Weak Formal Systems and Connections to Computational Complexity

Lecture Notes for a Topics Course
University of California, Berkeley
January–May 1988

Instructor: Sam Buss

Notes by: Frederick Teti
Maria Bonet
John Grosh
Martin Goldstern
Chrystopher Nehaniv
Eric Hughes
Stephen Carrier
Juan Bagaria
Jim Gloor
Alessandro Berarducci
Sam Buss

These lecture notes were written for a topics course in the Mathematics Department at the University of California, Berkeley during the winter semester. Each student was responsible for writing notes for three one hour lectures. I would like to take the opportunity to thank each of F. Teti, M. L. Bonet, J. Grosh, M. Goldstern, C. Nehaniv, E. Hughes, S. Carrier, J. Bagaria, J. Gloor and A. Berarducci for doing such a superb job of producing complete and exceptionally well-formatted lecture notes.

In addition to these lecture notes, the course covered two papers (listed below) which are not included for copyright reasons. The topics covered included:

Math 271 - Topics in Weak Formal Systems

Lecture Notes, Set #1
1988 January 20–27

Instructor: Sam Buss
Notes By: Frederick Teti

I. Propositional Sequent Calculus

A. Definitions

Symbols such as $p_0, p_1, \ldots, p, q, r$ represent propositional variables. (,) and the comma are used as punctuation. The logical connectives all have their usual meanings. σ typically stands for a truth assignment, that is, a function from the variables to a convenient two-element set such as {T,F}. The natural extension of σ to the set of formulas is denoted by $\bar{\sigma}$. Formulas are defined in the standard recursive fashion.

Definition: A CEDENT is a finite, possibly empty, sequence of formulas.

Definition: A SEQUENT is a string of the form $\Gamma \longrightarrow \Delta$ where Γ and Δ are cedents. Γ is the ANTECEDENT; Δ is the SUCCEDENT.

Given a truth assignment σ, we define the truth value of $\Gamma \longrightarrow \Delta$ by

$$\bar{\sigma}(\Gamma \longrightarrow \Delta) = \bar{\sigma}(\bigwedge A_i \to \bigvee B_i)$$

where the A_i are the terms of Γ, and the B_i are the terms of Δ. We write $\models \Gamma \longrightarrow \Delta$ if $\bar{\sigma}(\Gamma \longrightarrow \Delta) = T$ for all truth assignments σ.

B. Rules of Inference.

In the following, A and B range over formulas; Γ and Δ range over cedents. We assume for the nonce that the language for formulas comprises $\wedge$, $\vee$, and $\neg$. We write A, Γ to stand for the sequent whose first term is A and whose succeding terms are those of Γ. The notions of PRINCIPAL FORMULA and SUCCESSOR are defined by cases as the rules are listed.

1. Structural Rule: If $\Gamma^* \supseteq \Gamma$ (as a set), and $\Delta^* \supseteq \Delta$ (as a set), then the following syllogism is a valid inference.

$$\frac{\Gamma \longrightarrow \Delta}{\Gamma^* \longrightarrow \Delta^*}$$

   If A is a term of Γ (or of Δ), then the SUCCESSOR of A is the *first occurrence* of A in Γ^* (or in Δ^*). There is no principal formula associated with the structural rule.

2. ($\neg$-right) Rule:

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

   The indicated occurrence of the formula $\neg A$ is the SUCCESSOR of the indicated occurrence of the formula A. It is also the PRINCIPAL FORMULA associated with the ($\neg$-right) Rule. If B is a term of Γ, then the SUCCESSOR of B is the *first occurrence* of B in $\neg A, \Gamma$. If B is a term of Δ, then the SUCCESSOR of B is the *first occurrence* of B in Δ. (The successors of these so-called *side formulas* are all defined similarly and are henceforth omitted.)

3. ($\neg$-left) Rule:

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

   The indicated occurrence of the formula $\neg A$ is the SUCCESSOR of the indicated occurrence of the formula A. It is also the PRINCIPAL FORMULA assoiciated with the ($\neg$-left) Rule.

4. ($\wedge$-right) Rule:

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

The indicated occurrence of the formula $A \wedge B$ is the SUCCESSOR of the indicated occurrences of the formulas A and B. It is also the PRINCIPAL FORMULA associated with the ($\wedge$-right) Rule.

5. ($\wedge$-left) Rule:

$$\frac{A, \Gamma \longrightarrow \Delta}{A \wedge B, \Gamma \longrightarrow \Delta}$$

$$\frac{A, \Gamma \longrightarrow \Delta}{B \wedge A, \Gamma \longrightarrow \Delta}$$

The indicated occurrence of the formula $A \wedge B$ is the SUCCESSOR of the indicated occurrence of the formula A. The indicated occurrence of the formula $B \wedge A$ is the SUCCESSOR of the indicated occurrence of the formula A. In both variations of the Rule, the indicated occurrence of $A \wedge B$ or $B \wedge A$ is the PRINCIPAL FORMULA.

6. ($\vee$-right) Rule:

$$\frac{\Gamma \longrightarrow A, \Delta}{\Gamma \longrightarrow A \vee B, \Delta}$$

$$\frac{\Gamma \longrightarrow A, \Delta}{\Gamma \longrightarrow B \vee A, \Delta}$$

The indicated occurrences of the formulas $A \vee B$ and $B \vee A$ are both SUCCESSORS of the indicated occurrences of the formula A, respectively. They are also the PRINCIPAL FORMULAS of their own variations of the ($\vee$-right) Rule.

7. ($\vee$-left) Rule:

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

The indicated occurrence of the formula $A \vee B$ is the SUCCESSOR of both the indicated occurrence of A and the indicated occurrence of B. It is also the PRINCIPAL FORMULA associated with the ($\vee$-left) Rule.

8. Cut Rule:

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

The notions of successor and principal formula are not defined for the Cut Rule.

C. Metatheorems

Definition: A sequent is an AXIOM if it has the form $A \longrightarrow A$ for some formula A.

Definition: A sequent calculus PROOF is a finite tree with a unique root whose nodes are labelled with sequents in such a fashion that each leaf of the tree is labelled by an axiom, and all other sequents are inferred from their (1 or 2) children by a rule of inference. The sequent labelling the root of the tree is the ENDSEQUENT and is the sequent PROVED.

Definition: A CUT-FREE proof is a sequent calculus proof with no cut inferences.

Definition: The LENGTH of a sequent $\Gamma \longrightarrow \Delta$, written $|\Gamma \longrightarrow \Delta|$, is the number of characters it comprises.

Lemma (0) In any rule of inference but the Cut Rule and the Structural Rule, the length of the conclusion is greater than the length of any hypothesis.

proof: Omitted. $\dashv$

N.B. We will abuse terminology and say that the *formula* A is proved when in actuality the *sequent* $\longrightarrow A$ is proved.

Theorem (1) (Soundness) If $\Gamma \longrightarrow \Delta$ has a sequent calculus proof, then $\models \Gamma \longrightarrow \Delta$.

proof: It is a simple matter to check that all the axioms of the sequent calculus are valid, and that all the rules of inference preserve validity. Hence any provable sequent is valid. Details are left to the interested reader. $\dashv$

Theorem (2) (Completeness) If $\models \Gamma \longrightarrow \Delta$, then there is a cut-free proof of $\Gamma \longrightarrow \Delta$.

proof: We construct a tree with unique root whose nodes are labelled with sequents. Label the root of the tree with $\Gamma \longrightarrow \Delta$. Now suppose we have constructed the tree up to a certain height. Let $\Pi \longrightarrow \Lambda$ be any sequent on the periphery of the tree. Let A be the first non-atomic formula occuring in $\Pi \longrightarrow \Lambda$. We continue to construct the tree as described:

Case 1: B is $\neg C$ and $B \in \Lambda$

$$\begin{array}{c} C, \Pi \longrightarrow \Lambda^* \\ | \\ \Pi \longrightarrow \neg C, \Lambda^* \end{array}$$

where $\Lambda = \neg C, \Lambda^*$. For notational convenience, we assume that that B is the first formula of Λ. In truth, $\neg C, \Lambda^*$ may abbreviate something of the form $\Lambda_1, \neg C, \Lambda_2$. We make similar assumptions in the cases below.

Case 2: B is $C \wedge D$ and $B \in \Lambda$

$$\begin{array}{ccc} \Pi \longrightarrow C, \Lambda^* & & \Pi \longrightarrow D, \Lambda^* \\ \backslash & & / \\ & \Pi \longrightarrow C \wedge D, \Lambda^* & \end{array}$$

where $\Lambda = C \land D, \Lambda^*$.

Case 3: B is $C \lor D$ and $B \in \Lambda$

$$\begin{tikzcd} \Pi \longrightarrow C, D, \Lambda^* \arrow[d, dash] \\ \Pi \longrightarrow C \lor D, \Lambda^* \end{tikzcd}$$

where $\Lambda = C \lor D, \Lambda^*$.

Case 4: B is $\neg C$ and $B \in \Pi$

$$\begin{tikzcd} \Pi^* \longrightarrow C, \Lambda \arrow[d, dash] \\ \neg C, \Pi^* \longrightarrow \Lambda \end{tikzcd}$$

where $\Pi = \neg C, \Pi^*$.

Case 5: B is $C \lor D$ and $B \in \Pi$

$$\begin{tikzcd} C, \Pi^* \longrightarrow \Lambda \arrow[dr, dash] & & D, \Pi^* \longrightarrow \Lambda \arrow[dl, dash] \\ & C \lor D, \Pi^* \longrightarrow \Lambda & \end{tikzcd}$$

where $\Pi = C \lor D, \Pi^*$.

Case 6: B is $C \land D$ and $B \in \Pi$

$$\begin{tikzcd} C, D, \Pi^* \longrightarrow \Lambda \arrow[d, dash] \\ C \land D, \Pi^* \longrightarrow \Lambda \end{tikzcd}$$

where $\Pi = C \land D, \Pi^*$.

We stop constructing the tree when every formula at a leaf is atomic. The process must come to an end because the number of logical connectives in the sequents decreases as the tree is constructed.

Now assume there is a branch in the tree which ends at a leaf labelled with such a sequent $\Pi \longrightarrow \Lambda$ that no variable p appears in both the antecedent and the succedent. Define a truth asignment σ by the following rule:

$$\sigma(p) = \begin{cases} T & \text{if } p \in \Pi \\ F & \text{if } p \in \Lambda \\ \text{otherwise anything} \end{cases}$$

Claim: If Φ is any sequent on this branch, then $\bar{\sigma}(\Phi) =$F. Inspection of the six cases above reveals that falsehood is preserved downwards, and σ assigns the leaf of this branch the truth value F. Since $\Gamma \longrightarrow \Delta$ is at the bottom of this branch, it too must be assigned falsehood. Fortunately, $\Gamma \longrightarrow \Delta$ is valid by hypothesis, so no such branch exists. We conclude that for every sequent $\Pi \longrightarrow \Lambda$ at a leaf, the set $\Pi \cap \Lambda$ is non-empty.

We now discuss how to convert the tree into a cut-free proof. First of all, for any sequent $\Pi \longrightarrow \Lambda$ at a leaf, find a variable $p \in \Pi \cap \Lambda$ and tack above the leaf the sequent $p \longrightarrow p$ as axiom.

Now observe that the tree-pieces in cases 1, 2, 4, and 5 are valid inferences as they stand provided we restructure the bottom sequent so that the interesting formula is the first term of its cedent. To convert the configuration of case 3 into a string of valid inferences, we doctor it as follows:

$$\begin{tikzcd}
\Pi \longrightarrow C, D, \Lambda^* \arrow[d, dash] \\
\Pi \longrightarrow C \lor D, D, \Lambda^* \arrow[d, dash] \\
\Pi \longrightarrow D, C \lor D, \Lambda^* \arrow[d, dash] \\
\Pi \longrightarrow C \lor D, C \lor D, \Lambda^* \arrow[d, dash] \\
\Pi \longrightarrow C \lor D, \Lambda^*
\end{tikzcd}$$

The conversion of case 6 is similar; the only difference is the application of the ($\wedge$-left) Rule instead of the ($\vee$-right) Rule. There is also a considerable amount of restructuring as before. It is left to the doggedly dubious student to confirm that the converted tree (together with its labelling) is indeed a sequent calculus proof. $\dashv$

Corollary (3) (Cut-Elimination) If $\Gamma \longrightarrow \Delta$ has a sequent calculus proof, then it has a cut-free proof.

proof: This follows immediately from the Soundness and Completeness Theorems. $\dashv$

Lemma (4) (Subformula Property) In any rule of inference other than the Cut Rule, each formula in the hypothesis is a subformula of some formula of the conclusion. In particular, in a cut-free proof of $\Gamma \longrightarrow \Delta$, every formula appearing in the proof is a subformula of a formula in $\Gamma \longrightarrow \Delta$.

proof: By induction on the rules of inference. $\dashv$

D. Results Concerning the Enormity of Sequent Calculus Proofs

Definition: The SIZE of a proof is the number of occurrences of symbols in the sequents in the proof, i.e. the symbols inherent to the *tree* do not count. The size of a proof P is denoted $|P|$.

We engage in a stream-of-consciousness estimate on the size of the proof generated in the completeness theorem. Let $n = |\Gamma \longrightarrow \Delta|$. Then the length of any sequent in the unmodified tree is less than or equal to n, thanks to the Subformula Property.

There are fewer than or equal to n subformulas in $\Gamma \longrightarrow \Delta$. Given any subformula in $\Gamma \longrightarrow \Delta$, it is possible to determine whether it will appear in the antecedent or the succedent of a previous formula. Observe that the only rules of inference in which the principal formula jumps across the arrow are the $\neg$ Rules, so we only need to count the number of negation signs binding on the formula in question. So to specify a sequent in the tree, it suffices to say which subformulas of $\Gamma \longrightarrow \Delta$ occur in it (up to their order). We have to distinguish among different occurrences of a subformula in $\Gamma \longrightarrow \Delta$. So there are fewer than or equal to 2^n distinct sequents in the tree. So the unmodified tree has size less than or equal to $n2^n$.

We must add the structural inferences which reshaped the tree into a proof, and the conversions for cases 3 and 6. The structural inferences can at most double the size of the tree. Each *new* sequent in the case-3-and-6 conversions is at most *twice* the length of the original sequent, and there are *three* of them. Hence these coversions can introduce at most *six* extra tree-lengths. In total, muliplying by 8 gives an upper bound of $8n2^n$ on the size of a cut-free proof of $\Gamma \longrightarrow \Delta$. Note that this is just about on the same order as the size of a truth table.

How good is our upper bound? It's difficult to say. It is not known whether there is a subexponential upperbound on the size of proofs with cuts. If there is a polynomial bound, then NP = co-NP (defined at the end of the second set of notes).

To continue our investigation, we add the character $\supset$ to the language, as well as the following new rules of inference:

9. ($\supset$-right) Rule:

$$\frac{A, \Gamma \longrightarrow B, \Delta}{\Gamma \longrightarrow A \supset B, \Delta}$$

10. ($\supset$-left) Rule:

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

The manic student can verify in detail that the Completeness Theorem and the Subformula Property remain valid in this expanded system. In particular, our calculated upper bound still works.

Definition: A function $f : \mathbb{N} \to \mathbb{N}$ is $\mathcal{O}(g)$ for some $g : \mathbb{N} \to \mathbb{N}$ iff there exists a constant $c > 0$ such that for all $n \in \mathbb{N}$, $f(n) \leq c \cdot g(n)$.

Definition: A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is $o(g)$ iff for every constant $c > 0$, there is some $N \in \mathbb{N}$ such that for all $n > N$, $f(n) < c \cdot g(n)$.

Definition: A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is $\Omega(g)$ iff there exists a constant $c > 0$ such that for all $n \in \mathbb{N}$, $f(n) > c \cdot g(n)$.

Definition: A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is $\omega(g)$ iff for all constants $c > 0$, there is some $N \in \mathbb{N}$ such that for all $n > N$, $f(n) > c \cdot g(n)$.

Definition: A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is $\theta(g)$ iff f is both $\mathcal{O}(g)$ and $\Omega(g)$.

Let's now define some notation. Let $c_1, d_1, c_2, d_2, \ldots$, be propositional variables. Keeping in mind that our convention for serial conjunction is association from left to right, we define

$$F_i = \bigwedge_{k=1}^{i} c_k \vee d_k$$

$$\begin{array}{ll} A_1 = c_1 & B_1 = d_1 \\ A_{i+1} = F_i \supset c_{i+1} & B_{i+1} = F_i \supset d_{i+1} \end{array}$$

Let $\Gamma_n \longrightarrow \Delta_n$ be $A_1 \vee B_1, \ldots, A_n \vee B_n \longrightarrow c_n, d_n$. It is fairly easy to verify that $|\Gamma_n \longrightarrow \Delta_n|$ is $\mathcal{O}(n^2)$.

Lemma (5) $\Gamma_n \longrightarrow \Delta_n$ has a proof of size $\mathcal{O}(n^3)$.

proof: The sequent

$$F_i, A_{i+1} \vee B_{i+1} \longrightarrow c_{i+1}, d_{i+1}$$

has a proof with 13 sequents, and each sequent has $\mathcal{O}(i^2)$ symbols. Hence

$$F_i, A_{i+1} \vee B_{i+1} \longrightarrow F_{i+1}$$

has a cut-free proof of size $\mathcal{O}(i^2)$. By putting these sequents together with cuts and structural inferences, one gets a proof (with cuts) of size $\mathcal{O}(n^3)$ of $\Gamma_n \longrightarrow \Delta_n$. $\dashv$

Math 271, Set #2 **Instructor: Sam Buss**
Date: Jan,29-Feb,3 **Notes by: Maria Bonet.**

Theorem A–1: for arbitrary large m, there exists a valid sequent $\Gamma \longrightarrow \Delta$ of size m such that:

a) $\Gamma \longrightarrow \Delta$ has a proof with cuts of size $\mathcal{O}(m^{1.5})$

b) Any cut-free proof of $\Gamma \longrightarrow \Delta$ has $\geq 2^{\sqrt{m}}$ sequents.

References:

R. Statman in Annals of Mathematical Logic no.15 (p.225-287), 1978. G. Takeuti: private communication, 1987. The proof presented below is a slightly simplified version of Takeuti's proof.

Let's recall the following notation:

$$\begin{aligned} F_i &\equiv \bigwedge_{k=1}^{i} (c_i \vee d_i) \qquad \text{we associate from left to right} \\ A_1 &\equiv c_1 \\ A_{i+1} &\equiv F_i \supset c_{i+1} \\ B_1 &\equiv d_1 \\ B_{i+1} &\equiv F_i \supset d_{i+1} \end{aligned}$$

Let $\Gamma_n \longrightarrow \Delta_n$ be:

$$A_1 \vee B_1, \ldots, A_n \vee B_n \longrightarrow c_n, d_n$$

$|\Gamma_n \longrightarrow \Delta_n| = \mathcal{O}(n^2)$

To prove Theorem A–1 it suffices to show:

a)$\Gamma_n \longrightarrow \Delta_n$ has a proof with cuts of size $\mathcal{O}(n^3) = \mathcal{O}((n^2)^{1.5})$

b) A cut-free proof of $\Gamma_n \longrightarrow \Delta_n$ has $\geq 2^n$ sequents

Part a) was done in Lemma A–2(Fred Teti's Lemma 5)

Before proving Theorem A–1 we'll prove the following lemmas:

Lemma A–3: If P is a cut-free proof of $A \vee B, \Gamma \longrightarrow \Delta$ then there exists a cut-free proof P_1 of $A, \Gamma \longrightarrow \Delta$ with $|P_1| \leq |P|$ and the number of sequents in $P_1 \leq$ number of sequents in P.

Notation:# Seq.(P)= the number of sequents in P.

proof: Find all the direct ancestors of the indicated $A \vee B$. Change them to A. The result is a tree P_1 of sequents which can be modified to be a proof by discarding some of the branches:

It can fail to be a proof in a $\vee$-left inference. In P we have:

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

In P_1 it becomes:

$$\frac{A, \Gamma^* \longrightarrow \Delta \qquad B, \Gamma^* \longrightarrow \Delta}{A, \Gamma^* \longrightarrow \Delta}$$

In this case we have to discard $B, \Gamma^* \longrightarrow \Delta$ and everything above, and optionally one of the $A, \Gamma^* \longrightarrow \Delta$

(This proof could also fail if we were to allow axioms with non atomic formulas. In that case we could have the sequent $A \vee B \longrightarrow A \vee B$ as a leaf in P, but in P_1 we might get $A \longrightarrow A \vee B$. This is easily derived but at the cost of one extra inference.)

In the case of the structural rule, if in P we have, for example,

$$\frac{A \vee B, \Gamma \longrightarrow \Delta}{A \vee B, A \vee B, \Gamma \longrightarrow \Delta}$$

in P_1 it could become either one of the following choices:

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

both choices are valid inferences. QED(lemma A-3)

Lemma A–4: if P is a cut-free proof of $\Gamma \longrightarrow A \wedge B, \Delta$ then there is a cut-free proof P_1 of $\Gamma \longrightarrow A, \Delta$ with $|P_1| \leq |P|$ and #seq(P_1)$\leq$ #seq(P).

Lemma A–5: if P is a cut-free proof of $A \supset B, \Gamma \longrightarrow \Delta$, then there are cut-free proofs P_1 and P_2 of $B, \Gamma \longrightarrow \Delta$ and of $\Gamma \longrightarrow A, \Delta$ with $|P_i| \leq |P|$ and #seq.(P_i)$\leq$ #seq.(P).

Lemma A–6: Any cut-free proof of $\Gamma_n \longrightarrow \Delta_n$ has $\geq 2^n$ sequents.

Notation: $E_i \equiv A_i \vee B_i$

proof: the idea is to have to use many $\vee$-left rules and keep the two branches. Since there are n disjunctions in Γ_n this would lead to 2^n distinct branches in the proof tree. However, in general just having n disjunctions in Γ_n won't lead to 2^n branches, since if one of the two branches gets canceled all the time, the proof can become polynomial. Example:

$$D_1, \neg D_1 \vee D_2, \ldots, \neg D_n \vee D_{n+1} \longrightarrow D_{n+1}$$

has polynomial size tree-like cut-free sequent calculus proofs.

Let P be a cut-free proof of $\Gamma_n \longrightarrow \Delta_n$ with the fewest possible number of sequents. The last inference of P is either structural or a $\vee$-left operation with one of the $A_i \vee B_i$ as principal formula.

The final nonstructural inference of the proof is of the form:

$$\frac{E_1, \ldots, E_{i-1}, A_i, \ldots, E_n \longrightarrow c_n, d_n \qquad E_1, \ldots, E_{i-1}, B_i, \ldots, E_n \longrightarrow c_n, d_n}{E_1, \ldots, E_{i-1}, A_i \vee B_i, E_{i+1}, \ldots, E_n \longrightarrow c_n, d_n}$$

We are being imprecise in our notation: the E_i's can occur multiple times and in arbitrary order and, strictly speaking the $A_i \vee B_i$ should be the first formula in the lower sequent. However, we assume without loss of generality that no other occurence of $A_i \vee B_i$ is in the upper sequents. This is because P is the shortest possible proof, and by problem 4 (HW #1), a cut-free proof of $A, A \vee B, \Gamma \longrightarrow \Delta$ can be shortened to a cut free proof of $A, \Gamma \longrightarrow \Delta$.

It will suffice to show that any cut-free proof R_1 (S_1) of

$$E_1, \ldots, A_i, \ldots, E_n \longrightarrow c_n, d_n \quad (E_1, \ldots, B_i, \ldots, E_n \longrightarrow c_n, d_n)$$

can be shortened to get a proof of $\Gamma_{n-1} \longrightarrow \Delta_{n-1}$ with fewer sequents in the proof. Then,

$$\begin{aligned}
\#\text{seq.}(P) &\geq \#\text{seq}(R_1) + \#\text{seq}(S_1) \\
&\geq 2.\#\text{seq(shortest proof of } \Gamma_{n-1} \longrightarrow \Delta_{n-1}) \\
\#\text{seq.}(P) &\geq 2^n.
\end{aligned}$$

Let's see how we can shorten P_1 to a proof $\Gamma_{n-1} \longrightarrow \Delta_{n-1}$:

<u>case 1</u>: i=n.
Suppose R_1 is a cut-free proof of $E_1, \ldots, E_{n-1}, A_n \longrightarrow c_n, d_n$.
Since $A_n \equiv F_{n-1} \supset c_n$ by lemma 5, R_1 can be shortened to a cut-free proof R_2 of

$$E_1, \ldots, E_{n-1} \longrightarrow F_{n-1}, c_n, d_n \qquad \text{with } \#\text{seq.}(R_2) \leq \#\text{seq.}(R_1).$$

Since $F_{n-1} \equiv F_{n-2} \wedge (c_{n-1} \vee d_{n-1})$, by lemma 4 there is a cut-free proof R_3 of

$$E_1, \ldots, E_{n-1}, \longrightarrow c_{n-1} \vee d_{n-1}, c_n, d_n \qquad \text{with } \#\text{seq.}(R_3) \leq \#\text{seq.}(R_2).$$

The variables c_n, d_n occur only as indicated in the succedent. Hence c_n, d_n were introduced by weakening. So R_3 can be shortened to a cut free proof R_4 of

$$E_1, \ldots, E_{n-1} \longrightarrow c_{n-1} \vee d_{n-1}$$

and this can be shortened to a cut-free proof of

$$E_1, \ldots, E_{n-1} \longrightarrow c_{n-1}, d_{n-1}$$

and $\#\text{seq.}(R_1) \leq \#\text{seq.}(\text{the shortest proof of } \Gamma_{n-1} \longrightarrow \Delta_{n-1})$.

<u>Case 2</u>: $1 \leq i < n$.
Let R_1 be a cut-free proof of $E_1, \ldots, A_i, \ldots, E_n \longrightarrow c_n, d_n$
Idea: we are going to delete any occurrence of $c_i \vee d_i$ from the proof. Let,

$$\begin{aligned}
F^i_j &\equiv \bigwedge_{k=1,k\neq i}^{j} c_k \vee d_k \\
A^i_1 &\equiv c_1 \\
A^i_{j+1} &\equiv F^i_j \supset c_{j+1} \qquad j+1 \neq i \\
B^i_1 &\equiv d_1 \\
B^i_{j+1} &\equiv F^i_j \supset d_{d+1} \qquad j+1 \neq i \\
E^i_j &\equiv A^i_j \vee B^i_j
\end{aligned}$$

The goal from now on is to shorten the proof R_1 to a proof of:

$$E^i_1, \ldots, E^i_{i-1}, E^i_{i+1}, \ldots, E^i_n \longrightarrow c_n, d_n$$

Since A_i is $F_{i-1} \supset c_i$ by lemma A-5, R_1 can be shortened to a cut-free proof R_2 of

$$E_1, \ldots, E_{i-1}, c_i, E_{i+1}, \ldots, E_n \longrightarrow c_n, d_n$$

In R_2 replace every occurence of the subformula F_i by F_{i-1} to obtain a new tree of sequents R_3. This changes every F_j to F_j^i, A_j to A_j^i, B_j to B_j^i (if $i = 1$ replace each F_j by $\bigwedge_{\ell=2}^{j} c_\ell \vee d_\ell$ and replace E_2 by $c_2 \vee d_2$).

The only case where this is not a valid proof tree is where F_i was a principal formula of in inference in R_2. For instance

$$\frac{\Pi \longrightarrow F_{i-1}, \Lambda \qquad \Pi \longrightarrow c_i \vee d_i, \Lambda}{\Pi \longrightarrow F_i, \Lambda}$$

This becomes after the substitution:

$$\frac{\Pi \longrightarrow F_{i-1}, \Lambda \qquad \Pi \longrightarrow c_i \vee d_i, \Lambda}{\Pi \longrightarrow F_{i-1}, \Lambda}$$

To fix this modify R_3 by discarding $\Pi \longrightarrow c_i \vee d_i, \Lambda$ and the tree above. Now it's again a valid inference.

It's not possible to have F_i as principal formula in the antecedent in R_2, because it's preordained where the parts of $F_i \supset c_{i+1}$ will go. Also, we will never have $F_i \longrightarrow F_i$ as an axiom in R_2 because of our convention about having only atomic formulas in the antecedent and succedent of the formula. Hence the above changes to R_3 make it a valid proof.

So c_i occurs only in the antecedent of the endsequent of R_3. So, it must have been introduced by weakening. So we delete c_i everywhere in the proof to get a cut-free proof of:

$$E_1^i, \ldots, E_{i-1}^i, E_{i+1}^i, \ldots, E_n^i \longrightarrow c_n, d_n$$

Now for all $j > i$ we rename the variables c_j, d_j to c_{j-1}, d_{j-1}. This gives a cut-free proof of $\Gamma_{n-1} \longrightarrow \Delta_{n-1}$. QED(lemma A-6,Theorem A-1)

Propositional Proof Systems

Let Σ be any finite alphabet. $|\Sigma| \geq 2$. Let Σ^* be the set of finite strings from Σ. We'll suppose that Σ contains: p, 1, 0, (,), $\vee$, $\neg$, $\supset$, and comma.

Def: A propositional proof system is a function F computable in polynomial time with image the set of tautologies, and domain all Σ^*.

Example 1:

$$F_{cut-free}(w) = \begin{cases} A & \text{if } w \text{ is a valid cut-free proof of } \longrightarrow A \\ p \vee \neg p & \text{otherwise} \end{cases}$$

Example 2:

$$F_{TT}(w) = \begin{cases} A & \text{if } w \text{ codes a truth table proof of A} \\ p \vee \neg p & \text{otherwise} \end{cases}$$

Example 3:

$$F_{ZF}(w) = \begin{cases} A & \text{if } w \text{ is a ZF-proof that A is a tautology} \\ p \vee \neg p & \text{otherwise} \end{cases}$$

Def.: A decision problem is a subset of Σ^*.

Def.: A decision problem Q is in **P** iff there is a Turing Machine M such that for every x,
a) $x \in Q$ iff M accepts x.
b) for some polynomial p(n), M halts on input x within $p(|x|)$ steps.

Examples:
a) the set of palindromes is in **P**
b) set of cut-free proofs is in **P** (encoded as as string of symbols).

Feasible means doable on today's or next century's computers. **P** is the mathematical notion for the vague idea of 'feasible'. One might question whether they really coincide, since when the constants or exponents are very big in a polynomial function, the function doesn't seem very feasible.

Def: a decision problem Q is in **NP** iff there is $R(\ ,\ )$ in **P** and a polynomial $p(x)$ such that,
$\forall x(x \in Q \text{ iff } \exists w(|w| \leq p(|x|) \wedge R(x,w)).$

Def: Co-NP is the set of complements of members of **NP** . So, A$\in$**Co-NP** iff $\Sigma^*\backslash$A $\in$ **NP** .

Def.: $f \in$ **FP** iff $f : \Sigma^* \rightarrow \Sigma^*$ and there is a polynomial time Turing Machine M which starting with x on its input tape halts with $f(x)$ on its output tape in $\leq p(|x|)$ steps.

