Boolean Satisfiability
Lecture 8: Cutting Planes and other proof systems

Edward A. Hirsch[*]

February 19, 2024

Lecture 8

In this lecture we

- formulate what is a propositional proof system, what are polynomial simulations, and what lattice do we get of of them.
- introduce the Cutting Planes proof system,
- talk about Frege systems and the extension rule,
- denostrate short proofs of the propositional pigeonhole principle in Cutting Planes and Extended Frege.

Contents

[*]Ariel University, http://edwardahirsch.github.io/edwardahirsch

1 What is a proof system?

What is a mathematical proof? It should be something that can be verified by another mathematician without much creative work. It may be difficult to find a proof, but it should be fairly easy to check it. For us, "easy" means "in polynomial time". In real life we would prefer almost linear-time verification, otherwise we could not verify really long mathematical proofs...

Definition 1. Formally, a **proof system** for a language $L \subseteq \{0,1\}^*$ is a deterministic polynomial-time algorithm V that **v**erifies proofs. It must satisfy two conditions:

Completeness. $\forall x \in L\ \exists w \in \{0,1\}^*$ such that $V(x,w) =$ "yes".

Soundness. $\forall x \notin L\ \forall w \in \{0,1\}^*$ inevitably $V(x,w) =$"no".

Note that the polynomial (in "polynomial time") is taken over $|x| + |w|$ and not just $|x|$!

What "theorem" do we prove here? We prove that $x \in L$.

Note that we know a very efficient proof system for **SAT**: give a satisfying assignment w as a proof. It is very easy to verify it: just substitute the values into the formula.

The situation with **UNSAT** is more interesting.

Definition 2. A proof system V for L is **polynomially bounded** if there is a polynomial p such that for every $x \in L$, there exists a proof w of length $|w| \leqslant p(|x|)$ (the term "proof" here means that $V(x,w) = 1$).

Note that if L possesses a polynomially bounded proof system, then $L \in$ **NP** (this is clear immediately from the definitions). We do not know if there is a polynomially bounded proof system for **UNSAT**: this is equivalent to **NP** = **co-NP**!

Proof systems for **UNSAT** are called **propositional proof systems**. (Note that proof systems for **UNSAT** and proof systems for **TAUT**, the language of Boolean tautologies, are essentially the same: if F is a tautology, then $\overline{F}$ is unsatisfiable, and vice versa.)

We already know several proof systems for **UNSAT**: the resolution proof system and its subsystems (treelike resolution, regular resolution). We now introduce another proof system, which is stronger than resolution.

2 Cutting Planes

2.1 The definition of the Cutting Planes proof system

The **Cutting Planes** (or the *Cutting Plane*) proof system (**CP**) is based on the works of Ralph Gomory and Vašek Chvátal in the field of integer linear optimization.

First of all, we translate our CNF formula into a system of linear inequalities. A Boolean variable x is translated into a rational variable x, the negation of x is translated into $1 - x$. A clause $\ell_1 \vee \ell_2 \vee \ldots \vee \ell_k$ is translated into the inequality $\ell_1 + \ell_2 + \ldots + \ell_k \geqslant 1$, where ℓ_i's in the sum stand for the translations of the corresponding Boolean literals.

For example, the formula $(x \vee \overline{y}) \wedge (\overline{x} \vee \overline{y})$ is translated into the system

$$\begin{aligned} x + (1-y) &\geqslant 1, \\ (1-x) + (1-y) &\geqslant 1 \end{aligned}$$

or simply the system

$$\begin{aligned} x - y &\geqslant 0, \\ -x - y &\geqslant -1 \end{aligned}$$

(we open all the parentheses in CP, also we do not distinguish $0 \geqslant 1$, $-1 \geqslant 0$ etc).

W.l.o.g., we work with integer coefficients only.

