Logic in Computer Science

Overview

1. basic concepts
2. propositional Resolution system, feasible interpolation, SAT solvers,...
3. 3 main formalizations of proofs: Hilbert/Frege style, Gentzen's sequent calculus, Natural Deduction
4. the Cut-elimination Theorem and its applications
5. <u>Herbrand's Theorem</u>
6. bounds on the size of cut-free roofs and Herbrand's disjunction
7. first-order Resolution and automated theorem proving
8. selfreference, Gödel's theorems, Rosser's theorem,...
9. Natural Deduction and lambda calculus
10. Peano Arithmetic and its fragments
11. Bounded Arithmetic and the Polynomial Hierarchy

References

1. S. Buss, An introduction to proof theory, in Handbook of Proof Theory, edited by S. Buss, Elsevier North-Holland, 1998, pp 1-78. http://math.ucsd.edu/ sbuss/ResearchWeb/handbookI/index.html
2. C.L. Chang and R. C.-T. Lee: Symbolic Logic and Mechanical Theorem Proving, Academic Press, 1973
3. P. Pudlák: The lengths of proofs, in Handbook of Proof Theory, pp.547-637 http://www.math.cas.cz/∼pudlak/length.pdf
4. P. Hájek, P. Pudlák: Metamathematics of first order arithmetic, Springer-Verlag/ASL Perspectives in Logic, 1998 Kniha je ke stažení přes Project Euclid http://www.aslonline.org/books-perspectives cup Springer.html
5. C. Smorynski: The incompleteness theorems. Handbook of Mathematical Logic, J. Barwise Editor, 1977.
6. A.S. Troelstra and H. Schwichtenberg: Basic Proof Theory, Cambridge University Press

Of general interest:

- P. Pudlák, Logical Foundations of Mathematics and Computational Complexity, Springer, 2017

Acknowledgment

This course is partly based on S. Buss's chapter and in some slides I copied parts of his work.

1st lesson
basic concepts

- syntax — proof theory — Gotlob Frege, David Hilbert
- semantics — model theory — Alfred Tarski
- soundness and completeness of proof systems w.r.t. semantics (or vice versa)

1st lesson
basic concepts

- syntax — proof theory — Gotlob Frege, David Hilbert
- semantics — model theory — Alfred Tarski
- soundness and completeness of proof systems w.r.t. semantics (or vice versa)

computational aspects

- how big a proof of a given theorem must be
- how difficult it is to find it
- how strong a theory/proof system we need to prove the theorem

propositional logic

- variables, connectives, formulas, Boolean circuits
- satisfiability (=consistency)
- compactness of propositional logic

propositional logic

- variables, connectives, formulas, Boolean circuits
- satisfiability (=consistency)
- compactness of propositional logic

Exercise

1. *prove compactness of propositional logic*
2. *prove:*
   Let G be an infinite graph. If for every finite $H \subseteq G$, $\chi(H) \leq n$, then $\chi(G) \leq n$.

Resolution

Motivation — SAT solvers

Resolution

Motivation — SAT solvers

"...on 3 May 2016, Marijn Heule, Oliver Kullmann of Swansea University, UK, and Victor Marek of the University of Kentucky in Lexington have now shown that there are many allowable ways to colour the integers up to 7,824 — but when you reach 7,825, it is impossible for every Pythagorean triple to be multicoloured...

Theorem

The Ramsey number of Pythagorean triples, the least n such that for every 2-coloring there is a monochromatic triple, is 7,825.

Three computer scientists have announced the largest-ever mathematics proof: a file that comes in at a whopping 200 terabytes, roughly equivalent to all the digitized text held by the US Library of Congress.

. . .

The researchers have created a 68-gigabyte compressed version of their solution — which would allow anyone with about 30,000 hours of spare processor time to download, reconstruct and verify it — but a human could never hope to read through it. . . "

Three computer scientists have announced the largest-ever mathematics proof: a file that comes in at a whopping 200 terabytes, roughly equivalent to all the digitized text held by the US Library of Congress.

. . .

The researchers have created a 68-gigabyte compressed version of their solution — which would allow anyone with about 30,000 hours of spare processor time to download, reconstruct and verify it — but a human could never hope to read through it. . . "

Exercise

Write a CNF formula such that a satsifying assignment of it is the coloring they constructed.

complexity of resolution proofs

- If $NP \neq coNP$, then (if and only if) for every proof system there are sequences of tautologies without polynomial size proof.

complexity of resolution proofs

- If $NP \neq coNP$, then (if and only if) for every proof system there are sequences of tautologies without polynomial size proof.

The Pigeon-Hole Principle tautology

$\neg PHP_n^{n+1}$ says that $n+1$ pigeons map 1-1 to n holes.

complexity of resolution proofs

- If $NP \neq coNP$, then (if and only if) for every proof system there are sequences of tautologies without polynomial size proof.

The Pigeon-Hole Principle tautology

$\neg PHP_n^{n+1}$ says that $n+1$ pigeons map 1-1 to n holes.

The CNF with the clauses

- $\bigvee_{j \in [n]} p_{ij}$ for all $i \in [n+1]$
- $\neg p_{ij} \vee \neg p_{kj}$ for all $k \neq i$, $j \in [n]$
- $\neg p_{ij} \vee \neg p_{ik}$ for all $k \neq j$, $i \in [n+1]$
- $\bigvee_{i \in [n]} p_{ij}$ for all $j \in [n]$

complexity of resolution proofs

- If $NP \neq coNP$, then (if and only if) for every proof system there are sequences of tautologies without polynomial size proof.

The Pigeon-Hole Principle tautology

$\neg PHP_n^{n+1}$ says that $n+1$ pigeons map 1-1 to n holes.

The CNF with the clauses

- $\bigvee_{j \in [n]} p_{ij}$ for all $i \in [n+1]$
- $\neg p_{ij} \vee \neg p_{kj}$ for all $k \neq i$, $j \in [n]$
- $\neg p_{ij} \vee \neg p_{ik}$ for all $k \neq j$, $i \in [n+1]$
- $\bigvee_{i \in [n]} p_{ij}$ for all $j \in [n]$

Theorem (A. Haken, 1985)

Every resolution refutation of PHP_n^{n+1} has exponential size.

complexity of resolution proofs

- If $NP \neq coNP$, then (if and only if) for every proof system there are sequences of tautologies without polynomial size proof.

The Pigeon-Hole Principle tautology

$\neg PHP_n^{n+1}$ says that $n+1$ pigeons map 1-1 to n holes.

The CNF with the clauses

- $\bigvee_{j \in [n]} p_{ij}$ for all $i \in [n+1]$
- $\neg p_{ij} \vee \neg p_{kj}$ for all $k \neq i$, $j \in [n]$
- $\neg p_{ij} \vee \neg p_{ik}$ for all $k \neq j$, $i \in [n+1]$
- $\bigvee_{i \in [n]} p_{ij}$ for all $j \in [n]$

Theorem (A. Haken, 1985)

Every resolution refutation of PHP_n^{n+1} has exponential size.

