\forall$-reduction
- $\dfrac{C_1 \;\ldots\; C_l}{D}$ for any l,
  where the quantifier prefix Π is relaxed to a Σ_k^b-prefix Π' such that $\Pi'. \bigwedge_{i=1}^{l} C_i \models \Pi'.D \wedge \bigwedge_{i=1}^{l} C_i$

Genuine hardness results

Theorem [B., Hinde, Pich 17]

- For every odd k there exist QBFs that are easy in $Res + \forall red^{\Sigma_k^p}$, but require exponential-size proofs in $Res + \forall red^{\Sigma_{k-1}^p}$.
- There exist QBFs that require exponential-size proofs in $Res + \forall red^{\Sigma_k^p}$ for all k.

Theorem [B., Blinkhorn, Hinde 18]

Random QBFs (in a suitable random model) require exponential-size proofs in $Res + \forall red^{NP}$, $CP + \forall red^{NP}$ and $PC + \forall red^{NP}$.

Theorem [B., Bonacina, Chew 16]

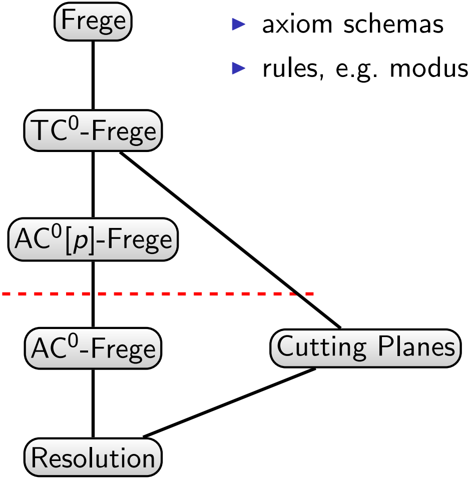
There exist QBFs that require exponential-size proofs in $AC^0[p]$-$Frege + \forall red^{NP}$.

Characterisations

Theorem [B. & Pich 16]

- super-polynomial lower bounds for *Frege* + $\forall$red $^{\mathsf{NP}}$ iff PSPACE $\not\subseteq$ NC^1
- super-polynomial lower bounds for *EF* + $\forall$red $^{\mathsf{NP}}$ iff PSPACE $\not\subseteq$ P/poly

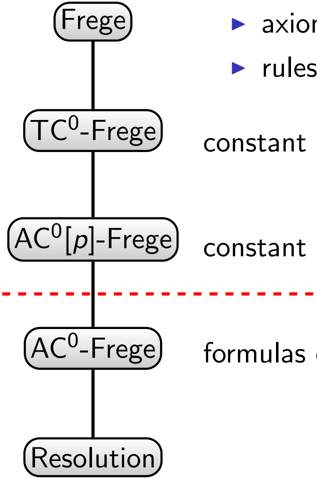
The current research frontier



Frege systems use:

- axiom schemas
- rules, e.g. modus ponens $\frac{A \quad A \to B}{B}$

The current research frontier



Frege systems use:

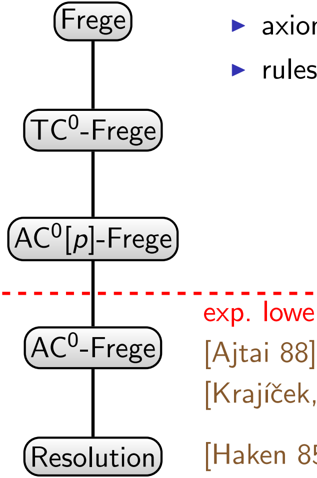
- axiom schemas
- rules, e.g. modus ponens $\frac{A \quad A \rightarrow B}{B}$

TC^0-Frege: constant depth with counting gates

AC^0[p]-Frege: constant depth with mod p gates

AC^0-Frege: formulas of constant depth

The current research frontier



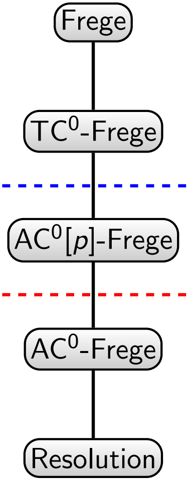
Frege systems use:

- axiom schemas
- rules, e.g. modus ponens $\frac{A \quad A \to B}{B}$

exp. lower bounds (propositional)
[Ajtai 88] [Pitassi, Beame & Impagliazzo 93]
[Krajíček, Pudlák & Woods 95]

[Haken 85]

The current research frontier



Frege systems use:

- axiom schemas
- rules, e.g. modus ponens $\frac{A \quad A \rightarrow B}{B}$

exp. lower bounds (QBF)
[B., Bonacina & Chew (ITCS'16)]

exp. lower bounds (propositional)
[Ajtai 88] [Pitassi, Beame & Impagliazzo 93]
[Krajíček, Pudlák & Woods 95]

[Haken 85]

Semantics via a two-player game

- We consider QBFs in prenex form
  Example: $\forall y_1 y_2 \exists x_1 x_2. (\neg y_1 \vee x_1) \wedge (y_2 \vee \neg x_2)$
- Two-player game between $\exists$ and $\forall$.
- $\exists$ wins a game if the matrix becomes true.
- $\forall$ wins a game if the matrix becomes false.
- A QBF is true iff there exists a winning strategy for $\exists$.
- A QBF is false iff there exists a winning strategy for $\forall$.

Response map

A response map R for a proof system $P + \forall\text{red}$ is a function

$$R : (L, \alpha) \mapsto \beta \qquad \text{where}$$

- L is a line in $P + \forall\text{red}$
- α is a total assignment to the existential variables of L
- β is a total assignment to the universal variables in L

such that if $L|_\alpha$ is not a tautology, then $L|_{\alpha \cup \beta}$ is false.