Consider the set **SAT** $=\{$A: A is satisfable$\}$. Cook showed that **SAT** is **NP** Complete. ie., for every $Q \in$ **NP** there is a many-one reduction f of Q to **SAT** , and $f \in$ **FP** . By a many-one reduction f of Q to **SAT** we mean a function f such that: $\forall x(x \in Q \leftrightarrow f(x) \in$ **SAT**).

Proposition: Let **TAUT** be the set of all tautologies. Then: a) **TAUT** is in **Co-NP** . b) **TAUT** is **Co-NP** Complete.
proof:
a) $\overline{\textbf{TAUT}} \in$ **NP** because:
$\varphi \in \overline{\textbf{TAUT}}$ iff $\varphi \notin$ **TAUT**
iff there is a truth assignment σ s.t. $\sigma(\varphi)$=F.
iff $\exists w|w| = \#$ of variables in φ,coding a truth assignment σ s.t. $\sigma(\varphi) = F$.
Since $\overline{\textbf{TAUT}} \in$ **NP** iff **TAUT** $\in$ **Co-NP** , **TAUT** $\in$ **Co-NP** .
b) It suffices to show that $\overline{\textbf{TAUT}}$ is **NP** -complete (A many-one reduction of Q to **TAUT** is the same as a many-one reduction of $\overline{Q}$ to $\overline{\textbf{TAUT}}$).
It suffices to give a many-one polynomial reduction of **SAT** to $\overline{\textbf{TAUT}}$:
$\varphi \in$ **SAT** $\iff \neg\varphi \notin$ **TAUT** $\iff \neg\varphi \in \overline{\textbf{TAUT}}$ So the reduction is $f : \varphi \mapsto \neg\varphi$. QED

Many-one reduction versus Turing reduction: In the many-one reduction you ask one question, and the answer you get from the oracle in one step is the total answer. In the Turing reductions you are allowed to ask the oracle as many times as you want, getting the answer in one step each time.

Recall we defined a Propositional Proof System to be a function $f \in$ **FP** such that $f : \Sigma^* \longrightarrow \Sigma^*$ and the image $f(\Sigma^*)$ is **TAUT** .

Def: a propositional proof system f is <u>Super</u> iff there is a polynomial $p(\ )$ s.t. $\forall x \in \textbf{TAUT}\ \exists w\ |w| \le p(|x|)$s.t. $f(w) = x$.

Theorem:[Cook-Reckhow 1974]
There exists a super propositional proof system iff **NP** =**Co-NP** .

proof:

($\Longrightarrow$) Suppose that f is super. Let p_f be the polynomial bound on the length of the proofs.

First note that **TAUT** $\in$ **NP** since,
$\forall x\ [x \in \textbf{TAUT} \iff \exists w |w| \le p_f(|x|)\text{s.t.} f(w) = x]$.

Now, let's prove that **NP** =**Co-NP** . Suppose that $Q \in$ **Co-NP** . Q is many-one reducible to **TAUT** . Since **TAUT** $\in$ **NP** , Q is many-one reducible to **SAT** by $g \in$ **FP** . So for all x,
$x \in Q \Leftrightarrow g(x) \in \textbf{SAT} \Leftrightarrow \exists w\ |w| \le |g(x)|, w$ is satisfying truth assignment of $g(x)$.

That shows that **Co-NP** $\subseteq$ **NP** . To see that **NP** $\subseteq$ **Co-NP** , suppose $Q \in$ **NP** . Then $\overline{Q} \in$ **Co-NP** . So $\overline{Q} \in$ **NP** and $Q \in$ **Co-NP** .

($\Longleftarrow$) Suppose that **NP** =**Co-NP** . So **TAUT** $\in$ **NP** . So there is a polynomial $p(\ )$ and $R(\ ,\ ) \in$**P** s.t., $\forall x(x \in \textbf{TAUT}$ iff $\exists w |w| \le p(|x|) s.t. R(x.w))$

Then the proof system is:

$$f(v) = \begin{cases} x & \text{if } v =< x, w > \text{ and } R(x, w) \\ p \vee \neg p & \text{otherwise} \end{cases}$$

QED.

Note: **P** =**NP** implies **NP** =**Co-NP** (because **P** is closed under complement). So **NP** $\neq$ **Co-NP** implies **P** $\neq$ **NP** . Therefore, if there is no super propositional proof system then **P** $\neq$**NP** .

Def: A propositional proof system g <u>simulates</u> a propositional proof system f iff there exists a polynomial p such that for all x and w with $f(w) = x$, there exists a w' satisfying $|w'| \le p(|w|)$ and $g(w') = x$.

Math 271 - Topics in Weak Formal Systems

Lecture Notes, Set #3 **Instructor: Sam Buss**
February 5–10, 1988 **Notes By: John Grosh**

Resolution Another Propositional Proof System.

Reference: G. S. Tseitin, "On the complexity of derivation in propositional calculus." circa. 1966, appears, Automation of Reasoning 2 pp. 466 - 483 Springer-Verlag 1983

Definitions

A *propositional variable* is denoted by p, q, r. Each propositional variable has a conjugate (or negative) denoted $\overline{p}$. Also $\overline{\overline{p}} = p$.

A *literal* is a propositional variable p or a conjugate $\overline{p}$.

A *clause* is a finite set of literals, where the meaning of the clause is the disjunction of the literals in the clause. For example $\{p_1, \overline{p}_2, p_3\}$ means $p_1 \vee \overline{p}_2 \vee p_3$.

If σ is a truth assignment, $\sigma(\overline{p}) =$ opposite of $\sigma(p)$. For a clause C, $\sigma(C) = T$ iff $\sigma(x) = T$ for some $x \in C$.

C is *satisfiable* if there is some truth assignment σ such that $\sigma(C) = T$. Note that only $\emptyset$ is not satisfiable as a clause.

More importantly, if $\mathcal{C}$ is a set of clauses, $\mathcal{C}$ *is satisfiable* if there is a truth assignment σ such that $\sigma(D) = T$ for all $D \in \mathcal{C}$.
We will never allow an empty set of clauses.

Resolution Rule

$$\frac{C_1 \cup \{x\} \qquad C_2 \cup \{\overline{x}\}}{C_1 \cup C_2}$$

Resolution has no axioms. Instead, we take $\mathcal{C}$ to be a set of hypotheses to which we apply the resolution rule.

Observation: If $\sigma(C_1 \cup \{x\}) = T$ and $\sigma(C_2 \cup \{\overline{x}\}) = T$ then $\sigma(C_1 \cup C_2) = T$.

Theorem 1 *If $\sigma(\mathcal{C}\ ) = T$ and if D can be inferred from $\mathcal{C}$ by repeated use of the resolution rule, then $\sigma(D) = T$.*

proof: repeated use of the above observation. □

Corollary 2 *If there is a resolution derivation of $\emptyset$ (the empty clause) from $\mathcal{C}$, then $\mathcal{C}$ is not satisfiable.* □

Resolution is a 'refutation' proof system; from a conjunctive normal form formula obtain a set of clauses, then derive the empty clause to refute the original formula.

Theorem 3 (Completeness) *If $\mathcal{C}$ is an unsatisfiable set of clauses, then there is a resolution derivation of the empty clause $\emptyset$ from $\mathcal{C}$.*

proof: (Reference: Davis, Putnam "A Computing Procedure For Quantification Theory" JACM 1 (1960) pp. 201–215. They prove a stronger result.)

Let the propositional variables in clauses in $\mathcal{C}$ be among $p_1 \ldots p_n$, $\bar{p}_1 \ldots \bar{p}_n$.

Goal: Get rid of one variable, say p_n, by deriving a nonempty set of clauses $\mathcal{C}^*$ from $\mathcal{C}$ such that neither p_n or $\bar{p}_n$ appears in any clause of $\mathcal{C}^*$ and $\mathcal{C}^*$ is unsatisfiable.

$\mathcal{C}$ contains four types of clauses.

(a) clauses that contain p_n and not $\bar{p}_n$.
(b) clauses that contain $\bar{p}_n$ and not p_n.
(c) clauses that contain neither p_n or $\bar{p}_n$.
(d) clauses that contain both p_n and $\bar{p}_n$.

Directions for forming $\mathcal{C}^*$.

(1) Put every clause of type (c) into $\mathcal{C}^*$.
(2) Throw away all clauses of type (d).
(3) For each clause C of type (a) and D of type (b), $\mathcal{C}^*$ contains the result of resolving C and D with respect to p_n.

We claim that $\mathcal{C}^*$ is not empty. We may assume that $\mathcal{C}$ is not empty since we don't allow the empty set of clauses. Then $\mathcal{C}^*$ is empty just in case both: (1) $\mathcal{C}$ has no type (c) clauses, and (2) either $\mathcal{C}$ has no type (a) clauses or it has no type (b) clauses. In this case, $\mathcal{C}$ is satisfiable with $\sigma(\bar{p}_n) = T$ or $\sigma(p_n) = T$ resp.

The next claim is that if $\mathcal{C}^*$ is satisfiable then $\mathcal{C}$ is also. Suppose $\sigma(\mathcal{C}^*) = T$. We let σ_{p_n} and $\sigma_{\bar{p}_n}n$ be the truth assignments extending σ such that $\sigma_{p_n}(p_n) = T$ and $\sigma_{\bar{p}_n}(\bar{p}_n) = T$.

Consider the following cases:

Case 1: $\mathcal{C}$ had no clauses of types (a) or (b). Then define $\sigma(p_n) =$ (anything) and σ satisfies $\mathcal{C}$.

Case 2: $\mathcal{C}$ had no clauses of type (a) [resp. type (b)] then $\sigma_{\bar{p}_n}$ [resp. σ_{p_n}] satisfies $\mathcal{C}$.

Case 3: $\mathcal{C}$ had clauses of both types (a) and (b). Suppose that neither σ_{p_n} nor $\sigma_{\bar{p}_n}$ satisfies $\mathcal{C}$. Then we must have clauses D_1 of type (a) and D_2 of type (b) such that $\sigma_{p_n}(D_2) = False$ and $\sigma_{\bar{p}_n}(D_1) = False$. Consider the resolution of D_1 and D_2 on the variable p_n. It will yield $D = D_1\backslash\{p_n\} \cup D_2\backslash\{\bar{p}_n\}$ where neither p_n or $\bar{p}_n$ occurs in D. By construction of $\mathcal{C}^*$ we know that $D \in \mathcal{C}^*$. So $\sigma(D) = T$. Then for some $x \in D$ (either a propositional variable or its negative) $\sigma(x) = T$. And $x \in D_1$ or $x \in D_2$. This is a CONTRADICTION since we forced σ_{p_n} and $\sigma_{\bar{p}_n}$ to extend σ. Hence one of σ_{p_n}, $\sigma_{\bar{p}_n}$ satisfies $\mathcal{C}$. So $\mathcal{C}$ is satisfiable.

We have now produced an unsatisfiable set $\mathcal{C}^*$ of clauses with no occurrences of p_n, $\bar{p}_n$. Completing the induction we will be able to produce a resolution derivation of the empty clause. □

So resolution is a proof system for Disjunctive Normal Form (DNF) formulas. Given a formula A in DNF, write the negation of A ($\neg$A) as a Conjunctive Normal Form (CNF) formula. Convert this to a set of clauses (each conjunct becomes a clause and each disjunct within a clause becomes a member of the clause). Refute $\neg$A as in the completeness theorem to show that A is a tautology.

It is desirable to have resolution be a proof system for arbitrary propostional formulas, not just DNF formulas. There are two methods for achieving this. The first method might be to convert our formulas to DNF first, then use resolution. The problem is that this may make the formula exponentially larger. So we discuss a second method called resolution with limited extension.

Resolution with Limited Extension

Resolution with limited extension will work as a proof system for general propositions.

Idea: Introduce new variables for each subformula. For each subformula B of our formula A we will have the variable p_B with the following requirements. (1) If B is atomic, say B is p_i, then $p_B = p_i$. (2) Otherwise the p_B are distinct for distinct subformulas of A.

Definition For a formula A we define a set of clauses LE(A), (the limited extension of A) as follows. For each subformula B of A,

Case 1: B is ¬C then $\{ p_B, p_C \}$, $\{ \bar{p}_B, \bar{p}_C \}$ ∈LE(A). It is not hard to see that truth assignment σ will satisfy these two clauses iff $\sigma(p_B) = \sigma(\bar{p}_C)$.
Case 2: B is C∨D then $\{ p_B, \bar{p}_C \}$, $\{ p_B, \bar{p}_D \}$, $\{ \bar{p}_B, p_C, p_D \}$ ∈LE(A). Again σ satisfies these three clauses iff $\sigma(p_B) = \sigma(p_C \vee p_D)$.
Case 3: B is C∧D then $\{ \bar{p}_B, p_C \}$, $\{ \bar{p}_B, p_D \}$, $\{ p_B, \bar{p}_C, \bar{p}_D \}$ ∈ LE(A).

Definition: The *size* of a clause is the cardinality of the clause. The *size* of a set of clauses is the sum of the sizes of its clauses.

Remark: |LE(A)| is $\mathcal{O}(|A|)$.

Theorem 4 *(a) A is satisfiable iff LE(A)* ∪ $\{ \{ p_A \} \}$ *is satisfiable.*
(b) A is valid iff LE(¬A) ∪ $\{ \{ \bar{p}_A \} \}$ *is not satisfiable iff there is a resolution derivation of* $\emptyset$ *from LE(¬A)* ∪ $\{ \{ \bar{p}_A \} \}$.

The proof proceeds by induction on the complexity of B, a subformula of A, showing that $\sigma(B) = \sigma(p_B)$ for all σ satisfying LE(A).□

We will next examine sizes of resolution proofs of the pigeonhole principle.

Pigeonhole principle: For each n, if $f : \{0, \dots, n\} \to \{0, \dots, n-1\}$ then f is not one-to-one.

For each i and j with $0 \leq i \leq n$ and $0 \leq j \leq n-1$ we will have the variable $p_{i,j}$ which 'means' $f(i) = j$.

$$\mathbf{PHP}_n \qquad \underbrace{\Big(\bigwedge_{0\leq i\leq n} \bigvee_{0\leq j\leq n-1} p_{i,j}\Big)}_{\text{f is total}} \longrightarrow \underbrace{\bigvee_{0\leq i<m\leq n} \bigvee_{0\leq j\leq n-1} p_{i,j} \wedge p_{m,j}}_{\text{f is not one-to-one}}$$

We've omitted the requirement that f be single-valued.

$$\neg\mathbf{PHP}_n \qquad \Big(\bigwedge_{0\leq i\leq n} \bigvee_{0\leq j\leq n-1} p_{i,j}\Big) \wedge \bigwedge_{0\leq i<m\leq n} \bigwedge_{0\leq j\leq n-1} (\bar{p}_{i,j} \vee \bar{p}_{m,j})$$

This expresses $\neg\mathrm{PHP}_n$ in conjunctive normal form.

$\mathcal{C}_{\neg\mathrm{PHP}_n}$ contains $\{p_{i,0},\ldots,p_{i,n-1}\}$ for each $i=0,\ldots,n$. And $\{\bar{p}_{i,j},\bar{p}_{m,j}\}$ for each i, j, m such that $0\le i<m\le n$ and $0\le j\le n-1$.

Theorem 5 (A. Haken) *There exists a constant $c>1$ such that any resolution derivation of $\emptyset$ from $\mathcal{C}_{\neg PHP_n}$ has $e^{c\cdot n}$ clauses.*

Reference for proof: A. Haken, "The Intractibility of Resolution." TCS 39 (1985) pp. 297–305.
also S. Buss and Gy. Turán ... to appear TCS.

Recall: Resolution proofs are sequences not trees. There were earlier results for 'tree-like' and 'regular' resolutions. (e.g. see G. S. Tsietin – 1966)

proof. Assume P is a derivation of $\emptyset$. We want a lower bound on the size of P. A clause is pictured as an $n\times(n+1)$ array of $+$'s, $-$'s and blanks. For example

$$\begin{array}{c c} & \begin{array}{cc} i_0 & i_1 \end{array} \\ \begin{array}{c} j_1 \\ \\ j_0 \end{array} & \underbrace{\left[\begin{array}{cccc} & & - & \\ & & & \\ & + & & \end{array}\right]}_{n+1} \end{array} \left.\vphantom{\begin{array}{c}1\\1\\1\end{array}}\right\} n \qquad \text{stands for the clause}\{p_{i_0,j_0},\bar{p}_{i_1,j_1}\}$$

Initial clauses in $\neg\mathrm{PHP}_n$ look like:
$$\begin{bmatrix} & + & \\ & + & \\ & + & \\ & + & \end{bmatrix} \qquad \begin{bmatrix} - & & - \\ & & \\ & & \\ & & \end{bmatrix}$$

Similarly a truth assignment σ is pictured as an array of 0's, and 1's where $0=\mathit{False}$ and $1=\mathit{True}$.

Definition: A truth assignment is *critical* if it has exactly n 1's with no two in the same column. (i.e. it codes a partial one-to-one function $f : \{0, \ldots, n\} \rightarrow \{0, \ldots, n-1\}$ with n values of f defined. One 'pigeon' is undetermined.)

Definition: The *0-column* of a critical truth assignment is the index of the column with no 1's in it.

Each critical truth assignment will be assigned a clause in P.

Lemma 6 *Let σ be a critical truth assignment. There is a clause C in P such that:*
(a) $\sigma(C) = F$
(b) C has exactly $\left\lfloor \frac{n}{2} \right\rfloor$ + 's in the 0-column of σ.

proof. Note that if, in a resolution inference $\frac{D_1 \quad D_2}{D}$, we have $\sigma(D) = F$ then $\sigma(D_1) = F$ or $\sigma(D_2) = F$ but not both. By tracing backwards through P we get a unique sequence of clauses $C_1, \ldots, C_t$ such that:

(1) C_t is the final clause $\emptyset$.
(2) C_{i+1} is (resolution) inferred from C_i and something else.
(3) C_1 is an initial clause.
(4) $\sigma(C_i) = F$ for all $i \leq t$.

Since σ is critical, C_1 must be the clause with n +'s in the 0-column of σ. Of course, C_t has no +'s in the 0-column of σ. Let C be the first C_i with $\leq \left\lfloor \frac{n}{2} \right\rfloor$ +'s in the 0-column of σ. Then C has exactly $\left\lfloor \frac{n}{2} \right\rfloor$ +'s in the 0-column of σ since the +'s disappear one at a time (by being resolved on).

Lemma 7 *Let σ be critical. Suppose C is a clause in P with $< \left\lfloor \frac{n}{2} \right\rfloor$ + 's in the 0-column of σ such that $\sigma(C) = F$, then there is a clause D before C in P such that $\sigma(D) = F$ and D has exactly $\left\lfloor \frac{n}{2} \right\rfloor$ + 's in the 0-column of σ.*

proof: same as previous lemma except letting C_t be D instead of $\emptyset$. $\square$

Definition: If σ is a critical truth assignment, we define C_σ to be the <u>first</u> clause in P satisfying:
(a) $\sigma(C_\sigma) = F$
(b) C_σ has exactly $\left\lfloor \frac{n}{2} \right\rfloor$ $+$'s in the 0-column of σ.

Definition: An FS1 (fixed set of 1's) is a set S of $\left\lfloor \frac{n}{4} \right\rfloor$ 1's in distinct rows and columns of the array.

Definition: *σ is compatible with S* if $\sigma(p_{i,j}) = T$ for all $p_{i,j} \in S$.
We will take C^S to be the first clause in P of the form C_σ for some σ compatible with S. C^S is called a *complex clause.*

Lemma 8 *Any complex clause has* $\geq (\left\lfloor \frac{n}{4} \right\rfloor + 1)$ *columns which either contain* $\geq \left\lfloor \frac{n}{2} \right\rfloor$ $+'s$ *or contain a* $-$.

proof: Let C^S be a complex clause, and S an FS1. Then $C^S = C_\sigma$ for some σ compatible with S. Here's an example:

$$C^S = C_\sigma = \left[\begin{array}{ccccc|c} - & & & + & + & \\ & - & & & & + \\ & + & & & + & \\ + & + & & & & + \\ & + & & & - & + \end{array}\right] \qquad \sigma = \left[\begin{array}{ccccc|c} 1 & & & & 0 & 0 \\ & 1 & & & & 0 \\ & & 1 & & & 0 \\ & & & 1 & & 0 \\ 0 & & & & 1 & 0 \end{array}\right]$$

$0\text{--}column$

Note that $-$'s (resp. $+$'s) in C_σ can appear only where 1's (resp. 0's) appear in σ. For pictorial convenience we showed σ with 1's on the diagonal and 0's elsewhere but of course the rows and columns may be scrambled.

To resume the proof of Lemma 8, assume C^S has $\leq \left\lfloor \frac{n}{4} \right\rfloor$ columns with either $\left\lfloor \frac{n}{2} \right\rfloor$ $+'s$ or a $-$.

Goal: Find a truth assignment τ such that τ is compatible with S and C_τ precedes $C_\sigma = C^S$ in P. This will CONTRADICT the definition of C^S. τ is obtained by swapping a column i of σ with column $n+1$ of σ where $n+1$ is the 0-column of σ.

Choose the column i such that τ will satisfy the following:

(1) τ is compatible with S.

(2) $\tau(C^S) = F$.

(3)C^S has $< \left\lfloor \frac{n}{2} \right\rfloor +'s$ in the column i (and hence in the 0-column of τ.

To do this, pick i such that:

(a) i is not a column containing a variable of S.

(b)i is not a column with a 1 in a row where the column of C^S corresponding to the 0-column of σ has a $+$.

(c) i is not a column in which C^S has a minus sign.

(d) i does not have $\geq \left\lfloor \frac{n}{2} \right\rfloor +'s$.

By counting, we find that there must be at least one such column i. Condition (a) excludes exactly $\left\lfloor \frac{n}{4} \right\rfloor$ possibilities for i (look at the size of S). Condition (b) excludes exactly $\left\lfloor \frac{n}{2} \right\rfloor$ possibilities for i since the column of C^S corresponding to the 0-column of σ has in it exactly $\left\lfloor \frac{n}{2} \right\rfloor$ +'s by the definition of $C^S = C_\sigma$. By the assumption of our lemma, (c) and (d) exclude at most $\left\lfloor \frac{n}{4} \right\rfloor$ values for i. Since we have $n+1$ columns there must be at least one choice of column for i.

Now our goal is achieved by Lemma 7. (Recall our requirements (2) and (3) for τ.) This finishes the proof of Lemma 8. □

We now resume the proof of Haken's theorem.

Put $g(n) = \max_C\{|\{S \in \mathrm{FS1} : C^S = C\}|\}$ and $h(n) = |\mathrm{FS1}|$. Then $h(n)/g(n)$ is a lower bound to the length of a resolution proof, since it is clearly a lower bound on the number of distinct complex clauses in the resolution proof. Let $k = \left\lfloor \frac{n}{4} \right\rfloor$. To compute $h(n)$ and $g(n)$ suppose we have a particular complex clause C. By Lemma 8 we can choose $k+1$ columns which contain a $-$ or at least $\left\lfloor \frac{n}{2} \right\rfloor$ $+$'s. To count the total number of $S \in \mathrm{FS1}$ we let the variable i denote the number of variables in S in the chosen $k+1$ columns. Then we have:

$$h(n) = \sum_{i=0}^{k} \binom{k+1}{i} \binom{m-k-1}{k-i} \frac{n!}{(n-k)!}$$

Similarly, to get the upper bound $g(n)$ on the number of $S \in \mathrm{FS1}$ such that $C^S = C$ we let i be the number of variables of S in one of the $k+1$ columns. In each of these $k+1$ columns there are at most $\left\lceil \frac{n}{2} \right\rceil$ variables which can be in such an S; this is because a $+$ in C excludes the corresponding variable from S and a $-$ in C implies that if S has a variable from that column it must be the variable corresponding to the $-$. Thus,

$$g(n) \leq \sum_{i=0}^{k} \binom{k+1}{i} \binom{m-k-1}{k-i} \left\lceil \frac{n}{2} \right\rceil^i \frac{(n-i)!}{(n-k)!}$$

So,

$$\begin{aligned} \frac{h(n)}{g(n)} &\geq \frac{\sum_{i=0}^{k} \binom{k+1}{i} \binom{m-k-1}{k-i}}{\sum_{i=0}^{k} \binom{k+1}{i} \binom{m-k-1}{k-i} \left\lceil \frac{n}{2} \right\rceil^i \frac{(n-i)!}{n!}} \\ &\geq \frac{\sum_{i=0}^{k} \binom{k+1}{i} \binom{m-k-1}{k-i}}{\sum_{i=0}^{k} \binom{k+1}{i} \binom{m-k-1}{k-i} \left(\frac{2}{3}\right)^i} \end{aligned}$$

since for $i \leq \left\lfloor \frac{n}{4} \right\rfloor$,

$$\left\lceil \frac{n}{2} \right\rceil^i \frac{(n-i)!}{n!} \leq \left(\frac{2}{3}\right)^i$$

The ratio of the $(i-1)$-th term over the i-th term in the summation in the denominator is

$$\frac{i(m-2k+i-1)}{\frac{2}{3}(k-i+1)(k-i+2)}$$

It is easily verified that this is less than 1 for $i \leq \frac{1}{25} \cdot \frac{n^2}{m}$, and hence the terms in the denominator are increasing while $i \leq \frac{1}{25} \cdot \frac{n^2}{m}$. Thus we can give a weaker lower bound (with smaller numerator and larger denominator):

$$\begin{aligned} \frac{h(n)}{g(n)} &\geq \frac{\sum_{i=\frac{1}{50}\frac{n^2}{m}}^{k} \binom{k+1}{i}\binom{m-k-1}{k-i}}{2 \cdot \sum_{i=\frac{1}{50}\frac{n^2}{m}}^{k} \binom{k+1}{i}\binom{m-k-1}{k-i}\left(\frac{2}{3}\right)^i} \\ &\geq \frac{1}{2}\left(\frac{3}{2}\right)^{\frac{1}{50}\frac{n^2}{m}} \end{aligned}$$

Hence the number of distinct (complex) clauses in P is at least exponential in n.□

Math 271 - Topics in Weak Formal Systems

Lecture Notes, Set #4 **Instructor: Sam Buss**
1988 February 12–24 **Notes By: Martin Goldstern**

This exponential lower bound (from the previous notes) for resolution is bad for automated theorem provers, since most theorem proving systems use resolution.

Could it be that this exponential length occurs only in a "few" pathological cases? No. Not "only" the pigeonhole principle needs these "long" proofs: Chvátal and Szemerédi ("Many hard examples for resolution") extended Haken's work to show that in some sense "almost all" sets of clauses formulas (are unsatisfiable and) have a shortest proof of exponential length.

We can still hope that in applications these "worst cases" don't happen. But even if we could somehow guarantee the existence of a polynomial-sized refutation, it might be hard (i.e. require exponential time) to find it.

There are several strategies that try to avoid this:

Strategy # 1: Use a restrictive form of resolution. This sounds paradoxical, since it may make proofs longer, but the proofs may be easier to find.

Example: A *linear resolution* of a set Δ of clauses is a sequence of clauses $C_1, \ldots, C_n$, where $C_1 \in \Delta$, and C_{i+1} is obtained by resolution from C_i and a clause in $\Delta \cup \{C_1, \ldots, C_{i-1}\}$. Why is this easier? When we want to do a resolution, we already know that we have to use C_i, so we have (roughly) only $\mathcal{O}(\|\Delta\|)$ options, instead of $\mathcal{O}(\|\Delta\|^2)$ options. It can be shown that linear resolution is complete, i.e. if Δ is an unsatisfiable set of clauses, then there is a linear refutation of Δ.

Strategy # 2: Restrict what sets of clauses Δ are allowed. The most common restriction is the restriction to *Horn clauses.*

Definition: A Horn clause is a clause with at most one unnegated propositional variable.
The Horn clause $\{p_0, \bar{p}_1, \ldots, \bar{p}_n\}$ corresponds to the formula $p_1 \wedge \ldots \wedge p_n \to p_0$, which is commonly written as

$$p_0 \leftarrow p_1, \ldots, p_n.$$

Not every proposition is expressible as a set of Horn clauses, e.g. $A \to B \vee C$.

In "expert systems", one usually has a "database" Δ consisting of Horn clauses. One such clause might be [1]

$$p_{\text{MEASLES}} \leftarrow p_{\text{FEVER}}, p_{\text{RASH}}, p_{\text{TEA}}.$$

To use this database, we input a set Γ of "observations", for example : $\Gamma = \{\{p_{\text{FEVER}}\}, \{p_{\text{RASH}}\}, \{p_{\text{TEA}}\}\}$, and the "query" γ: Does he have measles?, i.e. $\gamma = \{\bar{p}_{\text{MEASLES}}\}$. The answer will be "yes", iff $\Delta \cup \Gamma \cup \{\gamma\}$ is unsatisfiable. We use linear resolution to find an answer. The fact that $\Gamma \cup \Delta$ is satisfiable can help the search. In particular, $C_1 = \{\gamma\}$ in linear resolution.

It is not hard to show that sequent calculus (with cuts) simulates resolution. The proof will be left as an exercise. Notice that modus ponens is a form of resolution, and vice versa.

Definition: Let S and T be proof systems. Then we say that S *p-simulates* T ("polynomially simulates T"), if there is a polynomial-time algorithm which, given a T-proof of a formula A, produces an S-proof of A.

Note that the length of this S-proof is polynomially bounded by the length of the T-proof, since an algorithm that terminates after a "short" time can

[1] A few weeks ago a person with measles was at the math department's tea hour. The next day there was a note on the bulletin board, saying that anybody who had been to the tea, and developed a rash and fever during the next week probably got infected.

only produce "short" output. Hence every p-simulation is a simulation. Is every simulation a p-simulation? If not, then $P \neq NP$.

Frege Proof Systems

A *Frege ($\mathcal{F}$) proof system* consists of

1. A language $\mathcal{L}$, a finite complete set of propositional connectives.
2. A finite set of axiom schemata
3. A proof will be a sequence of propositions $A_1, \ldots, A_n$, where each A_i is either a substitution instance of an axiom, or inferred by MP (modus ponens) from some A_j and A_k, where $j, k < i$.
4. The proof system must be complete (and, of course, consistent).

What does MP mean? If $\rightarrow$ is in the language, then MP is the rule

$$\frac{A,\ A \rightarrow B}{B}.$$

If $\rightarrow$ is not in the language, let $\varphi(p, q)$ be a (fixed) formula equivalent to $p \rightarrow q$, and use the rule MP$_\varphi$:

$$\frac{A,\ \varphi(A, B)}{B}$$

The *size* of a Frege proof is the sum of the sizes of the formulas, where the size of a formula is the number of symbols in it.

Remark: The sizes of formulas in a Frege proof has no polynomial bound in terms of the size of the formula to be proved. Hence counting only the number of lines in a proof would not give the same notion of "long" proofs.

It does not really matter what particular Frege system we choose:

Theorem: If F_1 and F_2 are Frege systems with the same language, then F_1 simulates F_2.

Proof: Since F_1 is complete, it has for each axiom B of F_2 a proof of B. Any instance $B(p_1/A_1, \ldots, p_n/A_n)$ has an F_1-proof of size $\mathcal{O}(|A_1| + \cdots + |A_n|)$.