Try your favorite SAT solver.

A lower bound on tree-like resolution

Theorem

If a resolution refutation of $\neg PHP_n^{n+1}$ *has a form of a tree, then it has size* $> (\frac{3}{2})^{n-1}$.

A lower bound on tree-like resolution

Theorem

If a resolution refutation of $\neg PHP_n^{n+1}$ *has a form of a tree, then it has size* $> (\frac{3}{2})^{n-1}$.

Lemma

Let G *be a binary tree. Then there is an edge that splits* G *into two components* K_1, K_2 *such that*

$$\frac{1}{3}|G| \leq |K_1|, |K_2| \leq \frac{2}{3}|G|.$$

Proof - exercise.

A lower bound on tree-like resolution

Theorem

If a resolution refutation of $\neg PHP_n^{n+1}$ has a form of a tree, then it has size $> (\frac{3}{2})^{n-1}$.

Lemma

Let G be a binary tree. Then there is an edge that splits G into two components K_1, K_2 such that

$$\frac{1}{3}|G| \leq |K_1|, |K_2| \leq \frac{2}{3}|G|.$$

Proof - exercise.

Proof-idea: Arguing by contradiction, we assume that we have a tree-like refutation Π of size $\leq (\frac{3}{2})^{n-1}$. With suitable partial truth assignments to evaluate clauses in a refutation, we will apply Lemma as "binary search" to find an application of the resolution rule in which our assignment satisfies premises, but falsifies conclusion.

Proof

Let D be the set of pigeons $|D| = n + 1$, and R the set of holes $|R| = n$. Thus we have p_{ij} with $i \in D$, $j \in R$.

For a partial matching $M \subseteq D \times R$, we define a partial assignment $a(M)$ such that

- if $(i, j) \in M$, then $p_{ij} \to 1$,
- if $(i, j) \notin M$, and $i \in D(M)$ or $j \in R(M)$, then $p_{ij} \to 0$,
- otherwise $p_{ij} \to *$ (not defined).

Lemma

1. *No $a(M)$ falsifies any clause of $\neg PHP_n^{n+1}$;*
2. *If $|M| < n$ and $a(M)$ does not falsify a clause C, then there exists an extension $M \subseteq M'$, $|M'| = |M| + 1$ such that $a(M')$ satisfies C.*

Construct a sequence of tree-like resolution proofs, clauses and matchings as long as possible

- $\Pi_0 = \Pi, \Pi_1, \Pi_2 \dots$
- $C_0 = \perp, C_1, C_2 \dots$
- $M_0 = \emptyset, M_1, M_2 \dots$

such that

- Π_i is a proof of C_i from $\neg PHP_n^{n+1}$ and some clauses that are satisfied by $a(M_i)$,
- C_i is falsified by $a(M_i)$,
- $|M_{i+1}| \leq |M_i|$ and $|\Pi_{i+1}| \leq \frac{2}{3}|\Pi_i| + 1$.

Construct a sequence of tree-like resolution proofs, clauses and matchings as long as possible

- $\Pi_0 = \Pi, \Pi_1, \Pi_2 \ldots$
- $C_0 = \bot, C_1, C_2 \ldots$
- $M_0 = \emptyset, M_1, M_2 \ldots$

such that

- Π_i is a proof of C_i from $\neg PHP_n^{n+1}$ and some clauses that are satisfied by $a(M_i)$,
- C_i is falsified by $a(M_i)$,
- $|M_{i+1}| \leq |M_i|$ and $|\Pi_{i+1}| \leq \frac{2}{3}|\Pi_i| + 1$.

Construction. Given Π_i, C_i, M_i, let e an edge that splits the proof into two parts $\leq \frac{2}{3}|\Pi_i|$ and let C be the clause above e. Then

1. if C is falsified by $a(M_i)$, take the subproof above e as Π_{i+1} and set $C_{i+1} := C$, $M_{i+1} := M_i$;
2. otherwise let Π_{i+1} be the other part of Π_i together with clause C, put $C_{i+1} := C_i$, and extend M_i so that C is satisfied by $a(M_{i+1})$.

After k steps, we have

$$|\Pi_k| \leq (\dots((|\Pi|.\frac{2}{3}+1)\frac{2}{3}+1)\dots)\frac{2}{3}+1 =$$

$$|\Pi|.(\frac{2}{3})^k + (\frac{2}{3})^k + (\frac{2}{3})^{k-1} + \dots + \frac{2}{3} + 1 <$$

$$|\Pi|.(\frac{2}{3})^k + 3$$

Since $|\Pi_0| \leq (\frac{3}{2})^{n-1}$, for some $k < n-1$, $|\Pi_k| < 4$. One can easily see that this implies $|\Pi_k| = 3$ (and then it cannot not decrease anymore). Thus it is a single instance of the resolution rule:

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

where C_k is falsified by $a(M_k)$ and D, E are either satisfied by $a(M_k)$, or can be satisfied by an extension of it.

Contradiction. $\square$

2nd lesson

Algorithms for k-SAT

k-CNF is a CNF with all clauses of width at most k

2nd lesson

Algorithms for k-SAT
k-CNF is a CNF with all clauses of width at most k

The running time for all known algorithms (deterministic and probabilistic) for k-SAT is $2^{c_k n}$ for some constants $0 < c_k < 1$.

Example. The (modification of) PPSZ algorithm: $c_3 = 0.386\ldots$, $c_4 = 0.554\ldots$, $c_5 = 0.650\ldots$, ...

2nd lesson

Algorithms for k-SAT

k-CNF is a CNF with all clauses of width at most k

The running time for all known algorithms (deterministic and probabilistic) for k-SAT is $2^{c_k n}$ for some constants $0 < c_k < 1$.

Example. The (modification of) PPSZ algorithm: $c_3 = 0.386\ldots$, $c_4 = 0.554\ldots$, $c_5 = 0.650\ldots$, ...

Exponential-Time Hypothesis (ETH)

$\forall k \exists c_k > 0$ such that every algorithm for k-SAT has running time $\geq 2^{c_k n}$.

What about 2-SAT?

What about 2-SAT?

Exercise

Find a polynomial time algorithm for 2-*SAT!*

Hint: use Resolution!

Davis - Putnam algorithm

Let $\phi(p_1, \ldots, p_n)$ be a CNF

1. assign gradually values to $p_1, p_2, \ldots$
2. after each step, reduce clauses:
   2.1 delete the clauses that are satisfied by the partial assignment
   2.2 remove falsified literals from other clauses
3. if empty clause appears, backtrack if possible, otherwise stop
4. if a unit clause $\{p_i\}$, or $\{\neg p_i\}$ appears, assign p_i so that it is satisfied
5. continue with the next variable

Davis - Putnam algorithm

Let $\phi(p_1, \ldots, p_n)$ be a CNF

1. assign gradually values to $p_1, p_2, \ldots$
2. after each step, reduce clauses:
   2.1 delete the clauses that are satisfied by the partial assignment
   2.2 remove falsified literals from other clauses
3. if empty clause appears, backtrack if possible, otherwise stop
4. if a unit clause $\{p_i\}$, or $\{\neg p_i\}$ appears, assign p_i so that it is satisfied
5. continue with the next variable

The algorithm outputs a (partial) satisfying assignment or "UNSATISFIABLE".