Example: Resolution

- lines are clauses, e.g. $L = \underbrace{x_1 \vee \neg x_2}_{\textit{existential}} \vee \underbrace{u_1 \vee u_2}_{\textit{universal}}$
- map (L, α) to $(u_1/0, u_2/0)$.
- Response is independent of α.

Strategy extraction algorithm

Round-based strategy extraction

- Fix a response map R for $P + \forall\mathsf{red}$.
- Let π a $P + \forall\mathsf{red}$ refutation for $\Phi = \exists E_1 \forall U_1 \cdots \exists E_n \forall U_n\ \phi$.
- $\exists$ player chooses an assignment α_1 for E_1.
- $\forall$ player searches for the first line L in π which only contains variables from $E_1 \cup U_1$ and is not a tautology under α_1.
- $\forall$ responds by $R(L, \alpha_1)$.
- iteratively continue with E_2, U_2 ...

The cost of strategies

Definition

- Fix a winning strategy S for a QBF Φ and consider the size of its range (in each universal block).
- The cost of Φ is the minimum of this range size over all winning strategies.

Intuition

Strategies that require many responses of the universal player (in one block) are costly.

Example

Equality formulas

$$\exists x_1 \cdots x_n \forall u_1 \cdots u_n \exists t_1 \cdots t_n$$
$$\left(\bigwedge_{i=1}^{n} (x_i \vee u_i \vee \neg t_i) \wedge (\neg x_i \vee \neg u_i \vee \neg t_i) \right) \wedge \left(\bigvee_{i=1}^{n} t_i \right).$$

- The only winning strategy for these formulas is $u_i = x_i$ for $i = 1, \ldots, n$.
- The cost (=size of the range of the winning strategy) is 2^n.

Capacity

Capacity of lines and proofs

- Let L be a line in $P+\forall$red.
- The capacity of a line L is the size of the minimal range of $R(L,.)$ over all response maps R for $P+\forall$red.
- The capacity of a $P+\forall$red proof is the maximum of the capacity of its lines.

Example

- Clauses have capacity 1 (require only one response).
- Resolution proofs have always capacity 1.

The central connection

The Size-Cost-Capacity Theorem [B., Blinkhorn, Hinde 18]

For each $P + \forall\text{red}^{\text{ NP}}$ proof π of a QBF ϕ we have

$$|\pi| \geq \frac{cost(\phi)}{capacity(\pi)}.$$

Example: Equality formulas in resolution

$\exists x_1 \cdots x_n \forall u_1 \cdots u_n \exists t_1 \cdots t_n$
$[\bigwedge_{i=1}^{n} (x_i \vee u_i \vee \neg t_i) \wedge (\neg x_i \vee \neg u_i \vee \neg t_i)] \wedge \bigvee_{i=1}^{n} t_i$

- cost $= 2^n$
- capacity $= 1$
- $\Rightarrow$ proofs in $Res + \forall\text{red}$ are of size 2^n.

The central connection

The Size-Cost-Capacity Theorem [B., Blinkhorn, Hinde 18]

For each $P + \forall\text{red}^{\text{NP}}$ proof π of a QBF ϕ we have

$$|\pi| \geq \frac{cost(\phi)}{capacity(\pi)}.$$

Intuition on the proof

- cost counts the number of necessary responses of universal winning strategies
- these can be extracted from the proof (by the round-based strategy extraction algorithm)
- capacity gives an upper bound on how many responses can be extracted per line

The central connection

The Size-Cost-Capacity Theorem [B., Blinkhorn, Hinde 18]

For each $P + \forall\text{red}^{\text{NP}}$ proof π of a QBF ϕ we have

$$|\pi| \geq \frac{\mathit{cost}(\phi)}{\mathit{capacity}(\pi)}.$$

Remarks

- lower bound technique with semantic flavour
- works for all base systems P (under very mild assumptions)
- always produces 'genuine' QBF lower bounds on the number of $\forall$-reduction steps

In other QBF systems

Cutting planes

- capacity of lines is still 1
- the best response for a line

$$\underbrace{a_1 x_1 + \dots a_m x_m}_{existential} + \underbrace{b_1 u_1 + \dots b_n u_n}_{universal} \geq C$$

is to play $u_i = 0$ if $b_i > 0$ and 1 otherwise

Corollaries

- For each $CP + \forall$red proof π of a QBF ϕ we have $|\pi| \geq cost(\phi)$.
- Equality formulas require $CP + \forall$red proofs of size 2^n.

Polynomial Calculus (with Resolution)

Capacity is non-constant

- consider $x(1-u)+(1-x)u=0$
- winning strategy is $u=1-x$.
- requires 2 responses, hence capacity of the line is 2.

Lemma

If π is a $PC+\forall$red proof where each line contains at most M monomials, then $\mathit{capacity}(\pi) \leq M$.

Corollary

For each $PC+\forall$red proof π of a QBF ϕ we have $|\pi| \geq \sqrt{\mathit{cost}(\phi)}$.

Frege

Capacity can be exponential

- Consider $\bigvee_{i=1}^{n} [(x_i \vee u_i) \wedge (\neg x_i \vee \neg u_i)]$.
- The unique winning response is to play $u_i = x_i$ for all $i \in [n]$.
- Capacity of this line is 2^n.

Proposition

Equality formulas are easy in $\mathit{Frege} + \forall\text{red}$.

