Propositional Proof Complexity
An Introduction

Samuel R. Buss

Departments of Mathematics and Computer Science
University of California, San Diego

1 Preface and Acknowledgements

This article is an abridged and revised version of a 1996 McGill University technical report [14]. The technical report was based on lectures delivered by the author at a workshop in Holetown, Barbados and on the authors prepared overhead transparencies. The audience at this workshop wrote scribe notes which then formed the technical report [14]. The material selected for the present article corresponds roughly to the content of the author's lectures at the NATO summer school held in Marktoberdorf, Germany in July-August 1997.

The present document consists essentially of the first, second, and sixth parts of [14]. Part I (sections 2-6) of the present document on *Propositional Proof and Complexity* is based on the scribe notes by K. Regan and J. Torán; Part II (sections 7-14) on *Interpolation Theorems for Propositional Logic* is based on the scribe notes by E. Allender and A. Maciel; and Part III (sections 15-16) on *Cutting Plane Proof Systems*, is partly based on the scribe notes of C. Lautemann and C. Zamora-Cura. The last section of part III has been rewritten to take into account subsequent progress in interpolation theorems and lower bounds for cutting plane proof systems.

The preparation of this article was partially supported by NSF grants DMS-9205181 and DMS-9503247, and by grant INT-9600919/ME-103 from NSF and MŠMT (Czech Republic).

Part I
Propositional Proofs and Their Complexity[1]

2 Introduction to Lengths of Propositional Proofs

The contents of Part 1 consist of the following topics:

- Frege systems
- Resolution
- Abstract proof systems
- Extended Frege ($e\mathcal{F}$) and substitution Frege ($s\mathcal{F}$) systems
- Tautologies based on the Pigeon-Hole Principle (PHP)
- Best-known lower bounds, including a survey of the state of the art for bounds on proof lengths in various *restricted fragments* of Frege systems

For all of this work, we will use *propositional formulas*, which are built up out of the following:

- Variables: $p_1, p_2, p_3, \ldots$; or informally, $p, q, r, \ldots$
- Logical connectives: $\neg$, $\land$, $\lor$, $\to$, $\equiv$, $\oplus$,.... We will talk about systems with subsets of these connectives, and more abstractly, of systems with finite sets of connectives of any arities.
- Parentheses: (,).
- Propositional formulas: Typified by: p_1, $(\neg p_1)$, $(p_1 \to (p_2 \lor p_3))$. This uses "fully-parenthesized" syntax. We will often omit parentheses and rely on the usual rules of precedence: $\neg$ binds tighter than $\{\land, \lor\}$, which bind tighter than $\to$, which binds tighter than $\equiv$. Exclusive-or $\oplus$ is not assigned a precedence.
- The constants $\top$ (or *True*) for "true" and $\bot$ (or *False*) for "false." These are often identified with '1' for $\top$ and '0' for $\bot$, or vice-versa, or '-1' for $\top$ and '1' for $\bot$, or etc. Or we may dispense with these

[1]This part is based on notes prepared by K. Regan and J. Torán based on lectures delivered by S. Buss on March 6, 1995 at the McGill University Bellair's Research Institute in Holetown, Barbados.

constants altogether and define $\top \equiv (p_1 \vee \neg p_1)$, $\bot \equiv (p_1 \wedge \neg p_1)$. The differences will not be important, and we shall be free to choose whatever looks best at a given time.

Definition Let ϕ be a propositional formula, with variables $p_1, \ldots, p_n$. A *truth assignment* $\vec{a} \in \{0,1\}^n$ assigns a true/false value to each p_i, and induces a value $\phi(\vec{a})$. If $\phi(\vec{a}) = \top$ then we say that $\vec{a}$ *satisfies* ϕ, and we sometimes also write $\vec{a} \models \phi$.

A formula ϕ is a *tautology* (or: is *valid*) if it is true under all truth assignments; i.e., if $(\forall \vec{a} \in \{0,1\}^n)\, \vec{a} \models \phi$. Sometimes one writes simply $\models \phi$ to say that ϕ is valid. Two formulas ϕ and ψ are *equivalent* if $\phi \equiv \psi$ is a tautology, which is the same as saying that ϕ and ψ have the same set of satisfying truth assignments.

Let $TAUT$ stand for the language of tautologies, under a straightforward encoding scheme. Cook's Theorem [19] shows that $P = NP$ iff $TAUT \in P$; i.e., iff there is a deterministic polynomial-time algorithm for recognizing tautologies. $TAUT$ is complete for coNP under polynomial-time many-one reducibility; the NP complete language SAT is essentially the same as the complement of $TAUT$, going from ϕ to $\neg\phi$.

It's worth asking at this point: what methods do we use to test the validity of a formula? In school one learns the method of "Truth Tables," which takes exponential time. No "shortcut" method is known that does any better than exponential-time (i.e., time $2^{O(n)}$, where n is the number of variables) in the worst case. But it is only in extreme cases that we use the method of truth tables: the normal way to tell that something is valid is to *prove* it! Whether proofs with polynomial-size lengths exist, and whether they can be efficiently found, are the proof-theoretic versions of the questions of whether SAT has small circuits or belongs to P. This starts us on the road of examining specific *proof systems* for propositional formulas.

2.1 Frege systems

A *Frege proof system* $\mathcal{F}$ has a finite set of *schematic axioms* and *rules of inference*. The meaning of these terms is best conveyed by a concrete example:

Rules of inference: Only one, called *modus ponens* (MP):

$$\frac{P \qquad P \to Q}{Q}\ \text{MP}$$

Axioms:

1. $(P \wedge Q) \to P$
2. $(P \wedge Q) \to Q$

3. $P \to (P \vee Q)$
4. $Q \to (P \vee Q)$
5. $(P \to Q) \to ((P \to \neg Q) \to \neg P)$
6. $(\neg\neg P) \to P$
7. $P \to (Q \to P \wedge Q)$
8. $(P \to R) \to ((Q \to R) \to (P \vee Q \to R))$
9. $P \to (Q \to P)$
10. $(P \to Q) \to (P \to (Q \to R)) \to (P \to R)$.

Here it is important to note that P, Q, and R are not single formulas, but meta-symbols that can stand for *any* propositional formula. Commonly one would call the above "one rule and ten axioms," but formally each item stands for an infinite set of *instances* of the rule or axiom. Each instance is obtained by substituting some propositional formula for P, Q, and/or R. (This distinction between axioms and axiom schemas is similar in predicate logic, where one has quantifiers, and especially in formal systems of arithmetic. For instance, Peano Arithmetic (PA) can be given six single axioms plus the *axiom schema* of induction, but there is no finite axiom set whose arithmetical consequences are exactly those of PA.)

Since every propositional variable is itself a formula, one can form the "smallest instances" of each of the above by inserting the variables p, q, and r in place of P, Q, and R. Then every other instance of the rule or axiom is obtainable by a *substitution* σ of formulas for those variables. Generally, one can define *substitutions* of formulas for the variables in any given propositional formula ϕ; this notion will be prominent later on.

It is now clear how to abstract the definition of a Frege system $\mathcal{F}$ to say that it consists of

(1) a domain T of well-formed formulas over some propositional *language* $\mathcal{L}$,

(2) a finite set $\mathcal{A}$ of schematic axioms, and

(3) a finite set $\mathcal{R}$ of schematic rules.

Each rule $R \in \mathcal{R}$ has some arity $k \geq 1$. In the propositional case, we can identify the *language* $\mathcal{L}$ with the set of connectives allowed in the formulas in T. For instance, the above system has language $\{\neg, \wedge, \vee, \to\}$. In speaking of $\mathcal{F}$ as a Frege system, it is taken for granted that $\mathcal{L}$ is *propositionally complete*, meaning that every formula ϕ over the "standard basis" $\{\neg, \wedge, \vee\}$ has an equivalent formula ϕ' over $\mathcal{L}$.

A *Frege proof* Π in a system $\mathcal{F} = (\mathcal{L}, \mathcal{A}, \mathcal{R})$ is a sequence $(\psi_1, \psi_2, \ldots, \psi_m)$ such that for all i, either ψ_i is an (instance of an) axiom, or there exist $j_1, \ldots, j_k < i$ and a k-ary rule $R \in \mathcal{R}$ such that $\psi_i = R(\psi_{j_1}, \ldots, \psi_{j_k})$. Then Π is a proof *of* the *theorem* $\psi = \psi_m$, and we may variously write $\vdash \psi$, $\vdash_\Pi \psi$, $\vdash_\mathcal{F} \psi$, or $\mathcal{F} \vdash \psi$.

The system $\mathcal{F}$ is *sound* if every theorem is valid, and *complete* if every valid formula ψ has a proof. These properties are usually built into the term "Frege system."

Theorem 1 *There exist (many) sound and complete Frege proof systems, including the above.*

There is related notion of *implicational completeness*: Write $\phi \models \psi$ if for every truth assignment $\vec{a}$ to variables occurring in ϕ and/or ψ, $\vec{a} \models \phi \Rightarrow \vec{a} \models \psi$. Write $\phi \vdash \psi$ if adding ϕ as an axiom would allow one to construct a proof of ψ. (Note that neither ϕ or ψ need be valid by themselves.) Then $\mathcal{F}$ is *implicationally complete* if whenever $\phi \models \psi$, also $\phi \vdash_\mathcal{F} \psi$. When $\mathcal{F}$ has *modus ponens* among its rules of inference, this is easily seen to be equivalent to the simple notions of completeness, but it is possible to craft "pathological" Frege systems without MP that are complete but not implicationally complete. Similarly define $\mathcal{F}$ to be *implicationally sound* if whenever $\phi \vdash_\mathcal{F} \psi$, then $\phi \models \psi$. When we later generalize to *substitution Frege systems*, we will lose implicational soundness, because the substitution rule e.g. allows ψ to be ϕ with its variables renamed, but such ϕ will generally not be a consequence of ϕ.

Many of the notions defined in this and the next subsection extend naturally to systems of predicate logic, first-order arithmetics, and even to higher-order logics. Adding four "quantifier axiom schemas" to 1.–10. above yields a "Frege-style" system (rather, a Hilbert system) for first-order predicate logic that is sound and complete. However, when we move to arithmetics with $+$ and $\cdot$, and with a computable set of axioms and inference rules, at least one of soundness or completeness goes out the window—of course this is *Gödel's First Incompleteness Theorem*.)

2.2 Complexity of Proofs

Now we can highlight the three principal complexity notions for proofs:

Definition

(a) The *number of lines* or *steps* in a proof $\Pi = (\psi_1, \ldots, \psi_m)$ equals m.

(b) The *symbol-length* of the proof is $n = |\Pi| = \sum_{i=1}^{m} |\psi_i|$.

(c) The *depth* d of the proof is the maximum AND/OR depth of a formula ψ_i occurring in the proof.

By the *length* or *size* of a proof we usually mean the symbol-length. We write $\mathcal{F} \overset{n}{\vdash} A$ to mean that A has a proof in $\mathcal{F}$ of at most n symbols.

The AND/OR depth is one notion of the complexity of an individual formula ψ: Write ψ over the basis $\{\wedge, \vee, \neg\}$, and use DeMorgan's Laws to bring the $\neg$'s on the variables only. Then count the maximum number of alternations between $\wedge$ and $\vee$ in a path from the top operand to a variable in the formula. (Alternatively, by careful padding we can rewrite ψ in a "leveled" form such that all paths have the same sequence of $\wedge$ and $\vee$, and we can count the number of alternations in that.)

Theorem 1 yields an inductive procedure that cranks out a proof of any given tautology $\psi \in TAUT$. However, the induction causes exponential blowup in both symbol-length and the number of lines as a function of the length of ψ (or more specifically, as a function of the number of logical connectives in ψ). This proof is no better or worse than that obtainable by slogging through the truth table. Whether one can do better than this is open:

Open Problem: Do the tautologies have polynomial-size Frege proofs? I.e., is there a polynomial p such that for all $\psi \in TAUT$, there exists a proof Π of ψ of length at most $p(|\psi|)$?

If so, then $NP = \text{coNP}$! This is because a nondeterministic TM on input ψ can guess Π, and then in deterministic polynomial time verify that Π is correct—this only requires checking that each concrete axiom or inference in Π belongs to one of finitely many schemas. This would place the coNP-complete set $TAUT$ into NP.

2.3 Robustness of proof systems

The following "robustness theorems" of Cook and Reckhow [20, 39] show that the particular choice of a Frege system does not matter for our present purposes. First we consider Frege systems over the same language, such as $\{\wedge, \vee, \neg, \rightarrow\}$.

Theorem 2 ([20, 39]) *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be Frege systems over the same language. Then there is a constant $c > 0$ such that for all ϕ and n, if $\mathcal{F}_1 \overset{n}{\vdash} \phi$, then $\mathcal{F}_2 \overset{\leq cn}{\vdash} \phi$.*

Proof [2] For every schematic axiom A of $\mathcal{F}_1$, let η_A be the "smallest instance" of A as defined above. By completeness, there is an $\mathcal{F}_2$-proof π_A of η_A. Likewise, for every schematic rule R of $\mathcal{F}_1$, take the "smallest instance" $R(\eta_1, \ldots, \eta_k) = \eta_0$. Then there is an $\mathcal{F}_2$-proof π_R of η_0, in which $\eta_1, \ldots, \eta_k$ appear as hypotheses. The neat point (Lemma 2.5 in [20], proved simply by induction on formula structure) is that for every substitution σ,

[2]Inserted by KWR, following p40 of [20].

the formula $\sigma(\eta_A)$ has the $\mathcal{F}_2$-proof $\sigma(\pi_A)$ defined by applying σ to all formulas in π_A. (Here one can arrange that π_A has no variables other than those in η_A, or one can define σ to be the identity on other variables.) Likewise, $\sigma(\pi_R)$ is an $\mathcal{F}_2$ proof of $\sigma(\eta_0)$ from $\sigma(\eta_1), \ldots, \sigma(\eta_k)$.

Now let π_1 be a proof of ϕ in $\mathcal{F}_1$. For every instance ψ of a schematic axiom A in π_1, let σ be the (effectively unique) substitution such that $\sigma(\eta_A) = \psi$ and let the $\mathcal{F}_2$ proof π_2 have the sequence $\sigma(\pi_A)$ in place of the occurrence of ψ. For every application $R(\psi_1, \ldots, \psi_k) = \psi$ of a rule, there is a single substitution σ such that $\sigma(\eta_1) = \psi_1, \ldots, \sigma(\eta_k) = \psi_k$ and $\sigma(\eta_0) = \psi$. By induction we may suppose that $\psi_1, \ldots, \psi_k$ have already been proved in the $\mathcal{F}_2$-proof we are building. Hence they fill the roles of the hypotheses in the $\mathcal{F}_2$-proof sequence π_R, and so we need only splice the remainder of π_R into the segment of π_2 corresponding to the occurrence of ψ in π_1.

For the size analysis, let the constant K be the maximum of $|\pi_A|$ or $|\pi_R|$ over the finitely many schematic axioms and rules of $\mathcal{F}_1$. The key is that for every substitution σ,

$$|\sigma(\pi_A)| \leq |\pi_A| \cdot |\sigma(\eta_A)| \leq K \cdot |\sigma(\eta_A)|.$$

A similar inequality holds for instances of rules R and proof segments π_R. This says that for every occurrence of a formula ψ in π_1, the symbol-length of the segment of the $\mathcal{F}_2$-proof π_2 corresponding to that occurrence is linear in ψ. $\square$

In order to talk about simulations between Frege systems over *different* languages, we must first fix a translation from formulas ϕ of one system to "equivalent" formulas ϕ' of the other. There are two problems to overcome here:

(1) First, something more than formal equivalence of ϕ and ϕ' has to be meant, because all tautologies are formally equivalent. One needs a notion that the translated formula ϕ' has the same "meaning" or "structure" as ϕ.

(2) A "direct translation," by which one means substituting an equivalent composition of functions in $\mathcal{L}_\in$ for each $\mathcal{L}_\infty$-connective in the formula ϕ, may not be polynomial-time computable. Consider the languages $\mathcal{L}_\infty = \{\wedge, \oplus, \neg\}$ and $\mathcal{L}_\in = \{\wedge, \vee, \neg\}$, and ϕ over $\mathcal{L}_\infty$ of the form $\phi = \phi_1 \oplus \phi_2$. One can define the translation

$$\phi' = (\phi_1 \wedge \neg\phi_2) \vee (\neg\phi_1 \wedge \phi_2),$$

but doing this slavishly recursively leads to exponential blowup if ϕ has linear nesting depth.

