Introduction to Proof Complexity

Nicola Galesi
Dipartimento di Informatica
Università degli Studi di Roma "La Sapienza"

2009 August 15 – 16
NoNa Summer School
St Petersburg

Organization

Informal introduction and Overview

Informal introductions to P,NP,co-NP and themes from and relationships with Proof complexity

First Steps in Proof Complexity

Complexity theory and motivating problems

Proof systems (PS) and polynomially bounded PS

Polynomial Simulation between proof systems

Encoding of combinatorial principles as boolean formulae

The main problem of Proof Complexity

Organization

Resolution proof system

- Definitions
- Soundness and Completeness
- Treelike Resolution (TLR) and daglike Resolution (DLR)
- Complexity measure for Resolution: size, width and space.
- Examples
- Interpolation
- Davis Putnam (DPLL) Algorithm for SAT and TLR
- Search Problems and refutations in Resolution

Organization

Exponential Separation between TLR e DLR

- History and evolution of the results for TLR
- Prover-Delayer game: A two players game to model lower bounds for TLR
- Pebbling Games on DAG
- Peb(G): UNSAT formula encoding pebbling games on dag
- Poly size refutations in DLR for Peb(G)
- Exponential lower bounds for PEB(G) in TLR
- Open problems

Organization

Exponential lower bounds for DLR.

- From Resolution to Monotone Resolution. Polynomial equivalence wrt PHP.
- The Beame-Pitassi method: PHP requires exponential refutations in DLR.
- Synthesis of BP method: The width method of Ben-Sasson-Wigderson
- Application of width method - I : Random systems of linear equations
- Application of width method - II : Tseitin formulae.
- The “strange case” of Weak PHP: pseudowidth

Organization

Other measures and methods for Resolution

- Space complexity in Resolution: results
- Combinatorial characterization of width and relation with space
- Efficient Interpolation for Resolution
- DLR has Efficient Interpolation
- Automatizability and Efficient Interpolation
- DLR is not automatizable unless W[P] in RP
- Open Problems

Organization

Other proof systems and Open Problems

- Res[k]: Resolution on k-DNF
- Geometric Systems: Cutting Planes e Lovasz-Schriver
- Logic systems: Frege and bounded depth Frege
- Algebraic system: Polynomial Calculus and Hilbert Nullstellensatz
- Open problems: new ideas

First steps in Proof Complexity

Complexity theory (P,NP,co-NP)

Σ an alphabet. A decision problem is a subset of Σ^*.

Def. [P] A decision problem Q is in P if there is a TM M s.t.

- $\forall x \in \Sigma^*$: $x \in Q$ iff M accepts x
- For some polynomial p(), on inputs x, M halts within p(|x|) steps.

Def. [NP] A decision problem Q is in NP if there a relation R(*,*) in P and a polynomial p(), s.t.

$$\forall x \in \Sigma^* (x \in Q \text{ iff } \exists\, w: |w| \leq p(|x|) \text{ and } R(x,w))$$

Def. [co-NP] A decision problem Q is in co-NP if its complement is in NP

Complexity theory (SAT e TAUT)

SAT = {boolean frm A: A is satisfiable}

SAT $\in$ NP [… have a look]

SAT è NP-hard ($\forall$ Q $\in$ NP there is a many-one reduction

f:Q->SAT, f in FP) [have a look]

SAT is NP-complete

TAUT = {boolean frm A: A is tautology}

TAUT is co-NP complete

Proof.

(1) $\neg$TAUT $\in$ NP.

F $\in$ $\neg$TAUT iff F $\notin$ TAUT

$\exists$ assignment σ s.t. σ(F)=F [NP def]

Complexity theory (SAT e TAUT)

(2) ¬TAUT is NP-hard.

we give a poly time many-one reduction of SAT to ¬TAUT

$F \in$ SAT iff $\neg F \notin$ TAUT

iff $\neg F \in \neg$ TAUT

The reduction is then $F \to \neg F$

Big questions: Does NP = P ?, Does NP = co-NP ?

Exercise: 1. Prove that P=NP, implies NP=co-NP

2. Prove that UNSAT = TAUT

Proof Systems

Classical Definition

A propositional proof system is a surjective function f computable in polynomial time f: $\Sigma^* \rightarrow TAUT$.