Application: Hard random formulas in QBF

Random QBFs

- Pick clauses $C_i^1, \dots, C_i^{cn}$ uniformly at random
- for each C_i^j choose 1 literal from the set $X_i = \{x_i^1, \dots, x_i^m\}$ and 2 literals from $Y_i = \{y_i^1, \dots, y_i^n\}$.
- Define $Q(n, m, c)$ as

$$\exists Y_1 \dots Y_n \forall X_1 \dots X_n \exists t_1 \dots t_n . \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{cn} \left(\neg t_i \vee C_i^j \right) \wedge \bigvee_{i=1}^{n} t_i$$

Remarks

- All clauses contain existential and universal literals.
- Rightmost quantifier block is existential.

Hardness of the random QBFs

$$Q(n, m, c) = \exists Y_1 \dots Y_n \forall X_1 \dots X_n \exists t_1 \dots t_n . \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{cn} \left(\neg t_i \vee C_i^j \right) \wedge \bigvee_{i=1}^{n} t_i$$

Theorem

Let $1 < c < 2$ and $m \leq (1 - \epsilon) \log_2(n)$ for some $\epsilon > 0$.
With high probability, $Q(n, m, c)$ is false and requires size $2^{\Omega(n^\epsilon)}$ in QU-Resolution, $CP + \forall\text{red}$, and $PCR + \forall\text{red}$.

Proof idea

$Q(n, m, c)$ is false iff all QBFs $\Psi_i = \exists Y_i \forall X_i \bigwedge_{j=1}^{cn} C_i^j$ are false.

1. Show that Ψ_i is false whp.
2. Show that Ψ_i requires non-constant winning strategies whp.

Proof sketch

$\Psi_i = \exists Y_i \forall X_i \bigwedge_{j=1}^{cn} C_i^j$

1. Ψ_i is false whp.
    - each clause contains 2 existential and 1 universal variable
    - the formula is true iff the $\exists$ player can satisfy at least one variable in each clause
    - we therefore reduce the problem to 2-SAT and can use results of [Chvatal and Reed 92, Bollobás et al. 01 . . .]

Proof sketch

$\Psi_i = \exists Y_i \forall X_i \bigwedge_{j=1}^{cn} C_i^j$

2. Ψ_i requires non-constant winning strategies whp.

- use a result of [Creignou et al. 15] on satisfiability of random (1,2)-QBFs
- (1,2)-QBFs use clauses with 1 universal and 2 existential variables each and prefix $\forall X \exists Y$
- Then $\exists Y_i \forall X_i \bigwedge_{j=1}^{cn} C_i^j$ is false whp and $\forall X_i \exists Y_i \bigwedge_{j=1}^{cn} C_i^j$ is true whp.
- Therefore, winning strategies are non-constant whp.

Frege and Stronger Systems

Frege Systems

- Frege systems derive formulas using axioms and rules.
- Usually called Hilbert-style systems in texts on classical logic.

Definition

A Frege rule is a $(k+1)$-tuple $(\varphi_0, \varphi_1 \ldots, \varphi_k)$ of propositional formulas such that

$$\{\varphi_1, \varphi_2, \ldots, \varphi_k\} \models \varphi_0 \quad .$$

The standard notation for rules is

$$\frac{\varphi_1 \quad \varphi_2 \quad \cdots \quad \varphi_k}{\varphi_0} \quad .$$

A Frege rule with $k = 0$ is called a Frege axiom.

Frege Proofs

- A formula ψ_0 can be derived from formulas $\psi_1, \ldots, \psi_k$ by a Frege rule $(\varphi_0, \varphi_1 \ldots, \varphi_k)$ if there exists a substitution σ such that
$$\sigma(\varphi_i) = \psi_i \quad \text{for } i = 0, \ldots, k \ .$$
- Let $\mathcal{F}$ be a finite set of Frege rules.
- An $\mathcal{F}$-proof of a formula φ from a set of propositional formulas Φ is a sequence $\varphi_1, \ldots, \varphi_l = \varphi$ of propositional formulas such that for all $i = 1, \ldots, l$ one of the following holds:
  1. $\varphi_i \in \Phi$ or
  2. there exist numbers $1 \leq i_1 \leq \cdots \leq i_k < i$ such that φ_i can be derived from $\varphi_{i_1}, \ldots, \varphi_{i_k}$ by a Frege rule from $\mathcal{F}$.
- Notation: $\mathcal{F} : \Phi \vdash \varphi$

Frege Systems

- $\mathcal{F}$ is called complete if for all formulas φ

$$\models \varphi \quad \Longleftrightarrow \quad \mathcal{F} : \emptyset \vdash \varphi \ .$$

- $\mathcal{F}$ is called implicationally complete if for all formulas φ and sets of formulas Φ

$$\Phi \models \varphi \quad \Longleftrightarrow \quad \mathcal{F} : \Phi \vdash \varphi \ .$$

- $\mathcal{F}$ is a Frege system if $\mathcal{F}$ is implicationally complete.