Reckhow solved these problems in his thesis [39]. His translation scheme distinguishes systems over binary connectives and distinguishes $\{\equiv, \oplus\}$ from the other binary connectives. Reckhow showed that among the binary connectives apart from $\{\equiv, \oplus\}$, "direct translations" have polynomial-size overhead, and that translating *from* a formula over the standard basis $\{\wedge, \vee, \neg\}$ poses no special difficulty. The key is how to translate from formulas ϕ over arbitrary bases into the standard basis. He presented a uniform way to do this via the method of Spira [41] (see also [8, 9, 4]), which implicitly "re-balances" ϕ during the recursion. Reckhow called this an "indirect translation." The above combines to define a unique translation from any given language $\mathcal{L}_\infty$ to a given language $\mathcal{L}_\in$, and we can combine Reckhow's terms "direct" and "indirect" and call this the *natural translation* from $\mathcal{L}_\infty$ to $\mathcal{L}_\in$.

Theorem 3 ([39]) *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be any two (sound and complete) Frege systems, and let $\phi \mapsto \phi'$ be the natural translation from $\mathcal{F}_1$ to $\mathcal{F}_2$. Then there is a polynomial p such that for every $\mathcal{F}_1$-proof π of a tautology ϕ, there is an $\mathcal{F}_2$-proof π' of ϕ' such that $|\pi'| \leq p(|\pi|)$. Moreover, π' is computable in polynomial time given π.*

Proof (Sketch) The main task is to verify that the Spira-based translation into the standard basis has only polynomial blowup. Then one can adapt the proof of the last theorem. Consider first the case where $\mathcal{L}_\infty$ has binary connectives only, including $\oplus$ and/or $\equiv$, so that ϕ is a binary tree. The basic lemma, used earlier by Hartmanis and Stearns [26] to put CFLs in $DSPACE(\log^2 n)$, is that every binary tree T has a subtree S satisfying $\frac{1}{3}|T| \leq |S| \leq \frac{2}{3}|T|$.

To apply this lemma, make a new tree by adding a new root node labeled $\vee$, with two new $\wedge$ nodes as children. One $\wedge$ has S as one child, and the other child is what you get from $T \setminus S$ by substituting a '1' for the edge from the parent of S to $T \setminus S$. The other $\wedge$ node has for children the negation of S and the result of a '0' substitution on that edge into $T \setminus S$. This process is continued recursively on the four subtrees. At the bottom, we will be down to positively and negatively signed literals—all the $\oplus$ and $\equiv$ magically go away in the steps that simplify $T \setminus S$. The resulting tree has at most $2d$ levels of alternating $\vee$ and $\wedge$, where $d = \log_{3/2}|T| = O(\log |T|)$. The size is at most $O(|T|^2)$.

In the case of, say, a 6-ary connective in $\mathcal{L}_\infty$, one would use a "$\frac{1}{7}$, $\frac{6}{7}$" tree-splitting lemma to get a similar log-depth, poly-size translation, though with a larger polynomial blowup in size.

Once this translation is done, the rest is similar to the proof of Theorem 2. However, there are a lot of technical details and the proof is substantially more difficult. Among other things, one must prove that there are polynomial size proofs that the Spira translation of a conjunction $A \wedge B$ is equivalent to the conjunction of the Spira translations of A and B $\square$

The same rebalancing idea shows that polynomial-size formulas equal non-uniform NC^1. An interesting question is whether the quadratic blowup above in the binary $\oplus$ case above can be improved to (nearly) linear. Recent papers by Bshouty, Cleve and Eberly [9] and Bonet and Buss [4] show that with a more-involved scheme for chopping up the tree T, one can get a sub-quadratic simulation. See also the related paper by Kosaraju and Delcher [27], which is based on a notably different idea of "fracturing" a tree of size t into many little pieces, viz. $\sqrt{t}$-many subtrees, each of size roughly $\sqrt{t}$.

2.4 Resolution Proof Systems

A resolution *clause* is a finite set of literals, and stands for the disjunction of the literals. For convenience, we use * as a function to negate literals, so that $p_i^* = (\neg p_i)$ and $(\neg p_i)^* = p_i$.

Definition A clause α_3 is inferred from two clauses α_1, α_2 by the resolution rule

$$\frac{\alpha_1 \qquad \alpha_2}{\alpha_3}\,\text{Res}$$

precisely when

(a) there is a unique literal x such that $x \in \alpha_1$ and $x^* \in \alpha_2$, and

(b) $\alpha_3 = (\alpha_1 \cup \alpha_2) \setminus \{x, x^*\}$.

The resolution rule is a variant of modus ponens. In the case $\alpha_1 = \{x\}$ and $\alpha_2 = \{x^*\} \cup B$, where B is arbitrary, α_2 is equivalent to $\alpha_1 \to B$, and the resolution rule yields B just as MP does. The same kind of thing holds for bigger clauses α_1. The restriction that x be unique in (a) causes no loss of generality—consider for instance $\alpha_1 = (x \vee y \vee w)$ and $\alpha_2 = (x^* \vee y^* \vee z)$. Resolving on x alone would yield the useless tautological clause $(y \vee y^* \vee w \vee z)$, while eliminating x and y together to get $(w \vee z)$ is *unsound*.

Resolution is used to prove formulas ϕ that are in DNF, by obtaining a *resolution refutation* of the CNF formula $(\neg\phi)$, namely a sequence of resolution inferences that produces the empty clause. This refutation is also called a *resolution proof* of ϕ. The so-called *Davis-Putnam procedure* is an exponential-time deterministic algorithm that always produces a resolution proof of a given DNF tautology ϕ. This shows that resolution is *complete* for DNF formulas, and it is also clear that the resolution rule is *sound*.

Note that if the Davis-Putnam procedure ran in polynomial time, then P would equal NP. The fact that it runs in polynomial time for ϕ of clause size two is the proof that 2-SAT belongs to P. The Davis-Putnam procedure and various refinements of it work in polynomial time for other classes of formulas, and are used all the time as heuristics in practical applications.

(In the lectures, someone remarked that two main reasons for its popularity are that it's easy to program, and that "it's the kind of 'blind dumb search' on strings of symbols that computers are good at." Not to be too pejorative about this: there are many well-developed heuristics to guide this search, and they extend to general first-order resolution as well. Resolution is very important in practice.) It appears that Tseitin [43] was the first to study the lengths of individual resolution proofs and of shortest proofs for a given ϕ—note that this is a different matter from the running time of a deterministic algorithm that generates a resolution proof of ϕ.

One cause of theoretical interest in resolution is that in quite a few cases, both upper bounds and *matching* lower bounds of exponential proof size have been proved. More will be said about the complexity of resolution proofs later. But first, let us consider what seems to be the most abstract sensible idea of a proof system.

3 Abstract Proof Systems

The following definition was put forward by Cook et al. in much the same breath of research as the fundamental NP-completeness results.

Definition An *abstract propositional proof system* is a polynomial-time computable function f such that $Range(f) = TAUT$; i.e., the range of f is the set of all Boolean tautologies. An f*-proof* of a formula A is a string w such that $f(w) = A$.

Note that f need not be *polynomially honest*; i,e, there need not be a polynomial p such that for all w, $p(|f(w)|) > |w|$. If f is honest, then all proofs have polynomial size. The following example makes this point clear:

Example: A given Frege proof system $\mathcal{F}$ yields the function $f_{\mathcal{F}}(w) = A$ if w is a valid $\mathcal{F}$-proof of A, and $f_{\mathcal{F}}(w) = (p_1 \vee \neg p_1)$ otherwise.

One point of the general definition is that strong theories $\mathcal{F}$ such as PA or ZF can also be used as propositional proof systems; the only condition is that the theory is encoded in such a manner that validity of proofs can be checked in polynomial time. Two other examples are *cutting-planes proof systems*, originated by W. Cook, C. Coullard, and G. Turán in [21], and "quantified propositional logic" (see [23, 30]).

Definition A proof system f is *super* if every propositional tautology has a polynomial-size f-proof.

A super proof system can be modified to an equivalent proof system f' in which f' is polynomially honest after all. It is open whether super proof systems exist; in fact, we have the following equivalence:

Theorem 4 ([20]) *There exists a super proof system iff $NP = \text{coNP}$.*

Proof A super proof system would place the tautologies into NP, but $TAUT$ is coNP-complete under polynomial-time many-one reductions. For the converse, suppose $NP = \text{coNP}$, and let N be a polynomial-time NTM that recognizes $TAUT$. Then define $f(w) = A$ if $w = \langle A, c\rangle$, where c is an accepting computation of N on input A, and $f(w) = (p_1 \vee \neg p_1)$ otherwise. This f is super. $\square$

This theorem is one of the prime motivations for the study of propositional proof length—it justifies the study of upper and lower bounds on proof length for a variety of concrete propositional proof systems. Another motivation from the practical side comes from building efficient automated deduction systems.

Here it would be nice to have a theorem of the form "System $\mathcal{E}$ is super iff a super proof system exists." Of course it would be particularly nice if $\mathcal{E}$ was some concrete proof system, but it would also be interesting if such a system could be constructed by some kind of diagonal definition.

Definition [20] Let $\mathcal{E}$ and $\mathcal{E}'$ be proof systems with the same propositional language. Then say $\mathcal{E}'$ *simulates* $\mathcal{E}$ if there is a polynomial p such that for every tautology ϕ and $\mathcal{E}$-proof π of ϕ, there exists an $\mathcal{E}'$-proof π' of ϕ such that

$$|\pi'| \leq p(|\pi|).$$

Also say that $\mathcal{E}'$ *p-simulates* $\mathcal{E}$ if the mapping from π to π' is polynomial-time computable.

Similar definitions can be made when $\mathcal{E}$ and $\mathcal{E}'$ have different languages, provided a "natural translation" from $\mathcal{E}$ to $\mathcal{E}'$ is given.

Definition A proof system is *optimal* if it p-simulates every other proof system.

Open Problem:

(1) Does there exist an optimal proof system?

(2) Are Frege systems optimal?

Let E stand for $DTIME[2^{O(n)}]$, and NE for the corresponding nondeterministic time class. Krajíček and Pudlák [31] showed that $NE = E$ is a sufficient condition for (1). They also show that if $NE = coNE$, then there is a propositional proof system that is optimal in the weaker non-uniform sense of simulating every other proof system.

If (2) holds, then this would be a striking "complexity-theoretic conservation' result," saying that ordinary propositional logic is as efficient as anything powerful axioms like those in ZF set theory can do on propositional tautologies. Haken [25] proved that certain families of DNF tautologies require exponential length for resolution proofs, so resolution proof systems cannot be optimal even for DNF tautologies.

The next section describes a system that is suspected to be properly higher than the Frege systems under p-simulation.

4 Extended Frege Systems

Given an ordinary Frege system $\mathcal{F}$, an *extended Frege* proof, $e\mathcal{F}$ proof for short, is a sequence of formulas $A_1, A_2, A_3, \ldots, A_n$ such that for all i, either A_i follows from earlier formulas by a rule of $\mathcal{F}$, or A_i is an axiom instance of $\mathcal{F}$, or else A_i is an *extension formula* of the form

$$p_i \equiv \phi$$

where ϕ is any formula and p_i is a fresh "extension variable"; i.e., p_i occurs neither in ϕ nor in any of $A_1, \ldots, A_{i-1}$ nor in the last formula in the proof. We also speak of '$p_i \equiv \phi$' as inferred by the *extension rule*. The final formula A_n in the proof is not allowed to have any extension variables.

The idea is that p_i can now be used as an abbreviation for ϕ in all subsequent steps of the proof. This can reduce the proof size (number of symbols) greatly, though the number of steps is not reduced. Proofs using these rules can still be checked in polynomial time, so this is also an abstract proof system. This notion was studied by Tseitin [43], Statman [42], and by Cook and Reckhow [20, 39]. With the same provision about "natural translations" in case the underlying Frege systems are over different languages:

Theorem 5 ([20]) *Any two extended Frege proof systems p-simulate each other.*

Hence we can regard "$e\mathcal{F}$" as a single proof system, just as "$\mathcal{F}$" itself is a single Frege system, up to p-simulation equivalence. Since $e\mathcal{F}$ is an abstract proof system, it follows that if all tautologies have polynomial-size $e\mathcal{F}$ proofs, then $NP = \text{coNP}$. Statman proved that the size measure of $e\mathcal{F}$ proofs is essentially the same as the *step* measure of ordinary Frege proofs.

Theorem 6 ([42]) *For any Frege proof of step-length n of a formula A, A has an $e\mathcal{F}$-proof of size $O(n + |A|^2)$.*

The proof idea is that the $e\mathcal{F}$-proof introduces abbreviations for every "active" subformula in the Frege proof of A.

Open Problem:

(1) Can Frege proof systems (p-)simulate $e\mathcal{F}$ systems?

(2) Are $e\mathcal{F}$ proof systems optimal?

A major testing ground for problem (1) is the subject of the next section.

5 The Propositional Pigeonhole Principle

For each $n \geq 0$, we define a propositional formula PHP_n that expresses the *pigeonhole principle* for $n+1$ "pigeons" and n "holes." For each "pigeon" i and "hole" k, we allocate a propositional variable p_{ik}, with the intent that setting this variable *true* means that pigeon i has been placed into hole k. If we think of a function f from $[n+1] = \{0, \ldots, n\}$ into $[n] = \{0, \ldots, n-1\}$, then truth of p_{ik} signifies $f(i) = k$. The condition that every pigeon goes to some hole is encoded by the antecedent $\bigwedge_{i=0}^{n} \bigvee_{k=0}^{n-1} p_{ik}$. The "Pigeonhole Principle" states that then some hole must have more than one pigeon, expressed by $\bigvee_{k=0}^{n-1} \bigvee_{0 \leq i < j \leq n} (p_{ik} \wedge p_{jk})$. In full, for each $n \geq 1$, PHP_n is

$$\bigwedge_{i=0}^{n} \bigvee_{k=0}^{n-1} p_{ik} \quad \rightarrow \quad \bigvee_{k=0}^{n-1} \bigvee_{0 \leq i < j \leq n} (p_{ik} \wedge p_{jk}). \tag{1}$$

For example, with $r = n-1$, PHP_0 = "false $\rightarrow$ false," and:

$$\begin{aligned} PHP_1 &= p_{00} \wedge p_{10} \rightarrow p_{00} \wedge p_{10}, \\ PHP_2 &= (p_{00} \vee p_{01}) \wedge (p_{10} \vee p_{11}) \wedge (p_{20} \vee p_{21}) \rightarrow \\ & \quad (p_{00} \wedge p_{10}) \vee (p_{01} \wedge p_{11}) \vee (p_{00} \wedge p_{20}) \vee \\ & \quad (p_{01} \wedge p_{21}) \vee (p_{10} \wedge p_{20}) \vee (p_{11} \wedge p_{21}). \end{aligned}$$

The "pigeon-hole principle" is sometimes also called the "Dirichlet box principle." [3]

[3]Note that the left-hand side of (1) does not actually define a function from $[n+1]$ into $[n]$—it figuratively allows a "pigeon" to be assigned to more than one "hole"! To encode faithfully the statement that "every function from $[n+1]$ into $[n]$ is non-injective," we must conjoin to the left-hand side the clause

$$\bigwedge_{i=0}^{n} \bigwedge_{0 \leq k < \ell \leq n-1} (\neg p_{ik} \vee \neg p_{i\ell}).$$

Call the resulting statement PHP'_n. Note that this is intuitively a weaker assertion than PHP_n. A relevant technical counterpart to the idea of "weaker" is that from hypothesis PHP_n one can derive PHP'_n in just a few more proof lines and symbols; but the converse direction is not so clear. We are often sloppy about distinguishing between PHP_n and PHP'_n: this can usually be justified by defining

$$q_{ij} \equiv \text{`}g(i) = j\text{'} \equiv p_{ij} \wedge (\neg p_{i0} \wedge \neg p_{i1} \ldots \wedge \neg p_{i,j-1}).$$

The formulas PHP_n and the two other forms of the pigeon-hole principle discussed in the footnote are (transparently equivalent to) DNF formulas. Hence they are "in-bounds" for resolution, as well as for Frege proof systems. Kreisel was apparently the first to discuss the propositional forms of the pigeonhole principle, but the first in-depth analysis from the complexity point of view owes to the same paper by Cook and Reckhow that we have been discussing all along. Note that for each n, PHP_n has size proportional to n^3, so up to "polynomial scaling," it is OK to express complexity bounds in terms of n rather than the true size of PHP_n.

