CS 228 tut 3

7.12

P vs NP Fallacy

Suggested algo to check if F is satisfiable

X
1) Consider ¬F
2) Convert ¬F into CNF via Tseitin (G)
3) Check validity of G
   If yes, return unsat
   If no, return sat

Key: Tseitin encoding does NOT preserve validity.

$F \equiv (p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_2) \wedge (\neg p_1 \vee \neg p_2)$

F is unsat.

$\neg F$: $(p_1 \wedge p_2)^{p_3} \vee (p_1 \wedge \neg p_2)^{p_4} \vee (\neg p_1 \wedge p_2)^{p_5} \vee (\neg p_1 \wedge \neg p_2)^{p_6}$

on simplifying. this is VALID. (oops.)

G, which is Tseitin for ¬F

$(p_3 \vee p_4 \vee p_5 \vee p_6) \wedge (\neg p_3 \vee p_1) \wedge (\neg p_3 \vee p_2) \ldots$

m: p_3, p_4, p_5, p_6 are assigned 0.

G is not valid

7.20

Can ignore valid clauses in resolution

a, b) Subjective

c) If you can prove this, the answer to (b) is "it doesn't matter"

To prove:
If Σ is unsat, then it has a resolution derivation of $\perp$ WITHOUT using valid clauses.

Induction on number of atomic proposition in Σ.

(implicitly assumes Σ is finite. even if infinite, as will discuss later, can use ideas of "COMPACTNESS" and minimal unsat core)

Base Case: 1 atomic prop

$P, \neg P$, $P \vee \neg P$ ✓ accounted

Induction Step:

Suppose true for n vars

$P, P_1, \ldots, P_n$ occur in Σ

$\Sigma = \Sigma_0 \quad \Sigma_1 \quad \Sigma_* \quad \Sigma_{valid}$

clauses $\quad P \quad \neg P \quad$ no $P \quad P, \neg P$

with

Σ unsat: for every model, there is some formula that isn't satisfied

Σ_0': Σ_0, but delete p from clauses

Σ_1': Σ_1 " " $\neg p$ " "

Case I: $m(p) = 1$

All formulae in Σ_{valid}, Σ_0 are satisfied

$\Sigma_1' \cup \Sigma_*$ is unsat [prove by contradiction]

this uses only n variables, and p is not among them

Appeal to induction hyp to derive $\bot$

Case II: $m(p) = 0$

Symmetric, replace Σ_1' by Σ_0'.

So from ind. hyp.

$\Sigma_1', \Sigma_* \vdash \bot$ $\quad$ $\Sigma_0', \Sigma_* \vdash \bot$

WITHOUT using valid clauses

If any of these proofs use formulae SOLELY from Σ_*, then we're done.

So, we assume that Σ_1' and Σ_0' necessarily are used in the derivation

just $\neg p$ $\quad$ just p

$\Sigma_1, \Sigma_* \vdash \neg p$ $\quad$ $\Sigma_0, \Sigma_* \vdash p$

just copy the proofs verbatim is used.
$\neg R$ can't be a PIVOT

Finally, just combine. □

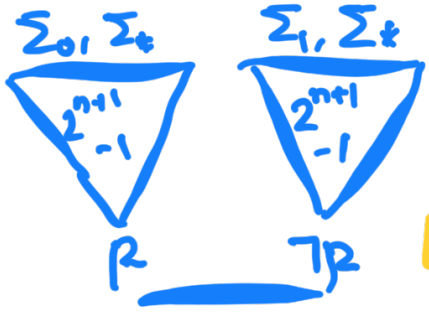
8.3 Size of Proof.

Key: Corollary of proof of completeness, induction again

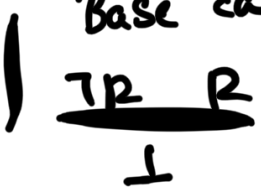
$\Sigma_0, \Sigma_1, \Sigma_*$ unsat, $n+1$ vars

$\Sigma_0', \Sigma_* \vdash \bot$ } unsat, n vars

$\Sigma_1', \Sigma_* \vdash \bot$



Base case



$: 3 = 2^{n+1} - 1$

[as we argued in prev question]

Counting size as # of formulae

size almost $\bot$

$2(2^{n+1}-1) + 1$

$2(2^{n+1}-1)$: 2 trees

$+1$: final step $= 2^{n+2} - 1$

8.8 Slim Proof

Key: Intuit what the notation is

trying to achieve ... [illegible]
and assignments to variables, and how they affect evaluation.