Any F_2-proof can be converted into an F_1-proof by adding, for each F_2-axiom instance B, an F_1-proof of it. This makes a linear increase in the size of the proof.

Here is an example of a Frege system:

> The logical connectives used are $\{\neg, \vee, \wedge, \rightarrow\}$. "$\rightarrow$" is associated from right to left, i.e. $\varphi \rightarrow \psi \rightarrow \chi$ is an abbreviation for $\varphi \rightarrow (\psi \rightarrow \chi)$ (which is equivalent to $\varphi \wedge \psi \rightarrow \chi$).
> The axiom schemata are

$$\varphi \rightarrow \psi \rightarrow \varphi$$

$$(\varphi \rightarrow \psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi)$$

$$(\neg\varphi \rightarrow \neg\psi) \rightarrow (\psi \rightarrow \varphi)$$

$$\varphi \rightarrow \psi \rightarrow \varphi \wedge \psi$$

$$\varphi \wedge \psi \rightarrow \varphi$$

$$\varphi \wedge \psi \rightarrow \psi$$

$$\varphi \rightarrow \varphi \vee \psi$$

$$\psi \rightarrow \varphi \vee \psi$$

$$(\varphi \rightarrow \chi) \rightarrow (\psi \rightarrow \chi) \rightarrow (\varphi \vee \psi \rightarrow \chi)$$

A *substitution Frege (s$\mathcal{F}$) proof system* consists of

1. A language $\mathcal{L}$, a finite complete set of propositional connectives.
2. A finite set of axiom schemata.
3. Two rules:
   - Modus Ponens
   - The substitution rule:
   $$\frac{A}{A(p/B)}$$
   (where $A(p/B)$ means: replace every occurrence of the variable p by the formula B).
4. The notion of "proof" is defined as usual.
5. The proof system must be complete (and, of course, consistent).

An *extended Frege (e$\mathcal{F}$) system* consists of

1. A language $\mathcal{L}$, a finite complete set of propositional connectives.
2. A finite set of axiom schemata
3. Two rules of inference:
   - Modus Ponens
   - Extension rule
4. An e$\mathcal{F}$ proof is a sequence $A_1, \dots, A_n$, where each A_i is either an axiom, or inferred from some previous A_j's by MP, or is of the form
   $$A_i = p \leftrightarrow B$$
   (read: "p abbreviates B"), where B is as formula and p is a variable not appearing in $A_1, \dots, A_{i-1}, A_n, B$. (If $\leftrightarrow$ is not in the language, choose a formula $\varphi_{\leftrightarrow}(p,q)$ equivalent to $p \leftrightarrow q$, and let $p \leftrightarrow B$ stand for $\varphi_{\leftrightarrow}(p, B)$.) Notation: Whenever we talk about extension Frege systems, a $\leftrightarrow$ binds stronger then any other propositional connective.

5. The proof system must be complete (and, of course, consistent).

(This is aimed at the way people *do* mathematics, e.g. we introduce the term "real numbers" by definition, and then don't refer to "cuts of rationals" every time.)

Theorem: Any two e$\mathcal{F}$ systems with the same language simulate each other.

Theorem: Any two s$\mathcal{F}$ systems with the same language simulate each other.

The proofs are exactly as for $\mathcal{F}$ systems.

Remark: In the above three theorems, the restriction that the systems have the same language is unnecessary; but the proofs are harder.

Theorem: Given a s$\mathcal{F}$ and an e$\mathcal{F}$ system in the same language, then the s$\mathcal{F}$ system simulates the e$\mathcal{F}$ system.

Before we start the proof, we need this

Lemma: There exists a polynomial p such that:

For all formulas A, for any e$\mathcal{F}$ proof P of A using only the extension rules $p_1 \leftrightarrow A_1, \ldots, p_k \leftrightarrow A_k$, there exists a Frege proof Q of

$$\Big(\bigwedge_{i=1}^{k} p_i \leftrightarrow A_i\Big) \to A,$$

where $|Q| \leq p(|P|)$.
(Remember that the $\leftrightarrow$ in $\bigwedge_{i=1}^{k} p_i \leftrightarrow A_i$ binds stronger than the $\wedge$. By definition, in this proof $\bigwedge_{i=p}^{q}$ associates from right to left, i.e. $\bigwedge_{i=p}^{q} \psi_i = \psi_p \wedge (\bigwedge_{i=p+1}^{q} \psi_i))$

Proof of the lemma: Let $P = B_1, \ldots, B_n$, where $B_n = A$. We define $Q_1, \ldots, Q_n$ (sequences of formulas), which together will give Q.

Case 1: B_j is an axiom. Let

$$Q_j = B_j \;+\; \text{a derivation of } B_j \rightarrow \Big(\bigwedge_{i=1}^{k} p_i \leftrightarrow A_i\Big) \rightarrow B_j$$
$$+\;\Big(\bigwedge_{i=1}^{k} p_i \leftrightarrow A_i\Big) \rightarrow B_j \text{ (inferred by MP).}$$

The formula $B_j \rightarrow (\bigwedge_{i=1}^{k} p_i \leftrightarrow A_i) \rightarrow B_j$ is an instance of the tautology $\varphi \rightarrow \psi \rightarrow \varphi$, which can be proved with a constant number of inferences (it is an axiom in the system given above). Hence $|Q_j| = \mathcal{O}(|P|)$.

Case 2: B_i is inferred by MP from B_s and $B_t = B_s \rightarrow B_j$. In this case, let Q_j be a derivation of

$$\left(\Big(\bigwedge_{i=1}^{k} p_i \leftrightarrow A_i\Big) \rightarrow B_s \rightarrow B_j\right) \rightarrow$$
$$\rightarrow \left(\Big(\bigwedge_{i=1}^{k} p_i \leftrightarrow A_i\Big) \rightarrow B_s\right) \rightarrow \left(\Big(\bigwedge_{i=1}^{k} p_i \leftrightarrow A_i\Big) \rightarrow B_j\right).$$

This is an instance of the tautology

$$(\varphi \rightarrow \psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi).$$

Use MP twice to get $(\bigwedge_{i=1}^{k} p_i \leftrightarrow A_i) \rightarrow B_j$. Again we have $|Q_j| = \mathcal{O}(|P|)$.

Case 3: B_j is inferred by extension, $B_j = p_m \leftrightarrow A_m$. Q_j will be the concatenation of

1. a derivation of

$$\Big(\bigwedge_{i=m}^{k} p_i \leftrightarrow A_i\Big) \rightarrow (p_m \leftrightarrow A_m)$$

(Note: This is an instance of $\varphi \wedge \psi \rightarrow \varphi$, and this is derivable with a constant number of inferences.)

2. a derivation of

$$\left(\left(\bigwedge_{i=m}^{k} p_i \leftrightarrow A_i\right) \to (p_m \leftrightarrow A_m)\right) \to$$

$$\to \left(\bigwedge_{i=m-1}^{k} p_i \leftrightarrow A_i\right) \to (p_m \leftrightarrow A_m)$$

(This is an instance of $(\varphi \to \psi) \to (\chi \wedge \varphi \to \psi)$.)

3. By MP,

$$\left(\bigwedge_{i=m-1}^{k} p_i \leftrightarrow A_i\right) \to (p_m \leftrightarrow A_m)$$

4. repeat 2. and 3. until you get

$$\left(\bigwedge_{i=1}^{k} p_i \leftrightarrow A_i\right) \to (p_m \leftrightarrow A_m)$$

In this case, $|Q_j| = \mathcal{O}(|P|^2)$.

The lemma is now proved by concatenating $Q_1, \ldots, Q_n$ to give Q, where $|Q| = \mathcal{O}(|P|^3)$.

Now we can start the proof of the theorem:
Let P be an e$\mathcal{F}$ proof of A using only extension rules $p_1 \leftrightarrow A_1, \ldots, p_k \leftrightarrow A_k$. The s$\mathcal{F}$ proof begins with Q. W.l.o.g. the $p_i \leftrightarrow A_i$'s are numbered in reverse order of how they appear in the proof, so p_i does not appear in A_j, for $j \geq i$. By the above lemma, there is a $\mathcal{F}$ proof Q which ends with

$$\left(\bigwedge_{i=1}^{k} p_i \leftrightarrow A_i\right) \to A,$$

which is the same as

$$(p_1 \leftrightarrow A_1) \wedge \left(\bigwedge_{i=2}^{k} p_i \leftrightarrow A_i\right) \to A.$$

Append a substitution inference to the end of Q to get

$$(A_1 \leftrightarrow A_1) \wedge \left(\bigwedge_{i=2}^{k} p_i \leftrightarrow A_i\right) \to A.$$

Now append a derivation of

$$\left(A_1 \leftrightarrow A_1 \wedge \left(\bigwedge_{i=2}^{k} p_i \leftrightarrow A_i\right) \to A\right) \to \left(\left(\bigwedge_{i=2}^{k} p_i \leftrightarrow A_i\right) \to A\right)$$

(this is an instance of $((\varphi \leftrightarrow \varphi) \wedge \psi \to \chi) \to (\psi \to \chi))$. Use MP to get

$$\left(\bigwedge_{i=2}^{k} p_i \leftrightarrow A_i\right) \to A.$$

Do this k times, until you get A. Hence we have a proof of size $\mathcal{O}(|P|^3)$.

Remark: It is an open problem, whether $\mathcal{F}$ systems simulate e$\mathcal{F}$ (or equivalently, s$\mathcal{F}$ systems). Of course we can transform every e$\mathcal{F}$ proof into an $\mathcal{F}$ proof, by replacing p_i everywhere with A_i, but that can make the proof exponentially large, for example if each p_i contains 2 occurrences of p_{i-1}.

Theorem: Given a s$\mathcal{F}$ and an e$\mathcal{F}$ system in the same language, then the e$\mathcal{F}$ system simulates the s$\mathcal{F}$ system.

Remarks: This was an open problem for some time. The first solution appeared in Dowd, "Model-theoretic aspects of $P \neq NP$", not yet published. The proof we will give is from Krajíček-Pudlak, "Propositional proof systems, the consistency of first order theories and the complexity of computations", 1987, preprint. There is a more general, "high-level" proof than the one presented below.

Proof: Let P be an s$\mathcal{F}$ proof, $P = A_1, \ldots, A_k$, using variables $p_1, \ldots, p_n$. Let $\vec{p} = (p_1, \ldots, p_n)$. Let q_{ij} (for $i = 1, \ldots, k$, $j = 1, \ldots, n$) be distinct variables. Let $\vec{q_i} = (q_{i1}, \ldots, q_{in})$, and assume all the q_{ij} are new, except that $\vec{q_k} = \vec{p}$. Write $A_i = A_i(\vec{p})$, and let $B_i = A_i(\vec{p}/\vec{q_i}) = A_i(\vec{q_i})$. We will construct a e$\mathcal{F}$ proof that proves the B_i's. This suffices since $B_k = A_k$.

Define vectors $\vec{\beta_i} = (\beta_{i1}, \ldots, \beta_{in})$ as follows:

Case 1: If A_i is an axiom or inferred by MP, then $\vec{\beta_i} = \vec{q_i}$.

Case 2: A_i is inferred by substitution from A_r, say $A_i = A_r(p_s/\alpha)$. Then $\beta_{ij} = q_{ij}$ for $j \neq s$, and $\beta_{is} = \alpha(\vec{p}/\vec{q}_i)$.

The e$\mathcal{F}$ proof is as follows:
First we introduce the q_{ij}'s by the extension rule:

$$\begin{aligned} q_{ij} \leftrightarrow (C_i \wedge \neg B_{i+1} \wedge \beta_{i+1,j}) &\vee (C_{i+1} \wedge \neg B_{i+2} \wedge \beta_{i+2,j}) \vee \cdots \\ &\vee \cdots \vee (C_{k-1} \wedge \neg B_k \wedge \beta_{kj}), \end{aligned}$$

where C_i is an abbreviation for $B_1 \wedge \cdots \wedge B_i$. It is easy to see that there is a polynomial-size proof of

$$C_r \wedge \neg B_{r+1} \rightarrow (q_{ij} \leftrightarrow \beta_{r+1,j}) \qquad (*)$$

whenever $r \geq i$.

Secondly: The e$\mathcal{F}$ proof derives $B_1, C_1, \ldots, B_k, C_k$. (Since $A_k = B_k$, this suffices.) Suppose $B_1, C_1, \ldots, C_{r-1}, B_r$ are already derived, $r \geq 0$. From B_r and C_{r-1} it is trivial to derive C_r. Now derive B_{r+1} from C_r according to the following cases:

Case (i) If A_{r+1} is an axiom, then so is B_{r+1}.

Case (ii) (This is the hardest case) If A_{r+1} is inferred from A_u and A_v by MP, where $A_v = A_u \rightarrow A_{r+1}$, then: From $(*)$ and C_r, get (using a constant number of lines)

$$\neg B_{r+1} \rightarrow (q_{uj} \leftrightarrow \beta_{r+1,j})$$

and

$$\neg B_{r+1} \rightarrow (q_{vj} \leftrightarrow \beta_{r+1,j}).$$

From this we get

$$\neg B_{r+1} \rightarrow (B_u \leftrightarrow A_u(\vec{p}/\vec{\beta}_{r+1}))$$

and

$$\neg B_{r+1} \rightarrow (B_v \leftrightarrow A_v(\vec{p}/\vec{\beta}_{r+1}))$$

(by induction on the length of A_u, A_v). These imply

$$\neg B_{r+1} \rightarrow A_u(\vec{p}/\vec{\beta}_{r+1}) \wedge A_v(\vec{p}/\vec{\beta}_{r+1}).$$

By tautological implication (again using a constant number of lines), essentially MP (remember that $A_v = A_u \rightarrow A_{r+1}$) we get

$$\neg B_{r+1} \rightarrow A_{r+1}(\vec{p}/\vec{\beta}_{r+1}),$$

and since $A_{r+1}(\vec{p}/\vec{\beta}_{r+1})$ is B_{r+1}, this is

$$\neg B_{r+1} \rightarrow B_{r+1},$$

from which we can get B_{r+1} with a constant number of lines.

Case (iii) $A_{r+1} = A_u(p_s/\alpha)$. By $(*)$,

$$\begin{array}{lcl} \neg B_{r+1} & \rightarrow & (q_{uj} \leftrightarrow \beta_{r+1,j}) \\ \neg B_{r+1} & \rightarrow & (B_u \leftrightarrow A_u(\vec{p}/\vec{\beta}_{r+1})) \\ \neg B_{r+1} & \rightarrow & A_u(\vec{p}/\vec{\beta}_{r+1}) \end{array}$$

But clearly

$$\begin{array}{lcl} A_u(\vec{p}/\vec{\beta}_{r+1}) & = & A_u(p_1/q_{r+1,1}, \ldots, p_s/\alpha(\vec{p}/\vec{q}_{r+1}), \ldots, p_n/q_{r+1,n}) \\ & = & A_{r+1}(\vec{p}/\vec{q}_{r+1}) = B_{r+1}. \end{array}$$

Again, from this we get $\neg B_{r+1} \rightarrow B_{r+1}$ and consequently, B_{r+1}.

This finishes the construction of the p-simulation of s$\mathcal{F}$ systems by e$\mathcal{F}$ systems.

In the following picture, $x \to y$ means "x simulates (in fact, p-simulates) y".

$$\begin{tikzcd}
 & \text{extended Frege Systems} \arrow[d] \arrow[r, leftrightarrow] & \text{substitution Frege Systems} \\
 & \text{Frege Systems} \arrow[d] \arrow[r, leftrightarrow] & \text{sequent calculus with cuts} \\
 & \text{resolution with limited extension} \arrow[dl] \arrow[dr] & \\
\text{cut-free sequent calculus} & & \text{resolution}
\end{tikzcd}$$

It is known that resolution with limited extension does not simulate Frege systems. The following questions are open: Does resolution simulate cut-free sequent calculus? Or vice versa? Are Frege systems super?

Theorem: (Cook-Reckhow, JSL 1979): The formulas PHP_n have polynomial-sized e$\mathcal{F}$ proofs.

Theorem: (Buss, JSL 1988) PHP_n also have polynomial-sized $\mathcal{F}$ proofs.

(The proofs of these theorems are not included in the scribe notes. See the references for proofs.)

This takes PHP_n off the list of potential candidates for "separating" $\mathcal{F}$ and e$\mathcal{F}$, leaving no "nice" sequence of formulas on it. However, there is a set of formulas φ_n (related to "self-consistency" assertions), such that:

If $\mathcal{F}$ and e$\mathcal{F}$ can be separated, then the φ_n separate them.

Math 271 - Topics in Weak Formal Systems

Lecture Notes, Set #5
1988 March 3-11

Instructor: Sam Buss
Notes By: Chrystopher Nehaniv

1. Circuits and Circuit Complexity

A **circuit** is a finite labelled directed acyclic graph. By "labelled" we mean that each vertex has a label. In particular, we allow disconnected circuits.

We define the **indegree** of a vertex to be the number of edges whose target is that vertex. Similarly, the **outdegree** of a vertex is defined to be the number of edges whose source is that vertex. For example, in the portion of a graph shown below, the vertex has indegree three and outdegree two.

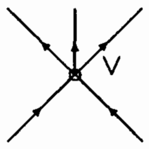
There are three kinds of vertices:

input vertices: These have indegree zero and are labelled with a Boolean input variable x (or a conjugate $\overline{x}$) or with one of the constants from {T,F}. Note that isolated vertices will thus be "unused inputs".

internal vertices: These have indegree and outdegree both strictly greater than zero, and are labelled with elements of $\{\neg, \vee, \wedge\}$ and possibly with other names. Internal vertices are also called **gates**.

output vertices: These have outdegree zero and indegree equal to one. As a matter of convention, an output vertex will be labelled with an output name in the case that there is more than one output vertex.

For example, in the circuit show below, the vertex labelled "$\vee$" has indegree three and outdegree two.



We think of things flowing in the direction of the arrows of a graph. That is, Boolean values travel in the direction of the arrows. At a gate, the function designated on the label of that gate is applied to the arriving Boolean values, and the result is the value that then travels away from the gate in the direction of the outgoing arrows.

Further, we define the **fanin of a gate** to be its indegree. The **fanout of a gate** is defined to be its outdegree. In our circuits, we shall allow that a vertex labelled with "$\wedge$" or "$\vee$" have arbitrary fanin and fanout; but we require that a vertex labelled "$\neg$" have indegree equal to one, although it may have arbitrary fanout. The **fanin of a circuit** is the maximum of the fanin of its gates. The **fanout of a circuit** is defined similarly to be the maximum fanout of its gates.

A circuit with **k** inputs and **n** outputs computes a Boolean function

$$f : \{T, F\}^k \rightarrow \{T, F\}^n$$

and, conversely, every Boolean function can be realized as a circuit (because it can be realized as a formula).

The **size of a circuit** C is $|C|$ = the number of edges in C. (Since we shall be interested only in polynomial size circuits, an alternative definition for size of a circuit would be the number of vertices. Note that the $(\#vertices)^2 \geq \#edges$).

Let S be a predicate, $S \subseteq \{0,1\}^*$, the set of finite words from the alphabet $\{0,1\}$. $\mathcal{C}$ is **a family of circuits recognizing** S iff $\mathcal{C} = \{C_0, C_1, C_2, C_3, \ldots\}$, where each C_i has inputs among $x_1, \overline{x}_1, \ldots, x_i, \overline{x}_i$

and C_i is a circuit which determines whether words of length i are in S. That is, for $w = w_1 \cdots w_i \in \{0,1\}^i$, C_i with inputs

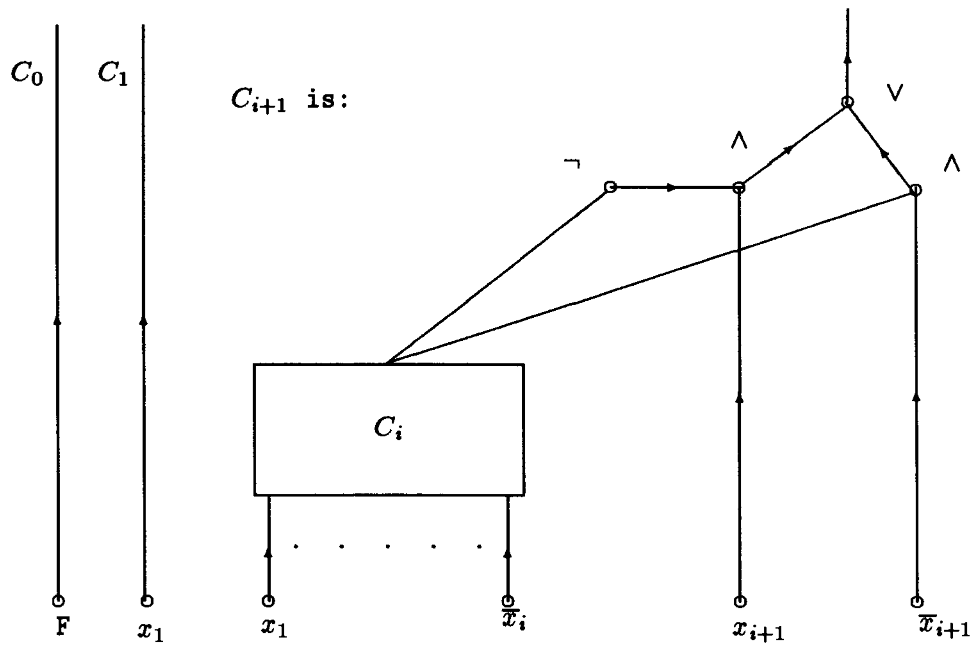
$$x_j = \begin{cases} \text{T} & \text{if } w_j = 1 \\ \text{F} & \text{if } w_j = 0 \end{cases}$$

outputs T iff $w \in S$.

Definition: S **has polynomial size circuits** iff $\exists$ C a family of circuits for S and there exists a polynomial p s.t. $\forall i$ $|C_i| \leq p(i)$.

Example: Parity is the set S of strings from the alphabet $\{0,1\}$ with an odd number of ones:

$$S = \{w \in \Sigma^* : w \text{ has an odd number of 1's}\}$$



Polynomial Size Circuits for Parity

For these circuits: for $i > 0$, we have $|C_{i+1}| = |C_i| + 7$. Hence, we see this set of circuits to be bounded in size by $p(i) = 7i + 1$.

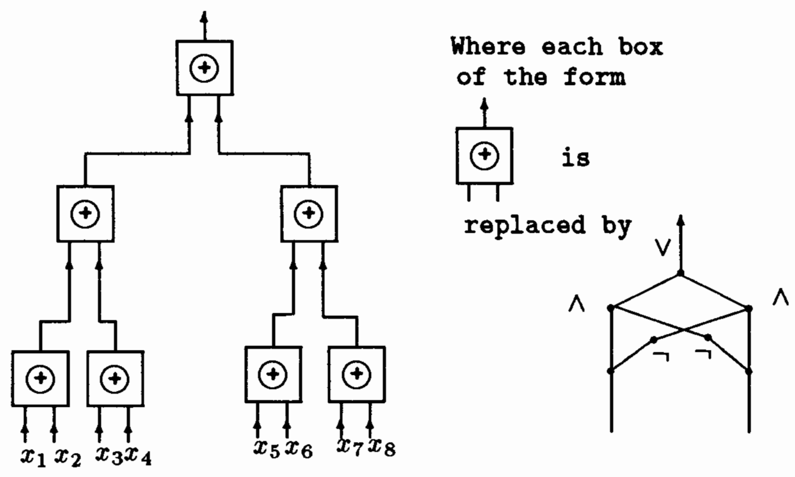
A formula is basically a circuit with fanout 1. More precisely, $\exists$ polynomial p s.t for every formula A, there exists an equivalent circuit C_A with fanout 1 and with $|C_A| \leq p(|\mathrm{A}|)$. Conversely, for every circuit C with fanout 1 and with one output there exists an equivalent formula A_C with $\mathrm{A}_C \leq p(|C|)$. Obviously p can be taken to be a linear function.

There is a natural correspondence between formulas and circuits with fanout 1 with one output. The obvious translation works. (NB: A circuit with fanout 1 could have arbitrary fanout at an input node, because fanout of a circuit is defined over *gates*).

The **depth** of a circuit is the maximal length over all paths through the circuit from an input to an output.

If we try duplicating subcircuits to make the above circuits for Parity have fanout 1, then we get into exponential size formulas. This is because we must have *two copies* of C_i to construct C_{i+1}.

However, there do exist polynomial circuits for parity with fanout 1. The following is a circuit for input of length 8. Use this type of circuit for input of any length n padding with F inputs to get $2^{\lceil \log n \rceil}$ inputs.



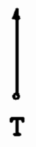
Where each box of the form
is
replaced by
$\vee$
$\wedge$
$\wedge$
$\neg$
$\neg$
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

(This has fanout 2, but it can be converted to a circuit of fanout 1 with only a polynomial increase in size. The crucial point is that the circuit has logdepth and bounded fanin - every gate has fanin 1 or 2.)

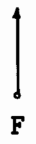
One might expect that having polynomial circuits corresponds to ease of computation; but consider that following:

Example: Polynomial Circuits for a Non-Recursive Set

Pick your favorite non-recursive set A$\subseteq \mathbb{N}$. Let $S = \{w : |w| \in A\}$. Clearly S is non-recursive because A is. S has circuits of size 1. Namely, let C_i be



`if` $i \in A$



`if` $i \notin A$

Thus we see that circuit complexity is a *non-uniform* notion of complexity, because we have separate circuits for each possible length of input.

2. Turing Machines and Circuits

To code circuits for input to a Turing Machine or another circuit, we shall code the circuit as a string of zeroes and ones, using two lists: $\{(vertex\#j,$ label on that vertex$)\}$ and $\{(from\_vertex, to\_vertex)\}$. This is done in a finite alphabet $\{0, 1, \wedge, \vee, \neg, x,$"$,$",(,)$\}$; and we can represent each letter of this alphabet as a string of four zeroes and ones.

The size of the coding of a circuit and the size of the circuit are polynomially related.

As before, **P** denotes polynomial time computable predicates and **FP** denotes polynomial time computable functions.

We defined our Turing Machines to be multitape TM's. Actually for polynomial time, we get the same results by using single-tape TM's. Indeed, if $f(x)$ is a function computable in time $t(x)$ on a multitape

Turing Machine, then $f(x)$ is computable in time $\mathcal{O}(t(x)^2)$ on a single-tape TM.

A **logspace TM** has a read-only input tape and multiple (or a single) work tapes, and there $\exists c$ constant, s.t. on input of length n, at most $c \log n + c$ work tape cells are used.

A *predicate* is **in Logspace** iff it is recognized by a logspace TM. Similarly, a *function* is in Logspace iff it is computed by a logspace TM.

Theorem 1 Logspace is a subclass of **P** (or **FP**, respectively).

pf: There are only $c^{\mathcal{O}(\log n)}$ instantaneous descriptions of a logspace machine on a given fixed input x of length n. In this context, an instantaneous description includes the TM's work tape contents, state, and input head position at some time i - but does not include input or output tape contents. So the TM either halts in polynomial time $n^{\mathcal{O}(c)}$ or goes into a loop (and never halts).$\Box$(Theorem 1)

Definition: If $A, B \subseteq \Sigma^*$ then a **many-one reduction of A to B** is a mapping $f : \Sigma^* \to \Sigma^*$ s.t $\forall x \in A \Leftrightarrow f(x) \in B$. So computing $f(x)$ allows the question $x \in A$? to be reduced to the question $f(x) \in B$?

This type of reduction is more restrictive than a Turing reduction, which allows us to use B as an oracle asked many times.

Definition: A decision problem A is **P-complete with respect to logspace many-one reductions** iff $A \in \mathbf{P}$ and $\forall B \in \mathbf{P}, \exists$ a many-one reduction of A to B computable in Logspace.

So if you know how to solve A, then with logspace reductions, you can solve *every* B in **P**.

Important: The many-one reduction here is from a *lower level of complexity* (namely, Logspace). Fact: Any $A \neq \emptyset, A \neq \Sigma^*, A \in \mathbf{P}$ is **P**-complete with respect to many-one *polynomial* time reductions.

Definition: The Circuit Value Problem (CVP) is the question, Given a fanin 2 circuit with no input variables (just fixed T,F's as inputs) is its output T? (Strictly speaking CVP is the set of codes of such circuits with output T).

Theorem 2 CVP is **P**-complete with respect to logspace reductions.

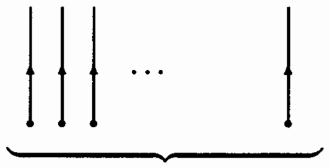
Proof: CVP$\in$**P**. This is easy. Think of the natural algorithm to evaluate the output of a circuit. Just cycle through assigning values to gates.

Given $B \in \mathbf{P}$, we want a logspace many-one reduction f_B of B to CVP. So given a the Turing Machine $\mathcal{M}$ accepting B in time $p(n)$ for p some polynomial and $n = |x|$, we want f_B s.t.

$$f_B(x) \in \text{CVP} \iff \mathcal{M} \text{ accepts } x$$

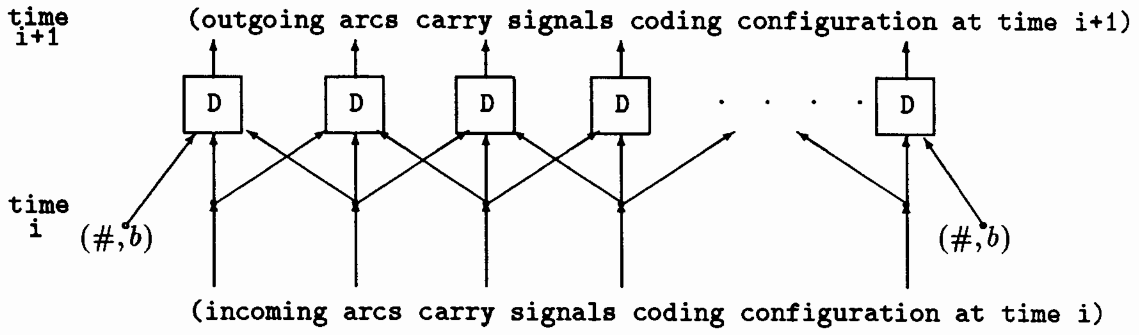
Given an input x to $\mathcal{M}$, f_B creates a circuit. The circuit will emulate the computation of $\mathcal{M}$ on input x. Without loss of generality, $\mathcal{M}$ is a single tape Turing Machine. A tape cell of $\mathcal{M}$ will be represented by a pair (q, γ), where γ is the symbol in the cell and q is either a state if the tapehead is at that cell or is some other symbol $\#$, to indicate that the tapehead is not at that cell.