Davis - Putnam algorithm

Let $\phi(p_1, \dots, p_n)$ be a CNF

1. assign gradually values to $p_1, p_2, \dots$
2. after each step, reduce clauses:
   2.1 delete the clauses that are satisfied by the partial assignment
   2.2 remove falsified literals from other clauses
3. if empty clause appears, backtrack if possible, otherwise stop
4. if a unit clause $\{p_i\}$, or $\{\neg p_i\}$ appears, assign p_i so that it is satisfied
5. continue with the next variable

The algorithm outputs a (partial) satisfying assignment or "UNSATISFIABLE".

Savings are achieved by unit clauses—no need to branch on p_i.

algorithms $\leftrightarrow$ proofs

Theorem

If Davis-Putnam algorithm ran on ϕ stops without producing a satisfying assignment, then the transcript of it is essentially a Resolution refutation of ϕ. Moreover, the graph of the proof is a tree.

I.e., DP produces either a satisfying assignment, or a Resolution proof of unsatisfiability.

Proof:

Let T be the transcript. It is

- binary tree (not full, in general)
- nodes labeled by variables p_i
- branches – partial assignments
- at each leaf there is a clause falsified by the assignment of the branch

Proof:

Let T be the transcript. It is

- binary tree (not full, in general)
- nodes labeled by variables p_i
- branches – partial assignments
- at each leaf there is a clause falsified by the assignment of the branch

We will inductively, starting from leaves,

1. assign clause to each node v so that they are falsified by the partial assignment produced by the branch up to v
2. show that branching corresponds to the an application of resolution

1. assign clause to each node v so that they are falsified by the partial assignment produced by the branch up to v
2. show that branching corresponds to the an application of resolution or copying a clause from above

$$\frac{v \mapsto C \quad u \mapsto D}{w \mapsto ?} \quad p_i$$

1. assign clause to each node v so that they are falsified by the partial assignment produced by the branch up to v
2. show that branching corresponds to the an application of resolution or copying a clause from above

$$\frac{v \mapsto C \quad u \mapsto D}{w \mapsto ?} \quad p_i$$

We have

$$C[\vec{a}\,0] = 0, D[\vec{a}\,1] = 0,$$

where 0 and 1 are assignments to p_i.

1. assign clause to each node v so that they are falsified by the partial assignment produced by the branch up to v
2. show that branching corresponds to the an application of resolution or copying a clause from above

$$\frac{v \mapsto C \quad u \mapsto D}{w \mapsto ?} \quad p_i$$

We have

$$C[\vec{a}\,0] = 0, D[\vec{a}\,1] = 0,$$

where 0 and 1 are assignments to p_i. Then

- if $p_i \notin C$, then $w \mapsto C$
- if $\neg p_i \notin D$, then $w \mapsto D$
- otherwise $p_i \in C$ and $\neg p_i \in D$ and $w \mapsto C \vee D$

1. assign clause to each node v so that they are falsified by the partial assignment produced by the branch up to v
2. show that branching corresponds to the an application of resolution or copying a clause from above

$$\frac{v \mapsto C \quad u \mapsto D}{w \mapsto ?} \quad p_i$$

We have

$$C[\vec{a}\,0] = 0, D[\vec{a}\,1] = 0,$$

where 0 and 1 are assignments to p_i. Then

- if $p_i \notin C$, then $w \mapsto C$
- if $\neg p_i \notin D$, then $w \mapsto D$
- otherwise $p_i \in C$ and $\neg p_i \in D$ and $w \mapsto C \vee D$

Clearly $root \mapsto \bot$. □

PPSZ algorithm for k-SAT

1. for some $l >> k$, derive clauses of width $\leq l$ and add them to the k-CNF
2. order variables randomly (!)
3. run Davis-Putnam

PPSZ algorithm for k-SAT

1. for some $l >> k$, derive clauses of width $\leq l$ and add them to the k-CNF
2. order variables randomly (!)
3. run Davis-Putnam

Analysis: show that unit clauses must occur often, or often arbitrary values can be used when branching. (Difficult.)

PPSZ algorithm for k-SAT

1. for some $l >> k$, derive clauses of width $\leq l$ and add them to the k-CNF
2. order variables randomly (!)
3. run Davis-Putnam

Analysis: show that unit clauses must occur often, or often arbitrary values can be used when branching. (Difficult.)

Theoretically best, but not practical—SAT solvers aim at linear time.

Complexity of the resolution proof system

- some tautologies require exponential proofs (refutations)
- but also it is difficult to find short proofs, unless $P = NP$

Complexity of the resolution proof system

- some tautologies require exponential proofs (refutations)
- but also it is difficult to find short proofs, unless $P = NP$

Theorem (Atserias-Müller, 2018)

- *If $P \neq NP$, then there is no algorithm that decides if a given CNF ϕ has a refutation of length m in time polynomial in $|\phi|$ and m.*
- *Consequently, if $P \neq NP$, then there is no algorithm that given a CNF ϕ and m constructs a refutation of length m in time polynomial in $|\phi|$ and m.*

Proof idea

Proof idea

Given CNF ϕ and m, construct CNF ψ such that

1. $\phi \in SAT \mapsto R_{\phi,m}$ has a refutation of length $\leq m$,
2. $\phi \notin SAT \mapsto R_{\phi,m}$ has no refutation of length $\leq m$

Proof idea

Given CNF ϕ and *m*, construct CNF ψ such that

1. $\phi \in SAT \mapsto R_{\phi,m}$ has a refutation of length $\leq m$,
2. $\phi \notin SAT \mapsto R_{\phi,m}$ has no refutation of length $\leq m$

$R_{\phi,m} :=$ "*ϕ has a resolution refutation of length $\leq m$*"

$R_{\phi,m} :=$ "*ϕ has a resolution refutation of length $\leq m$*"

1. Formalize resolution proofs using CNFs.

$R_{\phi,m} :=$ "*ϕ has a resolution refutation of length $\leq m$*"

1. Formalize resolution proofs using CNFs.
2. - If $\phi \in SAT$, then it is not refutable (using any sound proof); in particular, there is no refutation of length $\leq m$.
   - Hence $\neg R_{\phi,m}$ is true.
   - Moreover, we can prove this fact in Resolution using a polynomial size proof. Proving $\neg R_{\phi,m}$ in Resolution means constructing a refutation of $R_{\phi,m}$.

$R_{\phi,m} :=$ "*ϕ has a resolution refutation of length $\leq m$*"

1. Formalize resolution proofs using CNFs.
2. - If $\phi \in SAT$, then it is not refutable (using any sound proof); in particular, there is no refutation of length $\leq m$.
   - Hence $\neg R_{\phi,m}$ is true.
   - Moreover, we can prove this fact in Resolution using a polynomial size proof. Proving $\neg R_{\phi,m}$ in Resolution means constructing a refutation of $R_{\phi,m}$.
3. - If $\phi \notin SAT$ then ϕ is refutable.
   - If, moreover, ϕ has a refutation of length $\leq m$, then $R_{\phi,m}$ is satisfiable, hence there is no refutation of it.
   - If ϕ does not have a refutation of length $\leq m$, there is a refutation of $R_{\phi,m}$, but one can show that there is no subexponential size refutation.