In this translation, we keep the Boolean solutions, however, we may acquire extra solutions (say, negative or fractional). In order to make sure that the solutions are between 0 and 1, we introduce the axioms $x \geqslant 0$ and $-x \geqslant -1$ for every variable. In order to rule out non-integer solutions, we use the rounding rule:

$$\frac{\sum_i dc_i x_i \geqslant c}{\sum_i c_i x_i \geqslant \left\lceil \frac{c}{d} \right\rceil} \qquad \textbf{(Rounding)}$$

Note that rounding is done in the non-trivial direction: it throws away some non-integer solutions.

We also use another rule (nonnegative linear combinations) in our derivations:

$$\frac{A \geqslant a, \quad B \geqslant b}{\alpha A + \beta B \geqslant \alpha a + \beta b} \qquad (\alpha, \beta \geqslant 0) \qquad \textbf{(Linear combination)}$$

The proof of $F \in$ **UNSAT** in CP is a derivation starting from the translations of the clauses of F and the axioms, and deriving at each step a new inequality from previously derived inequalities using one of the two rules; the proof ends with the contradictory inequality $0 \geqslant 1$.

Note that similarly to resolution, we can define a normal (daglike) Cutting Planes proof system and a treelike CP (where we must derive an inequality again if we use it more than once).

It is clear that CP proofs are sound, as our rules derive semantic consequences: if previously derived inequalities had a common 0/1-solution, then this solution satisfies the newly derived inequality as well.

However, is CP complete? As resolution is complete, it suffices to show that every resolution proof can be transformed into a CP proof.

2.2 Simulation of resolution by Cutting Planes

We convert a resolution refutation into a CP refutation step by step. In fact, we **polynomially simulate** resolution in CP (that is, the proof becomes at most polynomially longer; the definition is given later in this lecture).

We start with the axioms and the initial clauses, and through our simulation we maintain the invariant: for every clause in the resolution proof, we derive its translation (see above!) in CP. For the ease of notation, we denote clauses and the sums of the translation of their literals by the same letters (that is, if $A = (x \vee y)$, then we also denote $A = x + y$).

Our invariant is trivially true for the original clauses. Once a new clause is derived using a resolution rule, we take the sum in CP:

$$\frac{A \vee x \qquad \overline{x} \vee B}{A \vee B} \quad \longrightarrow \quad \frac{A + x \geqslant 1 \qquad (1-x) + B \geqslant 1}{A + B \geqslant 1}$$

Unfortunately, what we get is not necessarily the translation of $A \vee B$, because A and B can have joint variables!

So denoting $A = A' \vee C$, $B = B' \vee C$, where now $A' \cap B' = \emptyset$, we actually get

$$A' + B' + 2C' \geqslant 1, \tag{A'B'2C}$$

and we do not have the coefficient 2 here in the proper translation. Let us get rid of it using the rounding rule.

First of all, we derive $A' + B' \geqslant 0$ as the sum of the axioms for the literals occurring in A' and B' (recall that we have the nonnegativity axioms). Then we add this inequality to (A'B'2C) and apply the rounding rule:

$$\cfrac{A' + B' + 2C' \geqslant 1 \qquad A' + B' \geqslant 0}{\cfrac{2A' + 2B' + 2C' \geqslant 1}{A' + B' + C' \geqslant 1}}$$

Now $A' + B' + C' \geqslant 1$ is a proper translation of $A \vee B$ (i.e., $A' \vee B' \vee C'$), and we are done.