There is a constant upper bound on how many (q, γ)'s are possible. So (q, γ)'s can be represented by k binary signals. Represent the configuration of $\mathcal{M}$ at time i by $k \cdot p(n)$ signals:



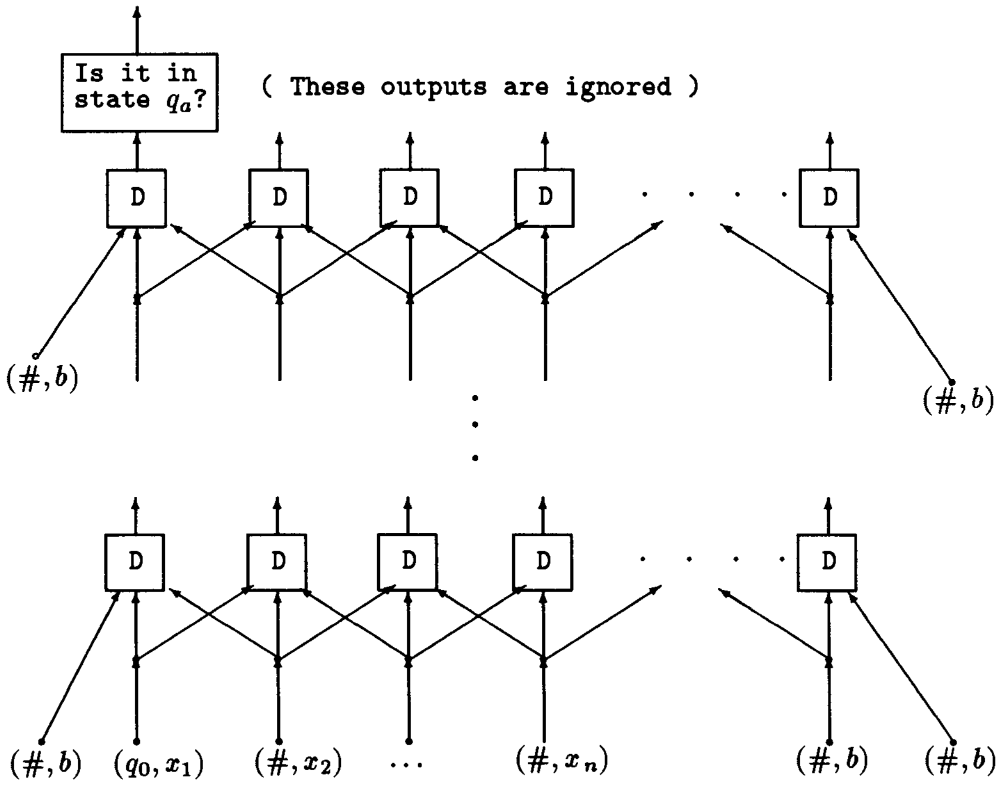
$p(n)$ arcs each carrying k signals coding a single (q, γ)

The circuit will be constructed to put the right values on these signals.


time i+1
(outgoing arcs carry signals coding configuration at time i+1)
D
D
D
D
D
time i
(#,b)
(#,b)
(incoming arcs carry signals coding configuration at time i)

Each of the $p(n)^2$ circuits labelled D takes 3k inputs coding the prior state of the tape cell and the prior states of its two neighboring tape cells. D is a fixed circuit depending on $\mathcal{M}$, such a D exists because circuits are complete.

The whole circuit $f_B(x)$ is as follows:



initial configuration of $\mathcal{M}$ on input $x_1, x_2, \ldots, x_n$

Some state q_a is a halting accepting state. W.l.o.g. $\mathcal{M}$ always halts at its starting position and never moves left of its starting position.

By our constraints, this circuit has output T iff $\mathcal{M}(x)$ accepts.

Furthermore, the circuit $f_B(x)$ can be computed in Logspace: D is just a fixed circuit. Run through all times $i = 0, \ldots, p(n) - 1$ and take positions $j = 0, \dots, p(n) - 1$, outputing the corresponding piece of the above circuit. Basically, we need to save i and j in the workspace, resulting in a use of $\mathcal{O}(\log(p(n)))$ work cells, i.e. $\mathcal{O}(\log n)$ work cells. □(Theorem 2)

Corollary 1 Any predicate B in **P** has polynomial size circuits.

Proof: (Immediate from the previous proof) $f_B(x)$ produces a circuit for determining if x is in B. Recall $f_B \in$ **FP**. The circuit $f_B(x)$ depended only on the length of x, although its inputs depended on x. This $f_B(x)$ gives a set of polysize circuits for B.□(Corollary 1)

Corollary 2 CVP $\in$ **Logspace** iff **P=Logspace**. [This is the whole point of "**P**-completeness". CVP is a hardest problem in **P**].

Lemma: If f and g are logspace computable functions then so is $g \circ f$.

Proof: The *problem* here is that we can't just compute $f(x)$ and then $g(f(x))$ because there's not enough room to write out $f(x)$ on a logspace work tape. The *solution* to this dilemma is to repeatedly re-compute $f(x)$ as we need it. *Details:* f and g are computable by $\mathcal{M}_f$ and $\mathcal{M}_g$ which are logspace machines. These have k_f and k_g work tapes respectively. Define $\mathcal{M}_{gof}$ to be a machine having $k_f + k_g + 2$ work tapes.

k_f of these tapes will be used in simulating $\mathcal{M}_f$. One tape will hold a counter giving location of $\mathcal{M}_f$'s output head. k_g tapes are used to simulate the actions of $\mathcal{M}_g$'s work tape contents. The last tape holds the location of $\mathcal{M}_g$'s input tape head.

$\mathcal{M}_{gof}$ acts as follows on input x: Repeatedly simulate a single step of $\mathcal{M}_g$ on input $f(x)$ by doing the following: Run $\mathcal{M}_f$ on input x, but instead of writing $f(x)$ on the output we just keep track of the output tapehead location. Run $\mathcal{M}_f$ on input x until it halts. Note the final symbol which would have been written to the location that $\mathcal{M}_g$'s input tapehead is at. That gives the symbol $\mathcal{M}_g$ should be reading on its input tape. So we can now simulate a single step of $\mathcal{M}_g$. When $\mathcal{M}_g$ would write on its output tape, $\mathcal{M}_{gof}$ does write on the output tape.

Claim: $\mathcal{M}_{gof}$ is a logspace TM.

k_f tapes use only logspace (since $\mathcal{M}_f$ does). $\mathcal{M}_g$ and $\mathcal{M}_f$ only run for polynomial time so I/O tapeheads only move polynomially many cells. So logspace suffices to specify their location. Finally $|f(x)|$ is less than or equal to $p(|x|)$, p a polynomial. $\mathcal{M}_g(f(x))$ uses $\mathcal{O}(\log p(|x|)$ space, which is $\mathcal{O}(\log(|x|))$ space. $\Box$(Lemma)

Proof of Corollary 2

$\Longleftarrow$: Trivial

$\Longrightarrow$: Given A $\in$ **P**, $\exists$logspace many-one reduction f from A to CVP. Thus $\mathcal{X}_A$ is $\mathcal{X}_{CVP} \circ f$ and is in Logspace. (Notation: $\mathcal{X}_Z$ is the characteristic function of Z.) $\Box$(Corollary 2)

3. Extended Frege Systems as Logics on Circuits

Recall that Extended Frege Systems allowed us to introduce abbreviations. In effect, this allows "circuits" to be handled. In fact, $e\mathcal{F}$ could have been defined as a logic on circuits just as Frege Systems are a logic on formulas. To see this: What can be expressed in a formula in an $e\mathcal{F}$-proof (as a function of the Boolean variables in the formula being proved)? Polysize $e\mathcal{F}$-proofs can express what can be defined by polysize circuits. Any formula in the $e\mathcal{F}$-proof can be expressed directly as a circuit. The symbols introduced by abbreviation (i.e. by the extension rule) correspond to gates with fanout > 1 being allowed.

Conversely, a circuit value can be defined by a polynomial size formula using symbols defined with extension: just introduce a new variable for each internal node of the circuit.

$e\mathcal{F}$ is essentially a logic on circuits. Since CVP is **P**-complete, we might expect a relationship between $e\mathcal{F}$-proof systems and polynomial size computations. There is indeed such a relationship [due to Cook].

4. Alternating Turing Machines

An **alternating Turning machine** $\mathcal{M}$ is defined to be a multitape machine with

- k tapes, of which k_1 are readable and k_2 writable.
- finite alphabet Γ
- finite set of states Q

- transition "function" $S : \Sigma^{k_1} \times Q \rightarrow \Sigma^{k_2} \times Q \times \{+1, 0, -1\}^k$ (where S is partial and possibly multivalued, i.e. a relation)

- states are designated as (exactly one of) universal, existential, deterministic, accepting and rejecting.

- S is *single-valued* on deterministic states.

Without loss of generality, we assume S is two-valued in every configuration involving a universal or existential state. When S has no value, the machine halts; we shall require that this happens exactly when the machine is in an accepting or a rejecting state.

A **nondeterministic TM** is an alternating TM with no universal states. A **co-nondeterministic TM** is one with no existential states.

Then **execution tree** of $\mathcal{M}$ is a (possibly infinite) tree of degree ≤ 2 at all nodes. Each node node is labelled with an **instantaneous description(ID)**, i.e. $\mathcal{M}$'s tapes' contents, tape head positions, and internal state. The root of the tree is labelled with the initial configuration of $\mathcal{M}$ on input x. If a node is labelled with configuration Φ, then there is exactly one child for each of successor configuration of Φ and there are no other children.

Inductive Definition: The configuration Φ of $\mathcal{M}$ **leads to acceptance**(resp. **rejection**) iff:

(a) if Φ is a halting configuration, then Φ leads to acceptance (resp. rejection) if Φ is an accepting (resp. rejecting) state.

(b) if Φ is deterministic and Ψ follows Φ in one step, then Φ leads to acceptance (resp. rejection) if Ψ does.

(c) if Φ is an existential state, then Φ leads to acceptance (resp. rejection) iff at least one (resp. all) of its successors does.

(d) if Φ is a universal state, then Φ leads to acceptance (resp. rejection) iff all (resp. at least one) of its successors does.

Math 271 - Topics in Weak Formal Systems

Lecture Notes, Set #6 **Instructor: Sam Buss**
March 14–18, 1988 **Notes By: Eric Hughes**

Run Times

Reference: Chandra–Kozen–Stockmeyer "Alternation" JACM 1981.

Definition: Run Time of an Alternating Turing Machine (the standard one) An alternating Turing Machine accepts in time t on input x iff when the execution tree is truncated at depth $t+1$, the truncated tree also accepts x (according to the above definition of acceptance).

Note: A leaf of a truncated tree need not be either accepting or rejecting. This may occur in an accepting truncated execution tree if the leaf is a descendant of an existential node.

Definition: M accepts input x if the initial configuration of M with input x leads to acceptance.

Definition: An alternating Turing machine (ATM) runs in time $t(n)$ iff for all x which M accepts, M accepts x within time $t(|x|)$.

Definition: An ATM M accepts x in space s iff the following subtree of the execution tree of M also accepts x. For each node of the tree whose ID has space $> s$, define that node to be not accepting and discard all of its children and their respective subtrees.

Definition: An ATM M runs in space $s(n)$ iff for all x which M accepts, M accepts x in space $s(|x|)$.

Definition: $t(n)$ is time constructible iff there exists a deterministic TM which on an input of length n runs for exactly $t(n)$ steps.

Definition: $s(n)$ is space constructible iff there exists a deterministic TM which on an input of length n visits exactly $s(n)$ work tape cells.

Generally we have $t(n) \geq n$ and $s(n) \geq \log n$. For $t(n)$ time constructible and $s(n)$ space constructible, the above definitions of time bounds and space bounds can be equivalently stated as follows.

Definition: (alternate) An ATM M runs in time $t(n)$ iff we can add a clock to M and require that for every n and for every branch of the execution tree, M halts before time $t(n)$ on the internal clock.

Definition: (alternate) An ATM M runs in space $s(n)$ iff we can add end-markers to each tape of M at distance $s(n)$ from each starting position and require that for every n and on every branch of the execution tree, the tape heads of M do not pass over the given markers.

Fact: Polynomials are both space and time constructible.

Let us recall the definition of **NP**.

Definition: NP Let Q be a decision problem, i.e. $Q \subseteq \Sigma^*$. Then $Q \in \mathbf{NP}$ iff $(\exists R \in \mathbf{P})(\exists \text{ polynomial } p(-))\forall x\,[x \in Q \leftrightarrow (\exists w, |w| \leq p(|x|))R(x,w)]$.

Definition: NP (alternate) Let Q be a decision problem. Then $Q \in \mathbf{NP}$ iff there is a polynomial time non-deterministic (i.e. no universal states) TM which recognizes Q.

Definition: Polynomial Time Hierarchy (original) Let Q be a decision problem. Then $Q \in \Sigma_k^{\mathbf{P}}$ iff $(\exists \text{ polynomials } p_1, \ldots, p_k)(\exists R \in \mathbf{P})\forall x\,[x \in Q \leftrightarrow (\exists y_1, |y_1| \leq p_1(|x|))(\forall y_2, |y_2| \leq p_2(|x|)) \ldots (Qy_k, |y_k| \leq p_k(|x|))R(x, \vec{y})]$. Since the pairing function is in **P**, we could also allow blocks of similar quantifiers in place of single quantifiers.

Definition: Let Q be as above. $Q \in \Pi_k^{\mathbf{P}}$ iff $\Sigma^* \setminus Q \in \Sigma_k^{\mathbf{P}}$.

Open Question: $\Sigma_k^{\mathbf{P}} \overset{?}{=} \Pi_k^{\mathbf{P}}$.

Proposition: If $(\exists k_0 \geq 1)\, \Sigma_{ko}^{\mathbf{P}} = \Pi_{ko}^{\mathbf{P}}$, then $(\forall s > k_0)\, \Sigma_{ko}^{\mathbf{P}} = \Sigma_s^{\mathbf{P}} = \Pi_s^{\mathbf{P}}$.
Proof: Left to reader.

References: Stockmeyer, TCS vol 3 (1977). Wrathall, TCS 3.

Definition: Σ_k**- and** Π_k**-Alternating Turing Machine** M is a Σ_k ATM if on all inputs x, on each branch of M, M makes some existential moves (where some can be zero), some universal moves, and so on, with at most $k-1$ alternations between existential and universal moves. Π_k ATM's are defined similarly.

Definition: Polynomial Time Hierarchy (alternate 1) Let Q be as above. Then $Q \in \Sigma_k^{\mathbf{P}}$ iff Q is accepted by a polynomial time Σ_k ATM. Also, $Q \in \Pi_k^{\mathbf{P}}$ iff Q is accepted by a polynomial time Π_k ATM.

Theorem: The original and first alternate definitions of $\Sigma_k^{\mathbf{P}}$ and $\Pi_k^{\mathbf{P}}$ are equivalent.

Proof:

A. $\Sigma_k^{\mathbf{P}} \subseteq$ Alt-1-$\Sigma_k^{\mathbf{P}}$

Let Q be defined by $(\exists y_1, |y_1| \leq p_1(|x|)) \ldots (Qy_k, |y_k| \leq p_k(|x|)) R(x, \vec{y})$. Then Q is accepted by a Σ_k ATM which first existentially guesses y_1, the universally chooses y_2, ...guesses/chooses y_k, and finally checks whether $R(x, \vec{y})$ holds and either accepts or rejects accordingly. The total runtime is $\sum_{i=1}^k p_i(|x|)$ plus the runtime of R.

B. Alt-1-$\Sigma_k^{\mathbf{P}} \subseteq \Sigma_k^{\mathbf{P}}$

Suppose M runs in time $p(n)$ and is a Σ_k machine. Then M accepts x iff $(\exists w_1, |w_1| \leq p(|x|))(\forall w_2, |w_2| \leq p(|x|)) \ldots (Qw_k, |w_k| \leq p(|x|)) A(x, \vec{w})$, where $A(x, \vec{w})$ says that when each w encodes quantified (i.e. $\exists$ or $\forall$)

choices, w_1 encodes the first block of $\exists$ moves of M(x), w_2 encodes the second block of $\forall$ moves, ..., w_k encodes the last block of $\exists$ or $\forall$ moves, then M accepts x on that branch of the execution tree of M(x). Here each w_i is a string of L's and R's which identify a branch of the execution tree. Finally, A is computed in polynomial time by simulating M(x) according to the w_i's.

Definition: Oracle Turing Machine An Oracle Turing Machine is a set of strings from Σ^*, called the oracle, and a deterministic TM with the following features added to it. It has three special states: query, query accepting, and query rejecting, called q, q_y, and q_n respectively. It has a query tape for writing out input to the oracle. We make requirements on the transition function. If the OTM is in the state q at time t, then at time $t+1$ it is in either the state q_y or in the state q_n, corresponding to whether or not the string on the oracle tape is in the oracle. If, at time t, the OTM is in the state q, then none of the tape heads change position from time t to time $t+1$.

Definition: Let X be a class of oracles. Then P^X is the class of predicates which are recognized by a polynomial time TM using some oracle from X.

Notation: If Ω is an oracle, then $P^\Omega = P^{\{\Omega\}}$.

Example: $P^{\mathbf{NP}} = P^{SAT}$, where SAT $= \{\phi : \phi$ is a satisfiable propositional formula $\}$, because SAT is **NP**-complete.

Definition: Polynomial Hierarchy (2nd alternate)

$$\begin{aligned}
\Delta_1^{\mathbf{P}} &\overset{\text{def}}{=} \mathbf{P} \\
\Sigma_1^{\mathbf{P}} &\overset{\text{def}}{=} \mathbf{NP} \\
\Delta_{k+1}^{\mathbf{P}} &\overset{\text{def}}{=} P^{\Sigma_k^{\mathbf{P}}} = P^{\Pi_k^{\mathbf{P}}} \\
\Sigma_{k+1}^{\mathbf{P}} &\overset{\text{def}}{=} \{\text{predicates Q } : (\exists \text{ polynomial } p)(\exists R \in \Delta_{k+1}^{\mathbf{P}}) \\
&\qquad (\forall x)[x \in \mathrm{Q} \leftrightarrow (\exists w, |w| \le p(|x|))R(x,w)] \\
\Pi_{k+1}^{\mathbf{P}} &\overset{\text{def}}{=} \text{co-}\Sigma_{k+1}^{\mathbf{P}} = \{\mathrm{Q} : \Sigma^* \setminus \mathrm{Q} \in \Sigma_{k+1}^{\mathbf{P}}\}
\end{aligned}$$

We now have three definitions of the polynomial hierarchy. The first definition was in terms of polynomial time, Σ_k-ATM's. The "first alternate" definition characterized $\Sigma_k^{\mathbf{P}}$ as the class of predicates definable with k polynomially bounded quantifiers in front of a polynomial time predicate. The "second alternate" definition intersperses polynomial computations with alternating quantifiers. We have already proved that the first two definitions are equivalent and we prove below that the second alternative definition is also equivalent.

Example: Given a propositional formula $\phi(p_1, \ldots, p_k)$, order the satisfying truth assignments lexicographically. Question: In the least satisfying assignment σ for this ϕ, what is $\sigma(p_k)$?

Claim: The set of ϕ such that $\sigma(p_k) = \mathrm{T}$ in the least satisfying assignment is a predicate in P^{SAT}. Indeed, the least satisfying assignment σ can be found by a polynomial time OTM with the oracle SAT.

Proof: To find this assignment, do a binary search on the truth table. First evaluate the predicate $(\exists\sigma)[\sigma(p_1) = F \wedge \sigma(\phi) = T]$. If so, set $v_{p_1} = \mathrm{F}$, otherwise set it to T. Next evaluate $(\exists\sigma)[\sigma(p_2) = F \wedge \sigma(\phi(p_1|v_{p_1})) = T]$. Set v_{p_2} accordingly, and so on for the rest of the variables.

This technique works for many problems.

Example: Travelling Salesman Problem Given a graph with edges labelled with distances and a given total distance. Question: Does there exist a way to traverse the graph with total distance travelled less than the given distance? This problem is **NP**-complete. Finding a journey with minimum possible distance is a problem in FP^{SAT}.

It is important to have an efficient Godel numbering for sequences. For the purposes of this discussion, let the alphabet $\Sigma = \{0, 1\}$. Supose each $w_i \in \Sigma^*$ and $\vec{w} = w_1, \ldots, w_k$. We shall define the Godel number of $\vec{w}$, denoted $\langle w_1, \ldots, w_k \rangle$, and this value will be $\in \Sigma^*$.

Construction: First write out the string "$w_1, w_2, \ldots, w_k,$", which is in the set $\{0, 1, \text{comma}\}^*$. Note the extra comma at the end of the string. Then apply the following map:

$$\begin{aligned} 0 &\mapsto 10 \\ 1 &\mapsto 11 \\ \text{comma} &\mapsto 01 \end{aligned}$$

This yields $\langle w_1, \ldots, w_k \rangle \in \Sigma^*$. Note that nothing maps to 00; this precludes problems with leading zeroes.

Example:

$$\begin{aligned} \langle\rangle &= \text{the empty word } = \emptyset. \\ \langle\emptyset\rangle &= \text{``,''} = 01. \\ \langle 0,1,11,101\rangle &= 1001110111110111101101. \end{aligned}$$

Note: $|\vec{w}| = 2\Sigma_{i=1}^{k}(1 + |w_i|)$. Thus $|\vec{w}| \leq 2($ number of elements in $\vec{w})(1 + \max\{|w_i|\})$.

Notation: $\vec{w} = w_1, \ldots, w_k$
$\text{length}(\vec{w}) = k$
$\beta(i, \vec{w}) = w_i$
$\langle\vec{w}\rangle * w_{k+1} = \langle w_1, \ldots, w_k, w_{k+1}\rangle$

Definition: (Polynomially bounded quantifiers) Here p is a polynomial.

$$(\exists y, |y| \leq p(|x|)), (\forall y, |y| \leq p(|x|))$$

Definition: (Logarithmically bounded quantifiers) Again, p is a polynomial.

$$(\exists i \leq p(|x|)), (\forall i \leq p(|x|))$$

Lemma: Quantifier Exchange Property Let $A(v, w, x)$ be a predicate. Let p, q be polynomials. Then the formula

$$(\forall i \leq p(|x|))\,(\exists j, |j| \leq q(|x|))A(i, j, x)$$

is equivalent to the following formula. Note that j^* is the Godel number of a sequence.

$$(\exists j^*, |j^*| \leq 2(q(|x|)+1)(p(|x|)+1))$$
$$(\forall i \leq p(|x|))(A(i, \beta(i+1, j^*), x) \wedge |\beta(i+1, j^*)| \leq p(|x|))$$

Proof: $\Longleftarrow$ Easy.

$\Longrightarrow$ There are only polynomially many values of i for which a corresponding j exists. We concatenate these values into a sequence j^*. The size bound on the Godel-number of sequences gives the polynomial bound on the length of j^*.

Corollary: Each of Σ_k^{P}, Π_k^{P}, and Δ_k^{P} are closed under logarithmically bounded quantification.

Proof: To show that Σ_k and Π_k are closed under logarithmically bounded quantification, we use induction on k and at each step use the quantifier exchange property to push the outer, logarithmically bounded quantifier one level inward and use the fact that Π_{k-1} and Σ_{k-1}, respectively, are closed under this operation. For the base case, we note that if $A(v, w, x)$ is polynomial time (optionally, relative to an oracle Ω), then so is the following predicate which appears in the statement of the above lemma, namely

$$(\forall i \leq p(|x|))(A(i, \beta(i+1, j^*), x) \wedge |\beta(i+1, j^*)| \leq p(|x|)).$$

This is because there are only polynomially many values of $i \leq p(x)$.

Theorem: The original definitions and the second alternate definitions of Σ_k^{P} and Π_k^{P} are equivalent.

Proof: Let 2nd-alt-Σ_k^{P} be the class given in the second alternate definition. By induction on k, it is clear that since $\Pi_{k-1}^{\mathrm{P}} \subseteq \Delta_k^{\mathrm{P}}$, then $\Sigma_k^{\mathrm{P}} \subseteq$2nd-alt-$\Sigma_k^{\mathrm{P}}$.

For the converse, let $\mathrm{Q} \in$ 2nd-alt-Σ_k^{P}. Then there is a predicate $R \in \Delta_k^{\mathrm{P}}$ and a polynomial p such that

$$\mathrm{Q} = \{x : (\exists w, |w| \leq p(|x|))\, R(x, w)\}$$

To show that $Q \in \Sigma_k^{\mathbf{P}}$, we need to show the existence of a predicate $R' \in \Pi_{k-1}^{\mathbf{P}}$ and a polynomial q such that

$$Q = \{x : (\exists w, |w| \leq q(|x|))\, R'(x, w)\}$$

We proceed by induction on k.

Case $k = 1$. $\Delta_1^{\mathbf{P}} = \mathbf{P}$ and $\Sigma_1^{\mathbf{P}} = \mathbf{NP}$ are the same for both definitions.

Case $k > 1$. By the definition of R, there is a polynomial time OTM M which runs in time $r(n)$ on all inputs of length n and uses an oracle $\Omega \in \Pi_{k-1}^{\mathbf{P}}$ and M accepts $\langle x, w\rangle$ iff $R(x, w)$ is true.

$$x \in Q \leftrightarrow (\exists w, |w| \leq p(|x|))\text{“M}(x, w) \text{ accepts”}$$

$$x \in Q \leftrightarrow$$
$$(\exists w, |w| \leq p(|x|))(\exists \vec{v} = v_1, \ldots, v_{r(|x|)})\text{“The } v_i\text{'s code an accepting computation of M}(x, w)\text{”}$$

At this point we note that each v_i uses $\leq r(|x|)$ space on each tape. We also know that the “v_i's code ...” iff v_1 is initial, $v_{r(|x|)}$ is final, and each $v_i + 1$ follows from v_i by one step of the OTM. For notational convenience, let QTC(v_i) be the query tape contents in the configuration coded by v_i;

note that $\mathrm{QTC}(v_i) \in \Sigma^*$.

```
x ∈ Q ↔
    $(\exists w, |w| \le p(|x|))$
    $(\exists v = \langle v_1, \ldots, v_{r(|x|)} \rangle, |v| \le \mathcal{O}(2(r(|x|)+1)^2))$
    $(\forall i \le r(|x|))$
        $(i = 1 \rightarrow$ "$v_i$ codes the initial configuration of M on
                input $\langle x, w \rangle$" )
        $\wedge(i = r(|x|) \rightarrow$ "$v_{r(|x|)}$ is an accepting configuration" )
        $\wedge(1 \le i < r(|x|) \rightarrow ($
            ( "$v_i$ is not in the query state" $\rightarrow$ "$v_{i+1}$ follows
            from $v_i$ by one deterministic step" )
            $\wedge$ (("$v_i$ is in the query state" $\wedge$ $\mathrm{QTC}(v_i) \in \Omega) \rightarrow$
                "$v_{i+1}$ is the same as $v_i$ except that $v_{i+1}$ is in state
                $q_y$" )
            $\wedge$ (( "$v_i$ is in the query state" $\wedge$ $\mathrm{QTC}(v_i) \notin \Omega) \rightarrow$
                "$v_{i+1}$ is the same as $v_i$ except that $v_{i+1}$ is in state
                $q_n$" )
        ) )
```

Everything in quotes is polynomial time, as well as is QTC. In fact, the only thing that is not is Ω. But $\Omega \in$ 2nd-alt-$\Pi^{\mathbf{P}}_{k-1}$, which is equal to $\Pi^{\mathbf{P}}_{k-1}$ by the induction hypothesis. So $\mathrm{QTC}(v_i) \in \Omega$ can be expressed as k polynomially bounded quantifiers, which begins with a universal quantifier, in front of a **P** predicate.

Next we use prenex operations to pull out quantifiers alternately from the predicates $\mathrm{QTC}(v_i) \in \Omega$ and $\mathrm{QTC}(v_i) \notin \Omega$. Represent these predicates by formulas O_y and O_n which are in $\Sigma^{\mathbf{P}}_{k-1}$ and $\Pi^{\mathbf{P}}_{k-1}$ respectively. We first pull out an existential quantifier from O_y, then a universal quantifier from O_n, then a universal quantifier from O_y, then an existential quantifier from O_n, an so on until we have an expression of the following form:

$$\begin{aligned} &x \in Q \leftrightarrow \\ &\quad (\exists w, |w| \le p(|x|))(\exists v, |v| \le r^*(|x|))(\forall i \le r(|x|)) \\ &\quad (\exists\forall\,\forall\exists \ldots Q\bar{Q})\text{"something polynomial time"} \end{aligned}$$

We then collapse the adjacent quantifiers using pairing and the β function. By k applications of the quantifier exchange property, we move the quantifier $(\forall i \le r(|x|))$ to the end of the expression, where it is absorbed into the **P** predicate. We now collapse the first three quantifiers (two originally and one pulled from O_y) to express Q as follows:

$$(\exists y_1, |y_1| \leq p_1(|x|)) \ldots (Q y_k, |y_k| \leq p_k(|x|)) \mathrm{Q}^*$$

where $\mathrm{Q}^* \in \mathbf{P}$. Thus Q is in $\Sigma_k^{\mathbf{P}}$.

First Order Theories of Number Theory and Fragments of Peano Arithmetic.

We will no longer work with strings from an alphabet, but will work with integers, i.e. we will move from Σ^* to $\mathbb{N}$. An integer $n \in \mathbb{N}$ can be represented as a string from $0, 1^*$ by binary representation and some fudging over leading zeroes.

Language: Logical: $\wedge \vee \neg \rightarrow \forall \exists =$
Non-logical: 0 S $+ \cdot \leq |x| \; \lfloor \frac{1}{2}x \rfloor \; \#$

$|x|$ will represent $\lceil \log_2(x+1) \rceil$, which is the length of the binary representation of x. We define $|0| = 0$. $|x|$ and $\lfloor \frac{1}{2}x \rfloor$ are not crucial to the theories, but it makes the axioms easier to state.

$x \# y = 2^{|x| \cdot |y|}$. # is pronounced 'smash' and was introduced by E. Nelson. It is an important symbol, since it is what allows polynomial growth rate for functions.

$$|x \# y| = |2^{|x| \cdot |y|}| = |x| \cdot |y| + 1$$

which is a polynomial in the lengths of x and y.

Claim: Any function $2^{p(|x|)}$, where p is a polynomial, can be expressed by a term in the language { 0, S, ·, #, x }.

Proof: For multiplication, $x \# y = 2^{|x| \cdot |y|}$. For addition, we note that $|1| = 1$ and thus that

$$(x \# 1) \cdot (y \# 1) = 2^{|x|} \cdot 2^{|y|} = 2^{|x| + |y|}$$

Furthermore, any term $t(\vec{x})$ in the language can be bounded by a function

$2^{p(|\vec{x}|)}$, where p is a polynomial.

$$|\vec{x}| = (|x|_1, |x|_2, \ldots, |x|_n), \text{where } \vec{x} = x_1, \ldots, x_n$$

This is easy to prove by induction on the complexity of terms.

In particular, polynomially and logarithmically bounded quantifiers will be expressible in the language. Our terms have the right growth rate for polynomial time computable functions. Without #, the lengths of terms would have linear growth rate rather than polynomial.

