Graduate Complexity Theory

at Carnegie Mellon University

Fall, 2017

Taught by Ryan O'Donnell

Handwritten lecture notes

(watch out, they contain a few bugs :)

+

11 homeworks and 2 tests

Complete set of 28 lecture videos on YouTube:
https://www.youtube.com/watch?v=pRnnEOAQZF8&list=PLm3J0oaFux3b8Gg1DdaJOzYNsaXYLAOKH

Course homepage:
http://www.cs.cmu.edu/~odonnell/complexity17/

Lecture contents below (homeworks+tests start on page 341)

#	pg. in this pdf	Title	Reading
1	4	Overview of the course	*Review:* Arora--Barak Chapters 1 (except 1.7), 2, and 4
2	12	Hierarchy theorems: time, space, and nondeterministic versions	*Reading:* Arora--Barak Chapters 3.1, 3.2; also 1.7 if you're interested in the O(T log T) simulation
3	21	Hopcroft--Paul--Valiant Theorem	*Reading:* The original paper
4	31	Circuits	*Reading:* Arora--Barak Chapters 6.1--6.7
5	40	Probabilistic complexity classes	*Reading:* Arora--Barak Chapters 7.1--7.5 (except not 7.5.2)
6	49	Quasilinear Cook--Levin Theorem	*Reading:* Section 2.3.1 in this survey by van Melkebeek, these slides by Viola
7	57	The Polynomial Time Hierarchy and alternation	*Reading:* Arora--Barak Chapters 5.1--5.3
8	66	Oracles, and the Polynomial Time Hierarchy vs. circuits	*Reading:* Arora--Barak Chapters 5.5, 6.4. *Bonus*: improving Kannan's Theorem.
9	77	Time/space tradeoffs for SAT	*Reading:* Arora--Barak Chapter 5.4
10	94	Intro to Merlin-Arthur protocols: MA and MA	*Reading:* Arora--Barak Chapter 8.2.0
11	102	More on constant-round interactive proof systems	*Reading:* Arora--Barak Chapter 8.2.4, Chapter 8 exercises
12	114	Approximate counting	*Reading:* Arora--Barak Chapter 8.2.1, 8.2.2
13	121	Valiant--Vazirani Theorem and exact counting (#P)	*Reading:* Arora--Barak Chapters 17.0, 17.1, 17.2.1, 17.3.2, 17.4.1
14	134	Toda's 1st Theorem, and the Permanent	*Reading:* Arora--Barak Chapters 17.4, 8.6.2, 17.3.1
20 (sic)	267	Permanent is #P-complete	*Reading:* PowerPoint slides
15	144	Algebraic circuit complexity	*Reading:* Arora--Barak Chapter 16.1. *Bonus*: "algebraic NP vs. P" vs. "Boolean NP vs. P".
16	162	Instance checking and the Permanent	*Reading:* Arora--Barak Chapter 8.6
17	172	IP = PSPACE	*Reading:* Arora--Barak Chapters 8.3, 8.4
18	182	Random restrictions and AC0 lower bounds	*Reading:* Arora--Barak Chapter 14.1
19	193	The Switching Lemma	*Reading:* My old notes on Razborov's proof
21	198	Monotone circuit lower bounds	*Reading:* Arora--Barak Chapter 14.3
22	209	Razborov-Smolensky lower bounds for AC0[p]	*Reading:* Arora--Barak Chapter 14.2
23	218	Toda's 2nd Theorem & lower bounds for uniform ACC	*Reading:* Arora--Barak Chapters 17.4.4, 14.4.2; and, B.2 of the Web Addendum (with correction)
24	227	Hardness vs. Randomness I	*Reading:* Arora--Barak Chapters 20.0, 20.1
25	240	Hardness vs. Randomness II	*Reading:* Arora--Barak Chapters 20.2
26	248	Hardness amplification	*Reading:* Arora--Barak Chapters 19.0, 19.1
27	254	Ironic Complexity	*Reading:* Arora--Barak Web Addendum

Additional resources

Textbooks:

- *Computational Complexity: A Modern Approach*, by Arora and Barak
- *Computational Complexity*, by Papadimitriou
- *Theory of Computational Complexity*, by Du and Ko
- *Complexity Theory*, by Wegener
- *Computational Complexity: A Conceptual Perspective*, by Goldreich
- *The Complexity Theory Companion*, by Hemaspaandra and Ogihara
- *Theory of Computation*, by Kozen
- *Computability and Complexity Theory*, by Homer and Selman
- *Structural Complexity I* and *II*, by Balcázar, Díaz, and Gabarró
- *Boolean Function Complexity: Advances and Frontiers*, by Jukna
- *The Nature of Computation*, by Moore and Mertens
- *Introduction to the Theory of Computation*, by Sipser

Lecture notes:

- Van Melkebeek: 2007, 2010, 2011, 2015, 2016
- Harsha: '11/'12, '12/'13, '13/'14
- Trevisan: 2001, 2002, 2004, 2008, 2010, 2012, 2014
- Sudan: 2002, 2003, 2007, 2009
- Spielman: 1998, 1999, 2000, 2001
- Moshkovitz: 2012, 2016
- Katz: 2005, 2011
- Umans
- Hansen 2010
- Vadhan 2002
- Cai
- Gács and Lovász
- Beame 2008
- Arora 2001
- Miltersen 2006
- Håstad
- M. Naor '04/'05
- Guruswami--O'Donnell 2009

Videos:

- Regan 2015

Lecture 1 - Intro to the course

Sunday, September 3, 2017 1:59 PM

Computational Complexity 15-855

Ryan O'Donnell

www.cs.cmu.edu/~odonnell/complexity17

- homework (#1 is out, due in 1 week)
- piazza (all announcements)
- Gradescope
- policies (homework, tests, grading, ...)
- textbook info
- prereqs

Complexity Theory: Showing natural algorithmic tasks cannot be done efficiently.

Algorithms Theory: " " " " can " " " .

(We're much much better at the latter than the former.)

(Why? 1. Cxty: proving a negative is harder than just exhibiting an alg.

2. Algs are amazing - lots of surprising, powerful, efficient algs. Hard to rule out!)

(But we'll try. It's a mature field - we know what we don't know.)

Today: ① Topics for the course.

(oppor. for →) ② "Reminders" of basic cxty ← (abbrev. for complexity)

<u>Topics</u>. Focus:

- time cxty
- circuit cxty
- randomness

(As I mentioned, it's a <u>very</u> mature field. Enormous amount known, many subareas. CCC '17 had so many topics: Algebraic cxty, proof cxty, logic, avg. case cxty, comm. cxty, inapproximability, crypto, property testing, quantum cxty.....
Not time for all of it. → I didn't even put <u>space</u> cxty!
This is a real "laying the groundwork" course.
Only a few results from 2000+; just one (Williams's Thm) from last 10 years!)

(Let's start with time complexity.)

<u>def</u>: $\underline{TIME(t(n))}$ = $\boxed{\text{all languages decidable in } O(t(n)) \text{ steps on inputs of length } n.}$

$L \subseteq \{0,1\}^*$ (or Σ^*) $\equiv$ $f: \{0,1\}^* \to \{0,1\}$ $(f(x)=1 \Leftrightarrow x \in L)$ $\equiv$ <u>decision</u> problem

↓

task with a yes/no answer

(can also consider search/sampling/promise/etc. problems, but we'll keep it simple for now) problems, but we'll keep it simple for now).

In what **Model**: multitape Turing Machine (TM)

(Let's say, for concreteness. There are several possibilities, and it matters somewhat, for log-factors, e.g., but we'll discuss later.)

→ $O(\cdot)$ baked into definition. We never care about const. factors.

Time Hierarchy Theorem (early '60s) (more time = more power)

$\Rightarrow$ $\mathrm{TIME}(n^2) \subsetneq \mathrm{TIME}(n^3)$, e.g.

i.e., $\exists L$ solvable in $O(n^3)$ time, not solvable in $O(n^2)$ time.