Theorem 7 ([20]) *The tautologies PHP_n have polynomial-size extended Frege proofs.*

The sketch of their proof is a good example of the effect of the extension rule for cutting down proof size. The conceptual pattern is a proof by reductio-ad-absurdum from the counterfactual hypothesis that f is 1-1.[4]

Proof Write $f : [n] \overset{1\text{-}1}{\to} [n-1]$, where $[n]$ is short for $\{0, \ldots, n\}$. Define $f_m : [m] \to [m-1]$ inductively for $m = n$ down to $m = 1$ by: $f_n = f$, and

This translation "factors through" to adapt the following proofs of PHP'_n into proofs of PHP_n. The adaptation also preserves constant-depth proofs.

Ajtai [1] worked with a still-weaker third form that restricts attention to functions f that are *onto* $[n]$, obtained by conjoining also

$$\bigwedge_{k=0}^{n-1} \bigvee_{i=0}^{n} p_{ik}$$

onto the left-hand side of (1). The neatly symmetrical form given by Ajtai for this, which we may call PHP''_n, is:

$$\neg \left[\left(\bigwedge_{i=0}^{n} \bigvee_{k=0}^{n-1} p_{ik} \right) \wedge \left(\bigwedge_{k=0}^{n-1} \bigvee_{i=0}^{n} p_{ik} \right) \wedge \left(\bigwedge_{i=0}^{n} \bigwedge_{0 \leq k < \ell \leq n-1} (\neg p_{ik} \vee \neg p_{i\ell}) \right) \wedge \left(\bigwedge_{k=0}^{n-1} \bigwedge_{0 \leq i < j \leq n} (\neg p_{ik} \vee \neg p_{jk}) \right) \right]. \quad (2)$$

The point offered by Ajtai is that for *negative* results such as those in his paper, using the weakest form (PHP''_n) makes the results apply to the others, while for *positive* results such as those given in these notes, one should use the strongest form (PHP_n).

However, there is sometimes a substantial difference between PHP''_n versus $\{PHP_n, PHP'_n\}$.

[4](Note by KWR): Sam speaks in these functional terms, and Cook and Reckhow actually used PHP'_n. However, the Cook-Reckhow proof works immediately for the most general case (i.e., for PHP_n), and I've re-worked what is on Sam's slides to make this plain. Intuitively speaking, we're allowing f to be a "multi-valued" function with domain $\{0, \ldots, n\}$ and values in $\{0, \ldots, n-1\}$, and the right-hand side of (1) is read as saying that f must have a "collision." I've written F in place of f for the possibly-multivalued case.

for $m \leq n-1,\ i \in [m]$:

$$f_m(i) = \text{if } f_{m+1}(i) < m \text{ then } f_{m+1}(i) \text{ else } f_{m+1}(m).$$

The idea is to prove the implication "f_m is 1-1 $\rightarrow$ f_{m-1} is 1-1" for each m, reducing things down to the assertion that $f_1 : \{0,1\} \rightarrow \{0\}$ is 1-1, which is clearly *absurdum*. Frege proofs can do reductio-ad-absurdum.

More generally, let F stand for a possibly-multivalued function with domain $[n]$ and values in $[n-1]$. Then define $F_n = F$ and inductively for $m < n$:

$$F_m(i) \mapsto k \quad \text{if } k < m \text{ and } [F_{m+1}(i) \mapsto k \vee (F_{m+1}(i) \mapsto m \wedge F_{m+1}(m) \mapsto k)].$$

If one lets B_m stand for the formula asserting that F_m has domain $[m]$, then $B_{m+1} \rightarrow B_m$ is immediate, since

$$F_{m+1}(i) \mapsto m \vee (\bigvee k < m)\ F_{m+1}(i) \mapsto k$$

follows from B_{m+1}. Now consider what happens if F_m has a collision; i.e., if there exist $i < j \in [m]$ and $k \in [m-1]$ such that $F_m(i) \mapsto k$ and $F_m(j) \mapsto k$. If both values can arise from the first disjunct, then $F_{m+1}(i) \mapsto k$ and $F_{m+1}(j) \mapsto k$, contradicting the assertion that F_{m+1} has no collisions. If both can arise from the second disjunct, then $F_{m+1}(i) \mapsto m$ and $F_{m+1}(j) \mapsto m$, ditto. If one value, say $F_m(j) \mapsto k$, comes from the first disjunct and the other comes from the latter, we have $F_{m+1}(j) \mapsto k$ and $F_{m+1}(i) \mapsto m$ and $F_{m+1}(m) \mapsto k$. This is also a contradiction since $i, j \leq m-1$. Thus we also get a reductio-ad-absurdum, starting from F_n having no collisions and ending with F_1 having no collisions as a multivalued function from $\{0,1\}$ to $\{0\}$, which contradicts the assertion B_1 that F_1 is total.

Now for the extended Frege proof, we introduce new "extension variables" q^m_{ik}, each intended to stand for "$f_m(i) = k$," or in the multi-valued case, "$F_m(i) \mapsto k$." Recall that we start with variables p_{ik} expressing "$f(i) = k$" or "$F(i) \mapsto k$," The extension rules that introduce the new variables are

$$\begin{aligned} q^n_{ik} &\equiv p_{ik} \text{ and for } m < n, & (3)\\ q^m_{ik} &\equiv q^{m+1}_{ik} \vee (q^{m+1}_{im} \wedge q^{m+1}_{mk}). & (4) \end{aligned}$$

The *whole point* is that by using these extension variables at each stage, the formal proof of each stage in the "reductio ad absurdum" *has the same symbol-length*. Indeed, mimicking the prose details given above for the multi-valued case, the symbols used in each stage are the same except for the superscripts m. Since each individual stage has polynomial symbol-length, the whole thing gets multiplied only by a factor of n, and is still polynomial size. □

However, if the above $e\mathcal{F}$ proof is converted into a Frege proof by removing the extension inferences and replacing the extension variables q_{ij}^m by the formulas they abbreviate, then the size of the formulas in the Frege proof would increase by a factor of 3 for the right-hand side of (4) at each stage, giving roughly 3^n symbols. That shows the difference between $\mathcal{F}$ and $e\mathcal{F}$ proofs. Note also here that the number of *lines* of the Frege proof is basically unaffected by the presence or absence of (4), in keeping with Statman's theorem.

For some time, the PHP_n formulas were considered a prime candidate for an exponential separation between $\mathcal{F}$ and $e\mathcal{F}$ proofs, which if established would be a major plank in Cook's program. But there is an alternative strategy that produces polynomial-sized Frege proofs of PHP_n. Intuitively speaking, it replaces the above use of induction by a clever stratagem for encoding *counting* into propositional formulas, and establishing some basic facts about counting via polynomial-sized Frege proofs. The stratagem involves the log-depth circuits for vector addition and counting from Ofman [35] and Wallace [44].

Theorem 8 ([11]) *The formulas PHP_n do have polynomial-sized Frege proofs.*

Proof Our proof is for PHP'_n, but it can be adapted for PHP_n.

For each ℓ, $0 \leq \ell \leq n-1$, define M_ℓ to be the number of i in the domain such that $f(i) \leq \ell$. Formally,

$$M_\ell := \left\| \left\{ i \in [n] : \bigvee_{0 \leq k \leq \ell} p_{ik} \right\} \right\|.$$

Again we take the hypothesis that f is 1-1 (or "has no collisions"). Thus $M_0 \leq 1$, since otherwise there would be a collision at $k = 0$. The objective is to prove the successive inequalities:

$$\begin{array}{rcl} M_0 & \leq & 1 \\ M_1 & \leq & M_0 + 1 \\ M_2 & \leq & M_1 + 1 \\ & \vdots & \\ M_{n-1} & \leq & M_{n-2} + 1. \end{array}$$

From this it follows that $M_{n-1} \leq n$. However, given that $f : [n] \to [n-1]$ is total on $[n]$, it follows that $M_{n-1} = n + 1$. This contradiction finishes the outline of the Frege proof. It remains first to give a polynomial-size Frege proof of each step, and then to show that the deduction $M_{n-1} \leq n$ also can be done in polynomial size.

First we need an encoding scheme in propositional logic for the numerical quantities M_ℓ. We write M_ℓ in standard binary notation as an $(a+1)$-bit number, where $a = \lfloor \log_2(n+1) \rfloor$, using leading zeroes if necessary. To these bits we will associate a sequence of formulas

$$\hat{m}^\ell = m^\ell_a, m^\ell_{a-1}, \ldots, m^\ell_0,$$

where each m^ℓ_i expresses whether the corresponding ith bit is 1 or 0.

To do this, it suffices to define formulas Count_i, $0 \le i \le a$, such that

(1) Each Count_i has exactly $n+1$ free variables $x_0, \ldots, x_n$.

(2) Count_i gives bit i (i.e., the place for 2^i) of the binary representation of the number of x_j that are true.

(3) The formulas Count_i behave correctly with regard to basic "intensional" properties of counting in binary, and these properties have polynomial-size Frege proofs.

(4) The size of Count_i is polynomially bounded in n.

The idea is that for the $x_0, \ldots, x_n$ we are going to substitute formulas for "$f(0) \le \ell, \ldots, f(n) \le \ell$." Then the resulting formulas Count^ℓ_i define the bits m^ℓ_i.

Now properties (1), (2), and (4) are immediate to arrange because each Count_i is a symmetric function of $n+1$ inputs, and every such function belongs to (non-uniform) NC^1. It remains to establish (3), and for this we have to say more concretely *how* Count_i is defined.

First idea: Define the sum of two a-bit numbers n_y and n_z, represented by sequences of formulas

$$\begin{aligned} \hat{y} &= y_a, y_{a-1}, \ldots, y_0 \\ \hat{z} &= z_a, z_{a-1}, \ldots, z_0, \end{aligned}$$

by $\mathit{Add}_0(\hat{y}, \hat{z}) := y_0 \oplus z_0$, and for bits $i > 0$,

$$\mathit{Add}_i(\hat{y}, \hat{z}) := y_i \oplus z_i \oplus \mathit{Carry}_i,$$

where

$$\mathit{Carry}_i = \bigvee_{j<i} \left[y_j \wedge z_j \wedge \bigwedge_{j<k<i} (y_k \oplus z_k) \right].$$

The *problem* with this is that the "Carry" formulas have log depth in the big ANDs and ORs. To see the net effect of this, let us follow through with our intended use of the formulas Add_i. This is a kind of "divide and conquer": For all i, j define the number

$$A^{ij} := \big|\big|\{x_k : x_k = \top \wedge j \cdot 2^i \le k < (j+1) \cdot 2^i\}\big|\big|.$$

In other words, this is the number of x_k that are true in the jth segment of size 2^i in $0\ldots n$. We can count up all these segments by "divide and conquer" recursion with basis

$$A^{0j} = \text{if } x_j \text{ then } 1 \text{ else } 0,$$

and induction

$$A^{i+1,j} = A^{i,2j} + A^{i,2j+1}.$$

To express A^{ij} by propositional formulas, we use the same encoding scheme as above, seeking formulas a_b^{ij} to represent the bits of A^{ij} in binary by the sequence

$$\hat{a}^{ij} = a_i^{ij}, \ldots, a_b^{ij}, \ldots, a_0^{ij}.$$

For $i = 0$, we have that a_0^{0j} is just x_j itself, and for bits $b > 0$, $a_b^{0j} = \bot$ (where we might encode $\bot$ by $(p_1 \wedge \neg p_1)$, say). Then the induction case is represented by

$$a_b^{i+1,j} = Add_b(\hat{a}^{i,2j}, \hat{a}^{i,2j+1}). \qquad (5)$$

Then $Count_i$ is given by a_i^{a0}, i.e., by $a_i^{\log_2 n,0}$.

Alas, when we analyze the size blowup of the recursion (5), we get the following: Each formula a_b^{ij} appears $\Theta(\log^2 n)$ times in $a_{b'}^{i+1,j'}$, for all $b' \geq b$, where $j' = \lfloor j/2 \rfloor$. Since we need to recurse up to $i = \log n$, we have in all $(\Theta(\log^2 n))^{\log n})$ occurrences of the base variables x_j. However, this equals $n^{\Theta(loglogn)}$, which is super-polynomial. Hence this scheme using the straightforward log-depth definition of Add_b is too big.

What we need to do to achieve polynomial size is to get the number of occurrences of variables "like" a_b^{ij} in something like (5) down from $O(\log^2 n)$ to *constant*; even $O(\log n)$ is not good enough. If our addition formula were constant-depth, we'd be OK. There are a couple ways to get the effect of constant=depth addition circuits. We shall use "carry-save-addition" circuits in our proof; but another possibility would be the use of a "redundant" or "signed bit" notation for integers instead of standard binary (cf. the results on constant-depth addition in these notations by Borodin, Cook, and Pippenger [7]). Yet another possibility would be to use a "ripple-carry circuit" for addition; this would work since the numbers concerned have size $O(\log n)$ bits.[5]

The idea used in the present proof is to achieve the same effect via *carry-save addition* (CSA), a trick that goes back to the 1960s. This converts three numbers n_0, n_1, n_2 into two numbers m_0, m_1 such that $n_0 + n_1 + n_2 = m_0 + m_1$. Each bit b of m_0 is the bitwise sum mod 2 of the bits b of n_0, n_1, and n_2, while bit b of m_1 is 1 if the sum of these bits is 2 or 3 (which would be the carry propagated into the next column in the standard

[5]This was an oversight in the original proof in [11]; however, there are still substantial advantages to the use of carry-save addition, since they allow addition of a vector of n-bit integers in logarithmic depth.

way of summing $n_0 + n_1 + n_2$), and 0 otherwise. Put another way, $m_{1b}m_{0b}$ equals $n_{0b} + n_{1b} + n_{2b}$ as a binary number between 0 and 3. We may also define "$m_{1,-1}$" to be 0.

Now define the following "double-barreled" recursion:

$$C^{0,j} = 0, \qquad S^{0,j} = \text{if } x_j \text{ then } 1 \text{ else } 0.$$

and for $1 \leq i \leq a-1$,

$$(C^{i+1,j}, S^{i+1,j}) = \mathit{CSA}(\mathit{CSA}(C^{i,2j}, S^{i,2j}, C^{i,2j+1}), S^{i,2j+1}). \tag{6}$$

The effect of the two applications of CSA is to give a constant-depth adder from four numbers into two. Finally, $C^{ij} + S^{ij}$ gives the desired quantity A^{ij}.

Now do the propositional translations $\hat{s}^{ij}$ and $\hat{c}^{ij}$ of C^{ij} and S^{ij} as before. The payoff is that in the propositional translation of (6), each s_b^{ij} and c_b^{ij} appears some constant k number of times in $s_b^{i+1,j'}$ and $c_{b+1}^{i+1,j'}$, where $j' = \lfloor j/2 \rfloor$ as before. Hence the base variables x_j appear at most $k^{\log n} = n^k$ times in $s_b^{\log n,0}$ and $c_b^{\log n,0}$. Finally, the propositional translation $\hat{a}^{ij}$ of A^{ij} is obtained by one application of the original $\mathit{Add}_b(\cdot,\cdot)$ formulas. Then from $\hat{a}^{ij}$ we obtain propositional representations of Count_i as before, and these have polynomial size.

It remains to verify that the formulas Count_i thus obtained are equivalent to the earlier definition in terms of the quantities A^{ij}, and that the nice properties listed above carry through. Then polynomial-sized Frege proofs of the sequence of inequalities for the M_ℓ can be put together. The details are long but straightforward and not so interesting, hence best omitted. $\square$

5.1 A note on circuit complexity and proof-system simulations

There is a natural connection between extended Frege systems and Boolean circuits, since both models allow the introduction of "abbreviations." The formulas in any line of an $e\mathcal{F}$-proof can be transformed into circuits, with the idea that the symbols that come from extension rules correspond to gates with fan-out greater than 1. Conversely a circuit can be transformed into a formula in an extended Frege proof by introducing new variables for each internal node of the circuit. Since the Circuit Value Problem is complete for the class P [33], one would expect some relationship between $e\mathcal{F}$-proofs and polynomial time computations.