Definition: Let F be a class of functions. Then f has growth rate of class F iff

$$(\exists p \in \mathrm{F})\,(\forall x)\,(|f(x)| \leq p(|x|))$$

Math 271 - Topics in Weak Formal Systems

Lecture Notes, Set #7
March 21–26, 1988

Instructor: Sam Buss
Notes By: Stephen Carrier

Language for a Fragment of Arithmetic

We have the language of PA plus some other symbols:

$$0 \quad S \quad + \quad \cdot \quad |x| \quad \lfloor \tfrac{1}{2}x \rfloor \quad \# \quad \leq$$

Where the nonobvious intended interpretations are:

$|x| = \lceil \log_2(x+1) \rceil =$ the number of digits in the binary representation of x

$$x \# y = 2^{|x|\cdot|y|}$$

We have three kinds of quantifiers:

- Regular unbounded quantifiers $\forall x$ and $\exists x$.
- Bounded quantifiers $\forall x \leq t$ and $\exists x \leq t$ for t any term.
- Sharply bounded quantifiers $\forall x \leq |s|$ and $\exists x \leq |s|$ for s any term.

We might have taken $(\forall x \leq t)(\cdots)$ to be an abbreviation of $\forall x(x \leq t \rightarrow \cdots)$ but we didn't. Instead we enlarged the syntax of first-order logic to include bounded quantification as a distinct syntactic construction. We assume there are sufficient logical axioms to make bounded quantifier formulas logically equivalent to what they would have been were they not simply themselves.

Sharply bounded quantification is just a special case of bounded quantification.

We will show that there is a natural correspondence between bounded quantification and polynomially bounded quantification on one hand and between sharply bounded quantification and logarithmically bounded quantification on the other hand.

Consider the polynomially bounded quantification $(\forall x, |x| \leq p(|z|))$. Let t be a term such that $t(z) = 2^{p(|z|)}$ (Such terms always exist). Then $(\forall x, |x| \leq p(|z|))(\cdots)$ is equivalent to $(\forall x \leq t(z))(x \neq t(z) \rightarrow \cdots)$. For the other direction, consider $(\forall x \leq s(z))(\cdots)$ where s is a term. Then there exists a polynomial p such that $|s(z)| \leq p(|z|)$ for all z. So $(\forall x \leq s(z))(\cdots)$ is equivalent to $(\forall x, |x| \leq p(|z|))(x \leq s(z) \rightarrow \cdots)$.

Similar transformations are obtainable between sharply bounded quantification and logarithmically bounded quantification.

Another Definition of the Polynomial Hierarchy

This definition will be syntactic in the sense that we will define classes of formulas. For every class of formulas there is the corresponding class of the predicates defined by those formulas. All the classes of predicates about to be indirectly defined will together be the same old polynomial hierarchy.

Definitions: A *bounded formula* is a formula in which only bounded quantification occurs. A *sharply bounded formula* is a bounded formula in which only sharply bounded quantification occurs.

Definition: Let Δ_0^b be the class of sharply bounded formulas. The classes Σ_i^b, Π_i^b are defined inductively on i as the smallest classes satisfying:

1. $\Sigma_0^b = \Pi_0^b = \Delta_0^b$
2. For $i \geq 1$, $\Sigma_i^b \supseteq \Pi_{i-1}^b$.

If A, $B \in \Sigma_i^b$ then $A \wedge B$, $A \vee B \in \Sigma_i^b$.

If $A \in \Sigma_i^b$ and $B \in \Pi_i^b$ then $\neg B \in \Sigma_i^b$ and $B \supset A \in \Sigma_i^b$.

If $A \in \Sigma_i^b$ then $\exists x \leq A$, $\forall x \leq |s| A$ are in Σ_i^b.

3. This is the dual to (2) obtained from simultaneously transposing all $\forall$ with $\exists$ and all Σ with Π.

The bounded hierarchy counts alternations of bounded quantifiers but ignores sharply bounded quantifiers; this is analogous to the arithmetic hierarchy which counts alternation of regular quantifiers but ignores bounded quantifiers. Because of the equivalence of term-bounds with polynomial and logarithmic bounds, we have already verified that the analogous quantifier exchange properties do hold.

Open Question: The bounded hierarchy can be denoted by $\Delta_0(\#)$ which is the class of bounded formulas in the usual arithmetic language expanded by including #. Is $\Delta_0(\#) = \Delta_0$?

Axioms of Bounded Arithmetic

We examine several forms of restricted induction. We are interested in theories much weaker than PA.

Definition: Let Ψ be a class of formulas.

Ψ-IND is the axiom scheme: For $A \in \Psi$:

$$A(0) \wedge \forall x(A(x) \rightarrow A(Sx)) \rightarrow \forall x A(x)$$

Definition: Ψ-PIND is the axiom scheme: For $A \in \Psi$:

$$A(0) \wedge \forall x(A(\lfloor \tfrac{1}{2}x \rfloor) \rightarrow A(x)) \rightarrow \forall x A(x)$$

(In the above two definitions $A = A(x, \vec{z})$ is allowed to have parameters $\vec{z}$.)

This is essentially induction on the length of x. Suppose we know $A(0)$ and $\forall x(A(\lfloor \frac{1}{2}x \rfloor) \rightarrow A(x))$. We can simulate PIND to deduce $A(100)$ in seven steps, as follows:

$$\begin{array}{lrrl}
 & A(0) & 0_{10} = & 0_2 \\
\text{hence} & A(1) & 1_{10} = & 1_2 \\
\text{hence} & A(3) & 3_{10} = & 11_2 \\
\text{hence} & A(6) & 6_{10} = & 110_2 \\
\text{hence} & A(12) & 12_{10} = & 1100_2 \\
\text{hence} & A(25) & 25_{10} = & 11001_2 \\
\text{hence} & A(50) & 50_{10} = & 110010_2 \\
\text{hence} & A(100) & 100_{10} = & 1100100_2
\end{array}$$

Our intuition should be that PIND is more 'feasible' than IND because when we have the power of the PIND hypothesis we can convert an induction proof to a brute-force proof (simulating the induction by hand for a particular value) with fewer steps. Simulating an IND proof of $A(\mathbf{n})$ takes n steps, whereas simulating a PIND proof of $A(\mathbf{n})$ takes only $|n|$ steps. Since PIND is more 'feasible' than IND we might expect that the Ψ-IND axioms imply the Ψ-PIND axioms; we shall prove such results below.

Definition: Ψ-LIND is the axiom scheme: For $A \in \Psi$:

$$A(0) \wedge \forall x(A(x) \rightarrow A(Sx)) \rightarrow \forall x A(|x|)$$

We notice that Ψ-IND $\Rightarrow$ Ψ-LIND. We will later see that Ψ-LIND $\Leftrightarrow$ Ψ-PIND for reasonable Ψ over some simple base theories.

In the theories we are considering $\forall y \exists x(|x| = y)$ is not a theorem. In words, exponentiation is not total. In such theories Ψ-LIND is not necessarily equivalent to Ψ-IND.

In addition to induction axioms we must have a base theory. The base theory is called BASIC and consists of the universal closures of:

(B1) $y \leq x \supset y \leq Sx$
(B2) $x \neq Sx$
(B3) $0 \leq x$
(B4) $x \leq y \wedge x \neq y \leftrightarrow Sx \leq y$
(B5) $x \neq 0 \supset 2 \cdot x \neq 0$
(B6) $y \leq x \vee x \leq y$
(B7) $x \leq y \wedge y \leq x \supset x = y$
(B8) $x \leq y \wedge y \leq z \supset x \leq z$
(B9) $|0| = 0$
(B10) $x \neq 0 \supset |2 \cdot x| = S(|x|) \wedge |S(2 \cdot x)| = S(|x|)$
(B11) $|S0| = S0$
(B12) $x \leq y \supset |x| \leq |y|$
(B13) $|x \# y| = S(|x| \cdot |y|)$
(B14) $0 \# y = S0$
(B15) $x \neq 0 \supset 1\#(2 \cdot x) = 2 \cdot (1\#x) \wedge 1\#(S(2 \cdot x)) = 2(1\#x)$
(B16) $x\#y = y\#x$
(B17) $|x| = |y| \supset x\#z = y\#z$
(B18) $|x| = |u| + |v| \supset x\#y = (u\#y) \cdot (v\#y)$
(B19) $x \leq x + y$
(B20) $x \leq y \wedge x \neq y \supset S(2 \cdot x) \leq 2 \cdot y \wedge S(2 \cdot x) \neq 2 \cdot y$
(B21) $x + y = y + x$
(B22) $x + 0 = x$
(B23) $x + Sy = S(x + y)$
(B24) $(x + y) + z = x + (y + z)$
(B25) $x + y \leq x + z \leftrightarrow y \leq z$
(B26) $x \cdot 0 = 0$
(B27) $x \cdot (Sy) = (x \cdot y) + x$
(B28) $x \cdot y = y \cdot x$
(B29) $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
(B30) $x \geq S0 \supset (x \cdot y \leq x \cdot z \leftrightarrow y \leq z)$
(B31) $x \neq 0 \supset |x| = S(|\lfloor \frac{1}{2}x \rfloor|)$
(B32) $x = \lfloor \frac{1}{2}y \rfloor \leftrightarrow (2 \cdot x = y \vee S(2 \cdot x) = y)$

With stronger induction axioms such as Σ_1^b-IND we could prove the associative law:

$$(x + y) + z = x + (y + z)$$

with a much weaker base theory. The BASIC axioms are so numerous because PIND is so weak and because there are so many functions to define.

Definitions: Let $S_2^i = \text{BASIC} + \Sigma_i^b\text{-PIND}$. Let $T_2^i = \text{BASIC} + \Sigma_i^b\text{-IND}$. Let

$$S_2 = \bigcup_i S_2^i \qquad T_2 = \bigcup_i T_2^i$$

Later we will prove that $S_2 = T_2$.

Theorem: For $i \geq 0$, $S_2^i \vdash \Sigma_i^b\text{-LIND}$

proof: Let $A(x) \in \Sigma_i^b$. Argue in S_2^i. We want to show

$$A(0) \wedge \forall x(A(x) \rightarrow A(Sx)) \rightarrow \forall x A(|x|)$$

Let $B(x)$ be the formula $A(|x|)$. B is also Σ_i^b. $|0| = 0$ is in BASIC so $S_2^i \vdash A(0) \rightarrow B(0)$. By this and axiom 31,

$$S_2^i \vdash \forall x(A(x) \rightarrow A(Sx)) \rightarrow \forall x(B(\lfloor \tfrac{1}{2}x \rfloor) \rightarrow B(x))$$

By Σ_i^b-PIND:

$$S_2^i \vdash B(0) \wedge \forall x(B(\lfloor \tfrac{1}{2}x \rfloor) \rightarrow B(x)) \rightarrow \forall x B(x)$$

So

$$S_2^i \vdash A(0) \wedge \forall x(A(x) \rightarrow A(Sx)) \rightarrow \forall x B(x)$$

Since $\forall x B(x)$ is $\forall x A(|x|)$ we have proved LIND for $A(x)$ and are done. •

The converse of this holds for $i \geq 1$. Precisely:

Theorem: For $i \geq 1$, $S_2^1 + \Sigma_i^b\text{-LIND} \vdash \Sigma_i^b\text{-PIND}$. Equivalently: $S_2^i \equiv \text{BASIC} + \Sigma_i^b\text{-PIND} \equiv S_2^1 + \Sigma_i^b\text{-LIND}$.

Introducing Function and Predicate Symbols

We will see that we can extend the language with new function and predicate symbols in such a way that the new symbols can be used in induction axioms.

Definition: A formula A said to be Δ_k^b with respect to a theory R if there exist formulas B, $C \in \Sigma_k^b$ such that

$$R \vdash (A \leftrightarrow B) \wedge (A \leftrightarrow \neg C)$$

Definition: A function $f : \mathbf{N}^k \to \mathbf{N}$ is Σ_i^b-defined by R if and only if there is a Σ_i^b-formula $A(y, x_1, \ldots, x_k)$ and a term t such that

$$\begin{array}{rl} \textit{(i)} & R \vdash \forall \vec{x}\, (\exists y \leq t) A(y, \vec{x}) \\ \textit{(ii)} & R \vdash \forall \vec{x} \forall y \forall y' (A(y, \vec{x}) \wedge A(y', \vec{x}) \to y = y') \\ \textit{(iii)} & \text{For all } \vec{n} \in \mathbf{N}^k,\ \mathbf{N} \models A(f(\vec{\mathbf{n}}), \vec{\mathbf{n}}) \end{array}$$

By a theorem of Parikh, the condition that a term t bounds y is superfluous for theories R which have only universal closures of bounded formulas as axioms. S_2^i, T_2^i, S_2, T_2 are such theories because induction axioms on bounded formulas can be re-expressed in bounded form. For example

$$\forall x[A(0) \wedge (\forall y \leq x)(A(y) \to A(Sy)) \to A(x))]$$

is stronger than IND(A) but can be proven from IND$(y \leq x \to A(y))$.

Theorem: Let R be one of S_2^i or T_2^i. Let $A(y, \vec{x})$ be a Σ_1^b-function definition in R, and let $R^* \equiv R + \forall \vec{x}\, (A(f(\vec{x}), \vec{x}))$. Let $\Sigma_i^b(f)$ be the class of formulas defined as the Σ_i^b-formulas were defined, except that the new symbol f is allowed in open formulas and in bounding terms. Then for $B(\vec{x})$ any $\Sigma_i^b(f)$-formula there exists a formula $B^*(\vec{x}) \in \Sigma_i^b$ such that:

$$R^* \vdash B(\vec{x}) \leftrightarrow B^*(\vec{x})$$

proof: By the definition of Σ_1^b-definable there is a term t that R^*-provably bounds f.

For every occurrence of f in the bounding term of a quantifier, such as $\forall z \leq s(f(\vec{r}))(\cdots)$ replace that term so that we have $\forall z \leq s(t(\vec{r}))(z \leq s(f(\vec{r})) \to \cdots)$. This uses the fact that since all the the original functions of S_2^i are nondecreasing in each argument, every term built from such functions is also nondecreasing in each argument.

If f occurs more than once in the same bounding term the transformations that remove occurrences of f can proceed in arbitrary order.

By this procedure we have obtained B_1 in which f does not occur in any quantifier bounds. Now use prenex operations to obtain $B_2 \in \Sigma_i^b$ in prenex-normal form. B_2 has the form:

$$(\mathrm{Q}x_1 \leq t_1) \cdots (\mathrm{Q}x_k \leq t_k)\, C(f(\vec{r}))$$

$C(f(\vec{r}))$ is R^*-equivalent to both

$$\begin{array}{ll} \textit{(i)} & (\exists y \leq t(\vec{r}))[A(y,\vec{r}) \wedge C(y)] \\ \textit{(ii)} & (\forall y \leq t(\vec{r}))(A(y,\vec{r}) \rightarrow C(y)] \end{array}$$

Replace $C(f(\vec{r}))$ by whichever one of these does not increase the number of quantifier alternations. Do this for every occurrence of f in the matrix of B_2.

Corollary: Fix $i \geq 1$. Let R be S_2^i (or resp. T_2^i). Let f be Σ_1^b-defined in R. Then the theory $R + \forall \vec{x}\, A(f(\vec{x}), \vec{x}) + \Sigma_i^b(f)$-PIND (resp. $\Sigma_i^b(f)$-IND) is conservative over R.

So Σ_1^b-defined function symbols can be introduced and used freely in induction axioms. We actually only showed this for one function symbol but it is easy to extend this argument to the case of many function symbols. A similar proof shows that Δ_1^b-defined predicates can be introduced and used freely in induction axioms. A formula analagous to B_2 is obtained in the same way. Δ_1^b-predicates have both Σ_i^b and Π_i^b formulas to express them, so an appropriate formula can always be found that won't increase the number of alternations of quantifiers.

Σ_1^b-definable functions for S_2^1

Theorem: S_2^1 can Σ_1^b-define the predecessor function:

$$P(x) = \begin{cases} x-1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \end{cases}$$

P is defined by $b = P(a) \overset{df}{\longleftrightarrow} M(a,b)$ where $M(a,b)$ is the formula

$$Sb = a \vee (a = 0 \wedge b = 0)$$

proof: First we note four simple theorems in S_2^1.

$S_2^1 \vdash x \le x$	(from B6)
$S_2^1 \vdash x \le Sx$	(from previous statement plus B1)
$S_2^1 \vdash \neg Sx \le x$	(from previous statement plus B2, B7)
$S_2^1 \vdash \neg Sx = 0$	(from previous statement plus B3)

We need to prove both

$$\begin{array}{ll} (\textit{uniqueness}) & S_2^1 \vdash \forall x \forall y \forall y'(M(x,y) \wedge M(x,y') \rightarrow y = y') \\ (\textit{existence}) & S_2^1 \vdash \forall x (\exists y \le x) M(x,y) \end{array}$$

(*uniqueness:*) First $S_2^1 + M(0,y) + M(0,y') \vdash y = y'$ obviously, since S_2^1 knows that 0 is the successor of no element. It suffices to show

$$S_2^i + (Sy = x) + (Sy' = x) \vdash y = y'$$

Argue in the so-extended theory:

$\vdash Sy = Sy' = x$	
$\vdash y \le y' \rightarrow y = y' \vee Sy \le y'$	(by B4)
$\vdash y \le y' \rightarrow y = y' \vee Sy' \le y'$	(by $Sy = x = Sy'$)
$\vdash y \le y' \rightarrow y = y'$	(because $\neg Sy' \le y'$)

Similarly $\vdash y' \le y \rightarrow y = y'$. We conclude $\vdash y = y'$ using (B6).

(*existence:*) $S_2^1 \vdash (\exists y \le 0) M(0,y)$ obviously. So by Σ_1^b-PIND it suffices to show:

$$S_2^1 \vdash (\exists y \le \lfloor \tfrac{1}{2}x \rfloor) M(\lfloor \tfrac{1}{2}x \rfloor, y) \rightarrow (\exists y \le x) M(x,y)$$

This is proved by division into cases:

$$S_2^1 \vdash x = S(2 \cdot \lfloor \tfrac{1}{2}x \rfloor) \vee x = 2 \cdot \lfloor \tfrac{1}{2}x \rfloor \qquad \text{(by B32)}$$

First case:

$$\left\{ \begin{array}{l} S_2^1 \vdash x = S(2 \cdot \lfloor \tfrac{1}{2}x \rfloor) \rightarrow M(x, 2 \cdot \lfloor \tfrac{1}{2}x \rfloor) \\ S_2^1 \vdash x = S(2 \cdot \lfloor \tfrac{1}{2}x \rfloor) \rightarrow 2 \cdot \lfloor \tfrac{1}{2}x \rfloor \le x \end{array} \right.$$

Second case:

$$\left\{ \begin{array}{l} S_2^1 \vdash x = 2 \cdot \lfloor \tfrac{1}{2}x \rfloor \wedge M(\lfloor \tfrac{1}{2}x \rfloor, y) \rightarrow M(x, S(2 \cdot y)) \quad (*) \\ S_2^1 \vdash M(\lfloor \tfrac{1}{2}x \rfloor, y) \rightarrow S(2 \cdot y) \le x \end{array} \right.$$

In (*), $S(2\cdot y)$ works because:

$$\begin{aligned} SS(2\cdot y) &= S(2\cdot y)+S0 \\ &= 2\cdot y + SS0 \\ &= 2\cdot y + 2\cdot S0 \\ &= 2\cdot(y+S0) \\ &= 2\cdot Sy \\ &= 2\cdot\lfloor \tfrac{1}{2}x\rfloor = x \end{aligned}$$

This uses the deep result that $2\cdot S0 = SS0$. Thus is the proof completed. •

That was a demonstration. Such details will be ommitted in the future. Be assured that somebody else has checked them and they work. Instead of treating such details for every function and predicate that we want to use we just exhibit their Σ_1^b and Δ_1^b-definitions and emphatically assert without proof that S_2^1 proves existence and uniqueness.

Facts: The following are Σ_1^b-defined functions and Δ_1^b-defined predicates in S_2^1 and all larger theories.

$$a < b \overset{df}{\longleftrightarrow} a \le b \wedge \neg a = b$$

By similar methods define $>$, $\ge$, $\ne$.

$$c = max(a,b) \overset{df}{\longleftrightarrow} c \ge a \wedge c \ge b \wedge (c = a \vee c = b)$$

Define *min* similarly.

$$Power2(a) \overset{df}{\longleftrightarrow} \text{“}a \text{ is a power of 2”} \overset{df}{\longleftrightarrow} S(|P(a)|) = |a|$$

$$c = Exp(a,b) \overset{df}{=} 2^{min(a,|b|)} \overset{df}{\longleftrightarrow} Power2(c) \wedge |c| = 1 + min(a,|b|)$$

$$b = Mod2(a) \overset{df}{\longleftrightarrow} 2\cdot\lfloor \tfrac{1}{2}a\rfloor + b = a$$

For the next two definitions, 2^b means $2^{min(b,|a|)}$. The most significant part is defined:

$$c = MSP(a,b) \overset{df}{=} \left\lfloor \frac{a}{2^b}\right\rfloor \overset{df}{\longleftrightarrow} 2^b\cdot c \le a \wedge 2^b\cdot(c+1) > a$$

The least significant part is defined:

$$d = LSP(a,b) \overset{df}{\longleftrightarrow} (\exists c \le a)(c = MSP(a,b) \wedge c \cdot 2^b + d = a) \vee |d| \le b$$

$$c = Bit(b,a) \overset{df}{=} Mod2(MSP(a,b))$$

$Bit(b,a)$ is the b^{th} bit in the binary representation of a. By the way, S_2^1 proves that a number is completely determined by its binary representation:

$$S_2^1 \vdash |a| = |b| \wedge (\forall i < |a|)(Bit(i,a) = Bit(i,b) \rightarrow a = b)$$

Now subtraction is definable:

$$c = a \dot{-} b \overset{df}{\longleftrightarrow} (c = 0 \wedge a \le b) \vee (b + c = a)$$

The definability of this function is mentioned so late because the machinery of bit-functions is needed to prove existence.

$$c = \left\lfloor \frac{a}{b} \right\rfloor \overset{df}{\longleftrightarrow} (\exists x < b)(b \cdot c + x = a) \vee (b = 0 \wedge c = 0)$$

$$Rem(a,b) \overset{df}{=} a \dot{-} b \cdot \left\lfloor \frac{a}{b} \right\rfloor$$

$$b|a \overset{df}{\longleftrightarrow} Rem(a,b) = 0 \wedge b \ne 0$$

$$Even(a) \overset{df}{\longleftrightarrow} Mod2(a) = 0$$

$$Odd(a) \overset{df}{\longleftrightarrow} Mod2(a) = 1$$

Protosequences

We won't try to rely on traditional Gödel numbers in S_2 because exponentiation is not total. Instead we code a sequence of numbers as follows: Write each number in binary representation, with the least significant figures on the right, as is traditional. Prepend each number with a comma, and concatenate them from right to left. Translate this three-symboled string into a unique string of zeros and ones by translating each of 0, 1, and comma into a couple of bits.

$$\begin{aligned} 0 &\mapsto 10 \\ 1 &\mapsto 11 \\ , &\mapsto 01 \end{aligned}$$

Such codes will be *sequences.* Because of technical difficulties with the formalization of sequences, we will formalize *protosequences* first. Protosequences are sequences which use equal-length binary representations for each of their elements.

$$Comma(b,a) \overset{df}{\longleftrightarrow} Even(b) \wedge Bit(b+1,a) = 0 \wedge Bit(b,a) = 1$$

$Comma(b,a)$ means that the b^{th} bit of a is the first (from the right) of a pair of bits that denote a comma.

$$\begin{aligned} c = Digit(b,a) \overset{df}{\longleftrightarrow} \; & (c = Bit(b,a) \wedge Even(b) \wedge Bit(b+1,a) = 1) \\ & \vee (c = 2 \wedge (Odd(b) \vee Bit(b+1,a) = 0)) \end{aligned}$$

$Digit(b,a)$ returns 0 or 1 respectively if the b^{th} bit is the first of a pair of bits which denote 0 or 1 respectively. Otherwise, $Digit(b,a)$ returns the value 2.

$$\begin{aligned} ProtosqSL(a,b,c) \; \overset{df}{\longleftrightarrow} \; & |a| + 2 = 2 \cdot c \cdot Sb \\ & \wedge (\forall y < |a|)[((2 \cdot b + 2) | (y+2) \rightarrow Comma(y,a)) \\ & \qquad \wedge (\neg (2 \cdot b + 2) | (y+2) \rightarrow Digit(2 \cdot y, a) < 2)] \end{aligned}$$

$ProtosqSL(a,b,c)$ says that a is a sequence of c numbers each represented by b pairs of bits.

$$\begin{aligned} b = Protosize(a) \overset{df}{\longleftrightarrow} \; & (Comma(2 \cdot b, a) \wedge (\forall i < b)(\neg Comma(2 \cdot i, b)) \\ & \vee (b = 0 \wedge (\forall i < |a|)(\neg Comma(2 \cdot i, b)) \end{aligned}$$

$Protosize(a)$ is the position of the first comma in a as measured by counting pairs of bits. If there is no comma in a then $Protosize(a) = 0$. If a is a protosequence then b is the number of pairs of bits used to code each element of the sequence.

$$Protolength(a) \overset{df}{=} \left\lfloor \frac{|a| + 1}{2 \cdot Protosize(a) + 2} \right\rfloor$$

If a is a protosequence then $\mathit{Protolength}(a)$ will be the number of elements in a.

$$\mathit{Protoseq}(a) \overset{df}{\longleftrightarrow} \mathit{ProtosqSL}(a, \mathit{Protosize}(a), \mathit{Protolength}(a))$$

$\mathit{Protoseq}(a)$ says that a is a protosequence.

$$\begin{aligned} b = \mathit{Proto}\beta\mathit{1}(a) \overset{df}{\longleftrightarrow}\ & |b| \leq \mathit{Protosize}(a) \\ & \wedge(\forall i < a)(i < \mathit{Protosize}(c) \rightarrow \mathit{Bit}(i, b) = \mathit{Bit}(2 \cdot i, a)) \end{aligned}$$

$\mathit{Proto}\beta\mathit{1}(a)$ is the first element of the protosequence a.

$$\mathit{Proto}\beta(b, a) \overset{df}{=} \mathit{Proto}\beta\mathit{1}(\mathit{MSP}(a, (2 \cdot \mathit{Protosize}(a) + 2) \cdot (b \dot{-} 1))$$

$\mathit{Proto}\beta(b, a)$ is the b^{th} element of the protosequence a.

So far we haven't used the # function in the definitions of extra functions or in the proofs of their existence and uniqueness. The # function will be important for proving the existence of codes for sequences under reasonable conditions.

Math 271 - Topics in Weak Formal Systems

Lecture Notes, Set #8
April 4–8, 1988

Instructor: Sam Buss
Notes By: Juan Bagaria

Earlier we showed: Ψ-PIND $\Leftarrow$ Ψ-LIND for Ψ a class Σ_i^b.

Theorem: Let $i \geq 1$.
(a) $S_2^1 + \Sigma_i^b\text{-LIND} \vdash \Sigma_i^b\text{-PIND}$
(b) $T_2^1 + \Sigma_i^b\text{-LIND} \vdash \Sigma_i^b\text{-PIND}$

Corollary:
(a) $S_2^1 + \Sigma_i^b\text{-LIND} \equiv S_2^i \quad (i \geq 1)$
(b) $T_2^1 \supseteq S_2^i \quad (i \geq 1)$

Proof: (a) $S_2^i \equiv S_2^1 + \Sigma_i^b\text{-PIND}$.
(b) Σ_i^b-LIND $\Longrightarrow$ Σ_i^b-LIND is obvious.

Proof of the Theorem: We prove (a). The proof of (b) is identical. We need $MSP(x,i) = \lfloor x \cdot 2^{-i} \rfloor$ and $\dot{-}$ to be Σ_1^b-definable. Let $A(x)$ be any Σ_i^b-formula. We want to show:

$$S_2^1 + \Sigma_i^b\text{-LIND} \vdash A(0) \wedge \forall x(\lfloor \tfrac{1}{2}x \rfloor) \supset A(x)) \supset \forall x A(x).$$

Let $B(x,u)$ be $A(MSP(x, |x| \dot{-} u))$. We will use LIND on B with respect to u. Note that B is in Σ_i^b. We have $S_2^1 \vdash A(0) \supset B(0)$ since $S_2^1 \vdash MSP(x,|x|) = 0$. Also

$$S_2^1 \vdash \forall x(A(\lfloor \tfrac{1}{2}x \rfloor) \supset A(x)) \supset \forall u(B(c,u) \supset B(c,u+1))$$

where c is a new variable symbol, since $\lfloor \frac{1}{2} MSP(x,u+1) \rfloor = MSP(x,u)$. So, using Σ_i^b-LIND,

$$S_2^1 + \Sigma_i^b\text{-LIND} \vdash A(0) \wedge \forall x(A(\lfloor \tfrac{1}{2}x \rfloor) \supset A(x)) \supset B(c,|c|).$$

But $S_2^1 + \Sigma_i^b$-LIND $\vdash B(c,|c|) \supset A(c)$ since $MSP(c,0) = c$. So,

$$S_2^1 + \Sigma_i^b\text{-LIND} \vdash A(0) \wedge \forall x(A(\lfloor \tfrac{1}{2}x \rfloor) \supset A(x)) \supset A(c)$$

and use $\forall$-introduction. q.e.d.

Theorem:
(a) $S_2^1 + \Sigma_i^b$-PIND $\equiv S_2^1 + \Pi_i^b$-PIND
(b) $S_2^1 + \Sigma_i^b$-IND $\equiv S_2^1 + \Pi_i^b$-IND
(c) $S_2^1 + \Sigma_i^b$-LIND $\equiv S_2^1 + \Pi_i^b$-LIND

Proof of (a)$\supseteq$: Let $A(x)$ be in Π_i^b. We want to show that $S_2^1 + \Sigma_i^b$-PIND proves induction on A. Let $B(x,u)$ be the formula $A(MSP(x,u))$. So, $S_2^1 \vdash A(0) \supset B(c,|c|)$, where c is a new variable. Also,

$$S_2^1 \vdash \forall x(A(\lfloor \tfrac{1}{2}x \rfloor) \supset A(x)) \supset \forall u(B(c,u+1) \supset B(c,u))$$

Equivalently:

$$S_2^1 \vdash \forall x(A(\lfloor \tfrac{1}{2}x \rfloor) \supset A(x)) \supset \forall u(\neg B(c,u) \supset \neg B(c,u+1)).$$

Note that $\neg B$ is in Σ_i^b. So,

$$S_2^1 + \Sigma_i^b\text{-PIND} \vdash \neg B(c,0) \wedge \forall u(\neg B(c,u) \supset \neg B(c,u+1)) \supset \neg B(c,|c|).$$

(Since PIND $\Longrightarrow$ LIND). And $S_2^1 \vdash B(c,0) \supset A(c)$. So,

$$S_2^1 \vdash A(0) \wedge \forall x(A(\lfloor \tfrac{1}{2}x \rfloor) \supset A(x)) \supset A(c)$$

and we are done by $\forall$-introduction. The other direction: (a)$\subseteq$ is similar. (b) and (c) are proved similarly. Idea: to do induction on $A(x)$ you instead do induction on $B(c,x) \equiv \neg A(c \dot- x)$.

