IAS/PCMI Summer Session 2000
Clay Mathematics Undergraduate Program
Advanced Course on Computational Complexity

Lecture 13: Polynomial-Size Frege Proofs of the Pigeonhole Principle

David Mix Barrington and Alexis Maciel
August 2, 2000

The pigeonhole principle states that there is no one-to-one function from a set of size n to a set of size $n-1$. In other words, if n pigeons are put into $n-1$ holes, then at least one hole will be occupied by more than one pigeon. This simple fact has an astonishing variety of applications in mathematics. It also corresponds to a tautology that has been used extensively in the study of the complexity of proof systems.

In this lecture, we will show that the pigeonhole principle has a polynomial-size Sequent Calculus proof (and thus polynomial-size Frege proofs). In contrast, the pigeonhole principle does not have subexponential-size Resolution proofs, something we will discuss in the next lecture.

1. Overview of the Proof

We first need to express the pigeonhole principle as a propositional tautology, a Boolean formula with free variables and no quantifiers. Suppose $n > m$. These numbers will correspond to the numbers of pigeons and holes, respectively. Let $[t]$ denote the set $\{1, 2, \ldots, t\}$. We will use a variable p_{ij} for every pair $i \in [n]$ and $j \in [m]$. The intended meaning is that $p_{ij} = 1$ indicates that pigeon i went to hole j. Now define the following formula written in sequent form:

$$\mathrm{PHP}^n_m = \ldots,\ \bigvee_{j\in[m]} p_{ij},\ \ldots\ \rightarrow\ \ldots,\ p_{ij} \wedge p_{kj},\ \ldots$$

where on the left, i ranges over all values in $[n]$ and, on the right, j ranges over all values i, j, k such that $i, k \in [n]$, $i < k$ and $j \in [m]$. This will be our pigeonhole principle tautology. (Note that PHP^n_m actually states something stronger than the usual pigeonhole principle: it states that there are no *relations* from $[n]$ to $[m]$. A relation from $[n]$ to $[m]$ would mean that a pigeon can be in more than one hole, at the same time. Proving that there are no relations also proves that there are no functions. In the Sequent Calculus, the function version of PHP_m^n can be derived from the relation version by simple weakening.)

How do we prove the pigeonhole principle? We are looking for a simple proof: one where each line is a small formula and one in which every step is simple, which eventually means that it can be carried out with a small Sequent Calculus proof. Here, small means of polynomial size.

Here is one such proof. This will be the *high-level outline* of our formal Sequent Calculus proof. Add the number of true p_{ij}'s. Since for every fixed i, this number is at least 1, the total will be at least n. Now suppose, by contradiction, that there is no more than one pigeon in every hole. Once again, add the number of true p_{ij}'s but this time, start by adding for every fixed j. Then we find that the total is no more than m and since $m < n$, we have a contradiction.

Our Sequent Calculus proof will "implement" the above argument, except for the contradiction part. There, instead, we will prove directly that if the total sum of p_{ij}'s is at least n, which is greater than m, then the sum for some j must be at least 2. More precisely, here are the four major steps of the Sequent Calculus proof. These contain formulas involving summations such as $\sum_{i\in[n]}\sum_{j\in[m]} p_{ij} \geq n$. We will define these formulas later.

1. $\displaystyle \bigvee_{j\in[m]} p_{ij} \rightarrow \sum_{j\in[m]} p_{ij} \geq 1 \qquad \text{(for } i \in [n])$

2. $\displaystyle \ldots, \sum_{j\in[m]} p_{ij} \geq 1, \ldots \rightarrow \sum_{i\in[n]}\sum_{j\in[m]} p_{ij} \geq n$

3. $\displaystyle \sum_{i\in[n]}\sum_{j\in[m]} p_{ij} \geq n \rightarrow \ldots, \sum_{i\in[n]} p_{ij} \geq 2, \ldots$

4. $\displaystyle \sum_{i\in[n]} p_{ij} \geq 2, \rightarrow \ldots, p_{ij} \wedge p_{kj}, \ldots \qquad \text{(for } j \in [m])$

On the left side in Step 2, i ranges over $[n]$, on the right side in Step 3, $j \in [m]$, and on the right side in Step 4, $i, k \in [n], i < k$. Note that some steps correspond to several sequents. From all of these sequents, we can obtain PHP_m^n by a series of cuts.