Proof idea of 2.

- take a satisfying assignment *a* of ϕ
- gradually derive clauses expressing that the clauses in the alleged refutation are satisfied
- contradiction cannot be satisfied, which is a contradiction

Proof idea of 2.

- take a satisfying assignment a of ϕ
- gradually derive clauses expressing that the clauses in the alleged refutation are satisfied
- contradiction cannot be satisfied, which is a contradiction

Proof idea of 3.

- take the canonical refutation of ϕ produced by the DP procedure
- “squeeze” the exponential size tree into a polynomial size proof assuming $\neg PHP^{2k}_k$
- finding an “error” in the squeezed proof would mean finding a collision in the PHP mapping
- but we know that $\neg PHP^{2k}_k$ can only be refuted by exponential size refutations

$\square$

Proving complexity lower bounds is hard.

Proving complexity lower bounds is hard.

Proving lower bounds on Resolution proofs is hard in Resolution.

Propositional Logic: 3 main types of proof systems

1. Frege systems
2. Sequent Calculus
3. Natural Deduction

Propositional Logic: 3 main types of proof systems

1. Frege systems
2. Sequent Calculus
3. Natural Deduction

We will consider now formulas in arbitrary basis.

Frege systems

- A Frege system is given by a finite number of deduction rules.
- We require that it is sound and complete.
- Furthermore, we usually also want it be implicationally complete.

Frege systems

- A Frege system is given by a finite number of deduction rules.
- We require that it is sound and complete.
- Furthermore, we usually also want it be implicationally complete.

A deduction rule is

$$\frac{\phi_1(A_1, \ldots, A_n), \ldots, \phi_k(A_1, \ldots, A_n)}{\psi(A_1, \ldots, A_n)}$$

where $\psi_1, \ldots, \psi$ are formulas with meta-variables $A_1, \ldots, A_n$.

Frege systems

- A Frege system is given by a finite number of deduction rules.
- We require that it is sound and complete.
- Furthermore, we usually also want it be implicationally complete.

A deduction rule is

$$\frac{\phi_1(A_1, \ldots, A_n), \ldots, \phi_k(A_1, \ldots, A_n)}{\psi(A_1, \ldots, A_n)}$$

where $\psi_1, \ldots, \psi$ are formulas with meta-variables $A_1, \ldots, A_n$.

An application of the rule is obtained by substituting specific formulas for meta-variables.

Frege systems

- A Frege system is given by a finite number of deduction rules.
- We require that it is sound and complete.
- Furthermore, we usually also want it be implicationally complete.

A deduction rule is

$$\frac{\phi_1(A_1,\ldots,A_n),\ldots,\phi_k(A_1,\ldots,A_n)}{\psi(A_1,\ldots,A_n)}$$

where $\psi_1,\ldots,\psi$ are formulas with meta-variables $A_1,\ldots,A_n$.

An application of the rule is obtained by substituting specific formulas for meta-variables.

If there are no assumptions ($k=0$), we call the rule an axiom schema, or simply an axiom.

Implicational completeness:

$$\text{If } \phi_1, \ldots, \phi_m \models \psi \text{ then } \phi_1, \ldots, \phi_m \vdash \psi$$

Implicational completeness:

$$\text{If } \phi_1, \ldots, \phi_m \models \psi \text{ then } \phi_1, \ldots, \phi_m \vdash \psi$$

Deduction theorem:

$$\text{If } \phi_1, \ldots, \phi_m \vdash \psi \text{ then } \vdash \phi_1 \wedge \cdots \wedge \phi_m \supset \psi$$

$\supset$ stands for implication

example

Modus ponens

$$\frac{A, A \supset B}{B}$$

plus axioms (p_i should be meta-variables):

$$\begin{array}{ll}
p_1 \supset (p_2 \supset p_1) & (p_1 \supset p_2) \supset (p_1 \supset \neg p_2) \supset \neg p_1 \\
(p_1 \supset p_2) \supset (p_1 \supset (p_2 \supset p_3)) \supset (p_1 \supset p_3) & (\neg\neg p_1) \supset p_1 \\
p_1 \supset p_1 \vee p_2 & p_1 \wedge p_2 \supset p_1 \\
p_2 \supset p_1 \vee p_2 & p_1 \wedge p_2 \supset p_2 \\
(p_1 \supset p_3) \supset (p_2 \supset p_3) \supset (p_1 \vee p_2 \supset p_3) & p_1 \supset p_2 \supset p_1 \wedge p_2
\end{array}$$

the sequent calculus

1.2.1. Sequents and Cedents. In the Hilbert-style systems, each line in a proof is a formula; however, in sequent calculus proofs, each line in a proof is a *sequent*: a sequent is written in the form

$$A_1, \ldots, A_k \longrightarrow B_1, \ldots, B_\ell$$

where the symbol $\longrightarrow$ is a new symbol called the sequent arrow (not to be confused with the implication symbol $\supset$) and where each A_i and B_j is a formula. The intuitive meaning of the sequent is that the conjunction of the A_i's implies the disjunction of the B_j's. Thus, a sequent is equivalent in meaning to the formula

$$\bigwedge_{i=1}^{k} A_i \supset \bigvee_{j=1}^{\ell} B_j.$$

Axiom $A \to A$

Weak Structural Rules

Exchange:left $\dfrac{\Gamma, A, B, \Pi \longrightarrow \Delta}{\Gamma, B, A, \Pi \longrightarrow \Delta}$ *Exchange:right* $\dfrac{\Gamma \longrightarrow \Delta, A, B, \Lambda}{\Gamma \longrightarrow \Delta, B, A, \Lambda}$

Contraction:left $\dfrac{A, A, \Gamma \longrightarrow \Delta}{A, \Gamma \longrightarrow \Delta}$ *Contraction:right* $\dfrac{\Gamma \longrightarrow \Delta, A, A}{\Gamma \longrightarrow \Delta, A}$

Weakening:left $\dfrac{\Gamma \longrightarrow \Delta}{A, \Gamma \longrightarrow \Delta}$ *Weakening:right* $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, A}$

The weak structural rules are also referred to as just *weak* inference rules. The rest of the rules are called *strong* inference rules. The *structural* rules consist of the weak structural rules and the cut rule.