Example of a Frege System

Axioms

$$p_1 \rightarrow (p_2 \rightarrow p_1)$$
$$(p_1 \rightarrow p_2) \rightarrow (p_1 \rightarrow (p_2 \rightarrow p_3)) \rightarrow (p_1 \rightarrow p_3)$$
$$p_1 \rightarrow p_1 \vee p_2$$
$$p_2 \rightarrow p_1 \vee p_2$$
$$(p_1 \rightarrow p_3) \rightarrow (p_2 \rightarrow p_3) \rightarrow (p_1 \vee p_2 \rightarrow p_3)$$
$$(p_1 \rightarrow p_2) \rightarrow (p_1 \rightarrow \neg p_2) \rightarrow \neg p_1$$
$$\neg\neg p_1 \rightarrow p_1$$
$$p_1 \wedge p_2 \rightarrow p_1$$
$$p_1 \wedge p_2 \rightarrow p_2$$
$$p_1 \rightarrow p_2 \rightarrow p_1 \wedge p_2$$

Modus Ponens

$$\frac{p_1 \qquad p_1 \rightarrow p_2}{p_2}$$

Simulations Between Proof Systems

Definition (Cook, Reckhow 79)

- A proof system Q p-simulates a proof system P ($P \leq_p Q$) if there exists a poly-time function f such that $P(\pi) = Q(f(\pi))$ for all π.
- P and Q are p-equivalent ($P \equiv_p Q$) if $P \leq_p Q$ and $Q \leq_p P$.

Equivalence of Classical Frege Systems

Theorem (Cook, Reckhow 79)

All Frege systems are polynomially equivalent.

Sketch of Proof

1. If F_1 and F_2 are Frege systems over distinct propositional languages L_1 and L_2, respectively, then we have to translate L_1-formulas into L_2-formulas.
   To obtain polynomial size formulas after the translation, we rebalance the formulas to logarithmic logical depth.
   This is possible by Spira's theorem.
2. Let F_1 and F_2 be two Frege systems using the same propositional language. Then the equivalence of F_1 and F_2 can be shown by deriving every F_1-rule in F_2 and vice versa. $\square$

A Game for Frege Systems

- developed by Pudlák and Buss 94
- played between Prover and Spoiler

Pudlák-Buss games

- Aim: prove that φ is a tautology.
- Spoiler claims that he knows a falsifying assignment α for φ.
- Prover asks the value of arbitrary formulas under α.
- Spoiler answers 0 or 1.
- Prover wins if he finds an immediate contradiction, e.g. he got answer 0 for $\theta_1 \wedge \theta_2$ and answer 1 for both θ_1 and θ_2.

Proofs and Games

Theorem

The minimal number of rounds in the game to prove φ is proportional to the logarithm of the minimal number of steps in a Frege proof of φ.

Proof

- We only show one direction.
- Let $\varphi_1, \ldots, \varphi_k$ be a Frege proof of φ.
- Prover first asks $\varphi = \varphi_k$ and gets answers 0.
- Prover then asks $\bigwedge_{i=1}^{k} \varphi_i$.
- If Spoiler answers 1, this immediately contradicts the previous answer.
- If Spoiler answers 0, Prover uses binary search to find the smallest i such that Spoiler answers 1 to $\bigwedge_{i=1}^{i} \varphi_i$ and 0 to $\bigwedge_{i=1}^{i+1} \varphi_i$.

Proof (cont'd)

- Prover uses binary search to find the smallest i such that Spoiler answers 1 to $\bigwedge_{i=1}^{i} \varphi_i$ and 0 to $\bigwedge_{i=1}^{i+1} \varphi_i$.
- Case 1: If no such i exists, Spoiler answered 0 to φ_1 which is an axiom, i.e. a substitution of a constant-size tautology like $A \vee \neg A$.
- Then Prover can find a contradiction in a constant number of queries.

Proof (cont'd)

- Prover uses binary search to find the smallest i such that Spoiler answers 1 to $\bigwedge_{i=1}^{i} \varphi_i$ and 0 to $\bigwedge_{i=1}^{i+1} \varphi_i$.
- Case 2: If this minimal i exists, then φ_{i+1} was derived by a Frege rule (e.g. Modus Ponens) from a constant number of formulas $\varphi_{i_1}, \dots, \varphi_{i_s}$ with $i_1 < \cdots < i_s < i + 1$.
- Prover asks the formulas $\varphi_{i_1}, \dots, \varphi_{i_s}$.
- If Spoiler answered 0 to any of these queries, this contradicts the answer to $\bigwedge_{i=1}^{i} \varphi_i$.
- If Spoiler answered 1 to $\varphi_{i_1}, \dots, \varphi_{i_s}$, then Prover finds a contradiction in a constant number of rounds, because a Frege rule is a substitution of a constant-size tautology. □

Lower Bounds by Games

Theorem

The minimal number of rounds in the game to prove φ is proportional to the logarithm of the minimal number of steps in a Frege proof of φ.

Strategy

Show lower bounds for Frege by devising good strategies for Spoiler.

Problem

Has not been done successfully for Frege systems.

Bounded-Depth Frege

The logical depth of a formula

is defined as the maximal number of alternations of logical operators in the formula.

Example

- Clauses have depth 1.
- Formulas in DNF or CNF have depth 2.

Bounded-depth Frege

Allow only formulas of logical depth d in the proof for a given constant d.

One of the strongest current lower bounds

Theorem

For any Frege system F and any integer d, there exists a constant $\delta > 0$ such that for large enough n, the size of a depth d F-proof of PHP_n^{n+1} is at least 2^{n^δ}.

History

- Ajtai (1988): First super-polynomial lower bound for PHP in bounded-depth Frege systems
- Uses the connection to bounded arithmetic
- Improved to exponential lower bounds by Pitassi, Beame & Impagliazzo 92 and independently by Krajíček, Pudlák & Woods 92
- Simplified proof by Ben-Sasson & Harsha 2010 using Pudlák-Buss games

Hard Formulas for Frege Systems?

Theorem (Buss 87)

The pigeonhole principle has polynomial-size proofs in Frege systems.