2. More Details

In this section, we explain how each of the above four steps can be carried out. The first point we need to address is that we need formulas for expressing statements such as $\sum_{j\in[m]} p_{ij} \geq 1$. We will do this by using an NC^1 circuit for iterated addition. We explain this in more detail in the next section. In the meantime, and for the remainder of this section, we will assume that we have a general formula for $\sum_{i\in[s]} A_i \geq a$ and that our choice of formula allows to give polynomial-size Sequent Calculus proofs of the following seven basic facts:

A. $\displaystyle\sum_{i\in[s-1]} A_i \geq a,\ A_s \ \rightarrow\ \sum_{i\in[s]} A_i \geq a+1$

B. $\displaystyle\sum_{i\in[s-1]} A_i \geq a \ \rightarrow\ \sum_{i\in[s]} A_i \geq a$

C. $\displaystyle\sum_{i\in[s]} A_i \geq a+1 \ \rightarrow\ \sum_{i\in[s-1]} A_i \geq a$

D. $\displaystyle\sum_{i\in[s]} A_i \geq a \ \rightarrow\ \sum_{i\in[s-1]} A_i \geq a,\ A_s$

E. $\displaystyle\sum_{i\in[s]} A_i + A + B + \sum_{j\in[t]} B_j \geq a \ \rightarrow\ \sum_{i\in[s]} A_i + B + A + \sum_{j\in[t]} B_j \geq a$

F. $\displaystyle\rightarrow\ \sum_{i\in[s]} A_i \geq 0$

G. $\displaystyle\sum_{i\in[s]} A_i \geq a \ \rightarrow \qquad\qquad \text{(if } a > s\text{)}$

The A_i's and B stand for arbitrary polynomial-size formulas and $a \geq 0$.

Now consider Step 1. In more general terms, what we need to prove is

$$\bigvee_{i\in[s]} A_i \ \rightarrow\ \sum_{i\in[s]} A_i \geq 1.$$

Here is a proof:

$$\begin{array}{rcl}
\displaystyle\bigvee_{i\in[s]} A_i & \rightarrow & \displaystyle\bigvee_{i\in[s-1]} A_i,\ A_s \\
\displaystyle\bigvee_{i\in[s-1]} A_i & \rightarrow & \displaystyle\sum_{i\in[s-1]} A_i \geq 1
\end{array}$$

$$
\begin{aligned}
\sum_{i\in[s-1]} A_i \geq 1 \ &\rightarrow \ \sum_{i\in[s]} A_i \geq 1 \\
A_s \ &\rightarrow \ \sum_{i\in[s]} A_i \geq 1
\end{aligned}
$$

The result is obtained by a series of cuts and one application of the contraction rule. A proof of the second of these sequents is obtained recursively (or its existence is proved inductively). The other sequents are easy to prove directly or by using the above basic facts.

The proof of Step 2 is a little more involved. We will prove the following:

$$
\sum_{i\in[s]} A_i \geq a,\ \sum_{j\in[t]} B_j \geq b \ \rightarrow \ \sum_{i\in[s]} A_i + \sum_{j\in[t]} B_j \geq a+b. \tag{1}
$$

The sequent of Step 2 can then be obtained by iterating the above sequent as follows:

$$
\begin{aligned}
\sum_{j\in[m]} p_{1j} \geq 1,\ \sum_{j\in[m]} p_{2j} \geq 1 \ &\rightarrow \ \sum_{i\in[2]}\sum_{j\in[m]} p_{ij} \geq 2 \\
\sum_{i\in[2]}\sum_{j\in[m]} p_{ij} \geq 2,\ \sum_{j\in[m]} p_{3j} \geq 1 \ &\rightarrow \ \sum_{i\in[3]}\sum_{j\in[m]} p_{ij} \geq 3 \\
&\ \vdots \\
\sum_{i\in[n-1]}\sum_{j\in[m]} p_{ij} \geq n-1,\ \sum_{j\in[m]} p_{nj} \geq 1 \ &\rightarrow \ \sum_{i\in[n]}\sum_{j\in[m]} p_{ij} \geq n
\end{aligned}
$$