Similarly Frege proofs can be connected to Boolean formulas, which are the same as Boolean circuits of fan-out 1 at each gate. Since the Boolean Formula Value Problem is in NC^1 [10, 17, 12], one tends to associate Frege-proofs with NC^1 computations. It would be very interesting to formalize these connections, showing for example that a polynomial simulation of $e\mathcal{F}$ proofs by $\mathcal{F}$ proofs is possible if and only if the complexity classes P and NC^1 coincide (in the nonuniform case).

There are no formal theorems behind the intuitions of the previous two paragraphs; it is merely a good motivational idea, which has proved fruitful in suggesting when contructions in circuit complexity can be transferred to lower bounds in proof complexity (the lower bound proofs often need substantial modification to be transferred in this way).

6 More Propositional Proof Systems and Simulations

In this section we consider variations of the Frege proof systems that arise when allowing different substitution rules. As we will see, all these systems are polynomially equivalent to the $e\mathcal{F}$ proof systems.

Definition A substitution Frege ($s\mathcal{F}$) proof system is a Frege system augmented with the substitution rule

$$\frac{\varphi(p)}{\varphi(\psi)}$$

that indicates that every occurence of the variable p in the formula φ, is replaced by the formula ψ.

As the next result shows, Frege systems with substitution are as powerful as extended Frege systems.

Theorem 9 *Given a $s\mathcal{F}$ and an $e\mathcal{F}$ system with the same language, then*

(a) [20] *the $s\mathcal{F}$ system p-simulates the $e\mathcal{F}$ system.*

(b) [23, 31] *the $e\mathcal{F}$ system p-simulates the $s\mathcal{F}$ system.*

Proof **(a)** Let A be a formula and P be an $e\mathcal{F}$ proof of A using the extension rules $p_1 \equiv B_1, \ldots p_k \equiv B_k$. Let $P = A_1, \ldots, A_n$ with $A_n = A$. It is not hard to see that for every A_j, $1 \leq j \leq n$, there is a polynomial size Frege proof of the formula

$$\left(\bigwedge_{i=1}^{k} p_i \equiv B_i \right) \rightarrow A_j.$$

In particular there is a Frege proof Q of size $O(|P|^3)$ of

$$\left(\bigwedge_{i=1}^{k} p_i \equiv B_i \right) \rightarrow A.$$

We can suppose that the $p_i \equiv B_i$ are numbered in reverse order of how they appear in P so that p_i does not appear in B_j for $j > i$. The last line of Q can be written as

$$(p_1 \equiv B_1) \wedge \left(\bigwedge_{i=2}^{k} p_i \equiv B_i \right) \rightarrow A,$$

substituting B_1 for p_1 and removing $B_1 \equiv B_1$ one gets

$$\left(\bigwedge_{i=2}^{k} p_i \equiv B_i \right) \rightarrow A.$$

This process can be repeated k times until A is obtained, and therefore this formula has a substitution Frege proof of size $O(|P|^3)$.

(b) Let $\varphi_1, \varphi_2, \ldots, \varphi_r$ be an $s\mathcal{F}$-proof of φ_r with propositional variables $p_1, \ldots, p_m$. We need to construct an $e\mathcal{F}$ proof of φ_r. The $e\mathcal{F}$-proof will have the set of extension variables $q_1^{i,k}, \ldots, q_m^{i,k}$ for $1 \leq i \leq k \leq r$, and will prove successively the formula

$$\neg\varphi_r \rightarrow \bigvee_{i=1}^{k} \neg\varphi_i(\vec{q}^{\,i,k})$$

for $k = r, r-1, \ldots, 2, 1$. This will suffice since when $k = 1$ we have an $e\mathcal{F}$-proof ending with

$$\neg\varphi_r \rightarrow \neg\varphi_1(\vec{q}^{\,1,1}),$$

and since $\varphi_1(\vec{q}^{\,1,1})$ is an axiom instance, φ_r follows in a few more steps.

We define next the extension rules for the variables $q_j^{i,k}$. This is done successively for $k = r, r-1, \ldots, 1$, and whenever we define the extension rule for $q_j^{i,k}$ we suppose that we have a short $e\mathcal{F}$-proof for

$$\neg\varphi_r \rightarrow \bigvee_{i=1}^{k+1} \neg\varphi_i(\vec{q}^{\,i,k+1}).$$

When $k = r$, $q^{i,k}$ is defined by the rule

$$q_j^{r,r} \equiv p_j$$

$$q_j^{i,r} \equiv \bot \quad \text{for } i < r.$$

Obviously

$$\neg\varphi_r \rightarrow \neg\varphi_r(\vec{q}^{\,r,r}).$$

For $k < r$ we have three cases depending on how φ_{k+1} has been obtained.

Case 1: If φ_{k+1} is an axiom then let

$$q_j^{i,k} \equiv q_j^{i,k+1} \quad \text{for all } i, j.$$

$\varphi_{k+1}(\vec{q}^{k+1,k+1})$ has an $e\mathcal{F}$-proof of one line, and the hypothesis

$$\neg\varphi_r \to \bigvee_{i=1}^{k+1} \neg\varphi_i(\vec{q}^{i,k+1})$$

immediately implies

$$\neg\varphi_r \to \bigvee_{i=1}^{k} \neg\varphi_i(\vec{q}^{i,k}).$$

Case 2: If φ_{k+1} is inferred by substitution

$$\frac{\varphi_l(p_1,\ldots,p_m)}{\varphi_{k+1}(\psi_1,\ldots,\psi_m)}$$

then define $q_j^{i,k}$ as follows: For $i \neq l$

$$q_j^{i,k} \equiv q_j^{i,k+1}$$

and for $i = l$,

$$q_j^{l,k} \equiv (\neg\varphi_l(\vec{q}^{l,k+1}) \wedge q_j^{l,k+1}) \vee (\varphi_l(\vec{q}^{l,k+1}) \wedge \psi_j(\vec{q}^{k+1,k+1})).$$

Put another way, the definition of $q_j^{i,k}$ in the case $i = l$ is:

$$q_j^{l,k} \equiv \begin{cases} q_j^{l,k+1} & \text{if } \neg\varphi_l(\vec{q}^{l,k+1}) \\ \\ \psi_j(\vec{q}^{k+1,k+1}) & \text{otherwise.} \end{cases}$$

By the way extensions are defined, from the formula $\varphi_l(\vec{q}^{l,k})$ we could infer $\varphi_l(\vec{q}^{l,k+1})$ and $\varphi_l(\psi_j(\vec{q}^{k+1,k+1}))$, but this last formula equals $\varphi_{k+1}(\vec{q}^{k+1,k+1})$. This establishes the implication

$$\neg\varphi_{k+1}(\vec{q}^{k+1,k+1}) \to \neg\varphi_l(\vec{q}^{l,k}).$$

Since $l \leq k$, this combines with the induction hypothesis to yield a short $e\mathcal{F}$ proof of

$$\neg\varphi_r \to \bigvee_{i=1}^{k} \neg\varphi_i(\vec{q}^{i,k}).$$

Case 3: If φ_{k+1} is inferred by modus ponens

$$\frac{\varphi_l \quad \varphi_{l'}}{\varphi_{k+1}}$$

then for $i \neq l$ and $i \neq l'$ define

$$q_j^{i,k} \equiv q_j^{i,k+1},$$

and in case either $i = l$ or $i = l'$, define

$$q_j^{i,k} \equiv \begin{cases} q_j^{i,k+1} & \text{if } \neg\varphi_i(\vec{q}^{\,i,k+1}) \\ & \\ q_j^{k+1,k+1} & \text{otherwise.} \end{cases}$$

By the way extensions are defined, from $\varphi_l(\vec{q}^{\,l,k}) \wedge \varphi_{l'}(\vec{q}^{\,l',k})$, we get $\varphi_l(\vec{q}^{\,k+1,k+1}) \wedge \varphi_{l'}(\vec{q}^{\,k+1,k+1})$, which entails $\varphi_{k+1}(\vec{q}^{\,k+1,k+1})$. Thus we have

$$\neg\varphi_{k+1}(\vec{q}^{\,k+1,k+1}) \rightarrow \neg\varphi_l(\vec{q}^{\,l,k}) \vee \neg\varphi_{l'}(\vec{q}^{\,l',k}),$$

and since l and l' are smaller than $k+1$, this combines with the induction hypothesis to yield an $e\mathcal{F}$-proof of

$$\neg\varphi_r \rightarrow \bigvee_{i=1}^{k} \neg\varphi_i(\vec{q}^{\,i,k}).$$

□

The above theorem shows that the substitution rule is rather powerful. The next two results show that the whole power of this rule can be achieved using two weak forms of substitution, True-False substitution and variable substitution.

Definition The True-False substitution rules,

$$\frac{\varphi(p)}{\varphi(\top)} \quad \text{and} \quad \frac{\varphi(p)}{\varphi(\bot)}$$

allow the substitution of the nullary constants $\top$ and $\bot$ (true or false) for the variable p in the formula φ.

Theorem 10 ([13]) *A Frege system augmented with T-F-substitution can p-simulate extended Frege systems.*

Proof The idea is that for any formula φ, and any variable p in φ, T-F-substitution can simulate in a polynomial number of steps any substitution of a formula ψ for p. This can be done obtaining the formulas $\varphi(\top)$ and $\varphi(\bot)$ (using two T-F-substitutions), and deriving the valid formulas $\psi \wedge \varphi(\top) \rightarrow \varphi(\psi)$ and $\neg\psi \wedge \varphi(\bot) \rightarrow \varphi(\psi)$. These can be derived using a number of lines that is linearly bounded in the number of connectives in $\varphi(p)$. From these formulas $\varphi(\psi)$ can be easily inferred in a constant number of lines. □

Another type of substitution that seems weak but is as powerful as the general substitution is the renaming rule. This rule allows to rename and identify different variables.

Definition The renaming rule is

$$\frac{\varphi(p)}{\varphi(q)}$$

(p and q are propositional variables).

Theorem 11 ([13]) *A Frege system augmented with the renaming rule can p-simulate extended Frege systems.*

Proof By the previous theorem it suffices to show that a Frege system with renaming can simulate Frege proofs with T-F-substitutions. Let A be a formula and P be a T-F-substitution Frege proof of A, and let $p_1, \ldots, p_k$ be the variables that appear in P. One can first prove without renaming the formulas $(p_1 \wedge \ldots \wedge p_k) \rightarrow A(\vec{p})$ and $(\neg p_1 \wedge \ldots \wedge \neg p_k) \rightarrow A(\vec{p})$. This can be done proving first the formula $A(\top, \ldots, \top)$ (for the first case) that does not have variables and therefore a proof only needs to prove its true subformulas and disprove the false ones. Then one can prove the formula $(p_1 \wedge \ldots \wedge p_k) \wedge A(\top, \ldots, \top) \rightarrow A(\vec{p})$. From these two, $(p_1 \wedge \ldots \wedge p_k) \rightarrow A(\vec{p})$ can be inferred in a constant number of steps.

Let D be the formula

$$\neg(p_1 \wedge \ldots \wedge p_k) \wedge (p_1 \vee \ldots \vee p_k).$$

We show that there is a Frege proof with renaming of the formula $D \rightarrow A$, (from this and the proofs of the formulas considered before, we can conclude that there is a proof for A). To do this we construct a new proof P' to simulate the proof P by proving $D \rightarrow B$ for each formula B in P. If B is inferred in P by a Frege inference, then $D \rightarrow B$ can easily be inferred from previous lines in P'. The remaining case is when B comes from a T-F-substitution like

$$\frac{B(p_i)}{B(\top)}.$$

By hypothesis that P' contains $D \rightarrow B(p_i)$ and we have to infer $D \rightarrow B(\top)$. Doing $k-1$ renaming inferences we can get $D(p_i/p_j) \rightarrow B(p_j)$ for all $j \neq i$. ($D(p_i/p_j)$ represents the replacement of p_i with p_j in D). Then one can get proofs for the formulas $p_j \wedge B(p_j) \rightarrow B(\top)$. Combining these one can infer $D_i \rightarrow B(\top)$, where D_i is the formula

$$\neg(p_1 \wedge \ldots \wedge p_{i-1} \wedge p_{i+1} \wedge \ldots \wedge p_k) \wedge (p_1 \vee \ldots \vee p_{i-1} \vee p_{i+1} \vee \ldots \vee p_k).$$

(Observe that $D(p_i/p_j)$ is equivalent to D_i). Now one can easily infer

$$D \wedge \neg D_i \rightarrow B(\top)$$

since it is easy to prove that the hypothesis of this formula holds only for two possible values of the variables and since $B(\top)$ is valid. With this and $D_i \to B(\top)$, $D \to B(\top)$ can be inferred in a constant number of lines. □

6.1 Tree-like versus non-tree-like proofs

A tree-like proof is one in which intermediate results (formulas) are used as hypotheses for an inference only once. As the next theorem shows, in the case of Frege systems, non-tree-like proofs can be transformed into tree-like without changing the size very much.

Theorem 12 ([28]) *If A has a (non-tree-like) $\mathcal{F}$-proof on size n then A has a tree-like $\mathcal{F}$-proof of size $p(n)$ for some polynomial p.*

Proof (Sketch) If the $\mathcal{F}$-proof has lines $B_1, \ldots, B_k = A$, the idea is to form a tree-like $\mathcal{F}$-proof which proves B_1, then $B_1 \wedge B_2$, then $B_1 \wedge B_2 \wedge B_3, \ldots$, until $B_1 \wedge B_2 \wedge \ldots \wedge B_k$ is proved. □

Bonet in [3] improved Krajíček's construction by giving better bounds on the size of the tree-like proof.

6.2 Best known lower bounds for (extended) Frege proofs

Theorem 13 *(a) Let $\mathcal{F}$ be an arbitrary Frege system. There are tautologies that require quadratic size $\mathcal{F}$ proofs.*

(b) Same holds for $e\mathcal{F}$ proof systems.

These results follow from the following theorem.

Theorem 14 ([13]) *Let A have n connectives and let m be the sum of the sizes of the subformulas of A without repetition, and suppose also that A is not an instance of a shorter tautology. Then*

(a) Any extended Frege proof of A has $\Omega(m)$ symbols.

(b) Any Frege proof has $\Omega(n)$ lines.

Proof (Sketch) Consider the formula A

$$\bot \vee (\bot \vee (\bot \vee (\ldots \vee (\bot \vee \top)) \ldots))$$

with n disjunctions. Each subformula of A must be "active" in any proof, where "active" means its principal connective is used by some inference. Otherwise, the subformula could be replaced everywhere by $\bot$, and we would still have a valid proof, but of a non-tautology. Each inference makes only a constant number of subformulas active, so proof size must be

$$\Omega(\sum ||\{B : B \text{ is a subformula of } A\}||) = \Omega(m).$$

□

Although the formula A in the above proof is an easy one, in fact these are the best known lower bounds on Frege proof size. These results constrast with the exponential lower bounds that are known for other (weaker) proof systems, like resolution.

Theorem 15 ([25]) *There is some constant $c > 0$ such that any resolution proof of the pigeonhole principle PHP_n requires at least 2^{cn} lines.*

Since as we have seen, PHP_n has polynomial size Frege proofs we get

Corollary 16 *Resolution does not p-simulate Frege proof systems.*

Exponential lower bound are known for Frege proof systems of constant depth. In a language $\{\wedge, \vee, \neg\}$; the depth of a formula is the number of alternations of $\wedge$'s and $\vee$s (after $\neg$ has been pushed to the literals). A constant depth Frege proof is one in which the formula depth is bounded by a constant. The following was independently obtained by Krajíček, Pudlák, and Woods, and by Pitassi, Beame, and Impagliazzo.

Theorem 17 ([32, 36]) *Depth d Frege proofs of PHP_n require $\Omega(2^{n^{\frac{1}{6^d}}})$ symbols.*

The next result shows that a Frege proof for a formula can be transformed into a new one where the symbol-size is related to line-size and the depth of the original proof.

Theorem 18 ([13]) *A depth d Frege proof with m lines can be transformed into a depth d Frege proof with $O(m^d)$ symbols.*

Corollary 19 *PHP_n requires depth d Frege proofs of $\Omega(2^{n^{\frac{1}{6^d}}})$ lines.*

Part II
Interpolation Theorems for Propositional Logic[6]

7 Introduction to Craig Interpolation

Let $\vec{p}$, $\vec{q}$ and $\vec{r}$ be vectors of variables and let $A(\vec{p},\vec{q})$ and $B(\vec{p},\vec{r})$ be two propositional formulas involving only the indicated variables. Suppose that $A(\vec{p},\vec{q})$ implies $B(\vec{p},\vec{r})$. Since B does not involve any variable from $\vec{q}$, whatever A says about $\vec{p}$ should be sufficient to imply B. This intuition is formalized by the Interpolation Theorem.

