Boolean Satisfiability
Lecture A: Frege proof systems

Edward A. Hirsch*

June 2, 2026

Lecture A

In this lecture we show that the following proof systems are polynomially equivalent:

- Frege systems that prove tautologies and Frege systems that refute unsatisfiable formulas.
- Daglike and treelike Frege systems.
- Frege systems that use different sets of sound (correct) rules and axioms (as long as these sets are complete and implicationally complete).
- Extended Frege and Extended Resolution.

Contents

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

1 Definition of Frege systems

Recall the definition from the previous lecture.

A Frege proof system proves Boolean tautologies (or refutes unsatisfiable formulas) in a language of formulas that use certain Boolean connectives (operations). It could be the set $\{\vee, \wedge, \neg\}$, where $\vee$ and $\wedge$ are binary and $\neg$ is unary, or some other set.

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 that we prove or refute). The rules must be sound (correct), that is,

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

(Here, X_i's are the abstract variables — $\Phi_1, \dots, \Phi_k, \Psi$ are formulas using these and only these 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 the following axioms

$$(P \Rightarrow P)$$

$$P \Rightarrow (Q \Rightarrow P)$$

$$(\neg Q \Rightarrow \neg P) \Rightarrow ((\neg Q \Rightarrow P) \Rightarrow Q)$$

$$(P \Rightarrow (Q \Rightarrow R)) \Rightarrow ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R))$$

$$(P \Rightarrow Q)) \Rightarrow ((P \Rightarrow (Q \Rightarrow R)) \Rightarrow (P \Rightarrow R))$$

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

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

One can use her either a 0-ary operation `False` or the negation $\neg$: $\neg\mathbf{F}$ is an equivalent for $(\mathbf{F} \Rightarrow \texttt{False})$.

Typical rules used in Frege systems formulate commutativity, associativity and distributivity laws and allow equivalent changes in the structure of the formula (such as replacing $A \vee A$ by A, and vice versa).

2 Notation for derivations

We introduce the following notation, which applies to any systems that derive lines step by step.

$F \vdash G$: G is derived from F in one step,

$F \vdash^* G$: G is derived from F in several (maybe zero) steps.

If there is no F, we mean that we use the axioms only: $\vdash^* G$.

In this notation, F and G can refer to *lists* of formulas.

3 Proof by contradiction and the deduction theorem

In Frege systems, proving a tautology F from the axioms and refuting its negation $\neg F$ are equivalent approaches.

Let us prove it for our example system.

Theorem 1 (Proof by contradiction)**.** $\vdash^* F$ *if and only if* $\neg F \vdash^* \texttt{False}$*. Moreover, the length of the shortest proof that* F *is a tautology is within a polynomial of the length of the shortest refutation of* $\overline{F}$*, and vice versa.*

This theorem is immediately implied by the deduction theorem (put $G = \texttt{False}$):

Theorem 2 (Deduction theorem)**.** $\Gamma \vdash^* (F \Rightarrow G)$ *iff* $\Gamma, F \vdash^* G$.
Moreover, the two (minimal) derivations sizes are within a polynomial of each other.

Proof. $\rightarrow$: Trivial use of modus ponens.

$\leftarrow$:

Consider the derivation $\Gamma, F \vdash^* G$: $\varphi_1,\ \varphi_2,\ \ldots,\ \varphi_s$.

Replace φ_i by $(F \Rightarrow \varphi_i)$: $(F \Rightarrow \varphi_1),\ \ldots,\ (F \Rightarrow \varphi_s)$.

Let us convert the obtained sequence of formulas into a valid proof by adding intermediate steps. Proceed from the left to the right. How was φ derived in the original proof? Consider the three possible cases:

φ_i **is** F**:** No changes are needed, as
(axiom) $(F \Rightarrow F)$.

φ_i **is derived by modus ponens:** $\dfrac{\varphi_j \quad \varphi_j \Rightarrow \varphi_i}{\varphi_i}$.