The search for hard formulas

- A number of combinatorial principles have been suggested, but most have poly-size Frege proofs.
- A good candidate from logic: reflection principles
- Problem: hard to analyze
- A promising approach: formulas from pseudo-random generators (Krajíček, Razborov)

Beyond Frege

Bounds on Proof Systems

Size of proofs

Let f be a proof system.

- $s_f(x) = \min\{|w| \mid f(w) = x\}$
- $s_f(n) = \max\{s_f(x) \mid |x| \leq n\}$
- f is t-bounded if $s_f(n) \leq t(n)$ for all $n \in \mathbb{N}$.
- If t is a polynomial, then f is called polynomially bounded.

Bounds on Proof Systems

Size of proofs

Let f be a proof system.

- $s_f(x) = \min\{|w| \mid f(w) = x\}$
- $s_f(n) = \max\{s_f(x) \mid |x| \leq n\}$
- f is t-bounded if $s_f(n) \leq t(n)$ for all $n \in \mathbb{N}$.
- If t is a polynomial, then f is called polynomially bounded.

Number of steps

- This measure only makes sense for proof systems where proofs consist of lines containing formulae or sequents.
- $t_f(\varphi) = \min\{k \mid f(\pi) = \varphi \text{ and } \pi \text{ uses } k \text{ steps}\}$
- $t_f(n) = \max\{t_f(\varphi) \mid |\varphi| \leq n\}$
- Obviously, it holds that $t_f(n) \leq s_f(n)$.

Extensions of Frege Systems

Extended Frege EF

Abbreviations for complex formulas: $q \leftrightarrow \psi$,
where q is a new propositional variable.

Extensions of Frege Systems

Extended Frege *EF*

Abbreviations for complex formulas: $q \leftrightarrow \psi$,
where q is a new propositional variable.

More precisely

An extended Frege proof of φ is a sequence $(\varphi_1, \ldots, \varphi_l = \varphi)$ of propositional formulas such that for each $i = 1, \ldots, l$ one of the following holds:

1. φ_i has been derived by a Frege rule or axiom;
2. $\varphi_i = q \leftrightarrow \psi$ where ψ is an arbitrary propositional formula and q is a new propositional variable that does not occur in φ, ψ and φ_j for $1 \leq j < i$.

Extensions of Frege Systems

Frege systems with substitution *SF*

Substitution rule: $\dfrac{\varphi}{\sigma(\varphi)}$

for arbitrary substitutions σ

The picture

- All Frege systems are p-equivalent.
- Frege $\leq_p$ *EF* $\equiv_p$ *SF*.

The Picture for Extensions of Frege

Current barrier

- lower bounds to size in Frege systems.

The Picture for Extensions of Frege

Current barrier

- lower bounds to size in Frege systems.

The following measures are equivalent

- number of steps in Frege;
- size in EF;
- number of steps in EF;
- size of SF.

The Picture for Extensions of Frege

Current barrier

- lower bounds to size in Frege systems.

The following measures are equivalent

- number of steps in Frege;
- size in EF;
- number of steps in EF;
- size of SF.

We can exponentially separate

- number of steps in EF;
- number of steps in SF.

Frege and EF

The following measures are equivalent

- number of steps in Frege;
- size in *EF*.

Frege and EF

The following measures are equivalent

- number of steps in Frege;
- size in EF.

Corollary

Proving lower bounds on the number of steps in Frege systems means proving lower bounds on the size of EF.

Intermezzo: Arithmetic Formulas

The language of arithmetic uses the symbols

$$0,\ S,\ +,\ *,\ \leq\ \ldots$$

- Σ_1^b-formulas are formulas in prenex normal form with only bounded $\exists$-quantifiers, i.e. $(\exists x \leq t(y))\psi(x, y)$.
- Σ_1^b-formulas describe NP-sets.
- Π_1^b-formulas: $(\forall x \leq t(y))\psi(x, y) \Rightarrow$ coNP-sets

Translating Π_1^b-Formulas into Propositional Formulas

Definition (Cook 75, Krajíček & Pudlák 90)

Let $\varphi \in \Pi_1^b$. Then there are propositional formulas $\|\varphi\|^n$, $n \in \mathbb{N}$ such that:

- $\|\varphi\|^n$ can be constructed in polynomial time from 1^n.
- $\|\varphi\|^n$ is a tautology $\iff$ $\mathbb{N} \models \varphi(a)$ for all $a \in \mathbb{N}$ of length $\leq n$

The Reflection Principle

Definition

The reflection principle of a propositional proof system P is defined by the arithmetic formula

$$RFN(P) = (\forall \pi)(\forall \varphi) Prf_P(\pi, \varphi) \to Taut(\varphi)$$

where

- Prf_P is a Σ_1^b-formula formalizing P-proofs
- $Taut$ is a Π_1^b-formula for propositional tautologies.

Very Strong Proof Systems

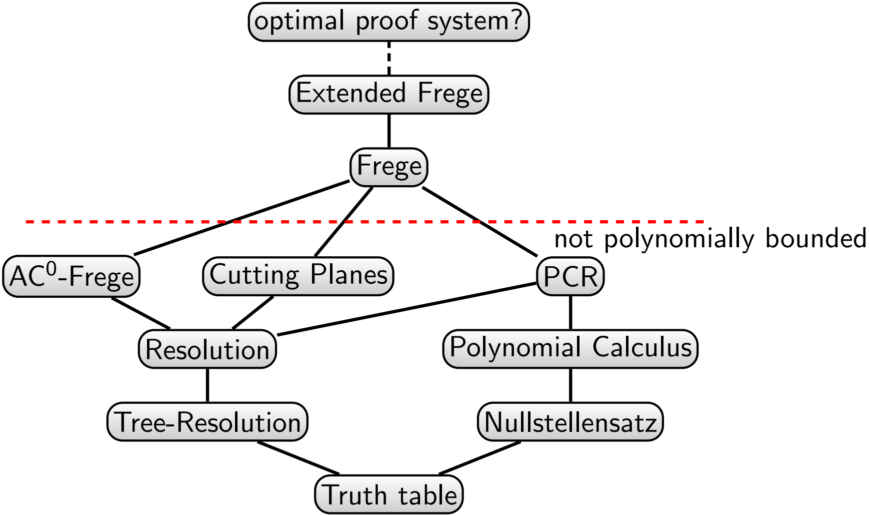
Theorem (Krajíček, Pudlák 89)

Every proof system P is simulated by a proof system of the form $EF + \Phi$.