Theorem 20 *Let $A(\vec{p},\vec{q})$ and $B(\vec{p},\vec{r})$ be propositional formulas involving only the indicated variables. Also, suppose that $A(\vec{p},\vec{q}) \rightarrow B(\vec{p},\vec{r})$ is a tautology. Then there is a propositional formula $C(\vec{p})$ not involving q's and r's such that*

$$A(\vec{p},\vec{q}) \rightarrow C(\vec{p}) \text{ and } C(\vec{p}) \rightarrow B(\vec{p},\vec{r})$$

are tautologies.

Proof Let $\tau_1,\ldots,\tau_n$ be the truth assignments to $p_1,\ldots,p_k$ for which it is possible to make $A(\vec{p},\vec{q})$ true by further assignment of truth values to $\vec{q}$. Let $C(\vec{p})$ say that one of $\tau_1,\ldots,\tau_n$ holds for $\vec{p}$, i.e.,

$$C(\vec{p}) = \bigvee_{i=1}^{n} (p_1^{(i)} \wedge p_2^{(i)} \wedge \cdots \wedge p_k^{(i)})$$

where

$$p_j^{(i)} = \begin{cases} p_j & \text{if } \tau_i(p_j) = \text{True} \\ \neg p_j & \text{if } \tau_i(p_j) = \text{False} \end{cases}$$

Then, clearly, $A(\vec{p},\vec{q}) \models C(\vec{p})$.

On the other hand, a truth assignment to $\vec{p}$ that satisfies $C(\vec{p})$ can be extended to a truth assignment to $\vec{p},\vec{q}$ that satisfies $A(\vec{p},\vec{q})$. Since $A(\vec{p},\vec{q}) \models B(\vec{p},\vec{r})$, every extension of this truth assignment to $\vec{p},\vec{q},\vec{r}$ must satisfy $B(\vec{p},\vec{r})$. Therefore, $C(\vec{p}) \models B(\vec{p},\vec{r})$. □

A stronger version of the interpolation theorem was proved by Craig for first-order logic [22]. Also, note that the above proof allows $C(\vec{p})$ to be exponentially big, as a function of the sizes of $A(\vec{p},\vec{q})$ and $B(\vec{p},\vec{r})$.

The following example exhibits a relationship between the size of interpolants and complexity theory.

[6]Part II is based on notes prepared by E. Allender and A. Maciel based on lectures delivered by S. Buss on March 6, 1995 at the McGill University Bellair's Research Institute in Holetown, Barbados.

Example 21 Consider the language FACT of all pairs (X, U) where X is the binary representation of a number and U is the prefix of some prime V that divides X. Let $p_1, \ldots, p_k$ code a pair (X, U). Let $A(\vec{p}, \vec{q})$ be a formula that is satisfiable if and only if $(X, U) \in \mathrm{FACT}$. For example, $A(\vec{p}, \vec{q})$ says that (i) $\vec{q}$ codes numbers $V_1, \ldots, V_m$ and Pratt primality witnesses for each V_i, and that (ii) $V_1 \cdots V_m = X$ and U is the prefix of some V_i. Let $B(\vec{p}, \vec{r})$ be a formula that is satisfiable if and only if $(X, U) \notin \mathrm{FACT}$. For example, $B(\vec{p}, \vec{r})$ says that (i) $\vec{r}$ codes numbers $V_1, \ldots, V_m$ and Pratt primality witnesses for each V_i, and that (ii) $V_1 \cdots V_m = X$ and U is not the prefix of any of the V_i. Then, $(X, U) \in \mathrm{FACT} \Leftrightarrow \exists \vec{q} A(\vec{p}, \vec{q}) \Leftrightarrow \neg \exists \vec{r} B(\vec{p}, \vec{r})$, so that

$$A(\vec{p}, \vec{q}) \to \neg B(\vec{p}, \vec{r})$$

is a tautology. Therefore, an interpolant $C(\vec{p})$ must express $(X, U) \in \mathrm{FACT}$, since $(X, U) \in \mathrm{FACT} \Rightarrow \exists \vec{q} A(\vec{p}, \vec{q}) \Rightarrow C(\vec{p})$ and $C(\vec{p}) \Rightarrow \forall \vec{r} \neg B(\vec{p}, \vec{r}) \Rightarrow (X, U) \in \mathrm{FACT}$. □

As a consequence, a polynomial upper bound on the size of interpolants for propositional logic would immediately translate into a polynomial upper bound on the size of formulas or circuits for the language FACT, depending on how the size of an interpolant is defined. Note that $\mathrm{FACT} \in NP \cap \mathrm{coNP}$; this is easily seen by using the formulas $A(\vec{p}, \vec{q})$ and $B(\vec{p}, \vec{r})$ above. On the other hand, it is not known if FACT has polynomial-size formulas or circuits. In fact, if this language has polynomial-size circuits, then so does factoring. The conjecture would therefore be that $\mathrm{FACT} \notin P/poly$. Generalizing this example gives the following:

Theorem 22 ([34]) *If there is a polynomial upper bound on the circuit size of interpolants in propositional logic, then*

$$\mathrm{NP/poly} \cap \mathrm{coNP/poly} = \mathrm{P/poly}.$$

Proof Let $\exists \vec{q} A(\vec{p}, \vec{q})$ express an NP/poly property $R(\vec{p})$ and let $\forall \vec{r} B(\vec{p}, \vec{r})$ express the same property in coNP/poly form. Then

$$\exists \vec{q} A(\vec{p}, \vec{q}) \Rightarrow \forall \vec{r} B(\vec{p}, \vec{r}),$$

which is equivalent to

$$A(\vec{p}, \vec{q}) \to B(\vec{p}, \vec{r})$$

being a tautology. Let $C(\vec{p})$ be a polynomial circuit size interpolant such that

$$A(\vec{p}, \vec{q}) \to C(\vec{p}) \text{ and } C(\vec{p}) \to B(\vec{p}, \vec{r})$$

are tautologies. Thus

$$\exists \vec{q} A(\vec{p}, \vec{q}) \Rightarrow C(\vec{p}) \text{ and } C(\vec{p}) \Rightarrow \forall \vec{r} B(\vec{p}, \vec{r}),$$

i.e., $R(\vec{p}) \Leftrightarrow C(\vec{p})$. Therefore, $R(\vec{p})$ has a polynomial-size circuit and thus is in P/poly. □

In the remainder of this lecture, we prove upper bounds on the size of interpolants in two restricted cases: when $A(\vec{p},\vec{q}) \to B(\vec{p},\vec{r})$ has a short cut-free proof in the propositional sequent calculus, and when $A(\vec{p},\vec{q}) \to B(\vec{p},\vec{r})$ has a short resolution refutation. We also prove a monotone version of the latter and one for resolution with limited extension.

8 The propositional sequent calculus

Let propositional formulas be built with the connectives $\top$, $\neg$, $\wedge$ and $\vee$ for true, negation, conjunction and disjunction. A *sequent* is of the form

$$A_1, A_2, \ldots, A_k \to B_1, B_2, \ldots, B_l$$

where $k \geq 0$, $l \geq 0$ and the A_i's and B_j's are formulas. The intended meaning of the sequent is

$$(A_1 \wedge A_2 \wedge \cdots \wedge A_k) \to (B_1 \vee B_2 \vee \cdots \vee B_l).$$

(An empty conjunction has value true; an empty disjunction has value false.)

The propositional sequent calculus PK is a proof system in which each line in a proof is a sequent. The *axioms* or *initial sequents* are

(1) $p \to p$, p any propositional variable

(2) $\to \top$

The rules of inference of PK are as follows. Γ and Δ represent formulas separated by commas (*cedents*).

(1) *Weak structural rules:*

$$\frac{\Gamma \to \Delta}{\Gamma' \to \Delta'}$$

provided every formula in Γ and Δ appears also in Γ' and Δ', respectively.

(2)

$$\wedge\text{:left}\ \frac{A, B, \Gamma \to \Delta}{A \wedge B, \Gamma \to \Delta} \qquad \wedge\text{:right}\ \frac{\Gamma \to \Delta, A \qquad \Gamma \to \Delta, B}{\Gamma \to \Delta, A \wedge B}$$

(3)

$$\vee\text{:left}\ \frac{A, \Gamma \to \Delta \qquad B, \Gamma \to \Delta}{A \vee B, \Gamma \to \Delta} \qquad \vee\text{:right}\ \frac{\Gamma \to \Delta, A, B}{\Gamma \to \Delta, A \vee B}$$

(4)

$$\neg\text{:left}\ \frac{\Gamma \to \Delta, A}{\neg A, \Gamma \to \Delta} \qquad \neg\text{:right}\ \frac{A, \Gamma \to \Delta}{\Gamma \to \Delta, \neg A}$$

(5) *Cut rule:*

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

PK is sound: if $\mathrm{PK} \vdash \Gamma \to \Delta$, then $\Gamma \to \Delta$ is valid. PK is also complete: if $\Gamma \to \Delta$ is valid, then there is a PK proof of $\Gamma \to \Delta$. In fact, PK is cut-free complete: if $\Gamma \to \Delta$ is valid, then there is a cut-free PK proof of $\Gamma \to \Delta$. Obviously, cut-free completeness implies completeness. The cut-free completeness of PK can be proved by induction on the number of propositional connectives occurring in $\Gamma \to \Delta$. (See [15].)

Note that $\land$:left and $\lor$:right are dual. The same is true of $\lor$:left and $\land$:right. In $\Gamma \to \Delta$, Γ is called the *antecedent*, and Δ is called the *succedent.* Let $V(A)$ denote the set of variables occurring in A. Let $|A|$ denote the number of symbols in formula A, and let $|A|_{\mathrm{dag}}$ denote that number of symbols in the circuit form of A. Let $|P|$ and $|P|_{\mathrm{dag}}$ denote the number of sequents in the tree form and in the circuit form of a proof P.

9 Interpolation for cut-free proofs

Theorem 23 *Let P be a cut-free PK proof of $A \to B$ where $V(A) \subseteq \{\vec{p}, \vec{q}\}$ and $V(B) \subseteq \{\vec{p}, \vec{r}\}$. Then, there is an interpolant C such that*

(i) $A \to C$ *and* $C \to B$ *are valid,*

(ii) $V(C) \subseteq \{\vec{p}\}$,

(iii) $|C| \leq 2|P|$ *and* $|C|_{\mathrm{dag}} \leq 2|P|_{\mathrm{dag}}$.

Therefore, tree-like cut-free proofs have interpolants of polynomial formula size and general cut-free proofs have interpolants of polynomial circuit size.

Note that the proof below will not only show that $A \to C$ and $C \to B$ are valid, but that $A \to C$ and $C \to B$ have short cut-free PK proofs. In addition, the theorem also holds for proofs P that have cuts only on formulas D such that $V(D) \subseteq \{\vec{p}, \vec{q}\}$ or $V(D) \subseteq \{\vec{p}, \vec{r}\}$. On the other hand, it is not known if similar bounds hold on the size of interpolants for general PK proofs.

Proof We will prove a slightly more general statement:

If P is a cut-free PK proof of $\Gamma \to \Delta$, if Γ is $\Gamma_1 \cup \Gamma_2$ and Δ is $\Delta_1 \cup \Delta_2$ (possibly reordered), if $V(\Gamma_1, \Delta_1) \subseteq \{\vec{p}, \vec{q}\}$ and $V(\Gamma_2, \Delta_2) \subseteq \{\vec{p}, \vec{r}\}$, then there is an interpolant C such that

(i) $\Gamma_1 \to \Delta_1, C$ *and* $C, \Gamma_2 \to \Delta_2$ *are valid,*

(ii) $V(C) \subseteq \{\vec{p}\}$,

(iii) $|C| \leq 2|P|$ *and* $|C|_{\mathrm{dag}} \leq 2|P|_{\mathrm{dag}}$.

The proof will be by induction on the number of inferences in P.

Base case: No inferences. If the initial sequent is $q_i \to q_i$, then take C to be $(\neg\top)$, since $q_i \to q_i, \neg\top$ and $\neg\top \to$ are valid. If the initial sequent is $r_i \to r_i$, then C can be $\top$. If the initial sequent is $p_i \to p_i$, then C will be $\top$, $(\neg\top)$, p_i, or $(\neg p_i)$ depending on how the two instances of p_i are split into $\Gamma_1, \Gamma_2, \Delta_1, \Delta_2$. For example, if Γ_1 and Δ_2 are p_i and Δ_1 and Γ_2 are empty, then C can be p_i. If the initial sequent is $\to \top$, then C will be $\top$ or $\neg\top$.

Induction step: Consider the last inference.

Case (1): The last inference is

$$\vee\text{:right}\ \frac{\Gamma \to \Delta, A, B}{\Gamma \to \Delta, A \vee B}$$

In this case, the interpolant for the upper sequent will still work for the lower sequent. By hypothesis, $A \vee B$ is either contained in Δ_1 or it is contained in Δ_2. If $A \vee B \in \Delta_1$, then an interpolant for the upper sequent is such that

$$\Gamma_1 \to \Delta_1^-, A, B, C \quad \text{and} \quad C, \Gamma_2 \to \Delta_2$$

are valid, where $\Delta_1^- = \Delta_1 - \{A \vee B\}$. If $A \vee B \in \Delta_2$, then an interpolant for the upper sequent is such that

$$\Gamma_1 \to \Delta_1, C \quad \text{and} \quad C, \Gamma_2 \to \Delta_2^-, A, B$$

are valid, where $\Delta_2^- = \Delta_2 - \{A \vee B\}$.

Case (2): The last inference is

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

If $A \wedge B \in \Delta_1$, apply the induction hypothesis twice to get interpolants C_A and C_B such that

$$\begin{array}{ll} \Gamma_1 \to \Delta_1^-, A, C_A \qquad & C_A, \Gamma_2 \to \Delta_2 \\[1ex] \Gamma_1 \to \Delta_1^-, B, C_B \qquad & C_B, \Gamma_2 \to \Delta_2 \end{array}$$

are valid. Now,

$$\frac{\Gamma_1 \to \Delta_1^-, A, C_A \vee C_B \qquad \Gamma_1 \to \Delta_1^-, B, C_A \vee C_B}{\Gamma_1 \to \Delta_1^-, A \wedge B, C_A \vee C_B}$$

and

$$\frac{C_A, \Gamma_2 \to \Delta_2 \qquad C_B, \Gamma_2 \to \Delta_2}{C_A \vee C_B, \Gamma_2 \to \Delta_2}$$

Therefore, $(C_A \vee C_B)$ is an interpolant.

If $A \wedge B \in \Delta_2$, apply the induction hypothesis twice to get interpolants C_A and C_B such that

$$\Gamma_1 \rightarrow \Delta_1, C_A \qquad\qquad C_A, \Gamma_2 \rightarrow \Delta_2^-, A$$

$$\Gamma_1 \rightarrow \Delta_1, C_B \qquad\qquad C_B, \Gamma_2 \rightarrow \Delta_2^-, B$$

are valid. Similarly to before,

$$\frac{C_A \wedge C_B, \Gamma_2 \rightarrow \Delta_2^-, A \qquad C_A \wedge C_B, \Gamma_2 \rightarrow \Delta_2^-, B}{C_A \wedge C_B, \Gamma_2 \rightarrow \Delta_2^-, A \wedge B}$$

and

$$\frac{\Gamma_1 \rightarrow \Delta_1, C_A \qquad \Gamma_1 \rightarrow \Delta_1, C_B}{\Gamma_1 \rightarrow \Delta_1, C_A \wedge C_B}$$

Therefore, $(C_A \wedge C_B)$ is an interpolant.

All other cases are handled similarly. In addition, the size bounds on C are easily verified. $\square$

10 Resolution

In this section we review the basic notions and definitions concerning resolution-based theorem proving.

Definition A *literal* is either a propositional variable p or a negated variable $\neg p$ (which will also be denoted $\overline{p}$).

Definition A formula in *CNF - Conjunctive Normal Form* is expressed as a set of *clauses* $\{C_1, \ldots, C_k\}$, where each C_i is a set of literals $\{l_1, \ldots, l_j\}$ interpreted as the disjunction of those literals, $l_1 \vee \ldots \vee l_j$ and the formula is interpreted as the conjunction of the clauses. We will assume throughout that no clause contains both p and $\overline{p}$.