The Cut Rule

$$\frac{\Gamma \longrightarrow \Delta, A \qquad A, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}$$

The Propositional Rules[1]

$\neg$*:left* $\dfrac{\Gamma \longrightarrow \Delta, A}{\neg A, \Gamma \longrightarrow \Delta}$ $\neg$*:right* $\dfrac{A, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \neg A}$

$\wedge$*:left* $\dfrac{A, B, \Gamma \longrightarrow \Delta}{A \wedge B, \Gamma \longrightarrow \Delta}$ $\wedge$*:right* $\dfrac{\Gamma \longrightarrow \Delta, A \qquad \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \wedge B}$

$\vee$*:left* $\dfrac{A, \Gamma \longrightarrow \Delta \qquad B, \Gamma \longrightarrow \Delta}{A \vee B, \Gamma \longrightarrow \Delta}$ $\vee$*:right* $\dfrac{\Gamma \longrightarrow \Delta, A, B}{\Gamma \longrightarrow \Delta, A \vee B}$

$\supset$*:left* $\dfrac{\Gamma \longrightarrow \Delta, A \qquad B, \Gamma \longrightarrow \Delta}{A \supset B, \Gamma \longrightarrow \Delta}$ $\supset$*:right* $\dfrac{A, \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \supset B}$

terminology

- *sequent* $\Gamma \to \Delta$
- *cedents: antecedent* Γ, *succedent* Δ
- in a rule,
  1. the new derived fromula is called *principal* or *main*,
  2. the formulas from which it was derived are *auxiliary*,
  3. the other formulas are *side* formulas.

E.g., in

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

A is the *main formula*, Γ, Δ are sequences of *auxiliary formulas*
A, B are auxiliary, $A \wedge B$ is principal, and formulas in Γ, Δ are side formulas.

the natural deduction calculus

$$\wedge\text{-intro} \quad \frac{A \qquad B}{A \wedge B} \qquad\qquad \wedge\text{-elim} \quad \frac{A \wedge B}{A} \qquad \frac{A \wedge B}{B}$$

$$\vee\text{-intro} \quad \frac{A}{A \vee B} \qquad \frac{B}{A \vee B} \qquad\qquad \vee\text{-elim} \quad \frac{A \vee B \qquad \begin{matrix}[A]\\ C\end{matrix} \qquad \begin{matrix}[B]\\ C\end{matrix}}{C}$$

$$\supset\text{-intro} \quad \frac{\begin{matrix}[A]\\ B\end{matrix}}{A \supset B} \qquad\qquad \supset\text{-elim} \quad \frac{A \qquad A \supset B}{B}$$

$$\frac{\bot}{A} \qquad\qquad \frac{\begin{matrix}[A \supset \bot]\\ \bot\end{matrix}}{A}$$

the complexity of the three calculi

- they are polynomially equivalent
- stronger than Resolution (e.g., polynomial size proofs of PHP)
- if $NP \neq coNP$, then there are sequences of tautologies that do not have poly-size proofs
- but no nontrival lower bounds have been proved
- the proof-search problem is hard under cryptographic assumptions (such as hardness of factoring)

Tait's calculus

- sequents with empty antecedent
- sequents are sets instead of sequences

Each line in a Tait calculus proof is a set Γ of formulas with the intended meaning f Γ being the disjunction of the formulas in Γ. A Tait calculus proof can be tree-like r dag-like. The initial sets, or logical axioms, of a proof are sets of the form $\Gamma \cup \{p, \overline{p}\}$. ı the infinitary setting, there are three rules of inference; namely,

$$\frac{\Gamma \cup \{A_j\}}{\Gamma \cup \{\bigvee_{i \in I} A_i\}} \quad \text{where } j \in I,$$

$$\frac{\Gamma \cup \{A_j : j \in I\}}{\Gamma \cup \{\bigwedge_{j \in I} A_j\}} \quad \text{(there are } |I| \text{ many hypotheses), and}$$

$$\frac{\Gamma \cup \{A\} \qquad \Gamma \cup \{\overline{A}\}}{\Gamma} \quad \text{the cut rule.}$$

3rd lesson
cut-free proofs

$$\text{Cut} \qquad \frac{\Gamma \to \Sigma, A \quad \Gamma, A \to \Sigma}{\Gamma \to \Sigma}$$

Theorem

The sequent calculus is complete without the cut rule.

3rd lesson
cut-free proofs

$$\text{Cut} \qquad \frac{\Gamma \to \Sigma, A \quad \Gamma, A \to \Sigma}{\Gamma \to \Sigma}$$

Theorem

The sequent calculus is complete without the cut rule.

Cut formulas $\leftrightarrow$ lemmas, propositions that use concepts not contained in the premises.

3rd lesson
cut-free proofs

$$\text{Cut} \qquad \frac{\Gamma \to \Sigma, A \quad \Gamma, A \to \Sigma}{\Gamma \to \Sigma}$$

Theorem

The sequent calculus is complete without the cut rule.

Cut formulas $\leftrightarrow$ lemmas, propositions that use concepts not contained in the premises.

Theorem

There are tautologies that have only exponentially long cut-free proofs, but have polynomial size proofs with cuts.

I.e., deep theorems cannot be proved without lemmas.

an application: the interpolation theorem

Theorem (Craig)

Consider formula in a language with $\bot, \top$. *If*

$$\models A \to B,$$

then there exists a formula C *such that*

1. $Var(C) \subseteq Var(A) \cap Var(B)$
2. $\models A \to C$ *and* $\models C \to B$

an application: the interpolation theorem

Theorem (Craig)

Consider formula in a language with $\bot, \top$. If

$$\models A \to B,$$

then there exists a formula C such that

1. $Var(C) \subseteq Var(A) \cap Var(B)$
2. $\models A \to C$ *and* $\models C \to B$

Proof 1

Let $A(\vec{x}, \vec{y})$, $B(\vec{x}, \vec{z})$. Define

$$C(\vec{x}) := \bigvee_{\vec{b} \in \{0,1\}^m} A(\vec{x}, \vec{b})$$

Proof 2

Prove a stronger theorem in the sequent calculus.

Theorem

Given a cut-free proof of a sequent $\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2$, one can construct in polynomial time a formula C such that

1. $Var(C) \subseteq Var(\Gamma_1, \Delta_1) \cap Var(\Gamma_2, \Delta_2)$
2. $\models \Gamma_1 \to \Delta_1, C$ *and* $\models C, \Gamma_2 \to \Delta_2$.

Proof 2

Prove a stronger theorem in the sequent calculus.

Theorem

Given a cut-free proof of a sequent $\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2$, one can construct in polynomial time a formula C such that

1. $\mathit{Var}(C) \subseteq \mathit{Var}(\Gamma_1, \Delta_1) \cap \mathit{Var}(\Gamma_2, \Delta_2)$
2. $\models \Gamma_1 \to \Delta_1, C$ *and* $\models C, \Gamma_2 \to \Delta_2$.

We will use the subformula property of cut-free proofs:

Every formula in a cut-free proof is a subformula of a formula in the last sequent.

Hence every sequent in the proof of $\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2$ has form $\Gamma_1', \Gamma_2' \to \Delta_1', \Delta_2'$ where

$$\mathit{Var}(\Gamma_1', \Delta_1') \subseteq \mathit{Var}(\Gamma_1, \Delta_1) \text{ and } \mathit{Var}(\Gamma_2', \Delta_2') \subseteq \mathit{Var}(\Gamma_2, \Delta_2)$$

Proof 2

Prove a stronger theorem in the sequent calculus.