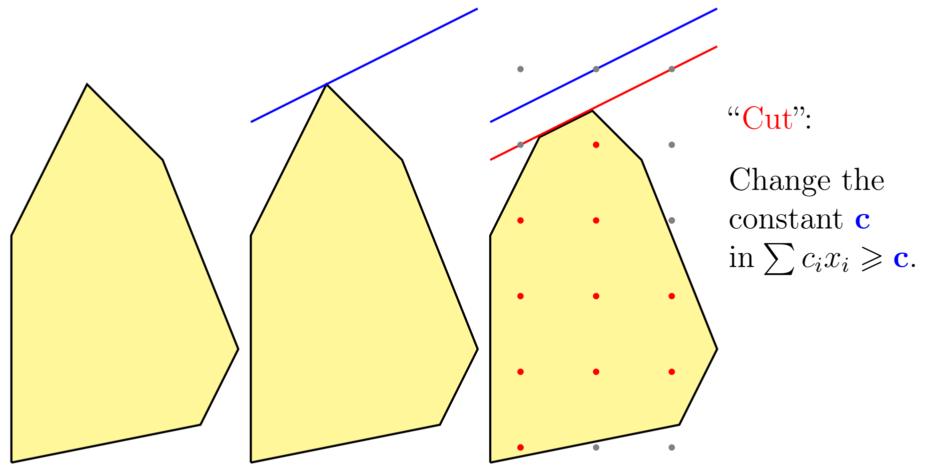
We eventually derive the empty clause, its translation is exactly the contradiction $0 \geqslant 1$ in CP.

2.3 Intuition from integer linear optimization

In order to optimize a (linear) function over integer points in a convex polytope, it is useful to reduce this polytope. In proof complexity we do not optimize any function, in fact, we prove that the polytope does not contain any integer (actually, 0/1) points.

When we take a nonnegative linear combination (see the picture: blue line), we do not change the polytope, as the new inequality is a semantic consequence of the old inequalities for all real solutions.

However, we can move this line by changing the constant part of the new inequality until we hit an integer point thus cutting the polytope (see the picture: red line):


"Cut":
Change the constant $\mathbf{c}$ in $\sum c_i x_i \geqslant \mathbf{c}$.

This is called **Gomory–Chvátal cuts**. There are also other cuts (Sherali–Adams, Lovász–Schrijver, etc) that we do not consider in this course.

2.4 PHP in Cutting Planes

To show that Cutting Planes has short proofs for some tautologies that are hard for the resolution proof system, we provide polynomial-size CP refutations for the propositional pigeonhole principle.

First, let us translate them:

Clauses	Inequalities
$x_{i1} \vee x_{i2} \vee \ldots \vee x_{i,m-1}$	$x_{i1} + x_{i2} + \ldots + x_{i,m-1} \geqslant 1$
$\overline{x_{ij}} \vee \overline{x_{i'j}}$	$x_{ij} + x_{i'j} \leqslant 1$

Our strategy is:

(1) Take the sum of the long clauses for $i = 1, \ldots, m$:

$$\sum_{ij} x_{ij} \boldsymbol{\geqslant m}.$$

(2) Prove that the short clauses for the hole j imply

$$x_{1j} + x_{2j} + \ldots + x_{m,j} \leqslant 1.$$

(3) Take the sum of these clauses for $j = 1, \ldots, m-1$:

$$\sum_{ji} x_{ij} \boldsymbol{\leqslant m - 1}.$$

(4) Take the sum of (1) and (3):

$$0 \geqslant 1.$$

It is clear that it remains to

- **prove that the short clauses for the hole j imply**

$$\boldsymbol{x_{1j} + x_{2j} + \ldots + x_{m,j} \leqslant 1}.$$

It would be easy to prove this for three pigeons:

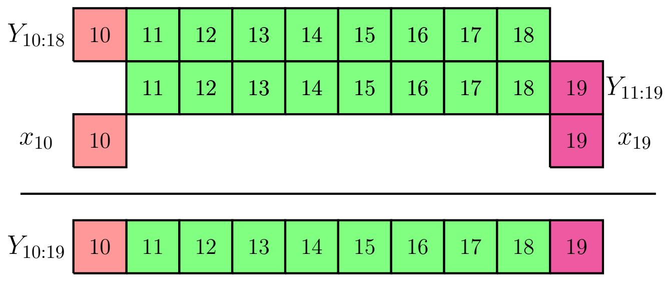
$$\frac{\dfrac{x_{ij} + x_{i'j} \leqslant 1 \qquad x_{i'j} + x_{i''j} \leqslant 1 \qquad x_{i''j} + x_{ij} \leqslant 1}{2 \cdot (x_{ij} + x_{i'j} + x_{i''j}) \leqslant 3}}{x_{ij} + x_{i'j} + x_{i''j} \leqslant 1}$$

We now proceed to the general case.