Then consider the two already derived formulas and insert three derivation steps:
(given) $F \Rightarrow \varphi_j$
(given) $F \Rightarrow (\varphi_j \Rightarrow \varphi_i)$
(axiom) $(F \Rightarrow \varphi_j) \Rightarrow ((F \Rightarrow (\varphi_j \Rightarrow \varphi_i)) \Rightarrow (F \Rightarrow \varphi_i))$
(modus ponens, twice) $F \Rightarrow \varphi_i$

φ_i **is an axiom or is in** Γ**:** Insert the following three steps:
(axiom) $\varphi_i \Rightarrow (F \Rightarrow \varphi_i)$
(axiom) φ_i
(modus ponens) $F \Rightarrow \varphi_i$ □

4 Implicational completeness

A Frege system is **complete**, if for every tautology F, we have $\vdash^* F$.

A Frege system is **implicationally complete**, if for every semantic implication[1] $(F \Rightarrow G)$, we can derive G from F, that is, $F \vdash^* G$.

The implicational completeness of our example system follows from its completeness and the deduction theorem: since $F \Rightarrow G$ is a tautology, $\vdash^* (F \Rightarrow G)$, thus $F \vdash^* G$.

It remains to show the completeness. Let us split a refutation of an unsatisfiable formula into two cases ($x = 0$, $x = 1$) and then combine the two refutations (that exist by induction):

- given an unsatisfiable F, assume that $F, x \vdash^* \texttt{False}$ and $F, \neg x \vdash^* \texttt{False}$;
- combine the two derivations: by the deduction theorem the first derivation can be transformed into $F \vdash^* \neg x$, now we can use the second derivation.

Where is the end (that is, the base) of this induction? This is when we considered all the variables occurring in F, that is, it remains to refute $F, \ell_1, \ell_2, \ldots, \ell_n$ for literals ℓ_i corresponding to variables x_i with the corresponding signs. Then $\neg F$ can be derived by the induction on the construction (structure) of F. (Left as an exercise.)

5 Equivalence of Frege systems

Theorem 3. *All sound and complete, implicationally complete Frege systems polynomially simulate each other.*

Proof. **Case 1.** In the case where the two systems use the same set of operations (or the operations can be rewritten without a blowup, for example, by the de Morgan rules), the proof is easy.

Consider two Frege systems, $\mathcal{F}_1$ and $\mathcal{F}_2$. Each rule of a Frege system is sound. Therefore, they are all semantic implications. By implicational completeness, each rule of $\mathcal{F}_1$ can be therefore simulated in $\mathcal{F}_2$. As these rules are constant-size objects, so the derivation is also constant-size. The same applies to the axioms (by completeness).

Let us simulate an $\mathcal{F}_1$ derivation in $\mathcal{F}_2$ step by step: just apply the above simulations to the particular use of the rules.

Case 2. If we change the operations, *deeply* nested formulas can blow up if a rewrite of these operations involve more than one occurrence of subformulas (for example, if we attempt to rewrite the equivalence into the de Morgan basis, $F \equiv G \iff (F \wedge G) \vee (\neg F \wedge \neg G)$, the

[1] Recall that semantic implication means that for all possible values of Boolean variables the implication is true; that is, the implication is a tautology.

formulas F (and G) appear twice in the expression, so if we continue to rewrite them, we can get an exponential blow-up; try translating some linear-depth formula like $(\dots((x_1 \equiv x_2) \equiv (x_3 \equiv x_4))\dots \equiv x_{n-1}\dots) \equiv x_n)$.

This can be overcome using the *indirect translation*, which reduces the formula depth. Consider a formula F.

- Let A be a subformula of $\approx$ half size (*see below what is size*) of F:
  $F = B(A)$ (think about a subtree A and remaining part B: "inner subtree" and "outer subtree").
- Replace F by the equivalent $(B(\texttt{True}) \wedge A) \vee (B(\texttt{False}) \wedge \neg A)$.
- Proceed inductively till depth-1 formulas (operations).

The operations themselves are translated at the lowest level, there is no recursion there, so no blowup at that point. The number of levels is *logarithmic*, so the size of the obtained formula is polynomial.

Let us check that the depth is logarithmic indeed: we use induction on the number of leaves (i.e., variables and constants $1/0$) in the formula. Let $D(F)$ be the minimum depth we can attain for a formula equivalent to the formula F, and let D_ℓ be the minimum depth we can attain for every formula with ℓ leaves. Our transformation shows that

$$D(B(A)) = \max(D(B(\texttt{True})), D(B(\texttt{False})), A, \neg A) + 2 \le \max(D(B), D(A)) + 3.$$

Since we took B and A to be the "halves" of the formula (and let us say it explicitly that now that "half size" means "half leaves"), $\max(\#\text{leaves}(B), \#\text{leaves}(A)) \le D(\#\text{leaves}B(A))/2 + 1$. Therefore, $D_\ell \le D_{\lceil \ell/2 \rceil + 1} + 3$ and thus we will reach a constant-size tree in a logarithmic number of steps, paying $+3$ for each step. (One can count only binary operations towards the depth and only variables towards the leaves, then the recurrence will be nicer without $\lceil\dots\rceil$, but the result is the same.)