(a genuine lower bound!)

$$\Rightarrow \quad P \subsetneq E \subsetneq EXP$$

$\mathrm{TIME}(\mathrm{poly}(n)) = \bigcup_c \mathrm{TIME}(n^c)$ $\qquad$ $\mathrm{TIME}(2^{O(n)})$ $\qquad$ $\mathrm{TIME}(2^{\mathrm{poly}(n)})$

$\mathrm{SPACE}(s(n)) = L$ decidable using $\leq s(n)$ tape cells

(Space is at least as valuable as time.)

$$\left.\mathrm{TIME}(f(n)) \subseteq \mathrm{SPACE}(f(n))\right\} \subseteq \mathrm{TIME}(2^{O(f(n))})$$

$$\Rightarrow \quad P \subseteq PSPACE \subseteq EXP$$

$$PSPACE = \mathrm{SPACE}(\mathrm{poly}(n))$$

$\subsetneq$

$\mathrm{SPACE}(\mathrm{poly}(n))$

$\subsetneq$

L

$=$

$\mathrm{SPACE}(\log n)$

$\exists$ Space Hier. Thm, $\Rightarrow L \subsetneq \mathrm{PSPACE}$

(Space is in fact more valuable than time. We'll show...)

[HPV'77]: $\mathrm{TIME}(t(n)) \subseteq \mathrm{SPACE}\left(\frac{t(n)}{\log t(n)}\right)$! (Need $t(n) \geq n$) (Need constructibility)

$\subsetneq \mathrm{SPACE}(t(n))$, by Space H.T.

(Then NP came into the picture & messed up paradise.)

Nondetermism – hypothetical kind of machine that makes guesses, said to succeed if it ever guesses right

$\mathrm{NP} = \mathrm{NTIME}(\mathrm{poly}(n))$. ('70s)

MANY alg problems "obviously" in NP, not obviously in P.

$P \subseteq NP$ (of course, $\mathrm{DTIME} \subseteq \mathrm{NTIME}$) $\boxed{P \overset{?}{=} NP}$ \$1MM Question

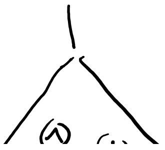
(epic failure to prove lower bounds...)

rem: NTIME Hier Thm. exists (and quite important; we'll prove it!)

$\Rightarrow NP \subsetneq NE \subsetneq NEXP$

$\uparrow$ $\mathrm{NTIME}(2^{O(n)})$

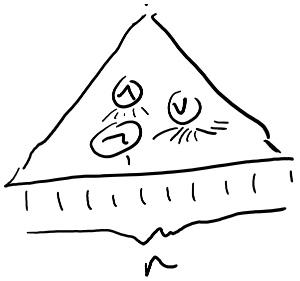
('80s) Circuits!



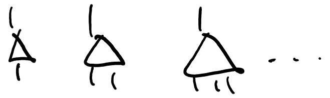
"Non-uniform" model

$\downarrow$

Circuits.



↓
diff "alg." for each input len. n.



def: <u>P/poly</u> = langs L decidable by poly(n) size circuit families

easy: $P \subsetneq P/\text{poly}$ (Inclusion is easy. Non-equality because...)

↖ contains undecidable langs!

(But these heavily exploit the catch that you just need a circuit to <u>exist</u> for each input len.)

Stronger than "$NP \not\subseteq P$": $NP \not\subseteq P/\text{poly}$ ("circuit")

|||

SAT not solvable by poly-size ckt families

(In '80s, people tried to prove this harder stuff! Why?

① "Feels" about equally difficult (?) Getting to have a diff. SAT alg for each input len. doesn't seem to help.

② Circuits are super-tangible, concrete, combinatorial. Feels easier to prove L.B.'s against them.)

[Hås'86, RS'87] : (we'll prove) ∃ language in P with no poly-size <u>constant-depth</u> circuits.

(in fact in L, in SPACE(1), reg. lang) 2, ⊕

[Razb. '85] CLIQUE (∈ NP) requires exponential-size AND/OR circuits (no NOTs) ("monotone")

(we'll prove)

(we'll prove) (no NOTs) ("monotone")

N. Blum '17: extended techniques to allow NOT gates!

$\therefore NP \not\subseteq P/poly$

[É. Tardos '87]: Razb. techniques apply to some langs in P. $\therefore P \not\subseteq P/poly$?