Sequent (1) above can be proved as follows:

$$
\begin{aligned}
\sum_{j\in[t]} B_j \geq b \ &\rightarrow \ \sum_{j\in[t-1]} B_j \geq b,\ B_t \\
\sum_{i\in[s]} A_i \geq a,\ \sum_{j\in[t-1]} B_j \geq b \ &\rightarrow \ \sum_{i\in[s]} A_i + \sum_{j\in[t-1]} B_j \geq a+b \\
\sum_{i\in[s]} A_i + \sum_{j\in[t-1]} B_j \geq a+b \ &\rightarrow \ \sum_{i\in[s]} A_i + \sum_{j\in[t]} B_j \geq a+b \\
\sum_{j\in[t]} B_j \geq b \ &\rightarrow \ \sum_{j\in[t-1]} B_j \geq b-1 \\
\sum_{i\in[s]} A_i \geq a,\ \sum_{j\in[t-1]} B_j \geq b-1 \ &\rightarrow \ \sum_{i\in[s]} A_i + \sum_{j\in[t-1]} B_j \geq a+b-1 \\
\sum_{i\in[s]} A_i + \sum_{j\in[t-1]} B_j \geq a+b-1,\ B_t \ &\rightarrow \ \sum_{i\in[s]} A_i + \sum_{j\in[t]} B_j \geq a+b
\end{aligned}
$$

Once again, the result is obtained by a series of cuts. Proofs for the second and fourth of these sequents are obtained recursively.

We now consider Step 3. As in the previous step, this sequent is obtained by applying and iterating the following more general sequent:

$$\sum_{i\in[s]} A_i + \sum_{j\in[t]} B_j \geq a+b \quad\rightarrow\quad \sum_{i\in[s]} A_i \geq a,\ \sum_{j\in[t]} B_j \geq b+1.$$

We leave the proof of this sequent as an exercise.

Finally, we take care of Step 4. We prove the more general sequent

$$\sum_{i\in[s]} A_i \geq 2 \quad\rightarrow\quad \ldots,\ A_i \wedge A_k,\ \ldots$$

where, on the right side, $i, k \in [s]$ and $i < k$. This is done as follows:

$$\begin{aligned}
\sum_{i\in[s]} A_i \geq 2 \quad&\rightarrow\quad \sum_{i\in[s-1]} A_i \geq 2,\ A_s \\
\sum_{i\in[s-1]} A_i \geq 2 \quad&\rightarrow\quad (\ldots,\ A_i \wedge A_k,\ \ldots)_{i,k\leq s-1} \\
\sum_{i\in[s]} A_i \geq 2 \quad&\rightarrow\quad \sum_{i\in[s-1]} A_i \geq 1 \\
\sum_{i\in[s-1]} A_i \geq 1,\ A_s \quad&\rightarrow\quad (\ldots,\ A_i \wedge A_s,\ \ldots)_{i\leq s-1}
\end{aligned}$$

A proof of the second sequent is obtained recursively and we leave the proof of the last sequent as an exercise.

Not to overburden the presentation, we have skipped over certain details such as identifying explicitly the base cases of the recursion in the construction of our proofs. We also have not said anything about the size of the proofs we constructed. We leave the task of filling in those details as an exercise.

3. Expressing Iterated Addition

Now that we have shown how to carry out the major steps of our proof of the pigeonhole principle, we need to go back and choose a formula for expressing statements $\mathrm{TH}_a(A_1,\ldots,A_s)$ of the form $\sum_{i\in[s]} A_i \geq a$. And our choice has to allow us to prove the basic facts we listed earlier.

Our formula will be based on an NC^1 circuit for the iterated addition problem. More precisely,

$$\mathrm{TH}_a(A_1,\ldots,A_s) = \mathrm{GEQ}(\mathrm{ITADD}(A_1,\ldots,A_s), a),$$

where $\mathrm{ITADD}(A_1, \ldots, A_s)$ actually stands for a sequence of formulas

$$\mathrm{ITADD}_{\log s}(A_1, \ldots, A_s),\ \ldots,\ \mathrm{ITADD}_0(A_1, \ldots, A_s),$$

each producing one bit of the binary representation of $\sum_{i\in[s]} A_i$. On the right hand side, a stands for the binary representation of a so that GEQ is actually a formula with $2\log s$ variables. The formula $\mathrm{GEQ}(x, y)$ evaluates to true if the numbers x and y satisfy $x \geq y$.