Theorem

Given a cut-free proof of a sequent $\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2$, one can construct in polynomial time a formula C such that

1. $Var(C) \subseteq Var(\Gamma_1, \Delta_1) \cap Var(\Gamma_2, \Delta_2)$
2. $\models \Gamma_1 \to \Delta_1, C$ *and* $\models C, \Gamma_2 \to \Delta_2$.

We will use the subformula property of cut-free proofs:

Every formula in a cut-free proof is a subformula of a formula in the last sequent.

Hence every sequent in the proof of $\Gamma_1, \Gamma_2 \to \Delta_1, \Delta_2$ has form $\Gamma_1', \Gamma_2' \to \Delta_1', \Delta_2'$ where

$$Var(\Gamma_1', \Delta_1') \subseteq Var(\Gamma_1, \Delta_1) \text{ and } Var(\Gamma_2', \Delta_2') \subseteq Var(\Gamma_2, \Delta_2)$$

Therefore, if we gradually construct interpolants in the proof, then for each interpolant we will have the condition $Var(C') \subseteq Var(\Gamma_1, \Delta_1) \cap Var(\Gamma_2, \Delta_2)$

If

$$\frac{\Sigma_1 \to \Pi_1 \quad \Sigma_2 \to \Pi_2}{\Sigma \to \Pi}$$

and C_1, C_2 are interpolants for $\Sigma_1 \to \Pi_1$, $\Sigma_2 \to \Pi_2$,

then one can construct an interpolant C for $\Sigma \to \Pi$.

If

$$\frac{\Sigma_1 \to \Pi_1 \quad \Sigma_2 \to \Pi_2}{\Sigma \to \Pi}$$

and C_1, C_2 are interpolants for $\Sigma_1 \to \Pi_1$, $\Sigma_2 \to \Pi_2$,

then one can construct an interpolant C for $\Sigma \to \Pi$.

E.g., if the rule is $\wedge$-introduction, the interpolant will be either $C_1 \wedge C_2$ or $C_1 \vee C_2$ depending on which side $\wedge$ is introduced.

If

$$\frac{\Sigma_1 \to \Pi_1 \quad \Sigma_2 \to \Pi_2}{\Sigma \to \Pi}$$

and C_1, C_2 are interpolants for $\Sigma_1 \to \Pi_1$, $\Sigma_2 \to \Pi_2$,

then one can construct an interpolant C for $\Sigma \to \Pi$.

E.g., if the rule is $\wedge$-introduction, the interpolant will be either $C_1 \wedge C_2$ or $C_1 \vee C_2$ depending on which side $\wedge$ is introduced.

Exercise. Prove at least a few cases. This will likely be a question on the exam!

Theorem

The sequent calculus is complete without the cut rule.

Theorem

The sequent calculus is complete without the cut rule.

HINT: use the fact that the rules, except of weakening and cut, are invertible. □

the cut-elimination procedure

Theorem

There exists a procedure (an algorithm) that gradually eliminates cuts one-by-one until the proof is cut-free.

Remarks

- we already know that the procedure must sometimes run at least for exponentially long time
- elimination of one cut may result in creation of multiple cuts, but using good bookkeeping we can show that the procedure terminates

example 1

cut with $A \wedge B$:

$$\dfrac{\dfrac{\Gamma \longrightarrow A \qquad \Gamma \longrightarrow B}{\Gamma \longrightarrow A \wedge B} \qquad \dfrac{A, B \longrightarrow \Sigma}{A \wedge B \longrightarrow \Sigma}}{\Gamma \longrightarrow \Sigma}\ \mathit{cut}$$

example 1

cut with $A \wedge B$:

$$\textit{cut}\ \dfrac{\dfrac{\Gamma \longrightarrow A \qquad \Gamma \longrightarrow B}{\Gamma \longrightarrow A \wedge B} \qquad \dfrac{A, B \longrightarrow \Sigma}{A \wedge B \longrightarrow \Sigma}}{\Gamma \longrightarrow \Sigma}$$

replaced with two cuts, one with A and one with B:

$$\textit{cut}\ \dfrac{\dfrac{\Gamma \longrightarrow B}{\Gamma \longrightarrow B, \Sigma} \qquad \textit{cut}\ \dfrac{\Gamma \longrightarrow A \qquad A, B \longrightarrow \Sigma}{\Gamma, B \longrightarrow \Sigma}}{\Gamma \longrightarrow \Sigma}$$

example 2

$$
\dfrac{
  \dfrac{
    \dfrac{\Gamma \longrightarrow A, A \qquad \Gamma \longrightarrow B}{\Gamma \longrightarrow A \wedge B, A}
    \qquad \Gamma \longrightarrow B
  }{
    \textit{contraction}\rightarrow \quad \dfrac{\Gamma \longrightarrow A \wedge B, A \wedge B}{\Gamma \longrightarrow A \wedge B}
  }
  \qquad
  \dfrac{A, B \longrightarrow \Sigma}{A \wedge B \longrightarrow \Sigma}
}{\Gamma \longrightarrow \Sigma}
$$

To eliminate the cut with $A \wedge B$ requires

- 1 cut with B
- 2 cuts with A

example 2

$$\mathit{contraction}\rightarrow\ \dfrac{\dfrac{\dfrac{\dfrac{\Gamma\longrightarrow A, A \qquad \Gamma\longrightarrow B}{\Gamma\longrightarrow A\wedge B, A} \qquad \Gamma\longrightarrow B}{\Gamma\longrightarrow A\wedge B, A\wedge B}}{\Gamma\longrightarrow A\wedge B} \qquad \dfrac{A, B\longrightarrow\Sigma}{A\wedge B\longrightarrow\Sigma}}{\Gamma\longrightarrow\Sigma}$$

To eliminate the cut with $A \wedge B$ requires

- 1 cut with B
- 2 cuts with A

Exercise

Draw the the proof!

bookkeeping

i.e., how to show that the procedure converges

Definition

- the rank of a formula = the number of connectives
- the rank of a cut = the rank of the cut formula

bookkeeping

i.e., how to show that the procedure converges

Definition

- the rank of a formula = the number of connectives
- the rank of a cut = the rank of the cut formula

Lemma

Let C be a cut of rank r such that all cuts above it (if any) have rank $< r$. Then the elimination of C results in having one less cut of rank r.

bookkeeping

i.e., how to show that the procedure converges

Definition

- the rank of a formula = the number of connectives
- the rank of a cut = the rank of the cut formula

Lemma

Let C be a cut of rank r such that all cuts above it (if any) have rank $< r$. Then the elimination of C results in having one less cut of rank r.