(Very stuck) Things we can prove:

thm: $\exists L \in P$ requiring circuits of size $3n - o(n)$

all 2-input gates

[N. Blum '84]

3.01 [FGHK'15].

Santhanam Theorem '07: $\forall c$ (e.g., $c = 1000$),

$\exists L \in$ (promise-) MA ← a slight randomized twist on NP

(↑ will discuss later)

s.t. L not computable by $O(n^c)$-size circuits.

$\not\rightsquigarrow$

$\approx NP \not\subseteq SIZE(n^{1000})$ vs. $\approx NP \not\subseteq P/poly$ (quantifiers reversed)

↳ $\exists L \in \approx NP$

$\forall c$ L not in $SIZE(n^c)$.

(relevant we're talking about non-uniform classes here)

vs. $\approx NP \not\subseteq TIME(n^{1000})$?! → $P \not\subseteq TIME(n^{1000})$ by THT!

Williams Theorem '11: $\exists L \in NEXP$ (!) not computable in

"$AC^0[6]$" ← poly-size, const. depth circuits

w/ AND, OR, NOT, mod-6 gates

(Rem: RS'87 ⇒ if you replace 6 by a prime, like 5 or 7, we know a lang. in P — a regular lang! — that works!)

Conj: $MAJ = \{x : \#\ 1\text{'s in } x \text{ is } \geq \frac{|x|}{2}\} \notin AC^0[6]$!

↘ $\in P$ (in L, even)

Randomness: (An extremely important topic for several reasons.

1. Heavily used in practical algs. Seems like it can give some speedups.
2. But doesn't seem like more than polynomial speedups
3. Some thms in cxty, despite not mentioning randomness, seem to require randomness to prove!)

BPP = langs decidable by poly-time randomized TMs (answer correct whp)

$P \subseteq BPP \subseteq \overset{?}{\sqcup} EXP$ (by trying out all "coin flips")

(= is possible, AFAWK! But doesn't seem plausible randomness buys you exponential time)

(we also know better upper bounds not using just TIME)

$\subseteq PSPACE$

$\subseteq NP^{NP}$ (nondet poly-time w/ oracle access for SAT)

?? • There are problems — eg. PIT ("polynom. identity testing") with $PIT \in BPP$ ✓ but $PIT \overset{?}{\in} P$ unknown.

But: • BPP = P has strong evidence:

"Hardness vs. Randomness Paradigm" [NW'94]

— strong ckt lower bounds for langs in expon. time, use their truth tables as PRGs "derandomization"

- strong ckt lower bounds for

$\underbrace{\text{use their truth tables as PRGs}} \rightarrow$ "derandomization"

- e.g.: [IW'97]: (will prove) If $\exists L \in E$ ← TIME$(2^{O(n)})$ requiring circuits of size $2^{\Omega(n)}$, then BPP = P.

- Also: derandomization ⇒ circuit lower bounds:

  [KI'03]: (will prove) If PIT $\in$ P then NEXP $\not\subseteq$ P/poly (or a similar L.B. for "algebraic circuits")

(That's some idea of where we're heading, but more along the way...) Re SAT & NP....

- if NP = P, what else $\in$ P? ⇝ PH, "poly-time hierarchy"
- what is cxty of counting the # of satisfying assignment to a CNF? Harder/easier than PH? (Toda's Theorem)
- does interaction help with proofs?
  randomness?
  both?
- why is proving P ≠ NP so hard?

Lecture 2 - Hierarchy Theorems & Models

T.H.T. at a high level

(e.g. $t_1(n)=n^2$, $t_2(n)=n^6$)

Assume "$t_2(n) \gg t_1(n)$".

Then $\exists L$ s.t. $L \in \mathrm{TIME}(t_2(n))$, $L \notin \mathrm{TIME}(t_1(n))$.

(more time = more langs)

Proof sketch

Fix "simple" $[\cdot]_{DTM} : \Sigma^* \to \{\text{Turing Machines}\}$

s.t. $\forall$ TMs M, $\exists x$ s.t. $M = [x]_{DTM}$.

Define L via a time-$t_2(n)$ machine D that decides it.