Fix j. Denote $Y_{s:s+t} := \sum_{i=s}^{s+t} x_{ij}$. We will prove by the induction on t that $Y_{s:s+t} \leqslant 1$.

The base is for $t = 1$. For the induction step, we will use the induction hypothesis for two overlapping segments $i : i + t$ and $i + 1 : i + t + 1$ and the original inequality $x_i + x_{i+t+1} \leqslant 1$.

Here is an example for the sake of clarity:



The formal proof of the induction step is

$$\frac{\dfrac{Y_{\boldsymbol{i:i+t}} \leqslant 1 \qquad Y_{\boldsymbol{i+1:i+t+1}} \leqslant 1 \qquad x_{\boldsymbol{i}j} + x_{\boldsymbol{i+t+1},j} \leqslant 1}{2 \cdot Y_{\boldsymbol{i:i+t+1}} \leqslant 3}}{Y_{\boldsymbol{i:i+t+1}} \leqslant 1}$$

We have thus proved:

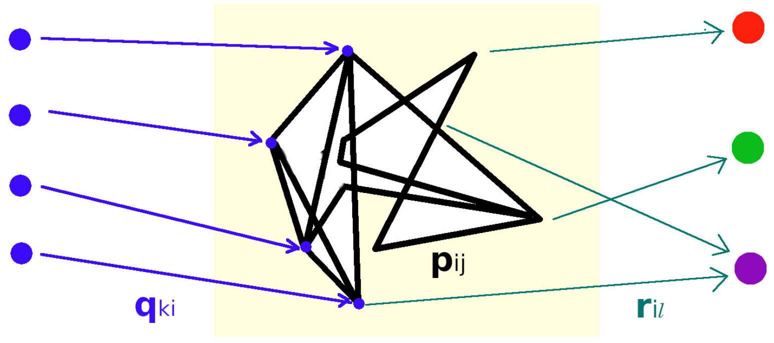
Theorem 1. ***PHP*** *has polynomial-size proofs in Cutting Planes.*

2.5 Cutting Planes summary and hard formulas

We learned that CP is better than the resolution proof system (for every formula, it has a proof of size $O(\ldots)$ of the shortest resolution proof, and for **PHP** it definitely has exponentially shorter proofs). However, there are hard formulas for CP as well. We delay the proof (of an exponential size bound) to a latter occasion, but here are the hard formulas:

Clique-coloring formulas express that a graph can contain an m-clique (complete subgraph K_m) and at the same time be colorable in $m-1$ colors:

- Consider n-vertex graph encoded by variables p_{ij} ("is there the edge (i,j)?").
- State that it contains an m-clique encoded by variables q_{ki} ("vertex i is the k-th node of the clique").
- State that this graph is $(m-1)$-colorable (the coloring is encoded by variables $r_{i\ell}$: "vertex i is colored by color ℓ").



Specifically, the clauses are:

$\bigvee_{\ell=1}^{m-1} r_{i\ell}$	for $1 \leqslant i \leqslant n$	vertex i is colored
$\overline{p_{ij}} \vee \overline{r_{i\ell}} \vee \overline{r_{j\ell}}$	for $1 \leq i < j \leq n$, $1 \leqslant \ell \leqslant m-1$	correctness of the coloring
$\bigvee_{i=1}^{n} q_{ki}$	for $1 \leqslant k \leqslant m$	the k-th node of the clique
$\overline{q_{ki}} \vee \overline{q_{k'i}}$	for $1 \leqslant i \leqslant n$ and $1 \leqslant k < k' \leqslant m$	all nodes are different
$\overline{q_{ki}} \vee \overline{q_{k',j}} \vee p_{ij}$	for $1 \leqslant k < k' \leqslant m$ and $1 \leq i < j \leq n$	edge between the nodes k and k'
$\overline{q_{ki}} \vee \overline{q_{kj}}$	for $1 \leqslant k \leqslant m$	there is only one k-th node

The last type of clauses is not needed for unsatisfiability, but may be useful when we prove upper bounds in other proof systems.