Theorem: $1 \leq i$. Let $A(x)$ be a Δ_i^b-formula with respect to S_2^i. (For $i = 1$ A is a Δ_1^b-defined predicate of S_2^1). Then, S_2^1 proves regular induction for A, i.e.

$$S_2^i \vdash A(0) \wedge \forall x(A(x) \supset A(x+1)) \supset \forall x A(x)$$

Corollary: $S_2^i \vdash \Sigma_{i-1}^b$-IND. So, $S_2^i \supseteq T_2^{i-1}$

Proof: Let $A_E(x) \in \Sigma_i^b$, $A_u(x) \in \Pi_i^b$ and $S_2^i \vdash A(x) \equiv A_E(x)$ and $S_2^i \vdash A(x) \equiv A_u(x)$ (from the definition of A being Δ_i^b).

Let $B(x,z) \equiv (\forall y \leq z+1)(A(x \dot{-} y) \supset A(x))$. Note that $B(x,z)$ is (equivalent to) a formula in Π_i^b. Let c and d be new variables.

Claim: $S_2^1 \vdash (\forall x \leq c)B(x, \lfloor \frac{1}{2}d \rfloor) \supset (\forall x \leq c)B(x,d)$

Proof: We argue informally inside S_2^1. Assume that $(\forall x \leq c)B(x, \lfloor \frac{1}{2}d \rfloor)$ holds. Let $x \leq c, y \leq d+1$ and suppose $A(x \dot{-} y)$ holds. We want to show $A(x)$. We have that $A(x \dot{-} y) \supset A(x \dot{-} \lfloor \frac{1}{2}y \rfloor)$ since $(x \dot{-} y) \dot{-} (x \dot{-} \lfloor \frac{1}{2}y \rfloor) \leq \lfloor \frac{1}{2}d \rfloor + 1$ And $A(x \dot{-} \lfloor \frac{1}{2}y \rfloor) \supset A(x)$ since $\lfloor \frac{1}{2}y \rfloor \leq \lfloor \frac{1}{2}d \rfloor + 1$ and so the claim is proved.

By the Claim and by $\Pi_i^b - PIND$ on $(\forall x \leq c)B(x,d)$ with eigenvariable d we have that S_2^i proves

$$(\forall x \leq c)B(x,0) \supset (\forall x \leq c)B(x,c)$$

Now,

$$S_2^1 \vdash \forall x(A(x) \supset A(x+1)) \supset \forall x B(x,0).$$

Also,

$$S_2^1 \vdash B(c,c) \supset (A(0) \supset A(c)).$$

So,

$$S_2^i \vdash A(0) \wedge \forall x(A(x) \supset A(x+1)) \supset A(c).$$

q.e.d.

Theorem:
(a) $\forall i \geq 1$, $T_2^i \supset S_2^i \supset T_2^{i-1}$.
(b) $S_2 \equiv T_2$

Proof: $S_2^i \supset T_2^{i-1}$ is the previous corollary. And T_2^i proves $\Sigma_i^b - LIND$ which implies $\Sigma_i^b - PIND$ by the first theorem above.

Remark: It is open whether S_2^2 and T_2^1 are actually distinct. Takeuti has shown that $S_2^0 \neq T_2^0$ by showing that the predecessor function is not definable in S_2^0.

$$\begin{tikzcd}
\Sigma_i^b\text{-IND} \arrow[r, Leftrightarrow] \arrow[d, Rightarrow] & \Pi_i^b\text{-IND} \arrow[r, Leftrightarrow] & \Sigma_i^b\text{-MIN} \arrow[r, Leftrightarrow] & \Delta_{i+1}^b\text{-IND} \\
\Sigma_i^b\text{-PIND} \arrow[r, Leftrightarrow] \arrow[d, Leftrightarrow] & \Pi_i^b\text{-PIND} \arrow[r, Leftrightarrow] & \Sigma_i^b\text{-LIND} \arrow[r, Leftrightarrow] & \Pi_i^b\text{-LIND} \\
\Sigma_i^b\text{-LMIN} \arrow[r, Leftrightarrow] \arrow[d, Rightarrow] & \text{strong } \Sigma_i^b\text{-replacement} \arrow[r, Leftrightarrow] & (\Sigma_{i+1}^b \cap \Pi_{i+1}^b)\text{-PIND} \\
\Sigma_{i-1}^b\text{-IND}
\end{tikzcd}$$

$$\Sigma_{i+1}^b\text{-MIN} \Longleftrightarrow \Pi_i^b\text{-MIN}$$

$$\Sigma_{i+1}^b\text{-replacement} \Longrightarrow \Sigma_i^b\text{-PIND} \Longrightarrow \Sigma_i^b\text{-replacement}$$

$$S_2^{i+1} \underset{\Sigma_{i+1}^b}{\succ} T_2^i$$

$$S_2^{i+1} \underset{\mathcal{B}(\Sigma_{i+1}^b)}{\succ} T_2^i + \Sigma_{i+1}^b\text{-replacement}$$

Relationships among axiomatizations for Bounded Arithmetic relative to the base theory S_2^1 with $i \geq 1$

The following are Σ_1^b-definable functions and Δ_1^b-definable predicates of S_2^1.

$b = Numones(a)$ $\Leftrightarrow$ "Number of ones in a's binary representation". i.e.

$$b = Numones(a) \Leftrightarrow \exists w, |w| \leq 2(|(|a|)| + 1)(|a| + 2) \text{ such that}$$
$$[PSqSL(w, |(|a|)|, |a| + 1) \text{ and}$$
$$Proto\beta(1, w) = 0 \text{ and}$$
$$\forall i < |a|(Proto\beta(i + 2, w) = Proto\beta(i + 1, w) + Bit(i, w)) \text{ and}$$
$$b = Proto\beta(|a| + 1, w)]$$

Notice that this is a Σ_1^b formula.Also notice that the first bounded quantifier in this formula can be expressed as $\exists w \leq ((4(a + 1))\#(2(|a| + 1)))^2$. This is the first place we needed the # function (although even here it can be eliminated).

A $proto - sequence$ has fixed length entries. Adding an extra element to the sequence,when the new element is larger than the sizes of the old elements, would require **all** entries to be "stretched". This is a problem with protosequences. So, instead we define a more general notion of sequences which have variable length entries.

Like protosequences, sequences will be a string of 0's and 1's and commas represented as a binary number by writing 11 in place of 1, 10 in place of 0 and 01 in place of ,(comma).

Notice: If $A(x,\vec{z})$ is a Δ_1^b-defined predicate, then $f(x,\vec{z}) =$ (number of $i \leq |x|)A(i,\vec{z})$ is a Σ_1^b-defined function of S_2^1. (Translate to: If $A \in P$, then $f \in FP$).

[**Proof:** Let $g(x,\vec{z}) = b \Leftrightarrow |b| \leq |x|+1$ and $(\forall i \leq |b|)[Bit(i,b) = 1 \equiv A(i,\vec{z})]$ So, g is Σ_1^b-defined. But $f(x,\vec{z}) = Numones(g(x,\vec{z}))$. q.e.d.]

Also Notice: If $A(x,\vec{z})$ is a Δ_1^b-defined predicate, then

$$f(x,\vec{z}) = \begin{cases} \mu i \leq |x| A(i,\vec{z}) & \\ |x|+1 & \text{if no such i exists} \end{cases}$$

This is a Σ_1^b-defined function.

[**Proof:** $f(x,\vec{z}) =$ (number of $i \leq |x|)\forall j \leq |x|(j \leq i \supset \neg A(j,\vec{z}))$. q.e.d.]

Other Σ_1^b-definable functions and Δ_1^b-definable predicates of S_2^1 are:

$Seq(w)$: w is a sequence.

$Len(w)$: length of w.

$\beta(i,w)$: the ith element of w.

$w * a$: adds a as a new element to w.

$v ** w$: concatenates v and w.

For example, the first can be defined by:

$$Len(w) = (\text{number of } i \leq |w|) Comma(i, w)$$

And the second by:

$$Seq(w) \Leftrightarrow (\forall i < |w|)(Even(i) \supset Comma(i, w) \vee Digit(i, w) \neq 2)$$

$$\text{and } \ (w = 0 \vee Comma(0, w))$$

We leave the definitions of β, $*$ and $**$ as an exercise.

Theorem: If $f : \mathbb{N}^k \longrightarrow \mathbb{N}$ is in FP, then S_2^1 can Σ_1^b-define f.

Proof: Let M be a deterministic Turing Machine which on input $\vec{x}$ runs in time $p(|\vec{x}|)$ and outputs $f(\vec{x})$.

WLOG, M has a single tape which extends in one direction only. An ID of M is coded by $\langle\langle q_1, a_1\rangle, \langle q_2, a_2\rangle, ...\rangle$ where a_i is the ith symbol on the tape and q_i= a state if tape head is at position i (=blank, otherwise). $f(\vec{x}) = z \Leftrightarrow \exists w, |w| \leq 2(p(|\vec{x}| + 1)^2 \ [w = \langle w_1, ..., w_{p(|\vec{x}|)}\rangle$ and w_1 codes initial ID of M($\vec{x}$), w_{i+1} follows w_i by one step and $w_{p(|\vec{x}|)}$ has z written on the tape].

All the conditions above are expressible by Δ_1^b-predicates. Use $\Sigma_1^b - LIND$ to prove that $w = \langle w_1, ..., w_i\rangle$ exists, by induction on i up to i=$p(|\vec{x}|)$.q.e.d.

Aside: We have two definitions of "f is an NP-function":

$$(1) \text{ the graph of } f \text{ is in } NP \cap co - NP$$

$$(1b) \text{ the graph of } f \text{ is in } NP$$

Note that (1) $\Rightarrow$ (1b). Also, (1b) $\Rightarrow$ (1) since $f(x) \neq y$ can be expressed as $\exists z(z \neq y \wedge f(x) = z)$

$$(2) \text{ the predicate } f(x) \geq y \text{ is in } \ NP$$

Another possible definition gives the class $\square_2^p$:

(3) f can be computed in polynomial time with an oracle from NP

Note that (1) $\Rightarrow$ (2) $\Rightarrow$ (3). However it is not known whether the converses hold.

Aside:

$$\begin{tikzcd}
 {} & & {} & & \\
 & \Delta_3^p \arrow[ul, phantom, sloped, "\subseteq"] \arrow[ur, phantom, sloped, "\subseteq"] & & & \square_3^p \arrow[lll] \\
\Pi_2^p \arrow[ur, phantom, sloped, "\subseteq"] & & \Sigma_2^p \arrow[ul, phantom, sloped, "\subseteq"] & & \\
 & \Delta_2^p \arrow[ul, phantom, sloped, "\subseteq"] \arrow[ur, phantom, sloped, "\subseteq"] & & & \square_2^p \arrow[lll] \\
\mathrm{coNP} = \Pi_1^p \arrow[ur, phantom, sloped, "\subseteq"] & & \Sigma_1^p = \mathrm{NP} \arrow[ul, phantom, sloped, "\subseteq"] & & \\
 & \Delta_1^p = P \arrow[ul, phantom, sloped, "\subseteq"] \arrow[ur, phantom, sloped, "\subseteq"] & & & FP = \square_1^p \arrow[lll]
\end{tikzcd}$$

Definition: $\square_i^p = P^{\Sigma_{i-1}^p}$ the set of functions computable in polynomial time with an oracle from Σ_{i-1}^p.

Theorem: If $f \in \square_i^p (i \geq 1)$, then S_2^i can Σ_i^b-define f.

This theorem was stated and proved above for $i = 1$. The general case with $i > 1$ is proved in a similar manner. Recasting this theorem in terms of predicate symbols gives the following Corollary. (The proof is an exercise.)

Corollary:
(a) If $A \in P$, then S_2^1 can Δ_1^b-define A.
(b) If $A \in \Delta_i^P$, then S_2^i can Δ_i^b-define A.

Every $A \in P$ can be expressed as Σ_1^b-formula which is provably equivalent to a Π_1^b-formula in S_2^1.

Later we shall show that the converse to the above Corollary holds; namely, if $A(x)$ is Δ_1^b with respect to S_2^1 then $A(x) \in P$. We show on the next page that a predicate is definable in $\mathbb{N}$ by a Σ_1^b (or Π_1^b respectively) formula iff it is NP (or co-NP respectively).It follows that any predicate S_2^1-provably in $NP \cap co-NP$ is actually polynomial time.

Recall: It is open whether $NP \cap co - NP = P$.

Remark: Defining a polynomial time function in S_2^1 by the above theorem may not give an *intensional* definition.**Example:**

$$f(x) = \begin{cases} 0 & \text{if x is a Godel number of an } S_2^1 \text{ proof of } 0 = 1 \\ 1 & \text{otherwise} \end{cases}$$

.

For this function, S_2^1 does not prove $\forall x(f(x) \neq 0)$

Theorem: A predicate $Q \subset \mathbb{N}$ *is in* NP iff there exists a Σ_1^b-formula $A(x)$ such that $\forall x(x \in Q \Leftrightarrow \mathbb{N} \models A(x))$

Proof: Recall that $Q \in NP$ iff there is $R \in P$ and a polynomial q such that

$$\forall x(x \in Q \Leftrightarrow \exists w, |w| \leq q(|x|)R(x,w))$$

To prove the theorem: $\Rightarrow$: Since $R \in P$, it is Δ_1^b-definable in S_2^1, in particular, there is a Σ_1^b formula $A(x,w)$ such that

$$\forall x,w[R(x,w) \Leftrightarrow \mathbb{N} \models A(x,w)]$$

So,

$$\forall x[x \in Q \Leftrightarrow \exists w, |w| \leq p(|x|)A(x,w)]$$

(Notice that the existential quantifier is polynomially bounded.It can be reexpressed as $(\exists w \leq t(x))[|w| \leq p(|x|) \wedge A(x,w)]$ which is a Σ_1^b formula.)
$\Leftarrow$: If $A(x)$ is a Σ_1^b formula, put A in prenex normal form,say:

$$(\exists y_1 \leq t_1(x))(\forall z_1 \leq |s_1(x)|)(\exists y_2 \leq t_2(x)).....B(x,\vec{y},\vec{z})$$

All universal quantifiers are sharply bounded and $B(x,\vec{y},\vec{z})$ is quantifier free. Reexpress bounded (respectively sharply bounded) quantifiers as polynomially (respectively logarithmically) bounded quantifiers and use quantifier exchange property to get a formula which shows that $A(x)$ expresses an NP predicate.q.e.d.

Theorem: More generally,

$$Q \in \Sigma_i^P \Leftrightarrow \exists A \in \Sigma_i^b \text{ such that } \forall x[x \in Q \Leftrightarrow \mathbb{N} \models A(x)]$$

(These theorems are due essentially to Stockmeyer and Wrathall. Kent and Hodgson have stronger versions than what I stated here.)

So, S_2^1 uses length induction on NP predicates.

Cook(1975) introduced a theory PV having function symbols for each polynomial time-function and length induction on polynomial-time predicates. It turns out that S_2^1 is conservative over PV.

Goals: To prove that:

*Every Σ_i^b-definable function of S_2^i is in $\Box_i^P$. So, every Σ_i^b-definable function of S_2^1 is a polynomial-time function.

**Any polynomial predicate A such that $S_2^1 \vdash \forall x A(x)$ has a polynomial size extended Frege proofs.[Cook] More precisely, given A Δ_1^b-defined by S_2^1 (and hence expressing a property in P) there are propositional formulas $\|A\|_n$ $n = 1, 2, 3, ..$ where $\|A\|_n$ says $(\forall x, |x| = n)$ $A(x)$ and $\|A\|_n$ will have polynomial size eF proofs (as a function of n).

To prove this we need some proof theory for first-order logic.

Sequent calculus for First-Order Logic

Language: $\wedge, \vee, \supset, \neg, \forall, \exists, \forall \leq, =$

Variables: Free: a,b,c,... Bound: x,y,z,...

In formulas, only free variables can occur free and only bound variables can occur bound. Terms have only free variables. Semiterms are like terms except that they contain both free and bound variables.

Non-logical symbols: $0, S, +, \cdot, \leq, |x|, \lfloor \frac{1}{2}x \rfloor, \#$

The sequents have the following rules:

$\vee$:left, $\vee$:right, $\wedge$:left, $\wedge$:right, $\supset$:left, $\supset$:right, $\neg$:left, $\neg$:right, cut rule and structural rule are identical to the propositional calculus rules.

$\forall$:left	$\dfrac{A(t), \Gamma \longrightarrow \Delta}{\forall x A(x), \Gamma \longrightarrow \Delta}$	where t is a term
$\forall$:right	$\dfrac{\Gamma \longrightarrow A(b), \Delta}{\Gamma \longrightarrow \forall x A(x), \Delta}$	where the eigenvariable b does not occur in the lower sequent
$\exists$:left	$\dfrac{A(b), \Gamma \longrightarrow \Delta}{\exists x A(x), \Gamma \longrightarrow \Delta}$	where b does not appear in the lower sequent
$\exists$:right	$\dfrac{\Gamma \longrightarrow A(t), \Delta}{\Gamma \longrightarrow \exists x A(x), \Delta}$	
$\forall \leq$:left	$\dfrac{A(t), \Gamma \longrightarrow \Delta}{t \leq s, (\forall x \leq s) A(x), \Gamma \longrightarrow \Delta}$	
$\forall \leq$:right	$\dfrac{b \leq s, \Gamma \longrightarrow A(b), \Delta}{\Gamma \longrightarrow (\forall x \leq s) A(x), \Delta}$	where b does not appear in the lower sequent
$\exists \leq$: left	$\dfrac{b \leq s, A(b), \Gamma \longrightarrow \Delta}{(\exists x \leq s) A(x), \Gamma \longrightarrow \Delta}$	where b does not occur in the lower sequent

$\exists \leq$: right $\qquad \dfrac{\Gamma \longrightarrow A(t), \Delta}{t \leq s, \Gamma \longrightarrow (\exists x \leq s) A(x), \Delta}$

Exercise: Show that $\longrightarrow (\exists x \leq s) A \equiv \exists x (x \leq s \wedge A)$ is provable with the above rules.

Gentzen defined LK (Logische Kalkul) as the system above without bounded quantifiers. Let LKB be $LK+$ bounded quantifiers.

Logical axioms: $A \longrightarrow A$ for A atomic.

Equality axioms:

$s_1 = t_1 \longrightarrow S(s_1) = S(t_1)$

and similarly for each unary function symbol.

$s_1 = t_1,\ s_2 = t_2 \longrightarrow s_1 + s_2 = t_1 + t_2$

and similarly for each binary function symbol.

$s_1 = t_1,\ s_2 = t_2,\ s_1 \leq s_2 \longrightarrow t_1 \leq t_2$

$s_1 = t_1 \longrightarrow t_1 = s_1$

$s_1 = t_1,\ t_1 = u_1 \longrightarrow s_1 = u_1$

$\longrightarrow s = s$

Math 271 - Topics in Weak Formal Systems

Lecture Notes, Set #9
April 11–15, 1988

Instructor: Sam Buss
Notes by Jim Gloor

Sequent Calculus for First-order Logic (con't)

In addition to the logical and equality axioms, we add **non-logical axioms** which are sequents based on the BASIC axioms for bounded arithmetic. The only formulas occurring in these axioms should be **atomic**. For example, we have...

(1') $y \leq x \longrightarrow y \leq Sx$
(2') $x = Sx \longrightarrow$
(3') $\longrightarrow 0 \leq x$
(4') $x \leq y \longrightarrow x = y, Sx \leq y$, and $Sx \leq y \longrightarrow x \leq y$,
and $Sx \leq y, x = y \longrightarrow$
(5') $SS0 \cdot x = 0 \longrightarrow x = 0$
(6') $\longrightarrow y \leq x, x \leq y$
(7') $x \leq y, y \leq x \longrightarrow x = y$
(8') $x \leq y, y \leq z \longrightarrow x \leq z$
etc.

In addition to the logical inferences, we add the following **induction rules**, where Φ is a class of formulas...

Φ–PIND Rule:

$$\frac{A(\lfloor \frac{1}{2}b \rfloor), \Gamma \longrightarrow A(b), \Delta}{A(0), \Gamma \longrightarrow A(t), \Delta},$$

where $A \in \Phi$, t is a term, and b is an eigenvariable which does not occur in the lower sequent. Note that this is sound in that if the upper sequent is valid, then the lower sequent is, too.

Φ-IND Rule:
$$\frac{A(b), \Gamma \longrightarrow A(Sb), \Delta}{A(0), \Gamma \longrightarrow A(t), \Delta},$$
where $A \in \Phi$, t is a term, and b is an eigenvariable which does not occur in the lower sequent.

Fact: *The Φ-IND rule is equivalent to the Φ-IND axiom (i.e., $\longrightarrow (A(0) \wedge \forall x\, (A(x) \supset A(Sx))) \supset \forall x\, A(x)$) and similarly for PIND.*

Proof: ($\Leftarrow$) This is clear.
($\Rightarrow$) First we derive without the induction rule the sequent
$$A(b), \forall x\, (A(x) \supset A(Sx)) \longrightarrow A(Sb).$$
Then letting c be a new variable, Φ-IND gives
$$A(0), \forall x\, (A(x) \supset A(Sx)) \longrightarrow A(c).$$
Applying $\forall$:right, we have
$$A(0), \forall x\, (A(x) \supset A(Sx)) \longrightarrow \forall x\, A(x),$$
and finally after two $\wedge$:lefts, a structural inference, and a $\supset$:right we derive
$$\longrightarrow (A(0) \wedge \forall x\, (A(x) \supset A(Sx))) \supset \forall x\, A(x).$$
A similar proof works for PIND. □

Note that in general (e.g. in a Hilbert-style calculus), the induction axioms and rules are *not* equivalent. The difference here is that we have the *side formulas* Γ and Δ in our induction rules.

Definition:
(a) S_2^i is the sequent calculus system LKB + BASIC axioms + Σ_i^b-PIND rule.
(b) T_2^i is the sequent calculus system LKB + BASIC axioms + Σ_i^b-IND rule.

These are equivalent to the previous definitions by the above fact.

Definition: The following notions are defined just as with the sequent calculus for propositional logic: **successor** of an occurrence of a given formula, **descendant** (chain of successors), **ancestor** (opposite of descendant), **direct descendant** (descendant in which given occurrence of formula remains unchanged), **direct ancestor** (opposite of direct descendant), **principal formula** of an inference, and **side formula** of an inference.
Note that in an induction inference there are two principal formulas–namely $A(0)$ and $A(t)$.

Definition (Takeuti): A cut inference in a proof P is **free** *unless* one of the cut formulas is a direct descendant either of a principal formula of an induction inference or of a formula in an equality or non-logical axiom.

Our goal is to prove the following cut-elimination theorem...

*** **Theorem** (Gentzen): *Suppose P is an S_2^i- (or T_2^i-) proof of $\Gamma \longrightarrow \Delta$. Then there is a free-cut free S_2^i- (or T_2^i-) proof, P^*, of $\Gamma \longrightarrow \Delta$ such that every induction formula in P^* is a substitution instance of an induction formula in P.*

Reference: Takeuti, *Proof Theory.* Our proof follows Takeuti's proof almost directly.

We will say that A is a subformula of B **in the wide sense** iff A is a substitution instance of a subformula of B where we substitute for both free and bound variables. If P^* is a free-cut free proof of $\Gamma \longrightarrow \Delta$, then every formula in P^* is either: (a) a subformula in the wide sense of $\Gamma \longrightarrow \Delta$, or (b) a substitution instance of a principal formula of an induction inference in P^*, or (c) it is used in an equality or non-logical axiom. So the next three corollaries follow directly from the main theorem.

Corollary: If $\Gamma \longrightarrow \Delta$ has an LKB-proof, then $\Gamma \longrightarrow \Delta$ has an LKB-proof in which every cut formula is atomic (i.e., is the direct descendant of a formula used in an equality axiom).

Corollary: If $\Gamma \longrightarrow \Delta$ has an LKB-proof not involving any equality axioms, then $\Gamma \longrightarrow \Delta$ has a cut-free proof.

Corollary: If every formula in $\Gamma \longrightarrow \Delta$ is in $\Sigma_i^b \cup \Pi_i^b$ and if $S_2^i \vdash \Gamma \longrightarrow \Delta$, then there is an S_2^i–proof of $\Gamma \longrightarrow \Delta$ in which every formula is in $\Sigma_i^b \cup \Pi_i^b$

In order to simplify the proof of Gentzen's theorem, we introduce the following new rule of inference...

Mix Rule:

$$\frac{\Gamma \longrightarrow \Delta \qquad \Pi \longrightarrow \Lambda}{\Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}(A),$$

where A occurs in both Π and Δ, and Π^* and Δ^* are obtained by deleting every occurrence of A from Π and Δ, respectively.

By use of structural rules, it is fairly easy to see that the mix and cut rules are equivalent.

Proof: For cut$\Rightarrow$mix we have

$$\frac{\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \Pi^* \longrightarrow A, \Delta^*, \Lambda} \qquad \dfrac{\Pi \longrightarrow \Lambda}{A, \Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}}{\Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}(\text{cut on } A)$$

and for mix$\Rightarrow$cut we have

$$\dfrac{\dfrac{\Gamma \longrightarrow A, \Delta \qquad A, \Gamma \longrightarrow \Delta}{\Gamma, \Gamma^* \longrightarrow \Delta^*, \Delta}(\text{mix on } A).}{\Gamma \longrightarrow \Delta}$$

□

We say that a mix inference in a proof, P, is **free** *unless* one of the occurrences of A in Π or Δ is the direct descendant of a principal formula of an induction inference or of a formula in an equality or non-logical axiom. Note that the property of "free-ness" is not altered when we switch from mixes to cuts and vice-versa.

Using this equivalence between cuts and mixes, it is clear that the following lemma suffices to prove Gentzen's theorem by working our way down a proof of $\Gamma \longrightarrow \Delta$ from the "leaves", eliminating free cuts/mixes as we go.

Lemma: *If P is an S_2^i- (or T_2^i-) proof of $\Gamma \longrightarrow \Delta$ that has only one free mix inference which occurs as its final inference, then there is a free-mix free S_2^i- (or T_2^i-) proof, P^*, of $\Gamma \longrightarrow \Delta$ such that every induction formula in P^* is a substitution instance of an induction formula in P.*

Before we begin the proof of the lemma, a few definitions...

Definition: Suppose P is as in the lemma and that the final inference of P is

$$\frac{\Gamma \longrightarrow \Delta \qquad \Pi \longrightarrow \Lambda}{\Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}(A).$$

Then we define

(a) for any formula, B, *Grade(B)* is the number of logical symbols in B.

(b) $Grade(P) = Grade(A)$.

(c) the **right-age** of P is the maximum length of a path beginning with $\Pi \longrightarrow \Lambda$ and going up toward the axioms such that a direct ancestor of A in Π is in the antecedent of each sequent on the path.

(d) the **left-age** of P is defined similarly for $\Gamma \longrightarrow \Delta$ (looking at direct ancestors of A in Δ).

(e) $Age(P) = right\text{-}age(P) + left\text{-}age(P)$.

Since *right-age(P)* and *left-age(P)* are always ≥ 1, we have that $Age(P) \geq 2$. Also note that Takeuti uses the term "rank" instead of "age" in his proof of the lemma.

Proof of Lemma (by double induction on *Age(P)* and *Grade(P)*): We will give a purely syntactical, constructive approach, manipulating proofs instead of looking at models.

Case 1: $Age(P) = 2$

1.1 $\dfrac{A \longrightarrow A \qquad \Pi \longrightarrow \Lambda}{A, \Pi^* \longrightarrow \Lambda}(A)$ is the final inference of P.

Obtain P^* by changing this inference to $\dfrac{\Pi \longrightarrow \Lambda}{A, \Pi^* \longrightarrow \Lambda}$ by a structural rule.

1.2 Dual where $\Pi \longrightarrow \Lambda$ is $A \longrightarrow A$.

1.3 A is introduced into Δ by a structural rule...

$$\frac{\dfrac{\Gamma_1 \longrightarrow \Delta_1}{\Gamma \longrightarrow \Delta} \qquad \Pi \longrightarrow \Lambda}{\Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}(A),$$

where $\Gamma_1 \subseteq \Gamma$ as sets and $\Delta_1 \subseteq \Delta$ as sets with A not occuring in Δ_1.
Change this to the structural rule $\dfrac{\Gamma_1 \longrightarrow \Delta_1}{\Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}$ to obtain P^*.

1.4 Dual where A is introduced into Π by a structural rule.

1.5 A is introduced into Δ and into Π by a logical inference. The type of logical inference is determined by the outermost connective of A...

1.5(i) A is $B \vee C$
Then P ends with

$$\frac{\dfrac{\Gamma \longrightarrow B, \Delta_1}{\Gamma \longrightarrow B \vee C, \Delta_1} \qquad \dfrac{B, \Pi_1 \longrightarrow \Lambda \qquad C, \Pi_1 \longrightarrow \Lambda}{B \vee C, \Pi_1 \longrightarrow \Lambda}}{\Gamma, \Pi_1 \longrightarrow \Delta_1, \Lambda}(B \vee C).$$

Note that this is indeed a free mix. From this form a proof, P_1, which ends

$$\frac{\Gamma \longrightarrow B, \Delta_1 \qquad B, \Pi_1 \longrightarrow \Lambda}{\Gamma, \Pi_1^* \longrightarrow \Delta_1^*, \Lambda}(B),$$

where Π_1^* and Δ_1^* are obtained from Π_1 and Δ_1, resp., by deleting all occurrences of B.
If this is a free mix, since $Grade(P_1) < Grade(P)$, we use the induction hypothesis to get a free-mix free proof, P_1^*, of $\Gamma, \Pi_1^* \longrightarrow \Delta_1^*, \Lambda$. If it is not a free mix, set $P_1^* = P_1$. Then since P is free-mix free "above" $\Gamma \longrightarrow B, \Delta_1$ and $B, \Pi_1 \longrightarrow \Lambda$, we have that P_1^* is free-mix free in either case. Form P^* as

$$\frac{P_1^*}{\Gamma, \Pi_1 \longrightarrow \Delta_1, \Lambda}.$$

1.5(ii)-(iv) A is $\neg B$, $B \wedge C$, and $B \supset C$ are all handled similarly.

1.5(v) A is $\exists x \leq t\, B(x)$
Then P ends with

$$\dfrac{\dfrac{\Gamma \longrightarrow B(s), \Delta_1}{s \leq t, \Gamma \longrightarrow \exists x \leq t\, B(x), \Delta_1} \quad \dfrac{b \leq t, B(b), \Pi_1 \longrightarrow \Lambda}{\exists x \leq t\, B(x), \Pi_1 \longrightarrow \Lambda}}{s \leq t, \Gamma, \Pi_1 \longrightarrow \Delta_1, \Lambda}(\exists x \leq t\, B(x)),$$

where b is an eigenvariable which doesn't occur in the lower sequent and w.l.o.g. no variable in the term s is used as an eigenvariable in any inference of P (if so, simply change variable names).
So by substitution of s for b, we get a free-mix free proof of

$$s \leq t, B(s), \Pi_1 \longrightarrow \Lambda.$$

Now form P_1, which ends

$$\dfrac{\Gamma \longrightarrow B(s), \Delta_1 \qquad s \leq t, B(s), \Pi_1 \longrightarrow \Lambda}{\Gamma, s \leq t, \Pi_1^* \longrightarrow \Delta_1^*, \Lambda}(B(s)),$$

where Π_1^* and Δ_1^* are obtained by deleting occurrences of $B(s)$ (and w.l.o.g. $B(s) \neq s \leq t$).
If this is a free mix, since $Grade(P_1) < Grade(P)$, we may use the induction hypothesis to get a free-mix free proof, P_1^*, with the same endsequent. If it is not a free mix, set $P_1^* = P_1$, which again gives a free-mix free proof.
A structural inference gives us $P^* \ldots$

$$\dfrac{P_1^*}{s \leq t, \Gamma, \Pi_1 \longrightarrow \Delta_1, \Lambda}.$$

1.5(vi)-(viii) A is $\forall x \leq t\, B(x)$, $\forall x\, B(x)$, and $\exists x\, B(x)$ are all handled similarly.