Definition A *resolution derivation* from $C_1, \ldots, C_k$ is a sequence $D_1, \ldots, D_l$ such that each D_i is either

1. one of the C_i's or

2. of the form $D_i = D_{j_1} \cup D_{j_2} \setminus \{x_r, \overline{x_r}\}$ where D_{j_1} and D_{j_2} have $x_r, \overline{x_r}$ as their only pair of complementary literals, and both j_1 and j_2 are less than i. (D_i is said to be the result of *resolving* D_{j_1} and D_{j_2}.)

Definition A *refutation* is a derivation whose last element is $\emptyset$ (denoted by the symbol $\square$).
This empty clause has the meaning of False; the only way it can be obtained in a derivation is for $C_{j_1} = \{x\}, C_{j_2} = \{\overline{x}\}$ for some literal x and clearly $x \wedge \overline{x}$ cannot be True.

If φ is a formula in DNF, then we may abuse notation slightly and speak of $\neg\varphi$ as being in CNF (whereas it would be more precise to consider the formula in CNF that results by applying DeMorgan's laws to $\neg\varphi$).

Although this form of proof is limited, it is in fact widely used and is therefore of interest. The following well-known result shows that, at least for proving DNF tautologies, resolution is a complete proof system.

Theorem 24 *Let φ be in DNF. Then $\models \varphi$ (i.e., φ is a tautology) iff there is a resolution refutation of $\neg\varphi$.*

Note that every resolution refutation has an associated graph, with nodes labeled by clauses, and edges from C to D if clause D is the result of resolving C with another clause. A resolution refutation is said to be *tree-like* if this graph is a tree. Since any refutation can be made tree-like by merely repeating parts of the derivation if need be, tree-like resolution is also a complete proof system for proving tautologies in DNF.

11 Interpolation for resolution

In this section, we show that a theorem analogous to Theorem 23 holds also for resolution refutations. If we look at Theorem 20 and consider the special case where $A(\vec{p},\vec{q})$ is in CNF, and $B(\vec{p},\vec{r})$ is in DNF, then saying that $A(\vec{p},\vec{q}) \rightarrow B(\vec{p},\vec{r})$ is a tautology is equivalent to saying that a set of clauses of the form $\{A_i(\vec{p},\vec{q})\} \cup \{B_j(\vec{p},\vec{r})\}$ is unsatisfiable. Thus the interpolant guaranteed by Theorem 20 has the form mentioned in the following theorem.

Theorem 25 ([37, 29]) *Let $\Gamma = \{A_i(\vec{p},\vec{q})\} \cup \{B_j(\vec{p},\vec{r})\}$ have a resolution refutation of length n. Then there is an interpolant $C(\vec{p})$ such that*

$$\bigwedge_i A_i(\vec{p},\vec{q}) \Rightarrow C(\vec{p})$$

and

$$C(\vec{p}) \Rightarrow \neg \bigwedge_j (B_j(\vec{p},\vec{r}))$$

and $C(\vec{p})$ has circuits of size $\leq 3n$.

If the refutation is tree-like, then $C(\vec{p})$ will have formulae of size $O(n)$.

Proof For each expression E in our resolution refutation, we will have a gate C_E in our circuit. (The circuit $C_\Box$ for the final clause in the refutation will be the circuit $C(\vec{p})$.)

1. If $E = A_i(\vec{p},\vec{q})$, then C_E is the constant 0.
2. If $E = B_j(\vec{p},\vec{r})$, then C_E is the constant 1.

3. If $E = F \cup G$ from $(F \cup \{p_i\}, G \cup \{\overline{p_i}\})$, then

$$C_E ::= (\overline{p_i} \wedge C_{F \cup \{p_i\}}) \vee (p_i \wedge C_{G \cup \{\overline{p_i}\}})$$

4. If $E = F \cup G$ from $(F \cup \{q_i\}, G \cup \{\overline{q_i}\})$, then

$$C_E ::= C_{F \cup \{q_i\}} \vee C_{G \cup \{\overline{q_i}\}}$$

5. If $E = F \cup G$ from $(F \cup \{r_i\}, G \cup \{\overline{r_i}\})$, then

$$C_E ::= C_{F \cup \{r_i\}} \wedge C_{G \cup \{\overline{r_i}\}}$$

(Note, circuit $C_\Box$ has inputs only for 0, 1, p_i, $\overline{p_i}$, as required.)

In order to finish the proof of the theorem, we must prove the following lemma. (This is clearly sufficient to prove the theorem, since the output gate of this circuit of $C_\Box$, and $\tau(\Box) = \bot$ for *every* truth assignment τ. Note that a truth assignment τ is the same as an input assignment to the circuit $C_\Box$.)

Lemma 26 *If τ is a truth assignment such that $\tau(E) = \bot$, then*

$$\tau(C_E) = \bot \Longrightarrow \exists i\ \tau(A_i) = \bot$$

$$\tau(C_E) = \top \Longrightarrow \exists i\ \tau(B_i) = \bot$$

(Note C_E does *not* compute expression E in general.)

Proof (of the lemma)

1. If $E = A_i(\vec{p}, \vec{q})$, then $\tau(C_E) = \bot$. The hypothesis of the lemma is that $\tau(A_i(\vec{p}, \vec{q}) = \bot$, so the claim holds trivially in this case.

2. $E = B_j(\vec{p}, \vec{r})$, then $\tau(C_E) = \top$ and $\tau(B_j(\vec{p}, \vec{r})) = \bot$, similar to case 1.

3. $E = F \cup G$ from $(F \cup \{p_i\}, G \cup \{\overline{p_i}\})$

$$C_E ::= (\overline{p_i} \wedge C_{F \cup \{p_i\}}) \vee (p_i \wedge C_{G \cup \{\overline{p_i}\}})$$

Since $\tau(E) = \bot$, we have that $\tau(F) = \tau(G) = \bot$

If $\tau(C_E) = \bot$, then

case a: $\tau(p_i) = \top$, then $\tau(C_{G \cup \{\overline{p_i}\}}) = \bot$ and $\tau(G \cup \{\overline{p_i}\}) = \bot$. By induction hypothesis, $\exists i\ \tau(A_i) = \bot$.

case b: $\tau(\overline{p_i}) = \top$, then $\tau(C_{F \cup \{p_i\}}) = \bot$ and $\tau(F \cup \{p_i\}) = \bot$. By induction hypothesis, $\exists i\ \tau(A_i) = \bot$.

If $\tau(C_E) = \top$, then

case a: $\tau(p_i) = \top$, then $\tau(C_{G\cup\{\overline{p_i}\}}) = \top$ and $\tau(G \cup \{\overline{p_i}\}) = \bot$. By induction hypothesis, $\exists i\ \tau(B_i) = \bot$.

case b: $\tau(\overline{p_i}) = \top$, then $\tau(C_{F\cup\{p_i\}}) = \top$ and $\tau(F \cup \{p_i\}) = \bot$. By induction hypothesis, $\exists i\ \tau(B_i) = \bot$.

4. $E = F \cup G$ from $(F \cup \{q_i\}, G \cup \{\overline{q_i}\})$

$$C_E ::= C_{F\cup\{q_i\}} \vee C_{G\cup\{\overline{q_i}\}}$$

Since $\tau(E) = \bot$, we know that $\tau(F) = \tau(G) = \bot$.

If $\tau(C_E) = \bot$, then $\tau(C_{F\cup\{q_i\}}) = \tau(C_{G\cup\{\overline{q_i}\}}) = \bot$.

case a: $\tau(q_i) = \top$, then $\tau(C_{G\cup\{\overline{q_i}\}}) = \bot$ and $\tau(G \cup \{\overline{q_i}\}) = \bot$. By induction hypothesis, $\exists i\ \tau(A_i) = \bot$.

case b: $\tau(\overline{q_i}) = \top$, then $\tau(C_{F\cup\{q_i\}}) = \bot$ and $\tau(F \cup \{q_i\}) = \bot$. By induction hypothesis, $\exists i\ \tau(A_i) = \bot$.

If $\tau(C_E) = \top$, then at least one of $\tau(C_{F\cup\{q_i\}})$ and $\tau(C_{G\cup\{\overline{q_i}\}})$ is true. Also, at least one of $\tau(F \cup \{q_i\})$ and $\tau(G \cup \{\overline{q_i}\})$ is true. Note that:

If $\tau(F \cup \{q_i\}) = \bot$ and $\tau(C_{F\cup\{q_i\}}) = \top$, then by inductive hypothesis, $\exists j\ B_j(\vec{p}, \vec{r}) = \bot$.

If $\tau(G \cup \{\overline{q_i}\}) = \bot$ and $\tau(C_{G\cup\{\overline{q_i}\}}) = \top$, then by inductive hypothesis, $\exists j\ B_j(\vec{p}, \vec{r}) = \bot$.

Thus we need only worry about the cases where

case a:

$$\tau(F \cup \{q_i\}) = \bot,\ \tau(C_{F\cup\{q_i\}}) = \bot;\ \tau(G \cup \{\overline{q_i}\}) = \top, \tau(C_{G\cup\{\overline{q_i}\}}) = \top$$

case b:

$$\tau(F \cup \{q_i\}) = \top,\ \tau(C_{F\cup\{q_i\}}) = \top;\ \tau(G \cup \{\overline{q_i}\}) = \bot, \tau(C_{G\cup\{\overline{q_i}\}}) = \bot$$

But note that $C_{F\cup\{q_i\}}, C_{G\cup\{\overline{q_i}\}}$, and $B_j(\vec{p}, \vec{r})$ don't have q_i as a variable. Thus if $\tau'(q_i) = \begin{cases} \bot & \text{if } \tau(q_i) = \top \\ \top & \text{otherwise} \end{cases}$ and $\tau = \tau'$ on all other variables, then we have that

case a: $\tau'(C_{F\cup\{q_i\}}) = \bot$ (since $\tau(F) = \tau'(F) = \bot$ and $\tau(F \cup \{q_i\}) = \top$ and $\tau(F \cup \{q_i\}) = \top$. By induction hypothesis, $\exists j\ \tau'(B_j) = \bot$, and $\tau'(B_j) = \tau(B_j)$.

case b: Similar to case a.

5. $E = F \cup G$ from $(F \cup \{r_i\}, G \cup \{\overline{r_i}\})$

$$C_E ::= C_{F \cup \{r_i\}} \wedge C_{G \cup \{\overline{r_i}\}}$$

   This is similar to case 4.

This completes the proof of the lemma and of the theorem.

□

12 Monotone circuits from resolution refutations

A modification of the proof of the preceding section yields *monotone* circuits for the interpolant, for a particular class of clauses being refuted.

Theorem 27 ([37, 29]) *Let $\Gamma = \{A_i(\vec{p}, \vec{q})\} \cup \{B_j(\vec{p}, \vec{r})\}$ have a refutation of length n, where* **either** *the $\vec{p}$ variables occur only positively in the A_i's* **or** *they occur only negatively in the B_j's. Then there is a monotone circuit $C(\vec{p})$ of size $O(n)$ such that for every τ,*

$$\tau(C(\vec{p})) = \bot \Longrightarrow \exists i \ \tau(A_i(\vec{p}, \vec{q})) = \bot$$

$$\tau(C(\vec{p})) = \top \Longrightarrow \exists j \ \tau(B_j(\vec{p}, \vec{r})) = \bot$$

Note: in order to get a monotone circuit, we need to assume that the p_l's either appear only positively in A_i or only negatively in B_j.

Proof We present the proof only in the case when the p_l's occur only negatively in the B_j's; the other case is similar.

We will build a circuit C_E for each expression E in the resolution refutation.

1. $E = A_i(\vec{p}, \vec{q})$, then C_E is the constant 0.
2. $E = B_j(\vec{p}, \vec{r})$, then C_E is the constant 1.
3. $E = F \cup G$ from $(F \cup \{p_i\}, G \cup \{\overline{p_i}\}$[7]

$$C_E ::= C_{F \cup \{p_i\}} \vee (p_i \wedge C_{G \cup \{\overline{p_i}\}})$$

4. $E = F \cup G$ from $(F \cup \{q_i\}, G \cup \{\overline{q_i}\})$

$$C_E ::= C_{F \cup \{q_i\}} \vee C_{G \cup \{\overline{q_i}\}}$$

[7]To prove the theorem when the p_i's occur only positively in the A_j's, define C_E to be $(p_i \vee C_{F \cup \{p_i\}}) \wedge C_{G \cup \{\overline{p_i}\}}$ in this case.

5. $E = F \cup G$ from $(F \cup \{r_i\}, G \cup \{\overline{r_i}\})$

$$C_E ::= C_{F \cup \{r_i\}} \wedge C_{G \cup \{\overline{r_i}\}}$$

Clearly, $C_\Box$ is a monotone circuit of size $O(n)$. As in the proof of the preceding theorem, we base our proof of correctness on a lemma that we prove by induction. The statement of the lemma for this monotone construction is more complicated than the statement of the corresponding lemma in the preceding result.

For any clause E appearing in the refutation, define two "sub-clauses" E^A and E^B as follows. E^A is the disjunction of all the literals occurring in E that involve q-variables and p-variables; that is, all of the literals involving r-variables are "erased". E^B is the disjunction of all of the literals occurring in E that involve r-variables and the *negative* p-literals; that is, all q-variables and all non-negated p-variables are "erased". Note that E^A and E^B are not necessarily disjoint.[8]

Note that the following lemma is clearly sufficient to prove the theorem, since $\Box^A = \Box^B = \Box$, and the output gate of the circuit is $C_\Box$.

Lemma 28

$$\tau(E^A) = \bot \textit{ and } \tau(C_E) = \bot \Longrightarrow \exists i \ \tau(A_i) = \bot$$

$$\tau(E^B) = \bot \textit{ and } \tau(C_E) = \top \Longrightarrow \exists i \ \tau(B_i) = \bot$$

Proof (of the lemma)

By induction on the structure of, where E appears in the resolution refutation.

1. If $E = A_i(\vec{p}, \vec{q})$, then for all $\tau, \tau(C_E) = \bot$, so only the first implication in the Lemma needs to be considered. Note also that $E^A = E$, and thus if the hypothesis for the first implication holds, then trivially $\tau(A_i) = \bot$.

2. $E = B_j(\vec{p}, \vec{r})$, then $\tau(C_E) = \top$ and $E = E^B$. Thus this is similar to the previous case.

3. $E = F \cup G$ from $(F \cup \{p_i\}, G \cup \{\overline{p_i}\})$

$$C_E ::= (C_{F \cup \{p_i\}}) \vee (p_i \wedge C_{G \cup \{\overline{p_i}\}})$$

Case a: $\tau(E^A) = \bot$ and $\tau(C_E) = \bot$. We consider two cases, depending on $\tau(p_i)$.

[8]To prove the theorem when the p_i's occur only positively in the A_j's, define E^A to be the result of erasing the r-literals and the *positive* p-literals from E, and define E^B to the result of erasing the q-literals from E. The rest of the argument is similar.

If $\tau(p_i) = \top$, then $\tau((G \cup \{\overline{p_i}\})^A) = \tau(G^A \cup \{\overline{p_i}\}) = \bot$. Also, $\tau(C_{G\cup\{\overline{p_i}\}}) = \bot$. Thus, by induction, there is some j such that $\tau(A_j) = \bot$.

If $\tau(p_i) = \bot$, then $\tau((F \cup \{p_i\})^A) = \tau(F^A \cup \{p_i\}) = \bot$. Also, $\tau(C_{F\cup\{p_i\}}) = \bot$. Again, the claim follows by induction.

Case b: $\tau(E^B) = \bot$ and $\tau(C_E) = \top$.

In this case, note that $(F \cup \{p_i\})^B = F^B$ and thus $\tau(F \cup \{p_i\})^B = \bot$. Since $C_E = \top$, there are the following two cases.

If $\tau(C_{F\cup\{p_i\}}) = \top$, then the induction hypothesis implies that for some $j, \tau(B_j) = \bot$.

Otherwise, $\tau(p_i \wedge C_{G\cup\{\overline{p_i}\}}) = \top$. Thus $\tau((G \cup \{\overline{p_i}\})^B) = \bot$. Again, the induction hypothesis yields the desired result.