$D(x)$: Let $M = [x]_{DTM}$, simulate $M(x)$

for $t_{1.5}(|x|)$ steps, $\boxed{t_1 \ll t_{1.5} \ll t_2}$

Do the opp.: if $M(x)$ acc $\to$ rej

if $M(x)$ rej $\to$ acc.

(If $M(x)$ doesn't halt, doesn't matter

say, acc.)

Assume doable in $t_2(|x|)$ steps,

$L \in \mathrm{TIME}(t_2(n))$? ✓

$L \notin \mathrm{TIME}(t_1(n))$?

Let Q be any TM running in $\le t_1(n)$ time.

Consider any x s.t. $[x] = Q$.

$Q(x)$ finishes in $t_1(|x|) < t_{1.5}(|x|)$ time.

$\therefore$ $D(x)$ finishes sim., does opp.

$\therefore$ $Q(x) \neq D(x)$, i.e. Q gets "$x \overset{?}{\in} L$" wrong.

$\square$

Issue 1: (this proof has a bug)

$\mathrm{TIME}(t_1(n))$ allows for $\underline{\underline{O}}(t_1(n))$ steps

(for reasons we'll see, we really want this in our def^{n}. So D has to "beat" $C \cdot t(n)$ time simultaneously $\forall C$.)

Need $t_1 = o(t_{1.5})$.

Also: need that $Q = [x]_{DTM}$ for infinitely many x. (Ok: just allow encodings to have arbit many 1's tacked on end)

Thus $C \cdot t_1(|x|) < t_{1.5}(|x|)$ eventually kicks in.

Issue 2: Need one <u>fixed</u> time-$O(t_2)$ machine U that can simulate <u>any</u> machine M for $t_{1.5}(n)$ steps.

- U has to be "universal" — able to handle M's any (constant) number of
  - states (OK: UTM)
  - tape symbols (OK: encoding)
  - tapes (!) (hmm)

Simulating $O(1)$-tape TM by 2-tape TM

Easy: time $T \longrightarrow$ time $O(T^2)$

Hard: [HS'64] time $O(T \log T)$

(see Arora-Barak) (Trick involves not zigzagging to go between head pos's, but keeping them in place & shifting tape around. Need <u>amortized analysis</u>.)

$\therefore$ Need $t_2 = t_{1.5} \log t_{1.5}$

Issue 3: Simulator needs to be able to clock out $t_2(n)$ time. (If $t_2(n)$ is some insane hard-to-compute fcn, might take a long time to fig. out how long to run for.)

def: $t(n)$ is time-constructible if ∃ a TM that on inputs of len. n, writes $1^{t(n)}$ in $O(t(n))$ time.

(Rem: All "normal-looking" fcns $t(n) \geq n$ are time-constr'ble.)

THT: Let $t_1(n) \geq n$. Let $t_2(n) = \omega(t_1 \cdot \log t_1)$ be time-constructible. Then ∃ L s.t. $L \in TIME(t_2)$, $L \notin TIME(t_1)$.

* * *

(Other hierarchy theorems?)

Space: Easy to sim. k-tapes, space $s(n)$ by 2-tapes, space $O(s(n))$.

∴ Space HT: Let $\log n \leq s_1(n) = o(s_2(n))$, where $s_2(n)$ space-constructible. Then $SPACE(s_1(n)) \subsetneq SPACE(s_2(n))$.

NTIME (Recall: nondet machines can "guess" bits (to their tape). String x is "overall" accepted if ∃ guesses that make the machine accept.)

(Sim. to space, for NTIME you can get eff. simul$^{\underline{n}}$)

[BGW'70]: nondet k-tape ⟿ nondet 2-tape
ook reibach egbreit

time T ⟶ time $O(T)$

(The trick, in words. On 2nd tape, guess all the T state/k-syms-under-heads over time. Then verify truth by sim'ing the k tapes sep'ly & sequentially, assuming guesses correct.)

So ... $\text{NTIME}(t_1) \subsetneq \text{NTIME}(t_2)$ if $t_1 = o(t_2)$?

No! Problem: NTIME not closed under complement!

(I.e. reversing acc/rej states of a nondet. machine does not make its language the complement! What to do..?)

↓ True for NSPACE since it is closed under complem.