Let $A \in TAUT$. Let P be a string.

If f(P) =A, then we interpret P as a PROOF of A.

f() is then a polytime function (in |P|) that efficiently verifies that P is in fact a proof of A.

the length of P, |P| (the size of the proof) has to be considered as a measure of the size of the tautology |A|

Proof Systems

Modern Definition

A proof system for a language L (TAUT) is a polynomial time algorithm (verifier) V such that

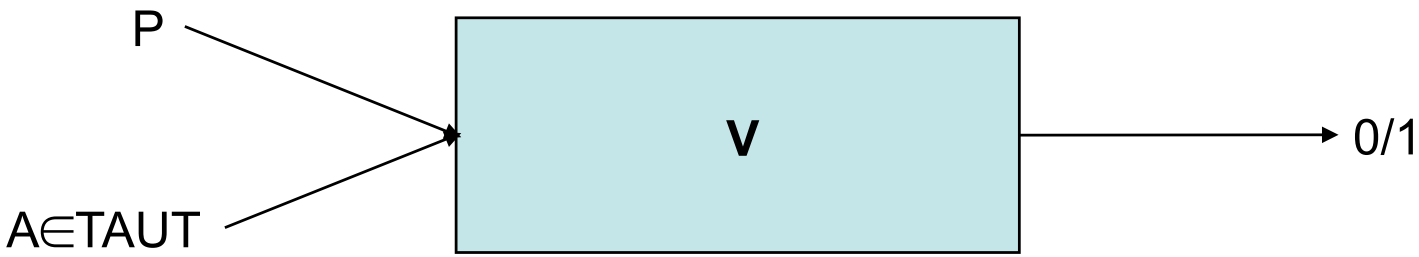
$\forall$A: (A $\in$ L iff $\exists$ a string P (a proof) s.t. V accepts (A,P))

we think of

- P as a proof that A is in L
- V as a verifier of the correctness of the proof

A propositional proof system is a proof system for TAUT .

Proof Systems



Intuition

Take your favorite inference system. You can think of V as an algorithm that efficently checks that the proof P terminates in A and follows from applications of the rules of your system.

Complexity

The main point is how big is |P| as a function of |A| ?

This affects the efficiency of V, as well.

Super Proof Systems

A proof system F is SUPER (p-bounded) if there is a polynomially bounded size proof for every tautology:

$\forall A \in TAUT\ \exists\ P : |P| \leq p(|A|)$ s.t. $f(P)=A$ ($V(P,A)=1$) for some polynomial p().

Thm [Cook-Rekhow,71] There exists a super proof system iff NP=co-NP.

Proof.

($\Rightarrow$) f is super

$\Rightarrow \forall A \in TAUT\ \exists\ x : |x| \leq p(|A|)$ s.t. $f(x)=A$

$\Rightarrow$ TAUT $\in$ NP

$\Rightarrow$ NP = co-NP [Exercise 3]

Super Proof Systems

($\Leftarrow$) Assume NP=co-NP.

$\Rightarrow$ TAUT $\in$ NP

$\Rightarrow$ there is a polynomial p() and a relation R(,) s.t.

$\forall x$ ($x \in$ TAUT iff $\exists$ w: $|w| \leq p(|x|)$ and $R(x,w)$).

Define f as follows:

$f(v) = x$ if $v = \langle x,w \rangle$ and $R(x,w)$

$f(v) = p \vee \neg p$ ow.

Corollary. If there is no super proof system, then NP $\neq$ P.

Exercise 3. TAUT$\in$ NP $\Leftrightarrow$ NP = co-NP

Exercise 4. f is super.

Main questions in Proof Complexity

By Cook-Reckhow Theorem, to prove, NP ≠ co-NP we have to prove that

there is no super proof systems

Assume we have a proof systems S. What exactly mean prove that S is not super ?

Find a tautology A ∈ TAUT and prove that the size of all the proofs of A in S are not bounded by any polynomial in the size of the formula A to be proved. Then it suffices to prove that it does hold for the shortest proof of A in S

Main questions in Proof Complexity

S is not super

There exists $A \in TAUT$ such that for all polynomials p and for all proof P of A in S, $|P|>p(|A|)$.

Stronger.