4. $E = F \cup G$ from $(F \cup \{q_i\}, G \cup \{\overline{q_i}\})$

$$C_E ::= C_{F\cup\{q_i\}} \vee C_{G\cup\{\overline{q_i}\}}$$

Case a: $\tau(E^A) = \bot$ and $\tau(C_E) = \bot$.

Note that $\tau(C_{F\cup\{q_i\}}) = \tau(C_{G\cup\{\overline{q_i}\}}) = \bot$. Also, either $\tau(F \cup \{p_i\}) = \bot$ or $\tau(G \cup \{\overline{p_i}\}) = \bot$. In either case, the induction hypothesis yields that for some j, $\tau(A_j) = \bot$.

Case b: $\tau(E^B) = \bot$ and $\tau(C_E) = \top$.

Note that $E^B = (F \cup \{q_i\})^B \cup (G \cup \{\overline{q_i}\})^B$, and thus $\tau((F \cup \{q_i\})^B) = \tau((G \cup \{\overline{q_i}\})^B) = \bot$. Also, since $\tau(C_E) = \top$, either $\tau(C_{F\cup\{q_i\}}) = \top$ or $\tau(C_{G\cup\{\overline{q_i}\}}) = \top$. In either case the induction hypothesis yields that for some j, $\tau(B_j) = \bot$.

5. $E = F \cup G$ from $(F \cup \{r_i\}, G \cup \{\overline{r_i}\})$

$$C_E ::= C_{F\cup\{r_i\}} \wedge C_{G\cup\{\overline{r_i}\}}$$

This is similar to case 4.

□

13 Lower bounds on proof length via monotone circuits

In this section we will use the results of the preceding section to show that known lower bounds on monotone circuit size provide lower bounds on the lengths of resolution refutations.

Here is how to build a set of clauses that encode the clique and coloring problems. Consider the following clauses:

$$\begin{array}{ll} \{q_{i,1}, q_{i,2}, \ldots, q_{i,n}\} & \text{for } 1 \le i \le k \\ \{\neg q_{i,m}, \neg q_{j,m}\} & \text{for } 1 \le m \le n \text{ and } 1 \le i < j \le k \\ \{\neg q_{i,m}, \neg q_{j,l}, p_{m,l}\} & \text{for } 1 \le m < l \le n \text{ and } 1 \le i, j \le k \end{array}$$

The above clauses encode a graph that contains a k-clique as follows:

- The q's encode a one-to-one function from $\{1, \ldots, k\} \to \{1, \ldots, n\}$ if we set $q_{i,j} = 1 \Leftrightarrow q(i) = j$. Thus the clause $\{q_{i,1}, \ldots, q_{i,n}\}$ which means $[(q(i) = 1 \vee \ldots \vee (q(i) = n)]$ says that $q(i)$ is defined (we could also add information saying that there is no more than *one* j such that $q(i) = j$, but that isn't needed for our purposes) and the clause $\{\neg q_{i,m}, \neg q_{j,m}\}$ (which is equivalent to $[q(i) = m \Rightarrow q(j) \ne m]$) ensures that the function is one-to-one.

- The p's encode a graph if we take $p_{m,l} = 1$ to mean that there's an edge between m and l. With this intuition, the last set of clauses $\{\neg q_{i,m}, \neg q_{j,l}, p_{m,l}\}$ is equivalent to $q(i) = m, q(j) = l \Rightarrow$ there is an edge between m and l.

Thus, $\{p_{m,l} : 1 \le m < l \le n\}$ encodes a graph containing a k-clique iff there exist assignments to the q variables making these clauses true.

Next, consider the sets of clauses that encode the property of being an l-partite graph:

$$\begin{array}{ll} \{r_{i,1}, \ldots, r_{i,l}\} & \text{for } 1 \le i \le n \\ \{\neg r_{i,a}, \neg r_{i,b}\} & \text{for } 1 \le i \le n, 1 \le a < b \le l \\ \{\neg r_{i,a}, \neg r_{j,a}, \neg p_{i,j}\} & \text{for } 1 \le a \le l \text{ and } 1 \le i < j \le n \end{array}$$

Here the explanation is as follows:

- The first set of clauses encode the coloring function: $r_{i,c} = 1$ means vertex i has color $c, (r(i) = c)$ and the second set of clauses means that each vertex has at most one color.

- Finally, we make this a proper coloring with the last set of clauses: If $r(i) = a$ and $r(j) = a$ then $p_{i,j} = 0$, (i.e., there is no edge between vertices i and j).

Claim: If $k = l + 1$ then these clauses (i.e., both sets together) are unsatisfiable.
Proof: Every assignment to the p's gives a graph. If all clauses are satisfiable, there is some assignment to the q's encoding a k-clique in the graph and at the same time an assignment to the r's that gives a proper

$(k-1)$-coloring of the graph. This is of course impossible.

Theorem 29 *Any resolution refutation of these clauses requires length* $2^{\Omega(\sqrt{k})}$*, if* $k \leq \sqrt[4]{n}$*.*

Proof It is known that any monotone circuit that evaluates to 1 on all of the k-cliques and evaluates to 0 on all of the $k-1$-partite graphs must have size at least $2^{\Omega(\sqrt{k})}$, for k in this range. (A nice proof is found in [6]; a slightly stronger lower bound appears in [2]. All of these use the proof technique developed in [38].) Since these clauses satisfy the restrictions in the hypothesis of Theorem 27, a lower bound on the length of a refutation follows immediately. $\Box$

It should be noted that strong lower bounds on the length of resolution refutations have been known since the work of [25]. For further discussion of the history of such results, see [18]. However, the proof presented above seems to be the first that explicitly makes use of circuit lower bounds. Further progress in this direction for the stronger "cutting planes" proof system also make use of these circuit lower bounds. These results are reported in [5, 37] and are discussed in section 16.

14 Interpolation for resolution with limited extension

It is natural to wonder if the results of the preceding sections can be extended to proof systems that are more powerful than resolution. One improvement in this direction involves the work on cutting planes [5] mentioned in the preceding paragraph. Another improvement is actually an immediate consequence of Theorem 25, and will be presented in this section.

The term "extension" refers to the process of taking some existing proof system, and extending it by allowing the introduction of new variables (say, σ_A) that can be used to represent the truth value of a propositional formula A. This allows short representations of long formulae, and implicitly allows a system such as resolution to deal with formulae that are not in CNF. The following paragraphs make this notion more precise, for the specific case of extending resolution in this way.

For every formula A, we will have a variable σ_A. In the case where A consists of a single propositional variable p, σ_A is just the formula p.

Now we will define, for each formula A, a set of clauses $LE(A)$.

Definition "Limited Extension." $LE(A)$ is defined inductively. If $A = p$, then $LE(A) = \emptyset$.

- $LE(\neg A) ::= \{\ \{\sigma_{\neg A}, \sigma_A\}\ \{\overline{\sigma_{\neg A}}, \overline{\sigma_A}\}\ \} \cup LE(A)$.
- $LE(A \wedge B) ::= \{\ \{\overline{\sigma_{A\wedge B}}, \sigma_A\}, \{\overline{\sigma_{A\wedge B}}, \sigma_B\},\ \ \{\sigma_{A\wedge B}, \overline{\sigma_A}, \overline{\sigma_B}\}\ \} \cup LE(A) \cup LE(B)$.
- $LE(A \vee B)\ ::=\ \{\ \ \{\overline{\sigma_A}, \sigma_{A\vee B}\}, \{\overline{\sigma_B}, \sigma_{A\vee B}\}, \quad \{\sigma_A, \sigma_B, \overline{\sigma_{A\vee B}},\}\ \ \} \cup LE(A) \cup LE(B)$.

Note that these clauses ensure that any truth assignment satisfying the clauses has σ_A equal to the negation of $\sigma_{\neg A}$, $\sigma_{A\wedge B}$ is equal to the logical and of σ_A and σ_B, etc.

Definition Let $\mathcal{A}$ be any set of formulae. Then define

$$LE(\mathcal{A}) ::= \bigcup_{\{A\in\mathcal{A}\}} LE(A).$$

Note that it is clear that $\mathcal{A}$ has a resolution refutation if and only if $\mathcal{A} \cup LE(\mathcal{A})$ has a resolution refutation.

Theorem 30 *Let $\Gamma = \mathcal{A} \cup \mathcal{B}$, where $\mathcal{A} = \{A_i(\vec{p}, \vec{q})\}$ and $\mathcal{B} = \{B_j(\vec{p}, \vec{r})\}$. Let $\Gamma \cup LE(\Gamma)$ have a resolution refutation of length n.*

Then there is an interpolant $C(\vec{p})$ such that

$$\bigwedge_i A_i(\vec{p}, \vec{q}) \Rightarrow C(\vec{p})$$

and

$$C(\vec{p}) \Rightarrow \neg \bigwedge_j (B_j(\vec{p}, \vec{r}))$$

and $C(\vec{p})$ has circuits of size $\leq 3n$.

Proof Note that $\Gamma \cup LE(\Gamma) = (\mathcal{A} \cup LE(\mathcal{A})) \cup (\mathcal{B} \cup \mathcal{LE}(\mathcal{B}))$. Note that the only variables that are shared between $(\mathcal{A} \cup LE(\mathcal{A}))$ and $(\mathcal{B} \cup LE(\mathcal{B}))$ are the variables in $\vec{p}$. Thus Theorem 25 gives us an interpolant with circuit size $O(n)$ for $(\mathcal{A} \cup LE(\mathcal{A}))$ and $(\mathcal{B} \cup LE(\mathcal{B}))$. That is,

$$(\bigwedge_i A_i(\vec{p}, \vec{q})) \wedge LE(\mathcal{A}) \Rightarrow C(\vec{p})$$

and

$$C(\vec{p}) \Rightarrow \neg(LE(\mathcal{B}) \wedge \bigwedge_j (B_j(\vec{p}, \vec{r})))$$

It suffices to observe now that this same $C(\vec{p})$ is also an interpolant for $\mathcal{A}$ and $\mathcal{B}$. But this is obvious, because of the following observations.

Let τ be any truth assignment with domain $\{\vec{p}, \vec{q}\}$ that satisfies $\mathcal{A}$. Then there is a unique extension of τ that satisfies $(\mathcal{A} \cup LE(\mathcal{A}))$. Thus if

$\tau(C(\vec{p})) = \bot$, must be the case that there is some i such that $\tau(A_i(\vec{p}, \vec{q})) = \bot$.

Similarly, if $\tau(C(\vec{p})) = \top$, there must be some j such that $\tau(A_i(\vec{p}, \vec{q})) = \bot$ (since otherwise this τ could be extended to satisfy $LE(\mathcal{B})$, in contradiction to C being an interpolant for $(\mathcal{A} \cup LE(\mathcal{A}))$ and $(\mathcal{B} \cup LE(\mathcal{B}))$). □

Part III
Cutting Plane Proof Systems[9]

15 Introduction to Cutting Plane Proofs

The cutting plane refutation system CP is an extension of resolution, where unsatisfiable propositional logic formulas in conjuctive normal form are recognized by showing the non-existence of boolean solutions to associated families of linear inequalities. The notes below cover equivalence between CP and its subsystem CP_2; and its relation with Frege Systems.

15.1 Preliminaries

The cutting planes system CP is a Refutation System for propositional logic formulas in conjuctive normal form (CNF). In CP the truth values TRUE and FALSE are interpreted by 1 and 0, and propositional formulas are expressed by systems of linear inequalities. The basic idea is that a clause $\{x_1, \ldots, x_k\}$ can be rewritten as an integer inequality $i_1 + \cdots + i_k \geq 1$; where $i_j := x_j$ if x_j is a positive literal, and $i_j := 1 - x_j$ otherwise. For example take the following CNF:

$$(x \vee y) \wedge (\neg x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee \neg y)$$

is represented by the family

$$\begin{array}{rcl} x + y & \geq & 1 \\ 1 - x + y & \geq & 1 \\ x + 1 - y & \geq & 1 \\ 1 - x + 1 - y & \geq & 1 \end{array}$$

[9]This part is based partly on notes prepared by C. Lautemann and C. Zamora-Cura based on lectures delivered by S. Buss on March 8, 1995 at the McGill University Bellair's Research Institute in Holetown, Barbados.

of linear inequalities, one for each clause.

We can simplify the last three of these inequalities to

$$\begin{array}{rcl} -x+y & \geq & 0 \\ x-y & \geq & 0 \\ -x-y & \geq & -1 \end{array}$$

More generally, for a clause C define $I(C)$ to be the inequality

$$\sum_i \alpha_i p_i \geq 1-m,$$

where

$$\alpha_i = \begin{cases} 1 & if \quad x_i \in C \\ -1 & if \quad \neg x_i \in C \\ 0 & if \quad neither\ in\ C \end{cases}$$

and m is equal to the number of negated variables in C.

The idea of CP proofs rests on the following fact:

Fact 1 *A set* Γ *of clauses is satisfiable if and only if the set of integer inequalities* $\{I_C : C \in \Gamma\}$ *is satisfiable, over* $\{0,1\}$.

More formally, we state that a line in a cutting plane proof is of the form:

$$\alpha_1 p_1 + \alpha_2 p_2 + \cdots + \alpha_k p_k \geq m$$

where $\alpha_i, m \in \mathbf{Z}$, and $p_1, \ldots, p_k$ are variables valued in the set $\{0,1\}$. We will call $\alpha_1 p_1 + \alpha_2 p_2 + \cdots + \alpha_k p_k$ a *CP expression.*

The system CP has as axioms:

$$\begin{array}{r} p \geq 0 \\ -p \geq -1; \end{array}$$

and as rules of inference:

- *Addition*:
$$\frac{E \geq a \quad , \quad F \geq b}{E+F \geq a+b,}$$
- *Multiplication* by $c \in \mathbf{N}$:
$$\frac{E \geq b}{c \cdot E \geq c \cdot b,}$$

- *Division* by $c \in \mathbf{N},\ c > 0,\ b \in \mathbf{Z}$, if $c \mid E$ with quotient E', then

$$\frac{E \geq b}{E' \geq \lceil \frac{b}{c} \rceil},$$

A formula B in conjuctive normal form has a *cutting plane refutation*, if there is a sequence $s_0, \ldots, s_m$ of linear inequalities, such that

- s_m is $0 \geq 1$,
- for all $i \leq m$, either s_i is a cutting plane axiom, or it is the translation of one of the clauses of B, or there exist $j, k < i$ such that s_i is obtained form s_j, s_k by one of the rules of inference.

A formula B is said to have a *cutting planes proof* if its negation has a cutting planes refutation.
Remark: It is possible to omit the multiplication rule without affecting the power of the cutting planes proof system. This is because multiplication (by an integer) can be simulated by repeated additions.

15.2 Power of *CP*

One of the central problems of propositional logic is to find useful methods for recognizing tautologies; since A is a tautology if and only if $\neg A$ is not satisfiable this is essentially the same as the problem of finding methods for recognizing satisfiable formulas. Three of the principal propositional proof systems are Frege Proof Systems, the Sequent Calculus System, and the Resolution Refutation Proof System. Therefore, in order to measure the power of CP it is important to compare how CP is related to some of these proof systems.

One sign of the importance and strength of cutting planes proofs is given by the following theorem proved by Cook, Coullard and Turan in [21], which shows how CP can p-simulate resolution.

Theorem 31 *The cutting planes proof system can* p-simulate *resolution.*

Proof. We will only sketch the proof. Given a refutation in resolution of a set Γ of clauses, translate it into a cutting planes refutation of $\{I_C\}_{C \in \Gamma}$.

A resolution inference[10]

$$\frac{C_1\ ,\ C_2}{C},$$

[10]Suppose that C and D are clauses and that $x \in C$ and $\neg x \in D$ are literals. The *resolution rule* applied to C and D is the inference

$$\frac{C\ ,\ D}{(C \setminus \{x\}) \cup (D \setminus \{\neg x\})}.$$

solving on some variable x, is simulated by the following (we can assume without loss of generality that $C_1 \setminus \{x\} = C$ and $C_2 \setminus \{\neg x\} = C$):

$$\frac{\sum \alpha_i p_i + x \geq 1 - m \, , \, \sum \alpha_i p_i - x \geq 1 - m - 1}{\sum 2\alpha_i p_i \geq 2 - 2m - 1}$$

$$\frac{\sum 2\alpha_i p_i \geq 2 - 2m - 1}{\sum \alpha_i p_i \geq 1 - m}$$

where we are using addition, and division rules.