(Immerman-Szelepcsényi)

Idea: Can do the opp. of a time $t(n)$ nondet. machine in $2^{t(n)}$ time

(Even det'ically. Just brute-force sim. all computation paths.)

(We'll now prove NTHT using "delayed diagonalization")

Nondet. THT: Let $t(n), T(n)$ be $\geq n$ and time-const'ble, with $t(n+1) = o(T(n))$. Then $\mathrm{NTIME}(t) \subsetneq \mathrm{NTIME}(T)$.

Proof (sketch): Will have $L \subseteq \{1\}^*$. Input 1^n will just be called "n". M_n will be the nondet. TM $[\mathrm{binary}(n)]_{NTM}$.

(by Žák'83, M.Li'84, improves Cook, simpl's SFM'78)

Inputs divided into consecutive intervals

$$[\underset{\substack{\uparrow \\ 0}}{\ell_1}, u_1], [\underset{\substack{\shortparallel \\ u_1+1}}{\ell_2}, u_2], [\underset{\substack{\shortparallel \\ u_2+2}}{\ell_3}, u_3], \ldots \quad \text{with } u_j \ggg \ell_j.$$

(Instead of the diagonalizing TM trying to differ from M_n on n, will just try to differ from M_n somewhere in $[\ell_n, u_n]$.)

Nondet TM $D(1^n)$:

- Find j s.t. $\ell_j \le n \le u_j$.
- If $n \ne u_j$: Simulate $M_j(n+1)$ for $T(n)$ steps. (Don't do opposite!)

  not n ↙

- Else if $n = u_j$: Deterministically brute-force sim. $M_j(\ell_j)$ for $\log_2 T(n)$ of its steps & do the opposite.

$(L = L(D).)$ $D \in \mathrm{NTIME}(T(n))$? ✓ (uses [BGW])

Suppose for contra that M is a time $t(n)$ NTM s.t. $M(n) = D(n)$ $\forall n$. Let j be such that $M = M_j$ (Need that trick that this holds for ∞'ly many n.)

By assumption, $M(\ell_j) = D(\ell_j)$.

What does $D(\ell_j)$ do? Sims $M_j(\ell_j+1)$ for $T(\ell_j)$ steps.

$M \nearrow$ ↓ $t(\ell_j+1)$ time $\nearrow o(\cdot)$

$$\therefore \quad D(\ell_j) = M(\ell_j+1) = D(\ell_j+1) \text{ by assump.}$$

What does $D(\ell_j+1)$ do? Sims $M(\ell_j+2)$ for $T(\ell_j+1)$ steps.

$$\therefore \quad D(\ell_j+1) = M(\ell_j+2) = D(\ell_j+2) \ldots\ldots$$

$$\cdots\quad D(u_j-1) = \underset{\underset{M(\ell_j)}{\parallel}}{M(u_j)} = D(u_j).$$

But what does $D(u_j)$ do? Simulates $M(\ell_j)$ for $\log_2 T(u_j)$ of its steps, (correctly) does opposite.

$\uparrow\!\lrcorner \geq \log_2(u_j)$. Take $u_j = 2^{t(\ell_j)}$.

(Remarks: constructibility used to show step 1 of D is possible. Since $t(n) \geq n$, this sequence grows so fast that Step 1 is linear time, too.)

$= t(\ell_j)$. So $D(u_j)$ succeeds in doing the opposite of $M(\ell_j)$ – contradiction. $\square$

Today: <u>[HPV'77] Theorem:</u>

Let $T(n) \geq n$ be "nice" (see HW1.2 footnote).
(appropriate constructibility)

Then $\mathrm{TIME}(T(n)) \subseteq \mathrm{SPACE}(T(n)/\log T(n))$.

E.g.: $\mathrm{TIME}(n) \subseteq \mathrm{SPACE}(\frac{n}{\log n})$.
(In fact, it's enough to prove this particular case, by padding.)

Cor: $\mathrm{TIME}(T(n)) \subsetneq \mathrm{SPACE}(T(n))$, by Space Hier!
(Space strictly <u>more</u> powerful than time.)