There exists A TAUT such that the shortest proof P of A in S is of size $|P|>\exp(|A|^{\varepsilon})$, with $\varepsilon>0$.

Notation and Positions

Usually, instead of a single tautology A we speak of families of (uniform) tautologies $\{F_n\}_{n\in N}$, where n is some parameter coming from the encoding. In general the size of F_n is polynomial in n, and hence wrt to proving a system is not super we usually use n instead of $|F_n|$.

Comparing strength of Proof systems

Question

Assume we have two proof systems S1 and S2. How we can say that "S1 is stronger than S2"

Answer: Find a family of tautologies F_n such that:

1. There are polynomial size proofs of F_n in S1
2. The shortest proof of F_n in S2 is not polynomially bounded in n (is exponential in n)

We say that **S2 is exponentially separated from S1**

Comparing strength of Proof systems

Question

Let us given two proof systems S1 and S2 defined over the same language. When can we say that if S1 is not super, then also S2 is not super ?

Answer: **S2 Polynomially simulates S1** (S2≥S1)

Iff there is a P-time computable function g:{0,1}*→{0,1}*, s.t. for all w in {0,1}* S1(w))=S2(g(w)). In other words

$$S1 \xrightarrow[P1]{} A, \text{ then } S2 \xrightarrow[P2]{} A, \quad |\mathrm{P2}| = \mathrm{p}(|\mathrm{P1}|)$$

Theorem.[Exercise 5]

If S1 is not super and S2≥S1, then S2 is not super

Separations and Incomparability of Proof systems

Defn

Two proof systems S1 and S2 are exponentially separated if there exists a family of formulas F over n variables such that

1. F admits polynomial size $O(n^{O(1)})$ proofs in S1

2. The shortest proof of F in S2 is exponentially long in n $\exp(n^{\varepsilon})$.

Defn

Two proof systems S1 and S2 are incomparable if there are two families of formulae that respectively separates exponetially S1 from S2 and S2 from S1.

k-CNF k-DNF

Propositional formulas are can be transformed into normal form called CNF conjuntive normal form and DNF disjunctive normal form.

CNF Conjuctions fo Disjunctions

$$\bigwedge_{i \in D} \bigvee_{j \in R} p_{i,j}$$

DNF Disjunctions of Conjunctions

$$\bigvee_{i \in D} \bigwedge_{j \in R} p_{i,j}$$

k-CNF all clauses have <=k literals

k-DNF all terms have <= k literals

Values and assignments

Consider a k-CNF F and a partial assignment α to its variables. F[α] is the formula resulting form F after applying the following semplifications:

- Delete all clauses containing literals set to 1 by α
- Delete from all clauses the literals set to 0 by α

Consider a k-DNF F and a partial assignment α to its a variable. F[α] is the formula resulting form F after applying the following semplifications:

- Delete all terms containing literals set to 0 by α
- Delete from all terms the literals set to 1 by α

A Concrete Example: Frege Systems

Rules

$\dfrac{.\qquad\qquad .}{A \to (B \to A)}$ [Axiom Scheme: Examples] ....

$\dfrac{.\qquad\qquad .}{A \to (A \lor B)}$

…………..

$\dfrac{A \quad A \to B}{B}$ [Modus Ponens]

Frege Systems

Proofs

A proof of a Tautology A in a Frege Systems is a sequence of fomulas

A1,A2,A3………,Am

such that

1. Am is exactly A
2. Each Ai is obtained either as instance of an axiom scheme, or from two previous formulas in the sequence by using (an instantiation of) the MP rule