Case 2: $Age(P) > 2$

2.1 $right\text{-}age(P) > 1$

__2.1.1__ A occurs in Γ or Λ

If A occurs in Γ, then change $\dfrac{\Gamma \longrightarrow \Delta \qquad \Pi \longrightarrow \Lambda}{\Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}(A)$ to the structural inference $\dfrac{\Pi \longrightarrow \Lambda}{\Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}$ to obtain P^*.
Similarly if A occurs in Λ.

__2.1.2__ Suppose $\Pi \longrightarrow \Lambda$ is inferred from $\Phi \longrightarrow \Psi$ and A is not the principal formula of this inference. Then P ends with

$$\dfrac{\Gamma \longrightarrow \Delta \qquad \dfrac{\Phi \longrightarrow \Psi}{\Pi \longrightarrow \Lambda}}{\Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}(A).$$

Form P_1 which ends with the mix $\dfrac{\Gamma \longrightarrow \Delta \qquad \Phi \longrightarrow \Psi}{\Gamma, \Phi^* \longrightarrow \Delta^*, \Psi}(A)$, where Φ^* is obtained by deleting all occurrences of A from Φ. Since *right-age*(P_1) = *right-age*$(P) - 1$ and since this must be a free mix, by the induction hypothesis there is a free-mix free P_1^* which has the same endsequent as P_1.
Now form P^* as

$$\dfrac{\dfrac{\dfrac{P_1^*}{\Phi^*, \Gamma \longrightarrow \Psi, \Delta_1}}{\Pi^*, \Gamma \longrightarrow \Lambda, \Delta_1}}{\Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}.$$

Here the first and third inferences are structural rules while the second is obtained from the original inference of $\Phi \longrightarrow \Psi$ to $\Pi \longrightarrow \Lambda$ (this is possible since A is not the principal formula of this inference).

__2.1.2'__ Suppose $\Pi \longrightarrow \Lambda$ is inferred from two sequents and A is not the principal formula of this inference. Then P ends with

$$\dfrac{\Gamma \longrightarrow \Delta \qquad \dfrac{\Phi_1 \longrightarrow \Psi_1 \qquad \Phi_2 \longrightarrow \Psi_2}{\Pi \longrightarrow \Lambda}}{\Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}(A).$$

This case is handled similarly to the previous one.

2.1.3 Γ doesn't contain A and A is the principal formula of the inference producing $\Pi \longrightarrow \Lambda$. Then since *right-age(P)* > 1, there are other occurrences of A in Π.

2.1.3(i) A is $B \vee C$
Then P ends with

$$\dfrac{\Gamma \longrightarrow \Delta \qquad \dfrac{B, \Pi_1 \longrightarrow \Lambda \qquad C, \Pi_1 \longrightarrow \Lambda}{B \vee C, \Pi_1 \longrightarrow \Lambda}}{\Gamma, \Pi_1^* \longrightarrow \Delta^*, \Lambda}(B \vee C).$$

Form P_1 and P_2 which end respectively with the free mixes

$$\dfrac{\Gamma \longrightarrow \Delta \qquad B, \Pi_1 \longrightarrow \Lambda}{\Gamma, B, \Pi_1^* \longrightarrow \Delta^*, \Lambda}(B \vee C)$$

and

$$\dfrac{\Gamma \longrightarrow \Delta \qquad C, \Pi_1 \longrightarrow \Lambda}{\Gamma, C, \Pi_1^* \longrightarrow \Delta^*, \Lambda}(B \vee C).$$

Since the right-age of both of these inferences equals *right-age(P)* -1, by the induction hypothesis there are free-mix free proofs, P_1^* and P_2^* with the same endsequents. Form P_3 as

$$\dfrac{\Gamma \longrightarrow \Delta \qquad \dfrac{\dfrac{P_1^*}{B, \Gamma, \Pi_1^* \longrightarrow \Delta^*, \Lambda} \qquad \dfrac{P_2^*}{C, \Gamma, \Pi_1^* \longrightarrow \Delta^*, \Lambda}}{B \vee C, \Gamma, \Pi_1^* \longrightarrow \Delta^*, \Lambda}}{\Gamma, \Gamma, \Pi_1^* \longrightarrow \Delta^*, \Delta^*, \Lambda}(B \vee C).$$

Now *right-age*$(P_3) = 1$ and *left-age*$(P_3) =$ *left-age(P)*. So $Age(P_3) < Age(P)$ and we use the induction hypothesis to get free-mix free P_3^* with the same endsequent as P_3.
Finally, by a structural inference, form P^* as

$$\dfrac{P_3^*}{\Gamma, \Pi_1^* \longrightarrow \Delta^*, \Lambda}.$$

2.1.3(ii)-(iv) A is $\neg B$, $B \wedge C$, and $B \supset C$ are all handled similarly.

2.1.3(v) A is $\exists x \leq t\, B(x)$
Then P ends with

$$\cfrac{\Gamma \longrightarrow \Delta \qquad \cfrac{b \leq t, B(b), \Pi_1 \longrightarrow \Lambda}{\exists x \leq t\, B(x), \Pi_1 \longrightarrow \Lambda}}{\Gamma, \Pi_1^* \longrightarrow \Delta^*, \Lambda}(\exists x \leq t\, B(x)),$$

where b is an eigenvariable and w.l.o.g. b doesn't occur in $\Gamma \longrightarrow \Delta$ (otherwise replace b everywhere on the right side of P with a new variable).
Form P_1 ending with the free mix

$$\frac{\Gamma \longrightarrow \Delta \qquad b \leq t, B(b), \Pi_1 \longrightarrow \Lambda}{\Gamma, b \leq t, B(b), \Pi_1^* \longrightarrow \Delta^*, \Lambda}(\exists x \leq t\, B(x)).$$

As before, we use the induction hypothesis to get a free-mix free P_1^* with the same endsequent. Now, using the fact that b is not in Γ or Δ, form P_2 as

$$\cfrac{\Gamma \longrightarrow \Delta \qquad \cfrac{\cfrac{P_1^*}{b \leq t, B(b), \Gamma, \Pi_1^* \longrightarrow \Delta^*, \Lambda}}{\exists x \leq t\, B(x), \Gamma, \Pi_1^* \longrightarrow \Delta^*, \Lambda}}{\Gamma, \Gamma, \Pi_1^* \longrightarrow \Delta^*, \Delta^*, \Lambda}(\exists x \leq t\, B(x)).$$

Once again, since *right-age*$(P_2) = 1$, we can use the induction hypothesis to get a free-mix free P_2^* with the same endsequent and a final structural inference gives us P^* ...

$$\frac{P_2^*}{\Gamma, \Pi_1^* \longrightarrow \Delta^*, \Lambda}.$$

2.1.3(vi)-(viii) A is $\forall x \leq t\, B(x)$, $\forall x\, B(x)$, and $\exists x\, B(x)$ are all handled similarly.

2.2 *left-age(P)* > 1

This is the exact dual of case 2.1.

Notice that throughout the proof, we have left out the statement "every induction formula in P^* is a substitution instance of an induction formula in P". It is easy to check that this is indeed the case, since the induction hypothesis always includes this statement for P_1^*, P_2^*, etc. and the "worst" we changed any individual formula was possibly to substitute one variable for another.

This completes the proof of the lemma. $\square$

Note again that the proof actually gives a constructive method for eliminating free mixes from a given proof. Of course, even a non-constructive proof would give *an* algorithm, although it would not be very feasible, being of the form "enumerate all proofs until you find a free-mix free one."

Carefully watching the sizes of the proofs formed in each step of the lemma shows that we have

$$|P^*| \leq \left. 2^{2^{\cdot^{\cdot^{\cdot^{2^{|P|}}}}}} \right\} \mathcal{O}(|\, P \,|),$$

although Buss is fairly sure that we actually have

$$|P^*| \leq \left. 2^{2^{\cdot^{\cdot^{\cdot^{2^{|P|}}}}}} \right\} \mathcal{O}(q),$$

where q is the maximum number of quantifiers in a formula of P.

Compare this with the propositional sequent calculus, where we had $|P^*| \leq 2^{\mathcal{O}(|P|)} \cdot |\, P \,|$. In fact, this could be extended to $|P^*| \leq 2^{\mathcal{O}(|\Gamma \longrightarrow \Delta|)} \cdot |\, \Gamma \longrightarrow \Delta \,|$, where $\Gamma \longrightarrow \Delta$ is the sequent being proved, and this was more-or-less optimal for tree-like proofs.

Statman [*Annals of Mathematical Logic* 15 (1978)] first showed that the upper bound given above is fairly good for tree-like free-mix free LKB-proofs, while Pudlák ["Cuts, Consistency Statements, and Interpretations," *JSL* 50

(1985)] gave another, high-level, proof that super-exponentiation is necessary. Although Pudlák discussed Herbrand consistency, his results apply to free-cut free proofs as well.

Math 271 - Topics in Weak Formal Systems

Lecture Notes, Set #10
Apr. 28 - May 4, 1988

Instructor: Sam Buss
Notes By: Alessandro Berarducci

References: The main theorem proved in these notes is essentially due to S.Cook, 'Feasibly constructive prrofs and the propositional calculus', 7 th ACM Symp. on Theory of Computing, 1975. An extension of Cook's work is due to J. Krajíček and Pudlak, 'quantified propositional calculus and fragments of bounded arithmetic', preprint, Jan. 88.

Goal: We will translate arguments formalizable in S^1_2 into arguments that can be formalized by polynomial-size extended Frege proofs.

Example: As an intuitive motivating example we consider counting. We know that in a Frege system we can count the number of true x's among $x_0, x_1, \ldots, x_{n-1}$ by using carry save addition (as in the $\mathcal{F}$- proof of PHP). In S^1_2 we can count by using the function *Numones(x)* which gives the number of 1's in the binary representation of x. There is no known way of directly mimicing the S^1_2 construction in a Frege system, hovever this can be done in an extended Frege system as follows: we will introduce new variables $p_{i,j}$ by the extension rule in such a way that $\vec{p_i}$ will code an integer a_i giving the number of true x_k's among $x_0, x_1, \ldots, x_{n-1}$ with $k \leq i$. So $\vec{p_0}$ will code either 0 or 1 according to whether x_0 is true or false and is defined by:

1. $p_{0,0} \leftrightarrow x_0$
2. $p_{0,j+1} \leftrightarrow p \wedge \neg p$

$\vec{p_{i+1}}$ will either add 1 or 0 according to whether x_{i+1} is true or false:

$$p_{i+1,j} \quad \leftrightarrow \quad \begin{array}{ll} p_{i,j} & \text{if} \quad (\neg x_{i+1}) \text{ or } (x_{i+1} \wedge \bigvee_{k<j} \neg p_{i,k}) \\ \neg p_{i,j} & \text{if} \quad x_{i+1} \wedge \bigwedge_{k<j} p_{i,k} \end{array}$$

The second clause expresses the fact that there is a carry in position j when 1 is added. Now $\vec{p_{n-1}}$ will code the number of true x_k's among $x_1, x_2, \ldots x_{n-1}$ and moreover there are polynomial-size $e\mathcal{F}$-proofs that this definition is equivalent to the carry save addition definition.

To carry out our goal we will define a map which translates a Π_2^b-formula into a family of propositional formulas (which have polynomial size with respect to the length of the integers being substituted for the free variables of the formula). We need some preliminary definitions.

Definition: Let t be a term of S_2^1. The bounding polynomial $q_t(n)$ of t is defined inductively by:

1. $q_0(n) = 1$
2. $q_a(n) = n$ for any variable a.
3. $q_{s(t)}(n) = q_t(n) + 1$ where s is the successor function.
4. $q_{s+t}(n) = q_s(n) + q_t(n)$
5. $q_{s \cdot t}(n) = q_s(n) + q_t(n)$
6. $q_{s \# t}(n) = q_s(n) \cdot q_t(n) + 1$
7. $q_{|t|}(n) = q_{\lfloor \frac{1}{2} t \rfloor}(n) = q_t(n)$

Proposition: If $t(a_1, \ldots, a_k)$ is a term and $x_1, \ldots, x_k$ are natural numbers of length $\leq n$, then $|t(\vec{x})| \leq q_t(n)$ (here $|t(\vec{x})|$ denotes the length of the binary representation of the value of $t(\vec{x})$).

Proof: By construction.

Definition: Let A be a bounded formula of S^1_2. The bounding polynomial q_A of A is inductively defined by:

1. $q_{s=t} = q_{s \leq t} = q_s + q_t$
2. $q_{A \wedge B} = q_{A \vee B} = q_{A \supset B} = q_A + q_B$
3. $q_{\neg A} = q_A$
4. $q_{(\exists x \leq t) A}(n) = q_t(n) + q_A(n + q_t(n)) = q_{\forall x \leq t A}(n)$

Proposition: The formula $A(x_1, \ldots, x_k)$ where $|x_i| \leq n$, only refers to numbers of length $\leq q_A(n)$.

Proof: By construction.

Note that there are fan-out 1 boolean circuits for computing the function symbols of the language of S^1_2. For example there is a circuit $\llbracket + \rrbracket_m$ which takes $2 \cdot m$ inputs (the code of two integers in binary) and outputs m binary symbols (their sum; any overflow is lost). Similarly we define the polynomial-size (in m) family of circuits $\llbracket 0 \rrbracket_m$, $\llbracket s \rrbracket_m$, $\llbracket \lfloor \frac{1}{2} \cdot \rfloor \rrbracket_m$, $\llbracket | \cdot | \rrbracket_m$, $\llbracket \# \rrbracket_m$, $\llbracket \cdot \rrbracket_m$. For $\llbracket \cdot \rrbracket_m$ we use carry-save addition. It is important that these circuits have fan-out 1 because we want to translate them into boolean formulas.

We are now ready to define, for each term t, a vector of m propositional formulas $\llbracket t \rrbracket^n_m$ giving the first m bits of the value of t when its free variables are assigned values of length $\leq n$ (in general we will assume that $m \geq q_t(n)$).

For each free variable a in t, the formulas $[\![t]\!]^n_m$ will have a sequence of n propositional variables $v^a_{n-1}, \ldots, v^a_0$ representing the value of a (an integer $< 2^n$). If m is bigger than the length of the value of t we expand the value of t with a sequence of leading 0's (represented by the truth-value false).

Definition:

1. $[\![0]\!]^n_m$ is a sequence of m false formulas (for eample $p \wedge \neg p$).
2. If a is a variable, $[\![a]\!]^n_m$ is a sequence of $m - n$ false formulas followed by $v^a_{n-1}, \ldots, v^a_0$.
3. $[\![s + t]\!]^n_m$ is $[\![+]\!]_m([\![s]\!]^n_m, [\![t]\!]^n_m)$ (the formulas corresponding to the circuit for addition applied to the output of $[\![s]\!]^n_m$ and $[\![t]\!]^n_m$).
4. Similar definitions work for the remaining cases.

Remark: Since a formula is essentially a circuit with fan-out 1, for each bit of $s + t$ we need to compute the values of s and t again. Also note that for fixed t, the size of $[\![t]\!]^n_m$ is polynomial in m.

The next goal is to define, given a Π^b_2-formula A, a propositional formula $[\![A]\!]^n_m$ (where $m \geq q_A(n)$). To do this we proceed as follows:

Definition: Given a formula B we assign to B special 'existential' propositional variables $\varepsilon^B_0, \varepsilon^B_1, \varepsilon^B_2, \ldots$, and special 'universal' propositional variables $\mu^B_0, \mu^B_1, \mu^B_2, \ldots$ (all new and distinct), with the convention that we will assign different sequences of existential variables to distinct occurrences of B but all occurrences of B use the same universal variables. From the context it will be clear which occurrences we are referring to.

Definition: A first order formula is in negation-implication normal form ($NINF$) if every negation is applied to an atomic subformula and there are no implications. For $A \in \Pi_2^b$ in $NINF$ and $m \geq q_A(n)$, we define the propositional formula $[\![A]\!]_m^n$ inductively as follows:

1. $[\![s = t]\!]_m^n$ is $EQ_{m-1}([\![s]\!]_m^n, [\![t]\!]_m^n)$ ($EQ_{m-1}(\vec{p}, \vec{q})$ has been defined in a previous lecture as $\bigwedge_{k=0}^{m-1}(p_k \leftrightarrow q_k)$).
2. $[\![s \leq t]\!]_m^n$ is $LE_{m-1}([\![s]\!]_m^n, [\![t]\!]_m^n)$
3. $[\![\neg A]\!]_m^n$ is $\neg[\![A]\!]_m^n$ for A atomic.
4. $[\![A \wedge B]\!]_m^n$ is $[\![A]\!]_m^n \wedge [\![B]\!]_m^n$
5. $[\![A \vee B]\!]_m^n$ is $[\![A]\!]_m^n \vee [\![B]\!]_m^n$
6. $[\![(\exists x \leq t)A(x)]\!]_m^n$ is $[\![b \leq t \wedge A(b)]\!]_m^n(\{\varepsilon_i^A/v_i^b\}_{i=0}^{n-1})$ where t is not of the form $|s|$ and b is a new free variable not appearing in $A(x)$.
7. $[\![(\forall x \leq t)A(x)]\!]_m^n$ is $[\![\neg b \leq t \vee A(b)]\!]_m^n(\{\mu_i^A/v_i^b\}_{i=0}^{n-1})$ where t is not of the form $|s|$ and b is a new free variable not appearing in $A(x)$.
8. $[\![(\forall x \leq |t|)A(x)]\!]_m^n$ is $\bigwedge_{k=0}^{m-1}[\![\neg \underline{k} \leq |t| \vee A(\underline{k})]\!]_m^n$ where $\underline{k}$ is a term with value k and length $\simeq \log k$ (use binary representations, for example $\underline{7}$ is $1 + 2 \cdot (0 + 2)$). Note that $|t| \leq m$ (by our assumption on m).
9. $[\![(\exists x \leq |t|)A(x)]\!]_m^n$ is $\bigvee_{k=0}^{m-1}[\![\underline{k} \leq |t| \wedge A(\underline{k})]\!]_m^n$

Proposition: For fixed $A \in \Pi_2^b$ the propositional formula $[\![A]\!]_m^n$ is polynomial-size in m (hence in m, n since $m \geq q_A(n)$). Moreover $[\![A]\!]_m^n$ expresses 'A is true' in the sense that if A is true for every assignment of its free variables to numbers of length $\leq n$, then for any asssignment of truth values to the universal variables and to the v_i^b's, there is a truth assignment to the existential variables that makes $[\![A]\!]_m^n$ true.

Definition We extend the map sending A to $[\![A]\!]_m^n$ to every formula in Π_2^b by first putting A in $NINF$ and then applying the previously defined map.

Definition: Given a propositional formula ψ with universal and existential variables, an $e\mathcal{F}$-proof of ψ is similar to an ordinary $e\mathcal{F}$-proof of ψ (as defined in a previous lecture), except that we allow the existential variables of ψ (but not its other variables) to be defined by an extension inference. More precisely a valid extension inference is of the form $p \leftrightarrow \varphi$ where: (a) p does not occur in φ and p has not occurred earlier in the proof; and (b) p does not occur in ψ or p is an existential variable in ψ.

Theorem: If $A(\vec{x}) \in \Pi_2^b$ and $S_2^1 \vdash \forall\vec{x}A(\vec{x})$, then there are polynomial-size (in n) $e\mathcal{F}$-proofs of $[\![A]\!]_{q(n)}^n$ where $q(n)$ is any polynomial $\geq q_A(n)$.

Remark: The degree of the polynomial bounding the size of the $e\mathcal{F}$-proof can be superexponential as a function of the size of the S_2^1-proof.

proof: We can assume without loss of generality that A is in $NINF$. Since $S_2^1 \vdash A(\vec{c})$ there is an S_2^1-proof of $A(\vec{c})$ in which every induction formula is in $NINF$ and in Σ_1^b. By the free cut free elimination theorem, there is a proof P in which every sequent has the form $A_1, \ldots, A_k \longrightarrow B_1, \ldots, B_l$ with $A_i \in \Sigma_1^b$, $B_j \in \Pi_2^b$ and each A_i, B_j in $NINF$. We will show by induction on the number of lines of P that for each sequent as above there are polynomial-size $e\mathcal{F}$-proofs of $[\![\neg A_1 \vee \cdots \vee \neg A_k \vee B_1 \cdots \vee B_l]\!]_{q(n)}^n$

Base case:

Logical axioms: The translation of a logical axiom $B \longrightarrow B$ (B atomic) is the propositional formula $[\neg B] \vee [B]$ which clearly has polynomial-size $e\mathcal{F}$-proofs (we omit superscripts and subscripts).

Equality axioms: Easy. For example $[\neg a = b \vee \neg c = d \vee a + c = b + d]\!]$ has short $e\mathcal{F}$-proofs.

Basic axioms of S_2^1: Consider for example $[\![(x+y)+z = x+(y+z)]\!]_{q(n)}^n$. We already claimed in a previous lecture that this formula has polynomial-size $\mathcal{F}$-proofs when we gave $\mathcal{F}$-proofs of PHP. The other cases are similar.

Induction step:

Case 1: $(\vee - right)$. Suppose P ends with

$$\frac{\Gamma \longrightarrow B, \Delta}{\Gamma \longrightarrow B \vee C, \Delta}$$

For notational convenience we will assume in the rest of the proof that Γ and Δ are single formulas.

1. $[\![\neg\Gamma \vee B \vee \Delta]\!]$ is $[\![\neg\Gamma]\!] \vee [\![B]\!] \vee [\![\Delta]\!]$
2. $[\![\neg\Gamma \vee (B \vee C) \vee \Delta]\!]$ is $[\![\neg\Gamma]\!] \vee [\![B]\!] \vee [\![C]\!] \vee [\![\Delta]\!]$

1) $\supset$ 2) has a simple $\mathcal{F}$-proof. Now the thesis follows easily from the induction hypothesis.

Case 2: $(\wedge - right)$.Similar.

Case 3: (Structural rule). A structural rule can be either a weakening inference or an exhange inference or a contraction:

3.1: (Weakening).

$$\frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow B, \Delta} \qquad \text{or} \qquad \frac{\Gamma \longrightarrow \Delta}{B, \Gamma \longrightarrow \Delta}$$

3.2: (Exchange).

$$\frac{\Gamma \longrightarrow \Delta_1, B, C, \Delta_2}{\Gamma \longrightarrow \Delta_1, C, B, \Delta_2}$$

3.3: (Contraction).

$$\frac{\Gamma \longrightarrow B, B, \Delta}{\Gamma \longrightarrow B, \Delta}$$

Weakening and exchange are easily handled by the methods of cases 1 and 2. The problem with the contraction rule is that different occurrences of B have different existential variables (but recall that all B's have the same universal variables). Suppose that the first B has existential variables $\varepsilon_1, \varepsilon_2, \ldots$, the second B has existential variables $\varepsilon'_1, \varepsilon'_2, \ldots$, and the B in the lower sequent has $\varepsilon''_1, \varepsilon''_2, \ldots$. By induction hypothesis there are polynomial-size $e\mathcal{F}$-proofs P_n of $[\![\neg\Gamma \vee B \vee B \vee \Delta]\!]^n_{q(n)}$. We form $e\mathcal{F}$-proofs Q_n by concatenating the following:

1. The proof P_n, followed by:
2. The definition of ε''_j by the following extension rule:
$$\varepsilon''_j \leftrightarrow ([\![B]\!](\vec{\varepsilon}) \wedge \varepsilon_j) \vee (\neg[\![B]\!](\vec{\varepsilon}) \wedge \varepsilon'_j)$$
3. A proof of $[\![\neg\Gamma]\!] \vee [\![B]\!](\vec{\varepsilon''}) \vee [\![\Delta]\!]$ from $[\![\neg\Gamma]\!] \vee [\![B]\!](\vec{\varepsilon}) \vee [\![B]\!](\vec{\varepsilon'}) \vee [\![\Delta]\!]$.

Case 4: ($\wedge$-right). P ends with

$$\frac{\Gamma \longrightarrow B, \Delta \qquad \Gamma \longrightarrow C, \Delta}{\Gamma \longrightarrow B \wedge C, \Delta}$$

We separate this inference into two steps:

$$\cfrac{\cfrac{\Gamma \longrightarrow B, \Delta \qquad \Gamma \longrightarrow C, \Delta}{\Gamma \longrightarrow B \wedge C, \Delta, \Delta}}{\Gamma \longrightarrow B \wedge C, \Delta}$$

The first inference is not included in our original definition of the sequent calculus but it is a valid derived rule and is easily handled by the method of cases 1 and 2. The second inference is a structural rule and is already handled by case 3. Note that we do not need two copies of Γ since $[\![\neg\Gamma]\!]$ has no existential variables so the two $[\![\neg\Gamma]\!]$'s are identical.

Case 5: ($\vee$-left). Similar to case 4.

Case 6: (Cut). P ends with

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

Note that B must be Σ_1^b. By induction hypothesis there are polynomial-size proofs P_n and R_n of $[\![\neg\Gamma \vee B \vee \Delta]\!]^n_{q(n)}$ and of $[\![\neg\Gamma \vee \neg B \vee \Delta]\!]^n_{q(n)}$. Now suppose that

- $[\![B]\!]$ has existential variables $\varepsilon_1, \varepsilon_2, \ldots$
- $[\![\neg B]\!]$ has universal variables $\mu_1, \mu_2, \ldots$
- The first $[\![\Delta]\!]$ has existential variables $\varepsilon'_1, \varepsilon'_2, \ldots$
- The second uses $\varepsilon''_1, \varepsilon''_2, \ldots$

We construct a proof Q_n of $[\![\Gamma \longrightarrow \Delta]\!]^n_{q(n)}$ by combining the following:

1. The proof P_n of $[\![\neg\Gamma]\!] \vee [\![B]\!](\vec{\varepsilon}) \vee [\![\Delta]\!](\vec{\varepsilon'})$
2. The proof R_n with each μ_i in R_n changed to ε_i. This gives a proof of $[\![\neg\Gamma]\!] \vee [\![\neg B]\!](\vec{\varepsilon}/\vec{\mu}) \vee [\![\Delta]\!](\varepsilon'')$
3. A proof of $[\![\neg\Gamma]\!] \vee [\![\Delta]\!](\vec{\varepsilon'}) \vee [\![\Delta]\!](\vec{\varepsilon''})$, obtained from the previous two proofs and tautological inference.
4. A proof of $[\![\neg\Gamma \vee \Delta]\!]$ obtained using the method of case 3 for the structural rule.

Case 7: $(\Sigma_1^b - PIND)$. P ends with

$$\frac{A(\lfloor \frac{1}{2} b \rfloor), \Gamma \longrightarrow A(b), \Delta}{A(0), \Gamma \longrightarrow A(t), \Delta} \quad \text{where b is the eigenvariable.}$$

Let $[\![t]\!]^n_{q(n)}$ be $\{\varphi_i^t | i < q(n)\}$, that is φ_i^t is the formula giving the i^{th} bit of t. Let v_i^b be variables specifying the value of b. Let $P_{n,k}$ be the proof of

$$[\![A(\lfloor\frac{t}{2^{k+1}}\rfloor) \wedge \Gamma \supset A(\lfloor\frac{t}{2^k}\rfloor) \vee \Delta]\!]$$

obtained by substituting the formulas φ^t_{i-k} for the variables v^b_i where φ^t_{i-k} is $p \wedge \neg p$ if $k > i$. Now put all the proofs $P_{n,k}$ together, for $k = q(n), \ldots, 1, 0$, using the method of case 6 for the cut inference. The resulting proof will be a proof of the translation of the lower sequent whose size is still polynomial in n. Note that this case depends heavily on the extension rule.

Case 8: ($\exists \leq$-right).

Case 8.a: (*sharply bounded*). P ends with

$$\frac{\Gamma \longrightarrow B(s), \Delta}{s \leq |t|, \Gamma \longrightarrow \exists x \leq |t| B(x), \Delta}$$

By induction hypothesis there are polynomial-size proofs P_n of $[\![\neg\Gamma \vee B(s) \vee \Delta]\!]^n_{q(n)}$. Recall that $[\![(\exists x \leq |t|)B(x)]\!]^n_{q(n)}$ is $\bigvee_{k=0}^{q(n)-1} [\![\underline{k} \leq |t| \wedge B(\underline{k})]\!]^n_{q(n)}$. Now there are simple $e\mathcal{F}$-proofs of $[\![s \leq |t| \supset s = \underline{0} \vee s = \underline{1} \vee \ldots \vee s = \underline{q(n)-1}]\!]$ and for each $k \leq q(n) - 1$ there are $e\mathcal{F}$-proofs $P_{n,k}$ of

$$[\![s = \underline{k}]\!] \supset ([\![B(s)]\!](\vec{\mu}^k, \vec{\varepsilon}^k / \vec{\mu}, \vec{\varepsilon}) \leftrightarrow [\![B(\underline{k})]\!])$$

where the $\vec{\varepsilon}, \vec{\mu}$ are the existential and universal variables of $B(s)$ and the $\vec{\varepsilon}^k, \vec{\mu}^k$ are the existential and universal variables of $B(\underline{k})$. Now we use tautological implication to combine the proofs $P_{n,k}$ to get a proof of

$$[\![\neg s \leq |t| \vee \neg\Gamma]\!] \vee \bigvee_{k=0}^{q(n)-1} [\![\underline{k} \leq |t| \wedge B(\underline{k})]\!] \vee [\![\Delta \vee \Delta \vee \ldots \vee \Delta]\!]$$

Finally we use the method of case 3 (structural rule) to get a polynomial-size $e\mathcal{F}$-proof of

$$[\![\neg s \leq |t| \vee \neg\Gamma \vee (\exists x \leq |t|)B(x) \vee \Delta]\!]$$

Case 8.b: (*not sharply bounded*). P ends with

$$\frac{\Gamma \longrightarrow B(s), \Delta}{s \le t, \Gamma \longrightarrow (\exists x \le t)B(x), \Delta}$$

By induction hypothesis there are polynomial-size $e\mathcal{F}$-proofs P_n of

$$[\![\neg\Gamma \vee B(s) \vee \Delta]\!]^n_{q(n)}$$

Let $\varepsilon^B_1, \varepsilon^B_2, \ldots$ be the existential variables for the outer quantifier in $(\exists x \le t)B(x)$. Let φ^s_i be the formula giving the i^{th}-bit of s. We form the desired $e\mathcal{F}$-proof Q_n by combining the following:

1. The definition of $\varepsilon^B_i \leftrightarrow \varphi^s_i$ via extension.
2. The proof P_n
3. A derivation of $[\![\neg s \le t \vee \neg\Gamma \vee (s \le t \wedge B(s)) \vee \Delta]\!]$
4. A derivation of $[\![\neg s \le t \vee \neg\Gamma \vee (\exists x \le t)B(x) \vee \Delta]\!]$ obtained by changing some of the φ^s_i to ε^B_i.