Rems: • Known to be very model-<u>in</u>dependent. ☺
(We'll do for multitape TMs, but also proven for RAMs, "pointer machines", . . . So it's really telling us sthg about TIME/SPACE, not just Turing Machine weirdness.)

• These ideas ++ ("Alternation, Σ_4, PH", <u>very</u> sharp Nondet (alt'ing) THT, . . .)
↓
[PPST '83]: $\mathrm{TIME}(n) \subsetneq \mathrm{NTIME}(n)$ (P≠NP for lin.time!!)
$n\sqrt{\log^* n}$ [Santh '01]
(<u>doesn't</u> extend upward → padding goes wrong direc.)

(seemingly) <u>De</u>pends on multitape TM model! ☹

Proof (sketch)

Let M be a time-O(T) machine. Need to sim. it by a space-$O(T/\log T)$ machine.

For simplic., we'll just show $O(\frac{T}{\log T}\cdot\log\log T)$

(qualitatively, just as good)

First: Let $B = T^{2/3}$, ← (not super-imp.; need $\geq \sqrt{n}, \leq n^{.99}$)

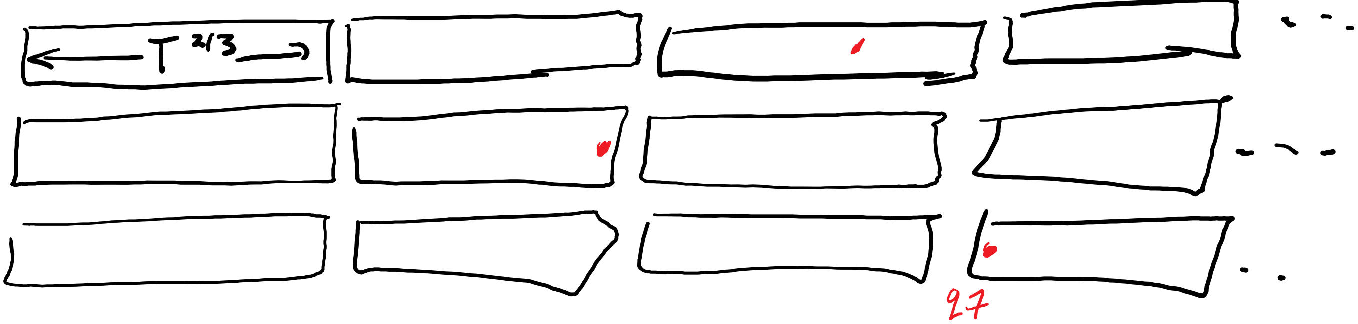
assume (HW #1.2) M is "B-block-respecting".

Let $V = \frac{O(T)}{B} = O(T^{1/3})$, # time "epochs"

(heads only go betw. blocks at epoch-transitions)

(High-level picture....)

M: e.g. k=3 tapes

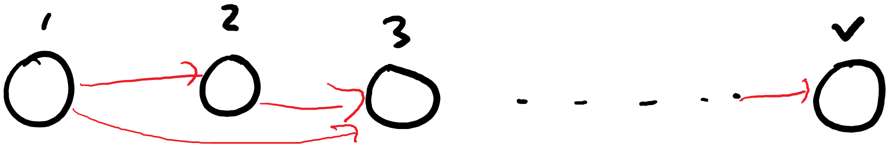


For epoch $t = 1, 2, \ldots, v$, define:

- state(t): TM state, head locations at end of epoch
  ↳ ⇒ what blocks worked on
- contents(t): contents of those blocks at end (where block-resp'ing comes in)

(Define something called) **Computation graph G**

V = epochs



Edges: • $\forall t$, $(t-1, t)$ (Reflects fact that to simul. t^{th} epoch, need to know state(t).)

(What else need to know to sim. t^{th} epoch? Prior contents of the blocks to be worked on.)

• $\forall t$, $\forall$ blocks b touched in epoch t, for "most recent" $s < t$ that touched b,

↳ add (s,t) to E.

G is a dag, w/ max in-degree $k+1 = O(1)$.

Idea: To simulate t^{th} epoch, only need to know

state(t), contents(s) $\forall$ preds s of t

$\downarrow$ $O(\log T)$ bits $\downarrow$ $O(B) = O(T^{2/3})$ bits each

Could sequentially compute for $t = 1, \dots, v$

$\longrightarrow v \cdot B = O(T)$ space.