Sketch of Proof

- Take as Φ the translations of the reflection principle of P.
- Let π be a P-proof of φ.
- Substituting the bits of π and φ into the reflection principle yields
$$\|\mathit{Prf}_P(\pi, \varphi)\| \to \|\mathit{Taut}(\varphi)\|$$
- Prove $\|\mathit{Prf}_P(\pi, \varphi)\|$ in EF.
- Prove $\|\mathit{Taut}(\varphi)\| \to \varphi$ in EF.
- Obtain an $EF + \Phi$ proof of φ which is poly-size in $|\pi|$, $|\varphi|$.

Simulations between important propositional proof systems



Does TAUT have Optimal Proof Systems?

Question (Krajíček, Pudlák 89)

Does TAUT have an optimal proof system?

Does TAUT have Optimal Proof Systems?

Question (Krajíček, Pudlák 89)

Does TAUT have an optimal proof system?

Some partial answers

- If NE $=$ coNE, then TAUT has optimal proof systems.
  [Krajíček, Pudlák 89]
- Optimal proof systems for TAUT imply complete sets for promise classes (e.g. NP $\cap$ Sparse, UP, disjoint NP-pairs).
  [Köbler, Messner, Torán 03]

Optimal Proof Systems and Easy Subsets

Definition

A class $\mathbb{C}$ of languages has a recursive P-presentation if there exists a recursively enumerable list $N_1, N_2, \ldots$ of deterministic polynomial-time clocked Turing machines such that $L(N_i) \in \mathbb{C}$ for $i \in \mathbb{N}$, and, conversely, for each $A \in \mathbb{C}$ there exists an index i with $A \subseteq L(N_i)$.

Theorem (Sadowski 02)

TAUT has a p-optimal proof system if and only if the class of all P*-subsets of TAUT has a recursive* P*-presentation.*

Proof Systems that Take Advice

Cook & Krajíček (JSL 07) consider non-uniform Frege proofs.

Proof Systems that Take Advice

Cook & Krajíček (JSL 07) consider non-uniform Frege proofs.

Definition (Karp, Lipton 80)

- An advice function is a mapping $h : \mathbb{N} \to \Sigma^*$.
- $h(n)$ is the advice string provided by h for input length n.
- For a language L, $L/h = \{x \mid \langle x, h(|x|)\rangle \in L\}$.

Proof Systems that Take Advice

Cook & Krajíček (JSL 07) consider non-uniform Frege proofs.

Definition (Karp, Lipton 80)

- An advice function is a mapping $h : \mathbb{N} \to \Sigma^*$.
- $h(n)$ is the advice string provided by h for input length n.
- For a language L, $L/h = \{x \mid \langle x, h(|x|)\rangle \in L\}$.
- For a complexity class C and a length bound $k : \mathbb{N} \to \mathbb{N}$, $C/k = \{L/h \mid L \in C, |h(n)| \leq k(n) \text{ for all } n\}$.

Proof Systems that Take Advice

Cook & Krajiček (JSL 07) consider non-uniform Frege proofs.

Definition (Karp, Lipton 80)

- An advice function is a mapping $h : \mathbb{N} \to \Sigma^*$.
- $h(n)$ is the advice string provided by h for input length n.
- For a language L, $L/h = \{x \mid \langle x, h(|x|)\rangle \in L\}$.
- For a complexity class C and a length bound $k : \mathbb{N} \to \mathbb{N}$, $\mathsf{C}/k = \{L/h \mid L \in \mathsf{C}, |h(n)| \leq k(n) \text{ for all } n\}$.
- $\mathsf{C}/\log = \bigcup\{\mathsf{C}/k \mid k(n) = O(\log n)\}$.
- $\mathsf{C}/\text{poly} = \bigcup\{\mathsf{C}/k \mid k(n) = n^{O(1)}\}$.

Proof Systems that Take Advice

Cook & Krajíček (JSL 07) consider non-uniform Frege proofs.

Definition (Karp, Lipton 80)

- An advice function is a mapping $h : \mathbb{N} \to \Sigma^*$.
- $h(n)$ is the advice string provided by h for input length n.
- For a language L, $L/h = \{x \mid \langle x, h(|x|)\rangle \in L\}$.
- For a complexity class C and a length bound $k : \mathbb{N} \to \mathbb{N}$, $\mathsf{C}/k = \{L/h \mid L \in \mathsf{C}, |h(n)| \leq k(n) \text{ for all } n\}$.
- $\mathsf{C}/\log = \bigcup\{\mathsf{C}/k \mid k(n) = O(\log n)\}$.
- $\mathsf{C}/\mathrm{poly} = \bigcup\{\mathsf{C}/k \mid k(n) = n^{O(1)}\}$.