3 The lattice of proof systems

Let us study relations between proof systems.

Polynomial simulations.

Definition 3. A proof system S **polynomially simulates** a proof system W (written $S \leqslant W$ or $W \to S$) iff S-proofs are at most as long as W-proofs (up to a polynomial p):

$$\forall F \in \mathbf{UNSAT} \quad |\text{the shortest } S\text{-proof of } F| \leqslant p(|\text{the shortest } W\text{-proof of } F| + |F|).$$

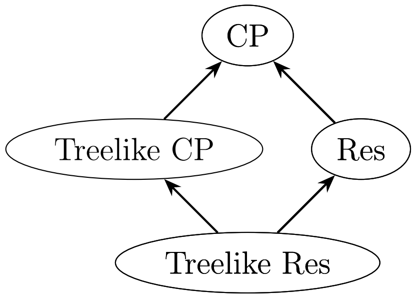
(For our purpose, it suffices to concentrate on proofs that include F itself, thus the "+|F|" term.)

S is **strictly stronger** than W (written $S < W$) if in addition $W \not\leqslant S$.

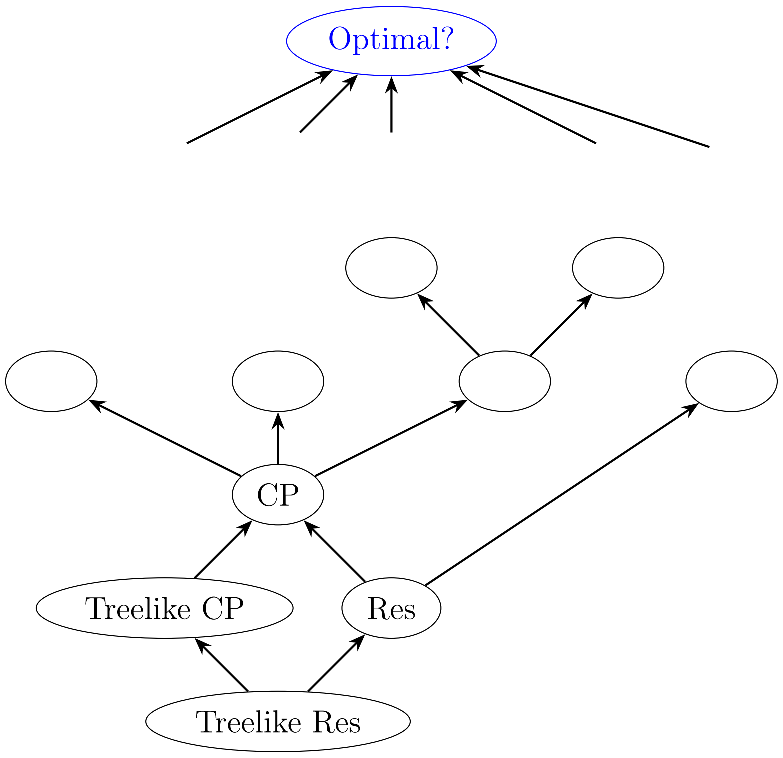
Usually $W \not\leqslant S$ is proved by providing a series of separating formulas that have short proofs in S but not in W.

Polynomial simulations define a non-strict *partial* order on proof systems, with separating formulas providing strict ordering between some proof systems. Thus proof systems form what we call "the lattice of proof systems".

We proved above that Cutting Planes polynomially simulate the resolution proof systems. Also there are no polynomial-size resolution refutations of **PHP** (we proved it for treelike resolution, but in fact an exponential lower bound is known for daglike resolution as well), while there are polynomial-size CP proofs for **PHP**. Thus Cutting Planes is strictly stronger than resolution; so we could already mark one edge in the lattice as strict. In fact, all edges in the following small picture are strict:


$$\begin{tikzcd}
 & \text{CP} & \\
\text{Treelike CP} \arrow[ur] & & \text{Res} \arrow[ul] \\
 & \text{Treelike Res} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

«S. Cook's program» is the intention to prove lower bounds for stronger and stronger proof systems. This way we develop new methods for proving lower bounds, which are the crux of the computational complexity theory.