So, each application of a resolution inference rule can be p-simulated, because the transformation function is polynomially time computable. $\Box$

It is perhaps not surprising that *CP* is more "efficient" than resolution, since addition of two inequalities may amount to a simultaneous resolution on many variables. To quantify the efficiency of *CP* over resolution, consider a combinatorial principle called the pigeonhole principle PHP_n; which states that there is no injection from $\{0, \ldots, n\}$ into $\{0, \ldots, n-1\}$:

$$\bigwedge_{0 \leq i \leq n} \bigvee_{0 \leq j < n} p_{ij} \supset \bigvee_{0 \leq i < i' \leq n} \bigvee_{0 \leq j < n} (p_{ij} \wedge p_{i'j})$$

The negation of PHP_n can be formulated by a propositional CNF formula denoted $\neg PHP_n$ of size $O(n^3)$. We will measure the *size* of a proof by the total number of logical connectives.

Theorem 32 *PHP_n has polynomial size cutting planes proofs.*

Proof. The $\neg PHP_n$ clauses become the inequalities:

$$\begin{array}{lcll} \alpha_i & : & p_{i,0} + \cdots + p_{i,n-1} \geq 1 \quad , & i = 0, \ldots, n \\ \beta_{i,m,j} & : & -p_{i,j} - p_{m,j} \geq -1 \quad , & 0 \leq i < m \leq n \quad and \quad 0 \leq j < n \end{array}$$

Now, firstly we derive inductively on k for each fixed j:

$$-p_{0,j} - p_{1,j} - \cdots - p_{k,j} \geq -1 \quad , \; k = 1, \ldots, n.$$

Base case: $\beta_{0,1,j}$ is what we want. Inductive step:

$$\sum_{i=0}^{k-1} \beta_{i,k,j} = -p_{0,j} - \cdots - p_{k-1,j} - kp_{k,j} \geq -k$$

this plus $[-p_{0,j} - \cdots - p_{k-1,j} \geq -1](k-1)$ and combined with the previous step we have

$$-kp_{0,j} - kp_{1,j} - kp_{1,j} - \cdots - kp_{k-1,j} - kp_{k,j} \geq -2k + 1.$$

Dividing by k gives:

$$-p_{0,j} - \cdots - p_{k,j} \geq \left\lceil \frac{-2k+1}{k} \right\rceil = -1$$

Summing all these formulas for all n values on j gives (with $k = n$):

$$\sum_{i,j} -p_{i,j} \geq -n,$$

and summing α_i for all $n+1$ values of i gives

$$\sum_{i,j} p_{i,j} \geq n+1.$$

Finally, summing the two very last inequalities we have

$$0 \geq 1.$$

□

Another theorem, which we will leave without proof, proved by Goerdt in [24] states a relation between Frege Proof Systems and CP.

Theorem 33 *Frege Proof Systems F can p-simulate the cutting planes proof systems.*

15.3 CP_k and CP

For an integer $k \geq 2$, the proof system CP_k is obtained from CP by restricting the division rule to division by k. The system CP_2 is quite strong, and the following theorem will show that CP_2 is p-equivalent to CP.

Theorem 34 *[16] For $k \geq 2$, CP_k* p-*simulates CP.*

Proof. Here, we only present the case $k = 2$, all other cases are similar. Suppose CP_2 is trying to simulate

$$\frac{m \cdot \alpha \geq n}{\alpha \geq \lceil \frac{n}{m} \rceil},$$

where α is a linear combination of variables. Let $2^{p-1} < m \leq 2^p$. Letting r_0 be equal to the sum of negative coefficients in α, and using addition of axioms, we get $\alpha \geq r_0$. Iterate:

- from $\alpha \geq r_i$ derive

$$\frac{(2^p - m)\alpha \geq (2^p - m)r_i}{2^p\alpha \geq n + (2^p - m)r_i},$$

here we are using addition with $m \cdot \alpha \geq n$,

$$\frac{2^p\alpha \geq n + (2^p - m)r_i}{\alpha \geq \left\lceil \frac{n+(2^p-m)r_i}{2^p} \right\rceil},$$

here we are using division by 2, p times.

- set
$$r_{i+1} = \left\lceil \frac{n + (2^p - m)r_i}{2^p} \right\rceil .$$

Note that

$$r_{i+1} \geq \frac{n}{m}\left(\frac{m}{2^p}\right) + r_i\left(\frac{2^p - m}{2^p}\right).$$

So, after polynomially many iterations we will have that

$$r_{i+1} > \frac{n}{m} - \frac{1}{m}.$$

Thus, $r_{i+1} = \left\lceil \frac{n}{m} \right\rceil$. □

The most interesting open question concerning CP is to exhibit a combinatorial family of tautologies requiring superpolynomial CP proof size.

16 Lower bound on the size of CP proofs with small coefficients

We conclude with a discussion of the the known exponential lower bound for the length of CP proofs for the k–clique tautology. This was proved first by Bonet-Pitassi-Raz [5] under the assumption that coefficients in the linear inequalities in a cutting planes proof are in unary notation; the general result was established by Pudlak [37]. Both proofs used the tautologies based on k-cliques; they also both use an interpolation theorem to make a reduction to to monotone (Boolean or real, resp.) circuits that separate k–cliques from $(k-1)$–cocliques, which by a strengthening of the lower bound result of Alon and Boppana[2] require exponential size.

16.1 The k-clique tautology

This tautology expresses the fact that a $(k-1)$-colorable graph on n vertices cannot contain a k-clique as a subgraph. In order to represent this fact by a propositional formula, we will use propositional variables $p_{i,j}$ to represent the presence of an edge between nodes number i and j in a graph. We use variables $q_{m,i}$ to code a 1-1 mapping from m elements into the nodes of the graph; we use variables $r_{i,l}$ to code a mapping from the nodes of the graph to a set of colors. Here $1 \leq i, j \leq n$ range over nodes in the graph, $1 \leq m \leq k$ ranges over a set of size k, and $1 \leq \ell \leq k-1$ ranges over a set of $k-1$ colors.

The following set of inequalities expresses the fact that the graph has a clique of size k encoded by the variables $q_{m.i}$:

$$\begin{aligned} \sum_i q_{m,i} &\geq 1 \qquad \text{for } 1 \leq m \leq k \\ \sum_m q_{m,i} &\leq 1 \qquad \text{for } 1 \leq i \leq k \\ p_{i,j} - q_{m,i} - q_{m',j} &\geq -1 \qquad \text{for } 1 \leq i < j \leq n,\ 1 \leq m, m' \leq k \text{ and } m \neq m' \end{aligned}$$

The next two kinds of inequalities express the condition that the variables $r_{i,l}$ code a $k-1$ coloring of the graph:

$$\begin{aligned} \sum_l r_{i,l} &\geq 1 \qquad \text{for } 1 \leq i \leq n \\ p_{i,j} + r_{i,l} + r_{j,l} &\leq 2 \qquad \text{for } 1 \leq i < j \leq n \text{ and } 1 \leq l \leq k-1 \end{aligned}$$

16.2 The exponential lower bound

Taken together, the above inequalities are inconsistent and therefore have a cutting planes refutation. Pudlak shows that any such cutting plances refutation can be transformed into a monotone *real* circuit which separates graphs with k-cliques from graphs with $(k-1)$-colorings. Here, a *monotone real circuit* is a circuit in which each gate computes a real function which is a nondecreasing function of both its inputs. The real monotone circuit performs the separation by outputing 1 for graphs which are not $(k-1)$-colorable and and outputing 0 for graphs without k-cliques.

Lemma 35 ([37]) *There is a polynomial $p(n)$ so that if the inequalies above have a cutting planes refutation of size n, then there is a real monotone circuit of size $p(n)$ which separates graphs with k-cliques from graphs with $k-1$-colorings.*

We omit the proof of the above lemma; the crucial property which makes the lemma hold is that when the inequalities are written with $\geq$ signs and with the variables on the left, then the inequalities in which both the $p_{i,j}$ and $q_{k,i}$ variables have positive coefficents on the $p_{i,j}$ variables (alternatively: it is also enough that the $p_{i,j}$ variables have negative signs when they occur in inequalities with the $r_{i,l}$ variables).

Lemma 36 ([37], based on [2]) *Let $k = \left\lfloor \frac{1}{8}(n/\log n)^{2/3} \right\rfloor$. Any real monotone circuit seperating graphs with k-cliques from $k-1$-colorable graphs has $2^{\Omega((n/\log n)^{2/3}}$ gates.*

From this we get immediately:

Theorem 37 ([37]) *For $k = \left\lfloor \frac{1}{8}(n/\log n)^{2/3} \right\rfloor$, and cutting planes refutation of the above inequalities requires $2^{\Omega((n/\log n)^{2/3}}$ steps.*

References

[1] M. Ajtai, *The complexity of the pigeonhole principle*, in Proceedings of the 29-th Annual IEEE Symposium on Foundations of Computer Science, 1988, pp. 346–355.

[2] B. Alon and R. Boppana, *The monotone circuit complexity of Boolean functions*, Combinatorica, 7 (1987), pp. 1–22.

[3] M. L. Bonet, *The Lengths of Propositional Proofs and the Deduction Rule*, PhD thesis, U.C. Berkeley, 1991.

[4] M. L. Bonet and S. R. Buss, *Size-depth tradeoffs for Boolean formulas*, Information Processing Letters, 29 (1994), pp. 151–155.

[5] M. L. Bonet, T. Pitassi, and R. Raz, *Lower bounds for cutting planes proofs with small coefficients*, Journal of Symbolic Logic, 62 (1997), pp. 708–728. An earlier version appeared in *Proc. Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, 1995, pp. 575-584.

[6] R. Boppana and M. Sipser, *The Complexity of Finite Functions*, MIT Press/Elsevier, 1990, pp. 757–804.

[7] A. Borodin, S. Cook, and N. Pippenger, *Parallel computation for well-endowed rings and space-bounded probabilistic machines*, Information and Control, 58 (1992), pp. 113–136.

[8] R. P. Brent, *The parallel evaluation of general arithmetic expressions*, J. Assoc. Comput. Mach., 21 (1974), pp. 201–206.

[9] N. H. Bshouty, R. Cleve, and W. Eberly, *Size-depth tradeoffs for algebraic formulae*, in Proceedings of the 32th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, 1991, pp. 334–341.

[10] S. R. Buss, *The Boolean formula value problem is in ALOGTIME*, in Proceedings of the 19-th Annual ACM Symposium on Theory of Computing, May 1987, pp. 123–131.

[11] ——, *Polynomial size proofs of the propositional pigeonhole principle*, Journal of Symbolic Logic, 52 (1987), pp. 916–927.

[12] ——, *Algorithms for Boolean formula evaluation and for tree contraction*, in Arithmetic, Proof Theory and Computational Complexity, P. Clote and J. Krajíček, eds., Oxford University Press, 1993, pp. 96–115.

[13] ——, *Some remarks on lengths of propositional proofs*, Archive for Mathematical Logic, 34 (1995), pp. 377–394.

[14] ——, *Lectures in proof theory*, Tech. Rep. SOCS 96.1, School of Computer Science, McGill University, 1996. Notes from a series of lectures given at the at the McGill University Bellair's Research Institute, Holetown, Barbados, March 1995. Scribe note written by E. Allender, M.L. Bonet, P. Clote, A. Maciel, P. McKenzie, T. Pitassi, R. Raz, K. Regan, J. Torán and C. Zamora.

[15] ——, *An introduction to proof theory*, in Handbook of Proof Theory, S. R. Buss, ed., North-Holland, 1998, pp. 1–78.

[16] S. R. Buss and P. Clote, *Cutting planes, connectivity and threshold logic*, Archive for Mathematical Logic, 35 (1996), pp. 33–62.

[17] S. R. Buss, S. A. Cook, A. Gupta, and V. Ramachandran, *An optimal parallel algorithm for formula evaluation*, SIAM J. Comput., 21 (1992), pp. 755–780.

[18] V. Chvátal and E. Szemerédi, *Many hard examples for resolution*, Journal of the ACM, 35 (1988), pp. 759–768.

[19] S. A. Cook, *The complexity of theorem proving techniques*, in Proceedings of the 3-rd Annual ACM Symposium on Theory of Computing, 1971, pp. 151–158.

[20] S. A. Cook and R. A. Reckhow, *The relative efficiency of propositional proof systems*, Journal of Symbolic Logic, 44 (1979), pp. 36–50.

[21] W. Cook, C. R. Coullard, and G. Turán, *On the complexity of cutting plane proofs*, Discrete Applied Mathematics, 18 (1987), pp. 25–38.

[22] W. Craig, *Linear reasoning. A new form of the Herbrand-Gentzen theorem*, Journal of Symbolic Logic, 22 (1957), pp. 250–268.

[23] M. Dowd, *Model-theoretic aspects of $P \neq NP$*. Typewritten manuscript, 1985.

[24] A. Goerdt, *Cutting plane versus frege proof systems*, in Computer Science Logic 1990, E. Börger, ed., Lecture Notes in Computer Science #552, 1992, pp. 174–194.

[25] A. Haken, *The intractability of resolution*, Theoretical Computer Science, 39 (1985), pp. 297–308.

[26] J. Hartmanis and R. Stearns, *On the computational complexity of algorithms*, Transactions of the American Mathematical Society, 117 (1965), pp. 285–306.

[27] S. Kosaraju and A. Delcher, *A tree-partitioning technique with applications to expression evaluation and term matching*, in Proceedings of the Thirteenth Annual Symposium on Foundations of Computing, IEEE Computer Society, 1990, pp. 163–172.

[28] J. Krajíček, *Lower bounds to the size of constant-depth propositional proofs*, Journal of Symbolic Logic, 59 (1994), pp. 73–85.

[29] ———, *Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic*, Journal of Symbolic Logic, 62 (1997), pp. 457–486.

[30] J. Krajíček and P. Pudlák, *The number of proof lines and the size of proofs in first-order logic*, Archive for Mathematical Logic, 27 (1988), pp. 69–84.

[31] ———, *Propositional proof systems, the consistency of first-order theories and the complexity of computations*, Journal of Symbolic Logic, 54 (1989), pp. 1063–1079.

[32] J. Krajíček, P. Pudlák, and A. Woods, *Exponential lower bound to the size of bounded depth Frege proofs of the pigeonhole principle*, Random Structures and Algorithms, 7 (1995), pp. 15–39.

[33] R. E. Ladner, *The circuit value problem is log space complete for P*, SIGACT News, 7 (1975), pp. 18–20.

[34] D. Mundici, *Tautologies with a unique Craig interpolant*, Annals of Pure and Applied Logic, 27 (1984), pp. 265–273.

[35] Y. Ofman, *On the algorithmic complexity of discete functions*, Doklady Akad. Nauk SSSR, 145 (1962), pp. 48–51. English translation in *Soviet Physics – Doklady* 7 (1963)589-591.

[36] T. Pitassi, P. Beame, and R. Impagliazzo, *Exponential lower bounds for the pigeonhole principle*, Computational Complexity, 3 (1993), pp. 97–140.

[37] P. Pudlák, *Lower bounds for resolution and cutting planes proofs and monotone computations*, Journal of Symbolic Logic, 62 (1997), pp. 981–998.

[38] A. A. Razborov, *Lower bounds on the monotone complexity of some Boolean functions*, Soviet Mathematics Doklady, 31 (1985), pp. 354–357.

[39] R. A. Reckhow, *On the Lengths of Proofs in the Propositional Calculus*, PhD thesis, Department of Computer Science, University of Toronto, 1976. Technical Report #87.

[40] J. Siekmann and G. Wrightson, *Automation of Reasoning*, vol. 1&2, Springer-Verlag, Berlin, 1983.

[41] P. M. Spira, *On time hardware complexity tradeoffs for Boolean functions*, in Proceedings of the Fourth Hawaii International Symposium on System Sciences, 1971, pp. 525–527.

[42] R. Statman, *Complexity of derivations from quantifier-free Horn formulae, mechanical introduction of explicit definitions, and refinement of completeness theorems*, in Logic Colloquium '76, R. Gandy and M. Hyland, eds., Amsterdam, 1977, North-Holland, pp. 505–517.

[43] G. S. Tsejtin, *On the complexity of derivation in propositional logic*, Studies in Constructive Mathematics and Mathematical Logic, 2 (1968), pp. 115–125. Reprinted in: [40, vol 2].

[44] C. S. Wallace, *A suggestions for a fast multiplier*, IEEE Transactions on Computers, EC-13 (1964), pp. 14–17.