Example

A->A

Complexity Measures in Frege Systems

Size of Proof

Total number of symbols used in the proof

P:= A1,...,Am, then |P|= |A1|+...+|Am|

Length of a Proof

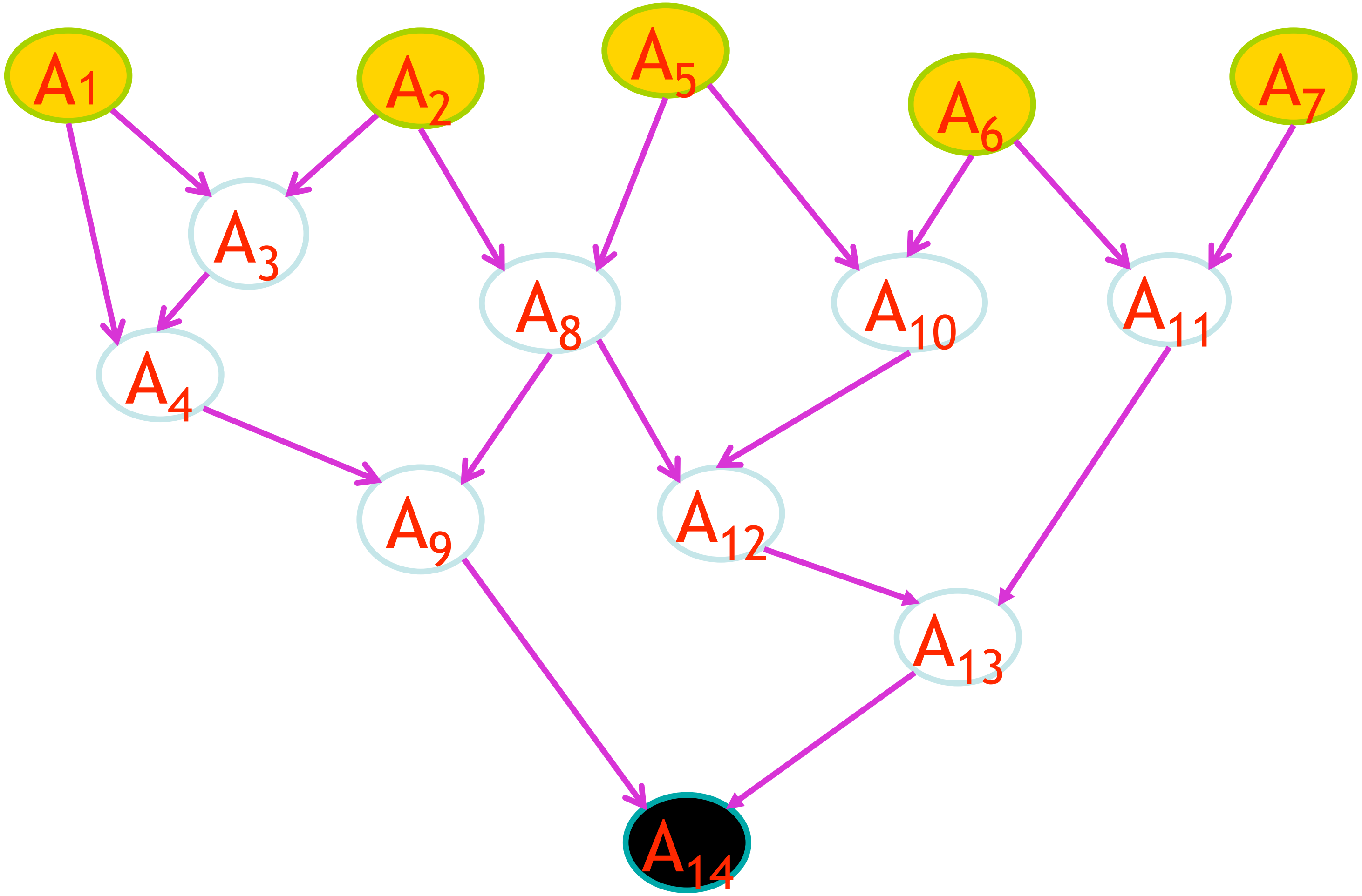
Number of lines of the proof

P:= A1,...,Am, then length of P = m

Thm[Cook-Rekchow] If a tautology A has a Frege proof of m lines, then A has a Frege proof of p(m) symbols, for some polynomial p().