Optimal proofs? One could also imagine an **optimal propositional proof system**, so to say, "proofs from The Book", that is, a propositional proof system that polynomially simulates every other propositional proof system.

Unfortunately, we are unable to prove that it exists, and I personally do not believe that it exists unless we relax the definition of a proof system. There may be many approaches working in different situations, but no universal approach! Of course, if **NP** = **co-NP**, then there is an optimal proof system (the one that has a polynomial-size proof for every formula). However, this assumption is believed to be false by many. On the contrary, we do not have a nice suitable assumption that would imply that no optimal proof systems exist; the only such complexity assumptions that we are aware of are not widely believable at all (probably, they are just false).

4 Frege and Extended Frege systems

4.1 Definition of Frege systems

A Frege[1] proof system proves Boolean tautologies (or refutes unsatisfiable formulas) in a language of formulas tha use certain Boolean connectives. It could be the set $\{\vee, \wedge, \neg\}$ or anything else.

An important thing is that the lines of a Frege proof are *formulas*, that is, treelike structures that use parentheses, we are not required to open them, and this may yield exponential savings.

A **Frege system** consists of a constant number of derivation rules of the form

$$\frac{\Phi_1 \quad \Phi_2 \quad \dots \quad \Phi_k}{\Psi},$$

where Φ_i, Ψ are propositional formulas using "abstract" Boolean variables (these are *not* the variables of the formula which we prove or refute). The rules must be sound, that is,

$$\forall X_1 \, \forall X_2 \dots \; (\Phi_1 \wedge \Phi_2 \wedge \dots \wedge \Phi_k \Rightarrow \Psi).$$

(Here, X_i's are the abstract variables.) Some of the rules (those with $k = 0$) are axioms.

A **Frege derivation** in a specific Frege system derives new formulas step by step by applying *instances* of the derivation rules of the system: such instance is obtained by substituting the abstract variables $X_1, X_2, \dots$ by Boolean formulas (the latter formulas use the variables appearing in the formula we prove).

For example, consider a system that uses only two connectives $\{\Rightarrow, \neg\}$, the implication and the negation, and has just three axioms:

$$\begin{aligned} P &\Rightarrow (Q \Rightarrow P) \\ (\neg Q \Rightarrow \neg P) &\Rightarrow ((\neg Q \Rightarrow P) \Rightarrow Q) \\ (P \Rightarrow (Q \Rightarrow R)) &\Rightarrow ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R)) \end{aligned}$$

and just one derivation rule (*modus ponens*):

$$\frac{P \qquad P \Rightarrow Q}{Q}.$$

One could use, for example, the instance

$$\boldsymbol{(x \Rightarrow y)} \Rightarrow ((z \vee (x \vee \neg y)) \Rightarrow \boldsymbol{(x \Rightarrow y)})$$

of the first axiom or the instance

$$\frac{(x \Rightarrow (z \Rightarrow y)) \qquad (x \Rightarrow (z \Rightarrow y)) \Rightarrow \boldsymbol{(x \Rightarrow \neg u)})}{\boldsymbol{(x \Rightarrow \neg u)}}$$

of the derivation rule.

[1] In honour of Gottlob Frege — but it does not mean that he defined these systems.

Frege systems can be formulated in two ways: as proof systems that derive tautologies from the (instances of) axioms of the Frege system (thus proof systems for the language **TAUT** of Boolean tautologies) and refutation systems that start with the axioms and also with the formula that we want to refute, and derive the contradiction `False` (thus proof systems for the language **UNSAT**). In both cases, soundness follows from the soundness of the rules (and axioms). The completeness depends on the set of rules and axioms (we usually consider complete Frege systems).