$C|_\ell$: assign ℓ to be true. What does the clause simplify to

$p \vee \neg q \vee r$.

$C|_p = T$
$C|_q = p \vee r$
$C|_{\bar{q}} = p \vee \neg q \vee r$

i) $F|_\ell$ has proof of $\bot$

$F \wedge \ell$

If derivation uses just F_*, then done
$F \wedge \ell$ $F_* \subseteq F$, use this

otherwise $F_1, F_* \vdash \bar{\ell}$

F_0, F_1, F_*, ℓ

Resolve ℓ with everything in F_1 to get F_1'. Now use F_1', F_* to complete proof

F_1 resolved with ℓ is $F_1|_\ell \subseteq F|_\ell$

m+1 n+1
m+n

8.2 $k \geq$ width(F)
slimest($F|_\ell$) $\leq k-1$
slimest($F|_{\bar{\ell}}$) $\leq k$

F_0, F_1, F_*

Prove slimest(F) $\leq k$

1) Look at the proof of $\bot$ from $F|_\ell$

max width $k-1$
F_1', F_*
(adding back $\neg\ell$
$F_1, F_* \vdash \neg\ell$
max width = k

For convenience, assuming $\ell = p$

$F_0, F_1, F_* \vdash \neg\ell$
$\leq k$ — use only these

2) Resolve $\neg\ell$ with everything in F_0, to get $F_0|_{\neg\ell}$ (width $k-1$) $[F_0']$

3) Finish off with the k-width proof of $\bot$ from $F|_{\neg\ell}$, which is $F_0|_{\neg\ell}, F_*$
↑ aka F_0'

Internalise the notation in terms of resolving with unit clause, and appreciate the (slightly) clever way of ordering

proofs.

9.12 Analysis of the HornSAT algorithm.

A change in truth value could only occur in the while loop.

while $m \neq ((p_1 \wedge \dots \wedge p_n) \Rightarrow p) \in H_s$ ← H_0

$m := m[p \to 1]$

two kinds of changes

in H_0:

since variables are only ever set to 1, H_0 will be permanently true

in H'

$p_1 \wedge \dots p_n \Rightarrow p_{n+1}$ ← hasn't been set to true yet

↑ all of these set to true from other clauses

↑ setting this to true makes this clause false

$T \Rightarrow \bot$ is wrong.

A clause (in H_s) changes truth value at most twice.

$T \longrightarrow \bot \longrightarrow T \not\longrightarrow$

↑ permanent (case 1)

Eg. $p_1 \Rightarrow p_2$, [starts vacuously true]

1) $\top \Rightarrow p_1$ [stands false]

$m[p_1 \to \top]$

$p_1 \Rightarrow p_2$ $\quad T \to \bot \to T$

2) $m[p_2 \to \top]$

9.15 Minimal unsat core of a 2-SAT formula.

Key: deduce from implication graph

How detect unsat?

For some atomic prop p, there is path from p to $\neg p$ as well as from $\neg p$ to p.

Use these paths you detect via any graph search algo!

Look up: detect scc's through DFS.

$p_1 \vee p_2$ $\quad \neg p_1 \vee p_2$ $\quad p_1 \vee \neg p_2$ $\quad \neg p_1 \vee \neg p_2$

p_1 to $\neg p_1$ →→

$\neg p_1$ to p_1 →→

Why minimal: Remove something from path, one chain of implication breaks

Will try to get back with more rigorous justification, and/or correct any inaccuracy/ loose ends.

↳ this might still contain p', $\neg p'$ in scc. iterate if this is case.

Or can we search cleverly to avoid this?

— In an scc, just identify shortest round trip $p \to \neg p \to p$.

If mistake/ gap raise it on chat! Please