Cor. No matter length or size wrt to prove Frege is NOT super

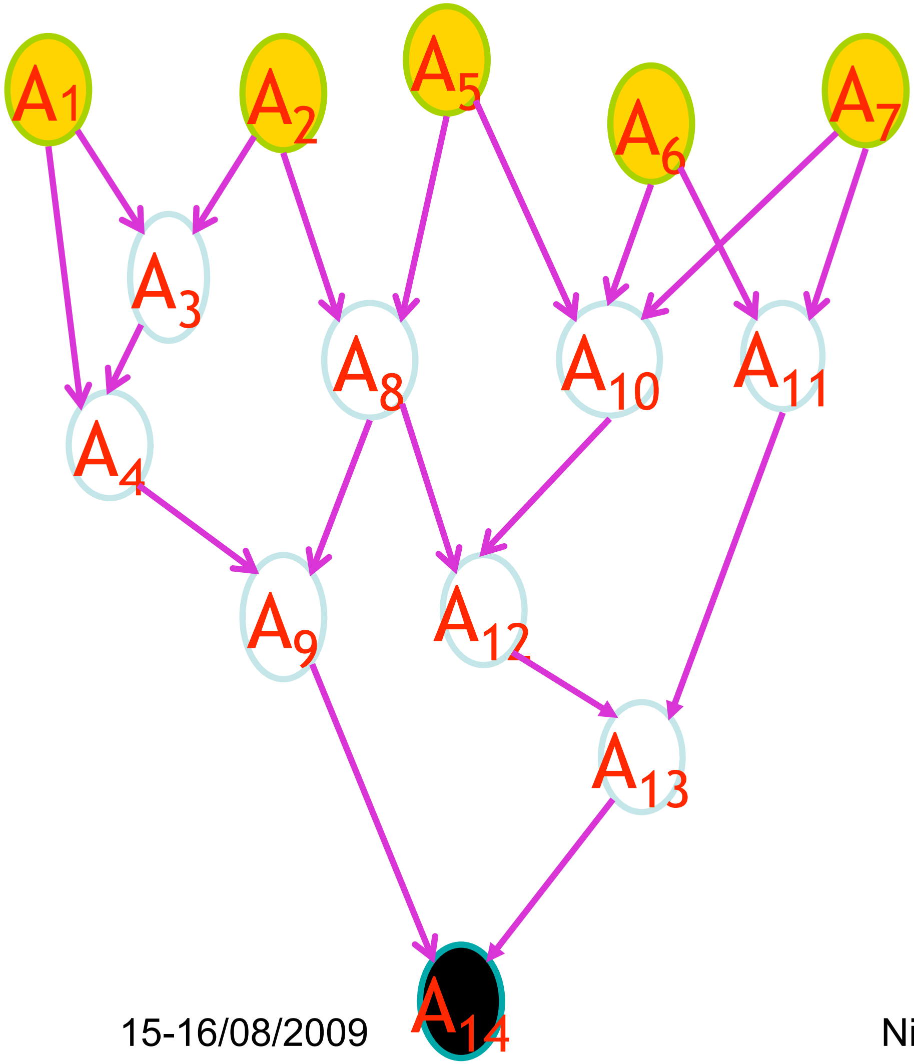
Proof Graph



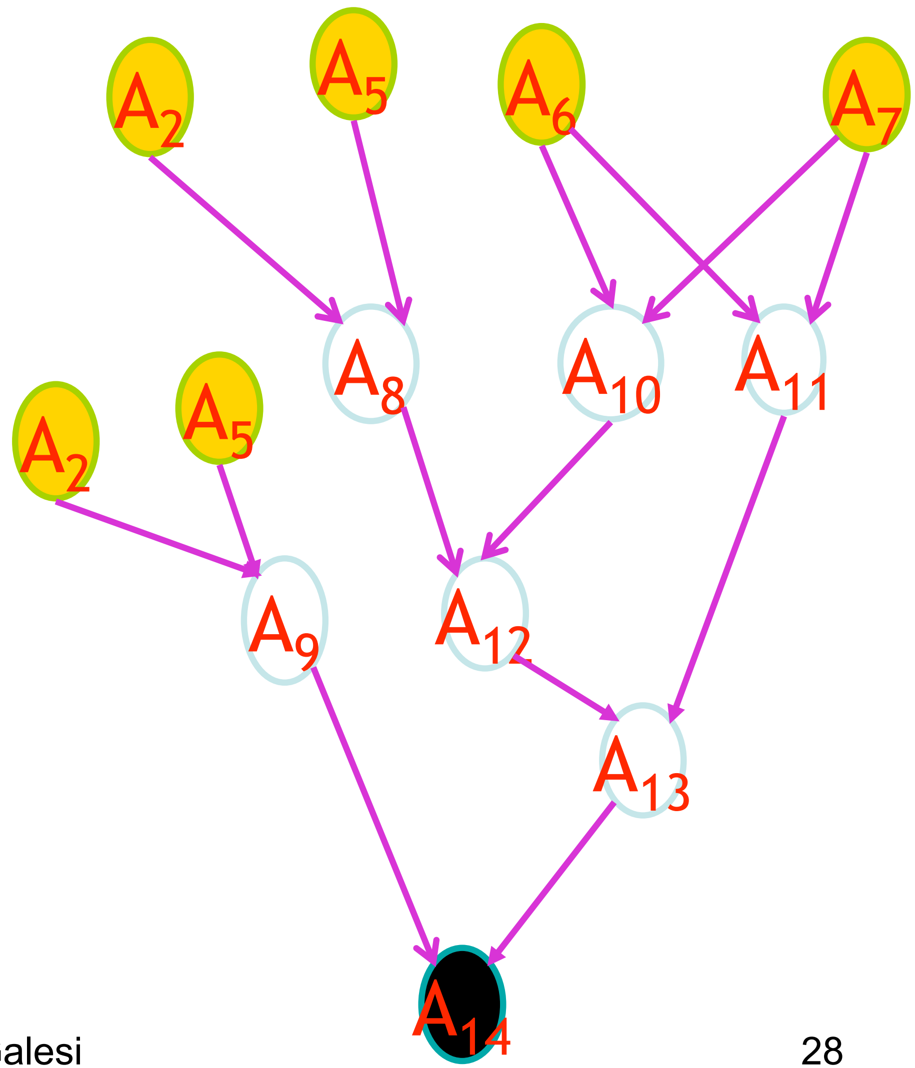
Length of Proof = number of nodes in the graph

Daglike and Treelike Proofs

daglike



treelike



Have tree and daglike proofs the same strenght ?

Question

Let S be a proof system. Is it true that treelike S polynomially simulates daglike S ?

Answer

It depends on the proof system.

1. For Frege systems this is true [Krajicek, next slides]
2. for Resolution it is false [next Chapter]

treelike and daglike Frege

Thm [Krajicek]

Treelike Frege system, polynomially simulates daglike Frege.

Proof

Let A1,.....,Am be a proof in daglike Frege.

let Bi =A1∧A2∧....∧Ai, for i=1,...,m

We get separated treelike proofs of the following formulas