To be absolutely correct, we must define a translation of Boolean formulas into formulas that use the set of connectives of our Frege system, and this translation is considered a part of that system. We will learn later that for "natural" Frege systems all these things do not matter:

- the set of connectives,
- the set of rules,
- proofs for **TAUT** versus proofs for **UNSAT**,
- daglike mode or treelike mode.

We delay these considerations to the next lecture.

Frege systems operate with arbitrary formulas instead of clauses, and they are already powerful enough. However, let us strengthen them even more.

4.2 Extended Frege systems

Tseitin's extension rule denotes a formula by a new variable (called **extension variable**), this is a very natural trick in a typical mathematical proof!

So it allows to introduce a new variable x along with additional axioms (called **extension axioms**) defining it: either

$$x \equiv \Phi$$

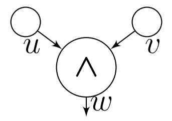
if your system uses the equivalence $\equiv$ or two axioms

$$(x \Rightarrow \Phi), \qquad (\Phi \Rightarrow x)$$

if it uses the implication $\Rightarrow$, or whatever other axioms that establish this equivalence.

These are not Frege-style axioms, they use the variables of the formula that we prove. An Extended Frege system combines derivations in a Frege system with extension axioms.

The extension rule is extremely powerful! We can actually encode Boolean circuits using this rule (and circuits generalize formulas, as dags generalize trees). Recall Tseitin's translation; for example, look at the example for the $\wedge$-gate:



$$\begin{array}{ll} (\overline{w} \vee u) & \wedge \\ (\overline{w} \vee v) & \wedge \\ (\overline{u} \vee \overline{v} \vee w) & \end{array} \qquad\qquad \boldsymbol{w \equiv (u \wedge v)}$$

Here w is exactly an extension variable, and the three clauses are its extension axioms.

We already know how to represent a circuit C by a CNF that uses these axioms:

$$\exists x_1, x_2, \dots x_n \; C(x_1, \dots, x_n) = 1 \iff \exists x_1, x_2, \dots x_n, w_1, \dots, w_m \;\; w_m \wedge \bigwedge_j A_j,$$

where w_i's are the extension variables, A_j are the extension axioms, and w_m is the extension variable corresponding to the output gate of C.

Actually, the extension rule is so powerful that Extended Frege systems are polynomially equivalent to Extended Resolution, the system obtained if we add the extension rule to the resolution proof system. That is, these proof systems polynomially simulate each other. In the next lecture we will demonstrate this fact.

4.3 Short proofs of PHP in Extended Frege

We now provide a polynomial-size proof for the propositional pigeonhole principle in an Extended Frege system.

The proof goes by induction on m. Namely, we assume the pigeonhole formula for m pigeons and $m-1$ holes, and derive from it the pigeonhole formula for $m-1$ pigeons and $m-2$ holes. Continuing this way, we eventually arrive at the formula for 2 pigeons and 1 hole

$$x_{1,1}^{(2)}$$

$$x_{2,1}^{(2)}$$

$$\overline{x_{1,1}^{(2)}} \vee \overline{x_{2,1}^{(2)}}$$

which is easily refutable even in resolution.

In the course of the proof, we introduce new variables. The new set of variables is denoted using a superscript.

- The original variables x_{ij} are denoted by $x_{ij}^{(m)}$.
- Given the variables $x_{ij}^{(n)}$ for n pigeons and $n-1$ holes, we introduce new variables for $n-1$ pigeons and $n-2$ holes by throwing away the last pigeon and the last hole and redirecting the pigeon that was occupying the removed hole to the hole that was occupied by the removed pigeon:
$$x_{i,j}^{(n-1)} \equiv x_{i,j}^{(n)} \vee (x_{n,j}^{(n)} \wedge x_{i,n-1}^{(n)}).$$
Other than that, the mapping of pigeons to holes remains the same as before.

To complete the proof, it remains to derive the **PHP** clauses for the new variables from the **PHP** clauses for the old variables.