Case 9: ($\forall \le$-left). Similar to case 8.

Case 10: ($\forall \le$-right)

Case 10a: (*Sharply bounded*). Suppose P ends with the inference

$$\frac{a \le |t|, \Gamma \longrightarrow B(a), \Delta}{\Gamma \longrightarrow (\forall x \le |t|)B(x), \Delta}$$

The free variable a is the *eigenvariable* and appears only as indicated. The induction hypthesis states that there are polynomial size $e\mathcal{F}$-proofs P_n of $[\![a \le |t| \wedge \Gamma \supset B(a) \vee \Delta]\!]^n_{q(n)}$. Recall that v^a_i is the propositional variable giving the i-th bit of the value of a. For $k \in \mathbb{N}$, let $\phi^{\underline{k}}_i$ be the formula specifying the i-th bit of the integer k; $\phi_i^{\underline{k}}$ is the formula given in the definition of $[\![\underline{k}]\!]$. For each $k \leq n$, let $P_{n,k}$ be the $e\mathcal{F}$-proof obtained from P_n by replacing each v_i^a with the formula $\phi_i^{\underline{k}}$. So $P_{n,k}$ is an $e\mathcal{F}$-proof of $[\![\underline{k} \leq |t| \supset B(\underline{k}) \vee \Delta]\!]$. Use tautological implication to combine the $P_{n,k}$'s to get an $e\mathcal{F}$-proof of

$$[\![\neg\Gamma]\!] \vee \bigwedge_{k=0}^{q(n)-1} [\![\underline{k} \leq |t| \supset B(\underline{k})]\!] \vee [\![\Delta \vee \cdots \vee \Delta]\!]$$

and as usual, use the method of Case 3, structural inference, to contract the multiple Δ's and thereby get an $e\mathcal{F}$-proof of $[\![\neg\Gamma \vee (\forall x \leq |t|)B(x) \vee \Delta]\!]$.

Case 10b: (*Nonsharply bounded*). Suppose P ends with the inference

$$\frac{a \leq t, \Gamma \longrightarrow B(a), \Delta}{\Gamma \longrightarrow (\forall x \leq t)B(x), \Delta}$$

where now t is not of the form $|s|$ for any term s. The induction hypothesis gives polynomial size, $e\mathcal{F}$-proofs P_n of $[\![a \leq t \wedge \Gamma \supset B(a) \vee \Delta]\!]^n_{q(n)}$. Recall that μ_i^B are the universal variables used to represent the value of x in $[\![(\forall x \leq t)B(x)]\!]$. We can transform P_n by replacing the propositional variables v_i^a with the universal variables μ_i^B and using a simple tautological implication to get an $e\mathcal{F}$-proof of $[\![\neg\Gamma \vee (\forall x \leq t)B(x) \vee \Delta]\!]$.

Case 11: ($\exists \leq$: left). This case is dual to Case (10) and is proved in exactly the same manner.

Q.E.D.

Math 271 - Topics in Weak Formal Systems

Lecture Notes, Set #11 **Instructor: Sam Buss**
May 6–9, 1988 **Notes By: Sam Buss**

As an application of the theorem on page 6 of the previous set of notes, we will prove a theorem of S. Cook which shows that the extended Frege proof systems are the strongest proof systems which can be proved consistent by S^1_2.

For the rest of the notes, let $e\mathcal{F}_0$ be an fixed extended Frege proof system with language $\{\neg, \vee, \wedge\}$. Earlier in the course we showed that any two extended Frege proof systems in the same language p-simulate each other, so the choice of axiom schemas for $e\mathcal{F}_0$ is not important. Every formula $[\![A]\!]$ is a formula in the language of $e\mathcal{F}_0$ and the theorem from the previous lecture notes applies to $e\mathcal{F}_0$.

Theorem 2 below states that if $\mathcal{G}$ is a propositional proof system such that S^1_2 proves that every consequence of $\mathcal{G}$ is a tautology, then $e\mathcal{F}_0$ p-simulates $\mathcal{G}$. Before we can state and prove this precisely, we need to see that S^1_2 can define metamathematical concepts such as "propositional formula" and "truth assignment":

Definition: We assume there is some natural way to assign Gödel numbers to propositional formulas and to truth assignments. We write $\ulcorner\phi\urcorner$ to denote the Gödel number of ϕ. For example: assign small Gödel numbers to logical symbols $\neg$, $\vee$ and $\wedge$ and to parentheses and assign $100+i$ to the variable p_i. If A is $\alpha_1\cdots\alpha_n$ then the Gödel number $\ulcorner A\urcorner = \langle\ulcorner\alpha_1\urcorner,\ldots,\ulcorner\alpha_n\urcorner\rangle$.

Truth assignments τ will be coded in an unusual but compact way. If τ is a truth assignment and A is a formula then the Gödel number of τ for evaluation of A is the integer $\ulcorner\tau\urcorner$ such that the i-th bit of its binary representation is equal to 1 if and only if the i-th propositional variable appearing in A is assigned "True" by τ. Note that the Gödel number of τ depends on the formula being evaluated; or conversely, a given integer can represent more than one truth assignment, depending in the formula being evaluated.

The formula $True(\ulcorner A\urcorner, \ulcorner\tau\urcorner)$ is a Δ_1^b formula of S_2^1 which asserts "$\ulcorner A\urcorner$ is a Gödel number of a propositional formula, $\ulcorner\tau\urcorner$ is the Gödel number of a truth assignment and $\overline{\tau}(A) = \top$." (We use $\top$ and $\bot$ as symbols for truth and absurdity, respectively.)

The property of m being the Gödel number of a propositional tautology is defined in S_2^1 by the following formula:

$$Taut(m) \Leftrightarrow (\forall \ulcorner\tau\urcorner \leq m)(True(m, \ulcorner\tau\urcorner)).$$

Note that because of our efficient way of coding truth assignments, the Gödel number of a truth assignment for A is always less than the Gödel number of A. Thus, since *True* is a Δ_1^b formula, *Taut* is a Π_1^b formula.

Lemma 1 *There is a polynomial $p(n)$ such that the following holds. If A is a propositional formula with variables $p_{i_1}, \ldots, p_{i_k}$ (in increasing order of subscript), then there is an $e\mathcal{F}_0$ proof of*

$$[\![True(\ulcorner A\urcorner, \ulcorner\tau\urcorner)]\!]_{q(|A|)}^{|A|} \;\leftrightarrow\; A(v_j^\tau / x_{i_j})$$

of size less than $p(|A|)$ where q is an appropriate bounding polynomial. The substitution (v_j^τ / x_{i_j}) substitutes the truth value $\tau(x_{i_j})$ which is given by the j-th bit of τ for the variable x_{i_j} in A (this is done for each j).

The import of Lemma 1 is that $e\mathcal{F}_0$ can give polynomial size proofs of the fact that (the translation of) the Δ_1^b formula *True* correctly defines the truth value of a given propositional formula. We shall not prove this lemma here; however, the essential idea is similar to the proof of the theorem on page 6 of the previous set of notes. Namely, the S_2^1 definition of *True* when translated into propositional form $[\![True]\!]$ can be effectively reasoned with by the extended Frege proof system. To properly prove Lemma 1, we would introduce a new Δ_1^b formula $\mathit{TrSbFmla}(\ulcorner A \urcorner, \ulcorner \tau \urcorner, i)$ which asserts that if τ is a truth assignment to the variables of the propositional formula A and if B is the subformula of A whose principle logical connective is the i-th symbol in A then $\overline{\tau}(B) = \top$. The extended Frege system $e\mathcal{F}_0$ can prove, successively for larger and larger subformulas B with principle connective the i_B-th symbol of A, that

$$[\![\mathit{TrSbFmla}(\ulcorner A \urcorner, \ulcorner \tau \urcorner, i_B)]\!] \leftrightarrow B(v_j^\tau / x_{i_j})$$

Theorem 2 *(Essentially due to Cook [STOC 1975].) Let $\mathcal{G}$ be a propositional proof system with language $\{\neg, \vee, \wedge\}$ and let G be a Σ_1^b-definition of $\mathcal{G}$. If S_2^1 proves*

$$\forall m \forall w (G(w) = m \rightarrow Taut(m))$$

then $e\mathcal{F}_0$ p-simulates $\mathcal{G}$.

Theorem 2 is a consequence of the next two lemmas:

Lemma 3 *If $A(b)$ is a Σ_1^b-formula with only b as free variable, then there is a polynomial $q(n)$ such that whenever $k \in \mathbf{N}$ and $\mathbf{N} \models A(k)$ there is an $e\mathcal{F}_0$-proof of $[\![A(\underline{k})]\!]^0_{q_{A(\underline{k})}}$ of size $\leq q(|k|)$.*

Proof This is very straightforward. The $e\mathcal{F}_0$-proof defines the existential variables of $[\![A(\underline{k}]\!]$ equal to the values that make $A(k)$ true and then for each subformula B of $[\![A(\underline{k}]\!]$ either proves or disproves B (in order of the complexity of B). $\square$

Lemma 4 *There is a polynomial p such that if $\mathcal{G}(w) = \ulcorner A \urcorner$ then there is an $e\mathcal{F}_0$-proof of A of length $\leq p(|x|)$.*

Lemma 4, of course, immediately implies Theorem 2.

Proof of Lemma 4 and Theorem 2. By the main theorem of the previous set of lecture notes, there is a polynomial p_1 such that $e\mathcal{F}_0$ has proofs of

$$[\![\neg G(w) = \ulcorner A \urcorner \vee \mathit{Taut}(\ulcorner A \urcorner)]\!]^n_{q(n)}$$

of size $\leq p_1(n)$, where q is a suitable bounding polynomial. Also, by Lemma 3, there is a polynomial p_2 such that $e\mathcal{F}_0$ has a proof of

$$[\![G(w) = \ulcorner A \urcorner]\!]^n_{q(n)}$$

of size $\leq p_2(|x|)$. This is because $\mathcal{G}$ is polynomial time computable and hence $|\ulcorner A \urcorner| \leq p'(|x|)$ for some polynomial p'. Thus there exists a polynomial p_3 such that there is an $e\mathcal{F}_0$-proof of

$$[\![\mathit{Taut}(\ulcorner A \urcorner)]\!]^n_{q(n)}$$

of size $\leq p_3(|x|)$. Note that p_1, p_2 and p_3 are all independent of A. Now by Lemma 1 there is a polynomial p (again independent of A) such that there is an $e\mathcal{F}_0$-proof of A of size $\leq p(|x|)$. $\Box$

Corollary 5 *If $e\mathcal{F}_1$ is an extended Frege system whose language includes $\{\neg, \vee, \wedge\}$ then $e\mathcal{F}_0$ can p-simulate $e\mathcal{F}_1$ in the following sense: There is a polynomial p such that for every tautology A in the language $\{\neg, \vee, \wedge\}$ if $e\mathcal{F}_1$ has a proof of A of size n then $e\mathcal{F}_0$ has a proof of A of size $\leq p(n)$.*

Proof It is straightforward to show that S^1_2 proves that every consequence of the extended Frege proof system $e\mathcal{F}_1$ is a tautology. This is done by using LIND to show, for $i \leq n$ that the i-th formula in the proof is a tautology. Since *Taut* is a Π^b_1-formula and S^1_2 implies Π^b_1-LIND; this is possible to carry out. The corollary now follows from Theorem 2. $\Box$

R. Reckhow [Toronto Ph.D. dissertation, 1975, "On the lengths of proofs in the propositional calculus"] defined a more general notion of *p-simulate* and *simulate* than we have used in this course. He also showed that any two extended Frege systems (with no restrictions on their languages) p-simulate each other. We shall not give the general definition of p-simulation here;

instead we note that Corollary 5 shows that there is no extended Frege proof system which is stronger than $e\mathcal{F}_0$ in terms of proving tautologies in the language $\{\neg, \vee, \wedge\}$. The next theorem proves another piece of Reckhow's theorem by showing that $e\mathcal{F}_0$ is also the weakest possible extended Frege proof system.

Theorem 6 *(Reckhow). Let $e\mathcal{F}_1$ be an extended Frege proof system with language $\mathcal{L}_1$. Then there is a translation σ of formulas in the language $\{\neg, \vee, \wedge\}$ into $\mathcal{L}_1$-formulas such that*

(a) *For all formulas A, $\sigma(A)$ is tautologically equivalent to A; and $|\sigma(A)|$ is $O(|A|)$.*

(b) *There is a constant c such that if A has an $e\mathcal{F}_0$-proof of size n then $\sigma(A)$ has an $e\mathcal{F}_1$-proof of size $\leq c \cdot n$.*

Proof We begin by describing the translation σ. Since $e\mathcal{F}_1$ is an extended Frege proof system, its language $\mathcal{L}_1$ is complete. In particular there are $\mathcal{L}_1$-formulas $\phi_\neg(p)$, $\phi_\wedge(p,q)$ and $\phi_\vee(p,q)$ such that $\phi_\neg(p)$ is tautologically equivalent to $\neg p$ and $\phi_\vee(p,q)$ is tautologically equivalent to $p \vee q$ and similarly for $\phi_\wedge$. We claim that there further exists formulas $\psi_\neg(p)$, $\psi_\wedge(p,q)$ and $\psi_\vee(p,q)$ which satisfy all of the above and additionally p (resp., p and q) appear exactly once in $\psi_\neg$ (resp., in $\psi_\vee$ and in $\psi_\wedge$).

It should be noted that other variables besides p (and q) may occur in the formulas. In this case, the other variables will be reassigned to be new variables that are used only as placeholders. (If the symbols $\top$ or $\bot$ are in $\mathcal{L}_1$ then either of them may be used as replacements for the extra placeholder variables.

An example of what is happening is as follows: suppose $\mathcal{L}_1$ contains only the "nand" symbol $|$. Then $\phi_\neg$ might be $p \mid p$. However this would not be acceptable as for $\psi_\neg$ since p occurs twice. If $\top$ were in the language we could take $p \mid \top$ as $\psi_\neg$. But in this case we have to use a new variable x and take $\psi_\neg$ to be $p \mid (x \mid (x \mid x))$. It is now easy to give formulas $\psi_\vee$ and $\psi_\wedge$.

So, if $\top$ is not $\mathcal{L}_1$ we will instead let $\top$ denote an $\mathcal{L}_1$-tautology which uses only the new variable x. Similarly, $\perp$ will denote some unsatisfiable $\mathcal{L}_1$ formula involving only the variable x.

Suppose $\phi_\neg$ contains n occurrences of the variable p (the only other variable in $\phi_\neg$ is w.l.o.g. x. Let $\phi_\neg^*$ denote the formula from $\phi_\neg$ obtained by replacing every x by $\top$; note that $\phi_\neg^*$ still is tautologically equivalent to $\neg p$. Now let $\phi_\neg^i$ be the formula obtained by replacing the first i occurrences of p in $\phi_\neg^*$ by $\top$ and the remaining $n-i$ occurrences by $\perp$. Note that each $\phi_\neg^i$ is either a tautology or is unsatisfiable. Since $\phi_\neg^0$ is a tautology and $\phi_\neg^n$ is unsatisfiable there must exist a k such that $\phi_\neg^k$ is a tautology and $\phi_\neg^{k+1}$ is unsatisfiable. Now let $\psi_\neg$ be the formula obtained from $\phi_\neg^*$ by changing the first k occurrences of p to $\top$, changing the last $n-k-1$ occurrences of p to $\perp$ and leaving the $k+1$-st p untouched. It is easy to see that $\psi_\neg$ satisfies the desired properties.

A similar but more complicated construction yields the formula $\phi_\wedge$. First let $\phi_\wedge^*$ be obtained by replacing each x in $\phi_\wedge$ by $\top$. Now let $\phi_\wedge^{i,j}$ be obtained from $\phi_\wedge^*$ by replacing the first i occurrences of p and the first j occurrences of q be $\top$ and changing the rest of the p's and q's to $\perp$. Let $\phi_\wedge^*$ contain r occurrences of p and s occurrences of q. We claim that there are values k and m such that the formulas $\phi_\wedge^{k,m}$, $\phi_\wedge^{k,m+1}$, $\phi_\wedge^{k+1,m}$ and $\phi_\wedge^{k+1,m+1}$ satisfy the following condition:

> ($\bigstar$) either (a) exactly three of them are tautologies and the other is unsatisfiable or (b) exactly one of them is a tautology and the other three are unsatisfiable

This is proved by noting that $\phi_\wedge^{0,0}$, $\phi_\wedge^{0,s}$, $\phi_\wedge^{r,0}$ and $\phi_\wedge^{r,s}$ satisfy condition ($\bigstar$) since $\phi_\wedge^*$ is equivalent to $p \wedge q$. We can now do a "binary search" to find k by noting that if $k_1 < k_2 < k_3$ and $m_1 < m_2$ and if $\phi_\wedge^{k_1,m_1}$, $\phi_\wedge^{k_1,m_2}$, $\phi_\wedge^{k_3,m_1}$ and $\phi_\wedge^{k_3,m_2}$ satisfy condition ($\bigstar$) then either $\phi_\wedge^{k_1,m_1}$, $\phi_\wedge^{k_1,m_2}$, $\phi_\wedge^{k_2,m_1}$ and $\phi_\wedge^{k_2,m_2}$ satisfy condition ($\bigstar$) or $\phi_\wedge^{k_2,m_1}$, $\phi_\wedge^{k_2,m_2}$, $\phi_\wedge^{k_3,m_1}$ and $\phi_\wedge^{k_3,m_2}$ satisfy condition ($\bigstar$). Similarly a binary search yields a value for m. Let $\chi(p,q)$ be the formula obtained from $\phi_\wedge^*$ by replacing the first k occurrences of p and the first m occurrences of q by $\top$ and replacing the last $r-k-1$ occurrences of p and the last $s-m-1$ occurrences by $\bot$. Then $\chi(p,q)$ contains exactly one occurrence of each of p and q and as a Boolean function of p and q assumes the value $\top$ exactly three times and $\bot$ once or vice-versa. Now $\psi_\wedge$ can be defined as

$$\psi_\neg(\chi(\psi_\neg(p),\psi_\neg(q))$$

where the $\psi_\neg$'s may be omitted as necessary to make $\psi_\wedge$ tautologically equivalent to $p \wedge q$.

The formula $\psi_\vee$ can easily be defined similarly or in terms of $\psi_\neg$ and $\psi_\wedge$.

Now we are ready to describe the translation σ. We define $\sigma(A)$ inductively on the complexity of A by:

(a) $\sigma(p_i)$ is p_i,

(b) $\sigma(\neg B)$ is $\psi_\neg(\sigma(B))$,

(c) $\sigma(B \wedge C)$ is $\psi_\wedge(\sigma(B),\sigma(C))$,

(d) $\sigma(B \vee C)$ is $\psi_\vee(\sigma(B),\sigma(C))$,

Since each p and q occurs at most once in $\psi_\neg(p)$, $\psi_\vee(p,q)$ and $\psi_\wedge(p,q)$ it is easy to verify that the size of $\sigma(A)$ is linear in the size of A.

The above completes the proof of part (a) of Theorem 6. The proof of part (b) is relatively straightforward; if an $e\mathcal{F}_0$ proof consists of the formulas $A_1, A_2, \ldots A_m$, then $e\mathcal{F}_1$ can emulate the proof by proving successively $\sigma(A_1), \sigma(A_2), \ldots, \sigma(A_m)$. The size of the resulting $e\mathcal{F}_1$ proof can be made linear in the size of the $e\mathcal{F}_0$-proof. $\square$

It should be noted that no reverse translation may exist of the type given in Theorem 6. For example, the formula $p \leftrightarrow q$ can not be expressed in the language $\{\neg, \vee, \wedge\}$ by a formula which contains only one occurrence each of p and q.

Math 271 - Topics in Weak Formal Systems

Homework #1
Due February 12, 1988

Instructor: Sam Buss
UC Berkeley

1. Give cut-free sequent proofs of:
   (a) $A \wedge B \longrightarrow A \vee B$.
   (b) $\longrightarrow A \vee \neg A$.
   Can this be proved without the use of a structural inference?
   (c) $A \vee (B \vee C) \longrightarrow (A \vee B) \vee C$
   (d) $(\neg A) \wedge (\neg B) \longrightarrow \neg(A \vee B)$
2. Suppose the propositional sequent calculus has been defined so that initial sequents $A \longrightarrow A$ must have A atomic. Let B be a formula of size n; give a polynomial upper bound on the length of the shortest cut free proof of $B \longrightarrow B$.
3. Expand the sequent calculus to allow the symbol $\leftrightarrow$ (logical equivalence) in the language. You should give additional rule(s) of inference and show that the soundness theorem, the completeness theorem, the cut elimination theorem and the subformula property all still hold.
4. Suppose P is a cut free proof of $A, A \vee B, \Gamma \longrightarrow \Delta$. Show there is a cut free proof P^* of $A, \Gamma \longrightarrow \Delta$ with $|P^*| < |P|$ and with the number of sequents in P^* less than or equal to the number in P. Use the convention that only atomic formulas are allowed in the initial sequents.

5. Cut free proofs may also be represented as sequences of formulas rather than as trees. (In a tree-like proof it may be necessary to rederive an intermediate sequent many times.) Prove that there exists a cut free proof *sequence* of the sequent $\Gamma_n \longrightarrow \Delta_n$ (from the proof of Theorem A-1) with size polynomal in n.

Research Problems. These are questions I do not know the answer to. Especially difficult problems are marked with an asterisk.

*6. Give either a non-polynomial lower bound or a sub-exponential upper bound on the length of sequent calculus proofs (with cuts) of valid sequents.

7. Try to give a $\Omega(n^3)$ lower bound on the size of such proofs. A quadratic lower bound is not too difficult to achieve.

8. When cut free proofs are coded as sequences instead of trees, is there an infinite family of valid formulas which require exponential size cut free proof sequences?

Math 271 - Topics in Weak Formal Systems

Homework #2
Due February 26, 1988

Instructor: Sam Buss
UC Berkeley

1. A set Δ of clauses contains x as a *pure literal* if x appears in some clause in Δ but $\overline{x}$ does not. Let Δ' be obtained by discarding every clause in Δ containing the pure literal x. Show that Δ' is satisfiable iff Δ is.

2. Suppose there is a resolution refutation of Δ with n inferences. Further suppose $C \subset D \in \Delta$. Prove that there is a resolution refutation of $(\Delta \setminus \{D\}) \cup \{C\}$ with $\leq n$ inferences. This justifies the *subsumption principle*; namely, you may discard any clause which is a proper superset of any other derived or initial clause.

3. *(Elimination of Tautologies)* Suppose there is a resolution refutation of the set $\Delta \cup \{p, \overline{p}\}$ which is n inferences long. Prove that there is a resolution refutation of Δ with $\leq n$ inferences.

4. Show that resolution with limited extension (polynomially) simulates the cut-free sequent calculus.

5. A (tree-like) resolution derivation is *regular* if each variable is resolved on at most once along each branch of the proof. Suppose there is a tree-like resolution derivation of C from Δ with n inferences. Prove that there is a regular, tree-like, resolution derivation of a subset of C from Δ with $\leq n$ inferences.

6. A *Horn* clause is a clause containing at most one unnegated propositional variable. Show that the result of applying resolution to Horn clauses is itself a Horn clause.

7. A *unit* resolution inference is one in which one of the resolvands (i.e., parent clauses) is a singleton. Suppose Δ is an unsatisfiable set of Horn clauses.

   (a) Show Δ contains a singleton or the empty clause.

   (b) Show that the empty clause can be derived from Δ by unit resolutions.

   (c) Give an example of an unsatisfiable set of clauses (not all Horn) from which there is no unit resolution derivation of the empty clause.

8. An *input* resolution derivation from Δ is a derivation in which every resolution inference has one its resolvands a clause in Δ.

   (a) Show that if there is an input derivation of the empty set from Δ then Δ contains a singleton clause or the empty clause.

   (b) Show that there is an input derivation of the empty clause from Δ if and only if there is a unit derivation of the empty clause from Δ.

   (c) In this case, is there always a derivation of the empty clause from Δ which is both a unit derivation and an input derivation?

Research Problems. These are questions I do not know the answer to. Especially difficult problems are marked with an asterisk.

9. Let $PHP_n^{n^2}$ express the pigeonhole principle that there is no one-to-one function mapping a set of cardinality n^2 into a set of cardinality n. How many inferences long are the shortest resolution proofs of these propositional formulas?

10. Does resolution simulate cut-free Gentzen systems (with respect to disjunctive normal form formulas)?

11. Does a cut-free Gentzen system simulate resolution (with respect to disjunctive normal form formulas)?

Math 271 - Topics in Weak Formal Systems

Homework #3
Due April 15, 1988

Instructor: Sam Buss
UC Berkeley

1. The *TF-substition rule* allows you to infer either $\phi(p/\top)$ or $\phi(p/\bot)$ from ϕ where p is a propositional variable and all of the occurences of p in ϕ must be replaced. $\top$ and $\bot$ represent some fixed tautology and unsatisfiable formula, respectively. Show that a Frege system plus the TF-substitution rule simulates a substitution Frege system in the same language.

2. The *renaming rule* allows you to infer $\phi(p/q)$ for any propositional variables p and q. Show that a Frege system plus the renaming rule simulates a substitution Frege system in the same language.

3. Let $||\phi||_{e\mathcal{F}}$ (resp., $||\phi||_{e\mathcal{F}}^{\text{fmla}}$) represent the minimum number of symbols (resp., formulas) in an extended Frege proof of a tautology ϕ. Show that there is a polynomial p such that $||\phi||_{e\mathcal{F}} \leq p(|\phi|, ||\phi||_{e\mathcal{F}}^{\text{fmla}})$.

4. $DTIME(t(n))$ is the class of predicates recognized by some deterministic multitape Turing machine that runs in time $t(n)$ for all inputs of length n. Prove that $DTIME(n^2)$ is a proper superset of $DTIME(2n)$. In the proof that you give, to what extent can the run time bounds n^2 and $2n$ be replaced by arbitrary bounds $t_1(n)$ and $t_2(n)$ with $t_1(n) > t_2(n)$?

5. Show that $DTIME(t(n)) = DTIME(c \cdot t(n))$ for $c \geq 1$ a constant and t time-constructible with $t(n) \geq 2n$ for all n.

6. $DSPACE(s(n))$ is the class of predicates recognized by some deterministic multitape Turing machine that uses work space $s(n)$ for all inputs of length n. Prove that $DSPACE(n^2)$ is a proper superset of $DSPACE(n)$. In the proof that you give, to what extent can the space bounds n^2 and n be replaced by arbitrary bounds $t_1(n)$ and $t_2(n)$ with $t_1(n) > t_2(n)$?

7. Show that $DSPACE(t(n)) = DSPACE(c \cdot t(n))$ for $c > 0$ a constant and t space-constructible.

8. Let $ATIME(t(n))$ be the class of predicates recognized by an alternating Turing machine that runs in time $t(n)$. Show that $ATIME(t(n)) \subseteq DSPACE(t(n))$ for t a time-constructible function. Hint: first show that $ATIME(t(n)) \subseteq DSPACE((t(n))^2)$.

*9. $NSPACE(s(n))$ is the class of predicates recognized by some nondeterministic Turing machine that runs in space $s(n)$ for all inputs of length n. Show that $NSPACE(t(n)) \subseteq ATIME((t(n))^2)$ for t a time-constructible function. Hint: try recursively computing the predicate $E_M(x, \Phi, \Psi, m)$ which states that there is a nondeterministic execution of M using space $t(|x|)$, of $\leq m$ steps, begining with the configuration Φ and ending with the configuration Ψ.

10. Use the above results to show that $NSPACE(t(n)) \subseteq DSPACE((t(n))^2)$ for t time-constructible. (Actually the assumption on t can be greatly weakened.)

You will probably wish to use the following theorem to prove 4 and 6:

Theorem. If a predicate is accepted by a time $t(n)$ bounded (resp, space $s(n)$ bounded) Turing machine with k work tapes, then it is accepted by a time $t(n) \log t(n)$ (resp, space $s(n)$) bounded Turing machine with two (resp, one) work tapes.

Research Problems. These are questions I do not know the answer to. Especially difficult problems are marked with an asterisk.

*11. Does a Frege system simulate an extended Frege system?

*12. Is $DSPACE(t(n))$ equal to $ATIME(t(n))$?

Math 271 - Topics in Weak Formal Systems

Homework #4 (Extra Credit)
Due April 15, 1988

Instructor: Sam Buss
UC Berkeley

1. Fill in the details of paragraph (a) on page 36 of chapter 2 of *Bounded Arithmetic.* Specifically, prove the existence and uniqueness conditions for the max and min functions in S^1_2.

2. Prove that S^1_2 can Δ^b_1-define every predicate in P using the fact that S^1_2 can Σ^b_1-define every polynomial time computable function.

3. Show that the *Numones* function can be Σ^b_1-defined without the use of the # function. You may omit proving the uniqueness and existence conditions in S^1_2.

4. The Ψ-*MIN* axioms are

$$(\exists x)A(x) \rightarrow (\exists x)[A(x) \wedge (\forall y < x)\neg A(y)]$$

for A a formula in Ψ. Show that relative to the base theory S^1_2, Σ^b_i-*MIN* is equivalent to Σ^b_i-*IND*.

Research Problems. These are questions I do not know the answer to. Especially difficult problems are marked with an asterisk.

5. It is known that Σ^b_{i+1}-replacement implies Σ^b_i-PIND which in turn implies Σ^b_i-replacement (relative to the base theory S^1_2). Do either of these implications reverse? Also, what connection is there between Σ^b_i-IND and Σ^b_{i+1}-replacement? Does either one imply the other?

6. Does S_2^i prove the Δ_{i+1}^b-PIND axioms? Here Δ_{i+1}^b means with respect to S_2^i.

*7. Redo the bootstrapping for S_1^1, the theory without the # function. Prove (or disprove) that S_1^1 can not prove its own bounded consistency.

Math 271 - Topics in Weak Formal Systems

Homework #5
Due May 6, 1988

Instructor: Sam Buss
UC Berkeley

1. Show that the sequent $A(b), (\forall x)(A(x) \supset A(x+1)) \longrightarrow A(b+1)$ is provable in LKB. For this problem and the problems below, you may assume without proof that $B \longrightarrow B$ is LKB-provable for all formulas B.

2. Prove that any formula provable in the sequent calculus version of S_2^i is provable in the theory S_2^i as originally defined. (In other words show that the PIND axioms imply the PIND rule.)

3. Show that $(\exists x \leq t)A \longrightarrow (\exists x)(x \leq t \wedge A)$ is LKB-provable.

4. Show that $(\exists x)(x \leq t \wedge A) \longrightarrow (\exists x \leq t)A$ is LKB-provable.

5. Prove Craig's interpolation theorem for LK by induction on the number of inferences in a cut free proof. (You may assume that the first order language contains only predicate symbols and no function symbols.)