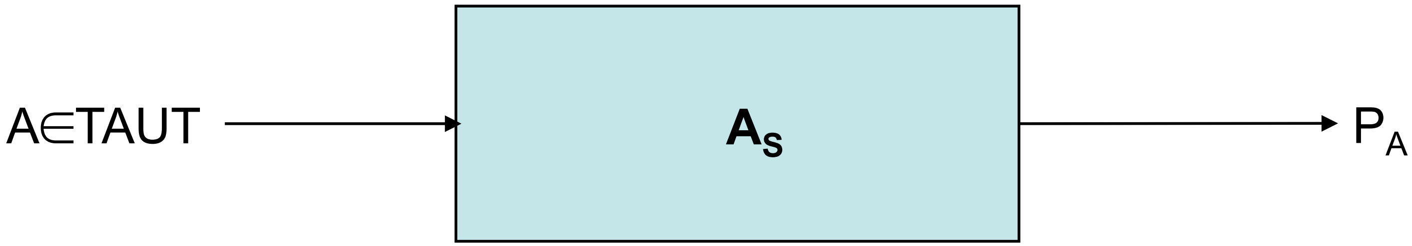
- B1
- Bi → B(i+1) for all i=1,...,m-1 [Exercise 6]

m applications of the Modus ponens gives a treelike proof of Am.

Properties of Proof Systems
Automatizability

Automatizability [Impagliazzo; Bonet,Pitassi,Raz]

A proof system S is automatizable if there is an algorithm A_S which in input a tautology A gives a proof in S of the A in time polynomially bounded in the shortest proof of A in S



Motivation

Devise algorithms for proof search in proof systems independently from the property to be p-bounded

Interpolation: general setting

Let U and V two disjoint NP-sets (as subset of $\{0,1\}^*$). By Cook SAT NP-completeness theorem we know that there exists two sequence of formulas $A_n(\mathbf{p},\mathbf{q})$ and $B_n(\mathbf{p},\mathbf{r})$ s.t.

