CSE 599S
Proof Complexity & its Applications
Lecture 1. Sept 30, 2020

What is a proof?

- Boole prop logic 1850's
- Frege quantifiers 1870's
- ...
- Gödel
- ...

Proofs as things computers could find

Davis and Putnam 1960
for all of 1st order logic
- based on propositional proofs

⇒ Cook "The complexity of theorem proving procedures"

Defined NP, NP-completeness

"26557 is composite" →

$L \in NP$ iff
$x \in L \Leftrightarrow \exists y . |y| \le |x|^{O(1)} \; V(x,y)$

(proof that $x \in L$)

"short easy-to-check proof that $x \in L$ — poly time

Other notion of proof:
Natural deduction
Math Induction ←
Truth tables
} easy to check

Definition (Cook-Reckhow 1979)
A proof system for L is a poly time verifier V s.t.

$x \in L \iff \exists y\ V(x,y)$ ←

$\Leftarrow$ Soundness
$\Rightarrow$ Completeness

L = TAUT, UNSAT

Truth tables: proofs are long but easy to check

Not original defn

f: onto map from $\Sigma^* \to L$
poly time computable

proof $\mapsto$ string being proved

Proof Complexity

Defn The complexity of a proof system V is smallest function $S: \mathbb{N} \to \mathbb{N}$ s.t. $x \in L \iff \exists y.\ |y| \in S(|x|).\ V(x,y)$

"How short can proofs always be as a function of the thing being proved."

Truth tables. For formula F in n vars

$S(n)$ $2^n \cdot |F|$ $2^{|F|} \cdot |F|$

Def^n V is poly bounded iff

$S(n)$ is $n^{O(1)}$

Cor: $L \in NP$ iff L has a poly bounded proof system.

Propositional Proof Systems

proof system for

$F \in$ TAUT - propositional logic tautologies

$\neg F$ UNSAT - unsatisfiable propositional logic formulas

$NP \stackrel{?}{=} coNP$

TAUT, UNSAT are coNP complete.

Thm $NP = coNP$ iff $\exists$ poly bounded propositional proof system.

CNF-UNSAT also coNP complete

Formula $\Rightarrow$ CNF formula

add extra vars for each subformula

Tseitin

$H = F \wedge G$ $\quad y_H \vee y_F \quad \overline{y_H} \vee y_G$

$\overline{y_H} \vee y_G \vee y_H$

Y output

U poly simulates V
proofs in U polynomial proofs in V.

Every algorithm that decides satisfiability of CNF formulas has a corresponding proof system

proof = trace of execution that failed to find SAT act.

Proof Systems involving inference

Proof line, Axioms, Inference rules

$L_1, L_2, \ldots L_t$

L_i either an axiom or follows from $L_{j_1} \ldots L_{j_k}$ $j_1 \ldots j_k < i$ by inference rule

---

d.a.g. graph of proof

axiom source L_i

$L_{j_1}, \ldots L_{j_k} \to L_i$ inference

tree-like general

Resolution for CNF-UNSAT

lines: clauses

axioms: input clauses of $F = \bigwedge_{i=1}^{m} C_i$

inference rule:

$$\frac{A \vee x \quad B \vee \bar{x}}{A \vee B}$$

refutation: derive empty clause $\perp$

Special case: tree resolution — inference graph is a tree

DPLL: Davis Putnam Logemann Loveland

remove all clauses with $\bar{x}$

shorten all clauses with x

backtrack

```
DPLL(F, A)                          | call DPLL(F, nil)
  while F contains a clause of size 1, x )
      F_{x←1}, A ← (A, x)
  If F is empty  halt and output A, yes
  If F contains ⊥ return (failure)
  else choose some literal x          nondet
       DPLL(F_{x←1}, (A, x))          recursive
       DPLL(F_{x←0}, (A, x̄))
  end
```

Thm DPLL tree size (recursion) = Tree resolution size

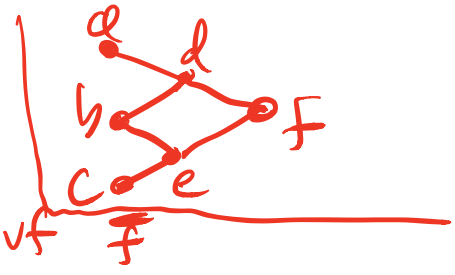
eg Pebbling formula

F a, b, c,

① $\bar{a} \vee \bar{b} \vee d$

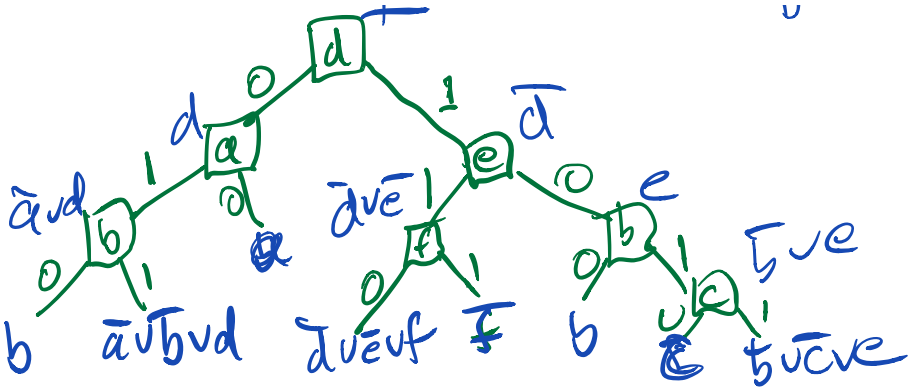
② $\bar{b} \vee \bar{c} \vee e$

③ $\bar{d} \vee \bar{e} \vee f$



1162

Dana Moshkovitz



CDCL unit-directed clause-learning solver