Proposition (Pippenger 79)

$L \in \mathsf{P}/\mathrm{poly}$ *iff L has poly-size circuits.*

All languages have optimal proof systems with advice

Theorem (Cook, Krajíček 07, B, Köbler, Müller 11)

Every language L has an optimal proof system f in FP/1.

Proof.

- Let $\langle \cdot, \ldots, \cdot \rangle$ be a polynomial-time computable tupling function on Σ^* which is length injective.
- f-proofs are of the form $w = \langle u, 1^T, 1^m \rangle$ with $u, T \in \Sigma^*$ and $m \in \mathbb{N}$.
- The advice bit $h(|w|)$ indicates whether the transducer T only outputs elements from L for inputs of length $|u|$.
- Now, if $h(|w|) = 1$ and $T(u)$ outputs y after at most m steps, then $f(w) = y$. Otherwise, $f(w) = \top$.
- If g is a proof system computed by a p-time transducer T, then f p-simulates g via the FP function $u \mapsto \left\langle u, 1^T, 1^{p(|u|)} \right\rangle$.

□

Summary

Lower Bounds

- Shown for bounded-depth Frege
- Open for Frege and stronger systems

Optimal proof systems

- Existence is open
- We have a number of interesting characterizations and consequences.
- They exist for stronger models of proof systems.

Proof Complexity – Further Connections

Motivations in Proof Complexity

Major motivations

- Separation of complexity classes
- Satisfiability algorithms (SAT-Solver)
- Proof Search – Automatizability
- Relations to bounded arithmetic
- Proving lower bounds is very challenging and interesting in its own right

Digression – Disjoint NP-Pairs

Definition (Grollmann, Selman 88)

(A, B) is a *disjoint NP-Pair (DNPP)* if $A, B \in \mathrm{NP}$ and $A \cap B = \emptyset$.

Example

Clique-Colouring pair (CC_0, CC_1)

$CC_0 = \{(G, k) \mid G \text{ contains a clique of size } k\}$

$CC_1 = \{(G, k) \mid G \text{ can be coloured with } k-1 \text{ colours }\}$

Definition (Grollmann, Selman 88)

$(A, B) \leq_p (C, D) \overset{df}{\iff}$ there exists a polynomial time computable function f such that $f(A) \subseteq C$ and $f(B) \subseteq D$.

P-Separable Pairs

Definition (Grollmann, Selman 88)

(A,B) is p-separable, if there exists a set $C \in \mathrm{P}$ such that $A \subseteq C$ and $B \cap C = \emptyset$.

Theorem (Lovász 79)

(CC_0, CC_1) *is p-separable.*

A Pair from Cryptography

The RSA pair

$$
\begin{aligned}
RSA_0 = \{(n, e, y, i) \mid \quad & (n, e) \text{ is a valid RSA key, } \exists x \; x^e \equiv y \bmod n \\
& \text{and the } i\text{-th bit of } x \text{ is } 0\} \\
RSA_1 = \{(n, e, y, i) \mid \quad & \ldots \quad \text{is } 1 \,\}
\end{aligned}
$$

Fact

If RSA is secure then (RSA_0, RSA_1) is not p-separable.

Canonical NP-Pairs

Definition (Razborov 94)

To a proof system P we associate a canonical pair:

$$\begin{aligned} Ref(P) &= \{(\varphi, 1^m) \mid P \vdash_{\leq m} \varphi\} \\ Sat^* &= \{(\varphi, 1^m) \mid \neg\varphi \text{ is satisfiable}\} \end{aligned}$$

Proposition

If P and S are proof systems with $P \leq S$, then $(Ref(P), Sat^*) \leq_p (Ref(S), Sat^*)$.

Proof.

$(\varphi, 1^m) \mapsto (\varphi, 1^{p(m)})$ where p is the polynomial from $P \leq S$. □

Canonical NP-Pairs

Definition (Razborov 94)

To a proof system P we associate a canonical pair:

$$\begin{aligned} Ref(P) &= \{(\varphi, 1^m) \mid P \vdash_{\leq m} \varphi\} \\ Sat^* &= \{(\varphi, 1^m) \mid \neg\varphi \text{ is satisfiable}\} \end{aligned}$$

Proposition

If P and S are proof systems with $P \leq S$, then $(Ref(P), Sat^*) \leq_p (Ref(S), Sat^*)$.

Proof.

$(\varphi, 1^m) \mapsto (\varphi, 1^{p(m)})$ where p is the polynomial from $P \leq S$. □

The converse does not hold.

Automatizability of proof systems

Definition

P is automatizable if there exists a deterministic algorithm with

input:	a formula φ
output:	a P-proof of φ (if it exists)
time:	polynomial in the length of the shortest P-proof of φ

Alternative characterization

P is automatizable if and only if there exists a polynomial time algorithm with

input:	$(\varphi, 1^m)$
output:	a P-proof of φ if $(\varphi, 1^m) \in Ref(P)$

Corollary

If P is automatizable then $(Ref(P), SAT^*)$ is p-separable.

Automatizability of proof systems

Proposition (B. 07)

There exists a proof system P that has a p-separable canonical pair. But P is not automatizable unless $\mathsf{P} = \mathsf{NP}$.

Proof.

Define the proof system P as:

$$P(\pi) = \begin{cases} \varphi & \text{if } \pi = (\varphi, T) \text{ where } T \text{ is a truth table of } \varphi \\ \varphi \vee \top & \text{if } \pi = (\varphi, \alpha) \text{ and } \alpha \text{ is a satisfying assignment for } \varphi \end{cases}$$

The following algorithm separates the canonical pair of P:

```
Input:  (φ, 1^m)
IF φ = ψ ∨ ⊤ or φ = ⊤ THEN output 1
IF m ≥ 2^{||Var(φ)||} THEN
  IF φ ∈ TAUT THEN output 1
output 0
```

□

Automatizability of proof systems

Proposition (Pudlák 03)

$(Ref(P), SAT^*)$ is p-separable iff there exists an automatizable proof system $Q \geq_p P$.