1. The size of A and B are polynomial in n

2. $U_n = U \cap \{0,1\}^* = \{\boldsymbol{\varepsilon} \in \{0,1\}^* : \exists \alpha\ A_n(\boldsymbol{\varepsilon},\boldsymbol{\alpha})$ true$\}$

3. $V_n = V \cap \{0,1\}^* = \{\boldsymbol{\varepsilon} \in \{0,1\}^* : \exists \beta\ B_n(\boldsymbol{\varepsilon},\boldsymbol{\beta})$ true$\}$

Intepolation: general setting

$U \cap V = \emptyset$ is equivalent to say that $A_n \rightarrow \neg B_n$ are tautologies.

By Craig's interpolation theorem exist I_n s.t.

$A_n \rightarrow I_n$ and $I_n \rightarrow \neg B_n$

This maens that the set

$$W = \bigcup_n \{\boldsymbol{\varepsilon} \in \{0,1\}^* : I_n(\boldsymbol{\varepsilon}) \text{ holds}\}$$

Separates U from V. i.e.

$U \subseteq W$ and $W \cap V = \emptyset$

Intepolation and complexity

Hence a lower bound on the complexity of the interpolant is a lower bound on the complexity of separating two disjoint NP-Sets.

Thm[Mundici] If W is computable by a polynomial size boolean circuit, then NP$\cap$co-NP$\subseteq$P/poly.

Feasible Intepolation

[Krajicek]
For a given proof system P, try to estimate the circuit size of an interpolant of an implication in terms of the size of shortest proof of the implication in P.

[Pudlak] Resolution admits feasible interpolation [Lecture III]

[Pudlak,Krajicek] Frege systems does not have feasible interpolation unless RSA cryptographic scheme is breakable [Lecture III]

Propositional Encoding

Relations

Assume to have a binary relations R(i,j) over some domain D.
I can think of modelling R through boolean variables $x_{i,j}$ such that $x_{i,j}$ = TRUE iff R(i,j) does hold.

Encoding statements over R

$\forall i \in D \; \exists j \in D$ R(i,j) is encoded by

$$\bigwedge_{i \in D} \bigvee_{j \in D} x_{i,j}$$

$\exists k \in D \; \forall i, j \in D$ R(i,k) $\rightarrow$ R(j,k)

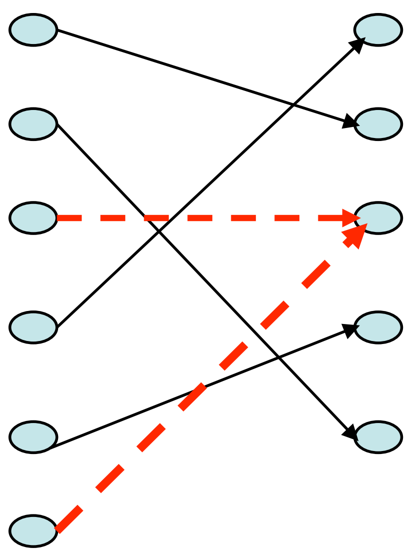
$$\bigvee_{k \in D} \bigwedge_{i,j \in D} \neg x_{i,k} \vee x_{j,k}$$

Encoding of Combinatorial principles

PigeonHole Principle

- There is no 1-1 function from [n+1] to [n].
- If a total mapping f maps [n+1] to [n], then there will be two elements in the dom(f) mapped to the same element in Rng(f)

$P_{i,j}$ = "pigeon i mapped by f into hole j"



Encoding of Combinatorial principles

PigeonHole Principle

If a total mapping f maps [n+1] to [n], then there are two elements in the dom(f) mapped to the same element in Rng(f)

if $\forall i \in [n+1]\ \exists\ j \in [n]\ f(i)=j \rightarrow \exists\ i \neq j \in [n+1]\ \exists k \in [n]\ (f(i)=k \wedge f(j)=k)$

$$PHP_n^{n+1} =_{def} \bigwedge_{i \in [n+1]} \bigvee_{j \in [n]} p_{i,j} \rightarrow \bigvee_{i,j \in [n+1]} \bigvee_{k \in [n]} (p_{i,k} \wedge p_{j,k})$$