Tait's strategy:

- eliminate a cut of the highest rank such that there is no cut above it with the same rank

Exercise

Find a different strategy for eliminating cuts that converges!

the complexity of interpolation

$\vdash \phi(\vec{p}, \vec{q}) \rightarrow I(\vec{p}) \rightarrow \psi(\vec{p}, \vec{r})$

the complexity of interpolation

$$\vdash \phi(\vec{p}, \vec{q}) \to I(\vec{p}) \to \psi(\vec{p}, \vec{r})$$

Denote by $\alpha(\vec{p}, \vec{q}) := \neg\phi(\vec{p}, \vec{q})$ and $\beta(\vec{p}, \vec{r}) := \psi(\vec{p}, \vec{r})$. Then

1. $\vdash \alpha(\vec{p}, \vec{q}) \vee \beta(\vec{p}, \vec{r})$
2. $\neg I(\vec{p}) \to \alpha(\vec{p}, \vec{q})$
3. $I(\vec{p}) \to \beta(\vec{p}, \vec{r})$

the complexity of interpolation

$$\vdash \phi(\vec{p},\vec{q}) \to I(\vec{p}) \to \psi(\vec{p},\vec{r})$$

Denote by $\alpha(\vec{p},\vec{q}) := \neg\phi(\vec{p},\vec{q})$ and $\beta(\vec{p},\vec{r}) := \psi(\vec{p},\vec{r})$. Then

1. $\vdash \alpha(\vec{p},\vec{q}) \vee \beta(\vec{p},\vec{r})$
2. $\neg I(\vec{p}) \to \alpha(\vec{p},\vec{q})$
3. $I(\vec{p}) \to \beta(\vec{p},\vec{r})$

Suppose that α_n, β_n, I_n are constructible in polynomial time. Denote by

$$A := \{\bar{u} | \exists \bar{v}\ \neg\alpha_i(\bar{u},\bar{v}), i \in \mathbb{N}\},$$
$$B := \{\bar{u} | \exists \bar{w}\ \neg\beta_i(\bar{u},\bar{w}), i \in \mathbb{N}\},$$
$$C := \{\bar{u} |\ I(\bar{u}), i \in \mathbb{N}\}.$$

the complexity of interpolation

$\vdash \phi(\vec{p}, \vec{q}) \to I(\vec{p}) \to \psi(\vec{p}, \vec{r})$

Denote by $\alpha(\vec{p}, \vec{q}) := \neg\phi(\vec{p}, \vec{q})$ and $\beta(\vec{p}, \vec{r}) := \psi(\vec{p}, \vec{r})$. Then

1. $\vdash \alpha(\vec{p}, \vec{q}) \vee \beta(\vec{p}, \vec{r})$
2. $\neg I(\vec{p}) \to \alpha(\vec{p}, \vec{q})$
3. $I(\vec{p}) \to \beta(\vec{p}, \vec{r})$

Suppose that α_n, β_n, I_n are constructible in polynomial time. Denote by

$A := \{\bar{u} | \exists \bar{v}\ \neg\alpha_i(\bar{u}, \bar{v}), i \in \mathbb{N}\}$,
$B := \{\bar{u} | \exists \bar{w}\ \neg\beta_i(\bar{u}, \bar{w}), i \in \mathbb{N}\}$,
$C := \{\bar{u} |\ I(\bar{u}), i \in \mathbb{N}\}$.

Then $A, B \in \mathrm{NP}$, $C \in \mathrm{P}$ and

1. $A \cap B = \emptyset$
2. $A \subseteq C$
3. $B \cap C = \emptyset$

$A, B \in \mathrm{NP},\ C \in \mathrm{P}$ and

1. $A \cap B = \emptyset$
2. $A \subseteq C$
3. $B \cap C = \emptyset$

$A, B \in \mathrm{NP}$, $C \in \mathrm{P}$ and

1. $A \cap B = \emptyset$
2. $A \subseteq C$
3. $B \cap C = \emptyset$

Conjecture

There exist $A, B \in$ **NP** *that cannot be separated by a set in* **P**.

$A, B \in \mathrm{NP},\ C \in \mathrm{P}$ and

1. $A \cap B = \emptyset$
2. $A \subseteq C$
3. $B \cap C = \emptyset$

Conjecture

There exist $A, B \in$ **NP** *that cannot be separated by a set in* **P***.*

Conjecture (stronger)

$\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$

$A, B \in \mathrm{NP},\ C \in \mathrm{P}$ and

1. $A \cap B = \emptyset$
2. $A \subseteq C$
3. $B \cap C = \emptyset$

Conjecture

There exist $A, B \in$ **NP** *that cannot be separated by a set in* **P**.

Conjecture (stronger)

$\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$

Corollary

Assuming the conjecture, interpolants cannot be constructed in polynomial time.

Proof.

From $A, B \in \mathrm{NP},\ A \cap B = \emptyset$, construct tautologies

$$\alpha_n(\vec{p}, \vec{q}) \vee \beta_n(\vec{p}, \vec{r})$$

$\square$

4th lesson
the graphs of proofs

- directed acyclic graph (DAG)
- nodes = labeled by
  1. formulas or sequents and
  2. rules applied
- arrows = indicate which assumptions used
- sources = axioms
- sink = the formula/sequent proved

4th lesson
the graphs of proofs

- directed acyclic graph (DAG)
- nodes = labeled by
  1. formulas or sequents and
  2. rules applied
- arrows = indicate which assumptions used
- sources = axioms
- sink = the formula/sequent proved

Example

$$\frac{A, A \supset B}{B} \qquad \begin{matrix} [A; ...] & & & & [A \supset B; ...] \\ & \searrow & & \swarrow & \\ & & [B; \textit{modus ponens}] & & \end{matrix}$$

trees and DAGs

Two forms of proofs

1. general, DAG-like
2. tree-like, useful for analyzing proofs

trees and DAGs

Two forms of proofs

1. general, DAG-like
2. tree-like, useful for analyzing proofs

The transformation from a DAG-like to tree-like may result in exponential blowup

trees and DAGs

Two forms of proofs

1. general, DAG-like
2. tree-like, useful for analyzing proofs

The transformation from a DAG-like to tree-like may result in exponential blowup

A similar distinction for Boolean circuits:

1. general Boolean circuits, DAG-like
2. tree-like, propositional formulas

Krajíček's idea

From a given proof in a weak proof system we may be able to construct an interpolant, or

Krajíček's idea

From a given proof in a weak proof system we may be able to construct an interpolant, or

From a given proof P and an assignment $\vec{a}$ to common variables we may decide which formula is a tautology.

Krajíček's idea

From a given proof in a weak proof system we may be able to construct an interpolant, or

From a given proof P and an assignment $\vec{a}$ to common variables we may decide which formula is a tautology. If

$$P \vdash \alpha(\vec{p}, \vec{q}) \vee \beta(\vec{p}, \vec{r})$$

and $\vec{p} \mapsto \vec{a} \in \{0, 1\}^n$, then

$$\models \alpha(\vec{a}, \vec{q}) \quad \text{or} \qquad \models \beta(\vec{a}, \vec{r})$$

Krajíček's idea

From a given proof in a weak proof system we may be able to construct an interpolant, or

From a given proof P and an assignment $\vec{a}$ to common variables we may decide which formula is a tautology. If

$$P \vdash \alpha(\vec{p}, \vec{q}) \vee \beta(\vec{p}, \vec{r})$$

and $\vec{p} \mapsto \vec{a} \in \{0,1\}^n$, then

$$\models \alpha(\vec{a}, \vec{q}) \quad \text{or} \quad \models \beta(\vec{a}, \vec{r})$$

We want to decide which of the two is true.