Proof.

Let $(Ref(P), SAT^*)$ be separated by $f \in \mathrm{FP}$, i.e.

$$\begin{aligned} (\varphi, 1^m) \in Ref(P) \quad &\Longrightarrow \quad f(\varphi, 1^m) = 1 \\ (\varphi, 1^m) \in SAT^* \quad &\Longrightarrow \quad f(\varphi, 1^m) = 0 \ . \end{aligned}$$

Define the system Q by

$$Q(\pi) = \begin{cases} \varphi & \text{if } \pi = (\varphi, 1^m) \text{ and } f(\varphi, 1^m) = 1 \\ \top & \text{otherwise} \ . \end{cases}$$

□

Weak Automatizability

Proposition (Pudlák 03)

$(Ref(P), SAT^*)$ is p-separable iff there exists an automatizable proof system $Q \geq_p P$.

Definition

A proof system P is weakly automatizable if there exists a proof system $Q \geq_p P$ such that Q is automatizable.

Corollary

A proof system P is weakly automatizable iff the canonical pair of P is p-separable.

Which Proof Systems are Automatizable?

A trivial positive example

The truth-table system is automatizable.

What about interesting systems?

Theorem (Krajíček & Pudlák 98)

Extended Frege systems are not weakly automatizable unless RSA is insecure.

Theorem (Bonet, Pitassi, Raz 00)

Frege systems are not weakly automatizable unless Blum integers can be factored in polynomial time
(a Blum integer is the product of two primes which are both congruent 3 modulo 4).

Theorem (Bonet, Domingo, Gavaldà, Maciel, Pitassi 04)

Bounded-depth Frege systems are not weakly automatizable under cryptographic assumptions.

Automatizability of Resolution

Theorem (Beame, Karp, Pitassi, Saks 02)

Tree-like Resolution is automatizable in quasi-polynomial time.
(Quasi-polynomial time $= n^{O(\log n)}$*)*

Theorem (Alekhnovich & Razborov 01,
Eickmeyer, Grohe & Grübner 08)

Resolution is not automatizable unless $FPT = W[P]$.

Open problem

Is Resolution weakly automatizable?

Motivations in Proof Complexity

Major motivations

- Separation of complexity classes
- Satisfiability algorithms (SAT-Solver)
- Proof Search – Automatizability
- Relations to bounded arithmetic
- Proving lower bounds is very challenging and interesting in its own right

Bounded Arithmetic

- first-order arithmetic theories
- weak subsystems of Peano arithmetic
- axiomatized by
  - a number of basic axioms describing the interplay of $+, \cdot, \leq, 0, 1, \ldots$ and
  - some controlled amount of induction

Most important examples

- $I\Delta_0$ (induction for all bounded formulas)
- PV (formalizes poly-time computations) [Cook 75]
- $S_2^1 \subseteq T_2^1 \subseteq S_2^2 \subseteq T_2^2 \subseteq \cdots \subseteq S_2 = T_2$ [Buss 86]

Propositional Translations

Bounded formulas

- A bounded universal quantifier is of the form $(\forall x)(|x| \le t \to \dots)$ with some term t.
- Π_1^b-formulas only contain bounded universal quantifiers.
- Π_1^b-formulas describe coNP-sets.

From first-order to propositional formulas

A Π_1^b-formula $\varphi(x)$ can be translated into a sequence of propositional formulas $\|\varphi\|^n$ such that

- $\|\varphi\|^n$ has polynomial size in n;
- for each $a \in \mathbb{N}$, $\mathbb{N} \models \varphi(a)$ iff $\|\varphi\|^{|a|}(a) \in \mathit{TAUT}$.

Bounded Arithmetic and Propositional Proof Systems

The correspondence

An arithmetic theory T corresponds to a propositional proof system P if the following conditions are satisfied:

- For $\varphi \in \Pi_1^b$, if $T \vdash (\forall x)\varphi$, then there are poly-size P-proofs of $\|\varphi\|^n$.
- T proves the correctness of P, i.e. $T \vdash RFN(P)$.

Example

S_2^1 corresponds to extended Frege EF.

This correspondence can be applied to

- construct short P-proofs (upper bounds);
- show lower bounds to the proof size for P [Ajtai 94];
- show simulations between proof systems.

Uniform vs. Non-uniform Concepts

	Complexity	Logic
uniform	P, NP, coNP, ... Turing machines	arithmetic theories Π_1^b formulas
non-uniform	AC^0, P/poly, NP/poly, ... Boolean circuits	proof systems propositional formulas

Our experience

Lower bounds in the non-uniform models are very hard.

Motivations in Proof Complexity

Major motivations

- Separation of complexity classes
- Satisfiability algorithms (SAT-Solver)
- Proof Search – Automatizability
- Relations to bounded arithmetic
- Proving lower bounds is very challenging and interesting in its own right

Summary

Proof Complexity

- is at the intersection of logic and complexity.
- uses concepts and intuition from algebra, geometry, ...

Main Objective

study lengths of proofs

Connections to other areas

- Separation of complexity classes
- Analysis of SAT algorithms
- Proof search – Automatizability
- First-Order Logic – Bounded Arithmetic
- Proving lower bounds is hard!