It remains to learn how to work with this representation. These technical details can be found in the dissertation of Robert Reckhow [1]. □

6 Treelike vs Daglike mode

As all derivation systems, Frege systems can be used in a daglike mode or in a treelike mode. However, for Frege systems these modes are equivalent.

In order to transform a daglike derivation into a treelike one, we keep the whole "history" of derivation in the last derived formula. So we transform a daglike derivation

F_1

F_2

…

F_n

into the treelike derivation

F_1

$F_1 \wedge F_2$

...

$F_1 \wedge F_2 \wedge \ldots \wedge F_n$

F_n

The last step is made using a (sound) derivation rule

$$\frac{\Phi_1 \wedge \Phi_2}{\Phi_1}.$$

Consider another step

$$\frac{F_i, \qquad F_j}{F_k}.$$

For simplicity, let it be an application of a binary rule

$$\frac{\Phi_1, \qquad \Phi_2}{\Psi}.$$

We derive $F_1 \wedge \ldots \wedge F_k$ from $F_1 \wedge \ldots \wedge F_{k-1}$. For this, we need to extract F_i and F_j from the conjunction, apply the rule and derive F_k without losing any previous formulas. So the new rule looks like

$$\frac{\Phi_1 \wedge \Phi_2 \wedge \Phi}{\Phi_1 \wedge \Phi_2 \wedge \Phi \wedge \Psi},$$

which is obviously correct, because the old rule was correct. To be absolutely formal, we should use the commutativity and associativity rules to bring F_i and F_j to the front of the big conjunction.

It is easy to see that the proof size is increased polynomially.

7 Extended Frege is polynomially equivalent to Extended Resolution

Theorem 4. *Extended Resolution polynomially simulates Extended Frege.*

Proof (Sketch). Consider an Extended Frege refutation.

Recall Tseitin's translation and introduce a new variable for every node of every (sub)formula appearing in an Extended Frege proof. Thus each formula is converted into a CNF.

Look at an application of a derivation rule. A rule such as $\frac{A\ B}{C}$ is translated into $\frac{x_A\ x_B}{x_C}$, where x_A, x_B, x_C are extension variables representing the corresponding formulas. The task is to derive x_C, that is, to use x_A, x_B, and the clauses defining x_A, x_B, x_C to derive x_C.

Then one can proceed by two methods.

Method 1. Since Frege systems are equivalent (and it is easy to see that then Extended Frege systems are equivalent as well), one can consider a specific Frege system and simulate its rules by

resolution. (In the lecture we simulated modus ponens, but one needs to prove axioms, i.e., 0-ary rules, as well.) Why it will work is explained below in Method 2, but one can simply do it.

Method 2. In general, rules are translated to derivations of (conjunctions of) implied clauses. As we already know (from the respective hw2 exercise), implied clauses can be derived using resolution (with the weakening rule, but we also know that at the end we can get rid of it).

Note that we have a variable for each subformula, so we can apply these rules to extension variables. Combine all these resolution refutations into one resolution refutation using extension axioms. □

The opposite direction is trivial: the resolution rule is a particular case of a Frege rule

$$\frac{\Phi \vee \Gamma, \qquad \Psi \vee \neg\Gamma}{\Phi \vee \Psi}$$

applied to very simple formulas (disjunctions of literals Φ and Ψ and a variable Γ). Commutativity, associativity, and structural rules (for removing duplicated literals) complete the job.

8 Takeaway

For Frege systems, nothing matters: different rules, proofs vs refutations, treelike vs daglike, all of them are polynomially equivalent.

If we use the extension rule, then even the (weak) Resolution system turns into a (very strong) Extended Frege system.

Historical notes and further reading

See the previous lecture.

References

[1] Robert A. Reckhow. On the lengths of proofs in the propositional calculus. PhD thesis, University of Toronto, 1976. https://www.cs.toronto.edu/~sacook/homepage/reckhow_thesis.pdf