For GEQ, we can use a formula based on the carry-look-ahead method presented in the Basic Lecture 2. For ITADD, we will use the *carry-save* or *three-for-two* method that was mentioned in one of the exercises of an earlier lecture. The idea is to divide the numbers in groups of three and for each group, compute two numbers whose sum is the sum of the original three. Iterating this process produces a logarithmic depth circuit that computes two numbers that we simply need to add. To simplify details, we will actually reduce the sum of four numbers to the sum of two. More precisely, the formula ITADD will be defined recursively as follows:

$$\begin{aligned}
\mathrm{ITADD}(A_1, \ldots, A_s) &= \mathrm{ADD}(\mathrm{ADD}_{s\to 2}(A_1, \ldots, A_s)) \\
\mathrm{ADD}_{s\to 2}(A_1, \ldots, A_s) &= \mathrm{ADD}_{4\to 2}(\mathrm{ADD}_{s/2\to 2}(A_1, \ldots, A_{s/2}), \mathrm{ADD}_{s/2\to 2}(A_{s/2+1}, \ldots, A_s)) \\
\mathrm{ADD}_{4\to 2}(w_1, w_2, w_3, w_4) &= \mathrm{ADD}_{3\to 2}(\mathrm{ADD}_{3\to 2}(w_1, w_2, w_3), w_4)
\end{aligned}$$

ADD is a formula that adds two numbers and $\mathrm{ADD}_{t\to 2}$ reduces the sum of t numbers to the sum of two numbers. The basic building block is $\mathrm{ADD}_{3\to 2}$. Its construction was explained in the earlier exercise referred to above.

Now that we have defined our formula for $\sum_{i\in[s]} A_i \geq a$, we need to establish the basic facts that we relied on in our proof of the pigeonhole principle. It is not hard to see that the formula TH is correct in the sense that $\mathrm{TH}_a(A_1, \ldots, A_s) = 1$ if and only if $\sum_{i\in[s]} A_i \geq a$ is true. And from this, of course, follows the fact that the sequents expressing our basic facts all correspond to tautologies. But this is not enough because we cannot prove that $\mathrm{TH}_a(A_1, \ldots, A_s)$ is correct without speaking of $\sum_{i\in[s]} A_i \geq a$, and this is exactly what we are trying to accomplish. What we need to do is prove those basic facts *directly*, relying not on the correctness of the TH formula, but on its structure.

Note also that we cannot simply stick our formula for $\sum_{i\in[s]} A_i \geq a$ in the proof and say that since the formula works, we are done. We do have to prove those basic facts since we used them in our proof of the pigeonhole principle.

The task of proving these basic facts is relatively straightforward but long in details. We omit it from this presentation and leave it as an exercise for the interested reader.

4. Exercises

1. Give a polynomial-size Sequent Calculus proof of the following sequent:

$$\sum_{i\in[s]} A_i \geq 1,\ B \ \rightarrow\ \ldots,\ A_i \wedge B,\ \ldots$$

   where i on the right side ranges over all values in $[s]$.

2. Give a polynomial-size Sequent Calculus proof of the following sequent:

$$\sum_{i\in[s]} A_i + \sum_{j\in[t]} B_j \geq a+b \ \rightarrow\ \sum_{i\in[s]} A_i \geq a,\ \sum_{j\in[t]} B_j \geq b+1.$$

3. Use the sequent of the previous exercise to prove the sequent of Step 3.

4. In three instances, we referred to a recursive process for constructing some of our proofs. Identify the base cases in these recursions and give a proof of the corresponding sequent.

5. Verify that our proof of the pigeonhole principle is of polynomial size.

6. Consider a formula MOD_2 that evaluates to 1 if and only an odd number of its arguments are true. Implement this formula with the familiar logarithmic-depth tree of binary MOD_2's. Give a proof of the following basic fact:

$$\mathrm{MOD}_2(A_1,\ldots,A_s) \ \rightarrow\ \mathrm{MOD}_2(A_1,\ldots,A_{s-1}),\ A_s$$

7. Give polynomial-size Sequent Calculus proofs of some of the basic facts we used in our proof.