Other PHP Statements [Exercise 7]

- Functional-PHP: Every function from [n+1] to [n] is non injective., i.e. Every pigeon is mapped to exactly one hole
- Onto-PHP: Functional-PHP + every hole gets a pigeon

Encoding of Combinatorial principles

Weak PigeonHole Principle

If a total mapping f maps [m] to [n] $m>n$, then there are two elements in the dom(f) mapped to the same element in Rng(f)

Complexity of Weak PHP

WeakPHP is "more" true than PHP. We will see that approximately for $m = \Omega(n^2/\log n)$ the PHP starts to behave differently for PHP . But the situation is different for different proof system and this represents an imporant questions in different proof systems

Encoding of Combinatorial principles

Negation of the PigeonHole Principle as CNF (UNSAT)

$$\neg PHP_n^{n+1} =_{def} \begin{cases} \bigwedge_{i \in [n+1]} (p_{i,1} \vee \cdots \vee p_{i,n}) \\ \bigwedge_{i \neq j \in [n+1]} \bigwedge_{k \in [n]} (\neg p_{i,k} \vee \neg p_{j,k}) \end{cases}$$

Encoding of Combinatorial principles

Linear Ordering Principle

Every linearly ordered finite set has a minimal element.

Let D a finite set linearly ordered. E.g. D=[n].

$x_{i,j}$=TRUE iff $i<j$ in the linear order

- If [n] is linearly ordered then there exists a minimal element in [n] ($\forall j \in [n]: i<j$)
- [n] linearly ordered iff
  - antisymmetry ($i<j \rightarrow \neg j<i$)
  - transitivity ($i<j \wedge j<k \rightarrow i<k$)

Encoding of Combinatorial principles

Linear Ordering Principle

$$LOP_n =_{def} \bigwedge_{\substack{i,j\in[n]\\ i\neq j}} (x_{i,j} \rightarrow \neg x_{j,i}) \wedge \bigwedge_{\substack{i,j,k\in[n]\\ i\neq j\neq k\neq i}} (x_{i,j} \wedge x_{j,k} \rightarrow x_{i,k}) \rightarrow \bigvee_{i\in[n]} \bigwedge_{j\in[n]} x_{i,j}$$

Negation of Linear Ordering Principle (UNSAT)

A finite set is linearly ordered but no element is minimal

$$\neg LOP_n =_{def} \begin{cases} \bigwedge_{\substack{i,j\in[n]\\ i\neq j}} (\neg x_{i,j} \vee \neg x_{j,i}) \\ \bigwedge_{\substack{i,j,k\in[n]\\ i\neq j\neq k\neq i}} (\neg x_{i,j} \vee \neg x_{j,k} \vee x_{i,k}) \\ \bigwedge_{i\in[n]} \bigvee_{j\in[n]} x_{j,i} \end{cases}$$

Encoding of Combinatorial principles

Tseitin Principle - Odd Charged Graph

The sum along nodes of the edges of a simple connected graph is even.

Encoding

Let G =(V,E) be a connected graph. Let m:V→{0,1} a labelling of the nodes of V s,t. $\sum_{v \in V} m(v) \equiv 1 \pmod 2$

Assign a variable x_e to each edge e in G.

For a node v in V $PARITY(v) = \bigoplus_{v \in e} x_e \equiv m(v) \pmod 2$

$$T(G,m) =_{def} \bigwedge_{v \in V} PARITY(v)$$

Random Formulae in CNF

Experiment:

Choose uniformly and independently m clauses with k variables from the space of all possible such clauses over n variables

$$(\neg x_4 \vee \neg x_2 \vee x_6) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_4 \vee x_5)$$

Fact

Let D=m/n be **density**. There exists a threshold value r* s.t.:

- **if r < r*:** F (n,m) è SAT w.h.p
- **If r > r*:** F(n,m) è UNSAT w.h.p.

Complexity of Random k-CNF

UNSAT Proofs:

Take a random k-CNF F with a density which w.h.p. guarentees UNSAT of F

Study complexity of Proofs for such a formula

Density m/n.

It is not difficult to see that the more the density grown over the threshold the easier will be to verify the UNSAT of F(n,m).
Hardness results hold only for weak proof systems
and for (almost always) constant densities.