Krajíček's idea

From a given proof in a weak proof system we may be able to construct an interpolant, or

From a given proof P and an assignment $\vec{a}$ to common variables we may decide which formula is a tautology. If

$$P \vdash \alpha(\vec{p}, \vec{q}) \vee \beta(\vec{p}, \vec{r})$$

and $\vec{p} \mapsto \vec{a} \in \{0,1\}^n$, then

$$\models \alpha(\vec{a}, \vec{q}) \quad \text{or} \quad \models \beta(\vec{a}, \vec{r})$$

We want to decide which of the two is true.

In terms of disjoint NP-sets:

Given a proof P of

$$A \cap B = \emptyset$$

and given $a \in A \cup B$, we want to decide which of the two

$$a \in A \qquad \text{or} \qquad a \in B$$

is true.

feasible interpolation for cut-free proofs

Theorem

Given a tree-like cut-free proof

$$P \vdash \neg\alpha(\vec{p}, \vec{q}) \to \beta(\vec{p}, \vec{r})$$

we can construct in polynomial time a formula $I(\vec{p})$ *s.t.*

$$\vdash \neg\alpha(\vec{p}, \vec{q}) \to I(\vec{p}) \to \beta(\vec{p}, \vec{r}),$$

feasible interpolation for cut-free proofs

Theorem

Given a tree-like cut-free proof

$$P \vdash \neg\alpha(\vec{p}, \vec{q}) \to \beta(\vec{p}, \vec{r})$$

we can construct in polynomial time a formula $I(\vec{p})$ s.t.

$$\vdash \neg\alpha(\vec{p}, \vec{q}) \to I(\vec{p}) \to \beta(\vec{p}, \vec{r}),$$

or equivalently

$$\vdash I(\vec{p}) \to \alpha(\vec{p}, \vec{q}),$$

$$\vdash \neg I(\vec{p}) \to \beta(\vec{p}, \vec{r})$$

feasible interpolation for cut-free proofs

Theorem

Given a tree-like cut-free proof

$$P \vdash \neg\alpha(\vec{p}, \vec{q}) \to \beta(\vec{p}, \vec{r})$$

we can construct in polynomial time a formula $I(\vec{p})$ s.t.

$$\vdash \neg\alpha(\vec{p}, \vec{q}) \to I(\vec{p}) \to \beta(\vec{p}, \vec{r}),$$

or equivalently

$$\vdash I(\vec{p}) \to \alpha(\vec{p}, \vec{q}),$$

$$\vdash \neg I(\vec{p}) \to \beta(\vec{p}, \vec{r})$$

Hence given $\vec{p} \mapsto \vec{a}$, we can decide in polynomial time which of the two is true

$$\models \alpha(\vec{a}, \vec{q}) \quad \text{or} \quad \models \beta(\vec{a}, \vec{r}).$$

Theorem

Given a general cut-free proof

$$P \vdash \neg\alpha(\vec{p}, \vec{q}) \to \beta(\vec{p}, \vec{r})$$

we can construct in polynomial time a circuit $C(\vec{p})$ *s.t.*

$$\models C(\vec{p}) \to \alpha(\vec{p}, \vec{q}),$$

$$\models \neg C(\vec{p}) \to \beta(\vec{p}, \vec{r})$$

Theorem

Given a general cut-free proof

$$P \vdash \neg\alpha(\vec{p}, \vec{q}) \rightarrow \beta(\vec{p}, \vec{r})$$

we can construct in polynomial time a circuit $C(\vec{p})$ s.t.

$$\models C(\vec{p}) \rightarrow \alpha(\vec{p}, \vec{q}),$$

$$\models \neg C(\vec{p}) \rightarrow \beta(\vec{p}, \vec{r})$$

Hence given $\vec{p} \mapsto \vec{a}$, we can decide in polynomial time which of the two is true

$$\models \alpha(\vec{a}, \vec{q}) \quad \textit{or} \quad \models \beta(\vec{a}, \vec{r}).$$

feasible interpolation for Resolution

Theorem

Given a Resolution proof P of contradiction from a set of clauses $\{A_i(\vec{p},\vec{q})\}_i \cup \{B_j(\vec{p},\vec{r})\}_j$, in symbols:

$$P : \{A_i(\vec{p},\vec{q})\}_i \cup \{B_j(\vec{p},\vec{r})\}_j \to \perp,$$

we can construct in polynomial time a circuit C s.t. for all assignments $\vec{a}$

$$C(\vec{a}) = 0 \to \{A_i(\vec{p},\vec{q})\}_i \text{ is unsatatisfiable}$$

$$C(\vec{a}) = 1 \to \{B_j(\vec{p},\vec{r})\}_j \text{ is unsatatisfiable}$$

splitting Resolution proofs

Theorem

Given a Resolution proof P of contradiction

$$P : \{A_i(\vec{p}, \vec{q})\}_i \cup \{B_j(\vec{p}, \vec{r})\}_j \to \bot,$$

and an assignment for $\vec{p} \mapsto \vec{a}$, we can construct in polynomial time two proofs

- P^A *a proof from* $\{A_i(\vec{a}, \vec{q})\}_i$,
- P^B *a proof from* $\{B_j(\vec{a}, \vec{r})\}_j$,

such that one of them is a proof of contradiction.

splitting Resolution proofs

Theorem

Given a Resolution proof P of contradiction

$$P : \{A_i(\vec{p}, \vec{q})\}_i \cup \{B_j(\vec{p}, \vec{r})\}_j \to \bot,$$

and an assignment for $\vec{p} \mapsto \vec{a}$, we can construct in polynomial time two proofs

- P^A *a proof from* $\{A_i(\vec{a}, \vec{q})\}_i$,
- P^B *a proof from* $\{B_j(\vec{a}, \vec{r})\}_j$,

such that one of them is a proof of contradiction.

Proof.

See my paper: *Lower bounds for resolution and cutting planes proofs and monotone computations.*

splitting Resolution proofs

Theorem

Given a Resolution proof P of contradiction

$$P : \{A_i(\vec{p}, \vec{q})\}_i \cup \{B_j(\vec{p}, \vec{r})\}_j \to \bot,$$

and an assignment for $\vec{p} \mapsto \vec{a}$, we can construct in polynomial time two proofs

- P^A *a proof from* $\{A_i(\vec{a}, \vec{q})\}_i$,
- P^B *a proof from* $\{B_j(\vec{a}, \vec{r})\}_j$,

such that one of them is a proof of contradiction.

Proof.

See my paper: *Lower bounds for resolution and cutting planes proofs and monotone computations.*
Missing argument: We need to show that after the substitution $\vec{p} := \vec{a}$ none of the chosen clauses disappears. This follows by induction. $\square$