Let us show how to do it with the "pigeon" axioms. We want to derive the clause

$$x_{i,1}^{(n-1)} \vee x_{i,2}^{(n-1)} \vee \dots \vee x_{i,n-2}^{(n-1)} \qquad \text{(NEW)}$$

whereas we have clauses

$$x_{i,1}^{(n)} \vee x_{i,2}^{(n)} \vee \ldots \vee x_{i,n-1}^{(n)} \tag{OLD}$$

To switch from the use of an extension variable to its defining formula, we use the rule

$$\frac{\Phi \vee \Psi \qquad \Psi \equiv \Delta}{\Phi \vee \Delta}$$

(surely, it is a correct Frege rule).

Using a repeated application of this rule, we can rewrite (NEW) as

$$x_{i,1}^{(n)} \vee x_{i,2}^{(n)} \vee \ldots \vee x_{i,n-2}^{(n)} \ \vee\ (x_{n,1}^{(n)} \wedge x_{i,n-1}^{(n)}) \vee (x_{n,2}^{(n)} \wedge x_{i,n-1}^{(n)}) \vee \ldots \vee (x_{n,n-1}^{(n)} \wedge x_{i,n-1}^{(n)}). \tag{NEW'}$$

Note that using the same rule we can also rewrite the obtained formula into (NEW).

Using the distributivity law (surely, it can be formulated as a correct Frege rule) we can rewrite (NEW') as

$$x_{i,1}^{(n)} \vee x_{i,2}^{(n)} \vee \ldots \vee x_{i,n-2}^{(n)} \ \vee\ \underbrace{(x_{n,1}^{(n)} \vee x_{n,2}^{(n)} \vee \ldots \vee x_{n,n-1}^{(n)})}_{\text{axiom!}} \wedge x_{i,n-1}^{(n)} \tag{NEW''}$$

and also rewrite this formula back into (NEW').

It can be easily seen that the latter formula can be derived from the two old axioms (OLD) for the pigeons i and m. For this, we use (also a correct) rule

$$\frac{\Phi_1 \vee \Phi_2 \qquad \Phi_3}{\Phi_1 \vee (\Phi_2 \wedge \Phi_3)}.$$

Now that we derived (NEW") from two instances of (OLD), we perform the steps rewriting (NEW") back into (NEW), and we are done with the pigeon axioms.

We can derive the "hole" axioms

$$\overline{x_{i,j}^{(n-1)}} \vee \overline{x_{i'j}^{(n-1)}}$$

in a similar fashion. This is left as a homework exercise in HW#2.

5 Takeaway

In this lecture we formally defined proof systems, polynomial simulations between them, looked at the obtained lattice and discussed the so-called *Cook's program* in the propositional proof complexity.

We defined Cutting Planes, Frege and Extended Frege systems, and demonstrated that the propositional pigeonhole principle has polynomial-size proofs in Cutting Planes and Extended Frege. (This is in contrast to the known exponential lower bounds for **PHP** refutations in the resolution proof system.)

In the next lectures we will further discuss Frege systems, show that Extended Frege is polynomially equivalent to Extended Resolution, and, it time permits, prove an exponential lower bound on the length of Cutting Planes proofs.

Historical notes and further reading

Formal definitions for proof systems and polynomial simulations were suggested by Steve Cook and Robert Reckhow in [1]. Our definitions are a little bit different, but they are equivalent for our purpose.

The material of these lecture notes can be mostly found in an old survey by Samuel R. Buss `https://mathweb.ucsd.edu/~sbuss/ResearchWeb/Barbados95Notes/index.html` .

For in-depth treatment and further historical remarks we refer the reader to a recent book

Jan Krajíček: *Proof complexity.*
Encyclopedia of Mathematics and Its Appplications, Vol.170, Cambridge University Press, Cambridge - New York - Melbourne, (2019), 530pp.
Draft version: `https://www.karlin.mff.cuni.cz/~krajicek/prfdraft.html`

References

[1] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. J. Symb. Log., 44(1):36–50, 1979.