(But to do one particular epoch, only need to know k contents,...)

Idea 2: Perhaps could selectively delete some contents(·), only recompute "if needed".

* * *

Assume WLOG low-space simulating machine "knows" G.

How? Try all G, reusing space! $\to \tilde{O}(v) = \tilde{O}(T^{1/3})$

G def'd by state(t), t=1...v

(If you "guessed" (are trying) the wrong G, you'll notice it.)

* * *

Idea 2 can be implemented, only ever need to keep $O(\frac{v}{\log v} \cdot \log\log v)$ different contents(·)'s around. $\to O(\frac{T^{1/3}}{\log T} \log\log T)$.

(Boils down to ....)

(Boils down to . . . .)

$\log T$

Total space: ↗ $\times O(T^{2/3})$!

Pebbling

Game played on a dag, with multiple "sources", 1 "sink". E.g.

Rules:

(computing conts(t)) →

- Can put pebble on a vtx if all its incoming vtcs have pebbles
  (So can always put pebbles on sources.)

(erasing an old contents(t)) →

- Can always remove a pebble.

(final epoch)

- Goal: get pebble on sink, use as few pebbles as poss

(Actually play it on above.
Can do it in 4 pebbles. Get volunteer from audience!
Can you do it in 3? Not sure...)

(Obvious: can always pebble using ≤ #vtcs many pebbles.)

thm: (HPV) In dag with v vertices, max in-degree $C = O(1)$, can pebble with $O\left(\frac{v}{\log v}\right)$ pebbles. (Hidden const. is C^2, I think)

<u>Pf sketch</u> for $O\left(\frac{v}{\log v} \cdot \log\log v\right)$. (Close enough for our purposes.)

Rem: may take $\exp(v)$ <u>steps</u>.

① Hmwk #1c: <u>Depth reduction</u>.

edges $= m = O(v)$. (at most $C \cdot v$)

depth $= d \leq m = O(v)$

"k" $= \log d \sim \log v$

Take "r" $= \log k \sim \log\log v$.

$\therefore$ by deleting $\frac{r}{k} m = O\left(\frac{v}{\log v} \log\log v\right)$ ~~edges~~, can reduce depth to $\leq \frac{d}{2^r} = O\left(\frac{v}{\log v}\right)$

(just del. both → vertices
→ endpts of each edge)

Let P be those vtcs ("Permanent Pebble Positions")

⊛ Any path avoiding P has len. $\leq O\left(\frac{v}{\log v}\right) =: L$.

Pebbling strategy:

- topo sort $P \rightsquigarrow u_1, u_2, \ldots, u_{|P|} = \text{sink}$
- for $i = 1 \ldots |P|$,

  Depth-First-Pebble(u_i, dontDelete $= P$)

```
D.F.Pebble(u, dontDelete)
    for $w \in$ preds(u)    // $\leq c$
        if w not pebbled    (e.g., maybe in P, already done)
            D.F.Pebble(w, dontDel $\cup$ preds(u))
    Pebble w
    remove all except $\{w\} \cup$ Don'tDelete
```

Claim/ex: If every path to u avoiding Don'tDel has len. $\leq \ell$, then D.F.Peb uses $\leq c \cdot \ell + |\text{Don'tDel}| + 1$ pebbles // $\therefore c \cdot L + |P| + 1$ overall.

(Intuition: each rec. call from DFP spends c,

(Intuition: each rec. call from DFP spends C, depth of rec. $\leq l$ b/c can stop at Don'tDel)

overall.

✓

Final issue: Given G,

pebbling *strategy* must be comp'ble in low space.

(rec: strat. len could be ≈ 2^v steps)

Sol$^{\underline{n}}$ 1: Check all details.

(Our proof, including HW depth-reduction, very explicit. I'm pretty sure doable in $\tilde{O}(v)$ space $= \tilde{O}(T^{1/3})$. ☺)

Sol$^{\underline{n}}$ 2: *Any* p-pebble strat. easily comp'd in NSPACE($p \log p$) — just guess it!

↓

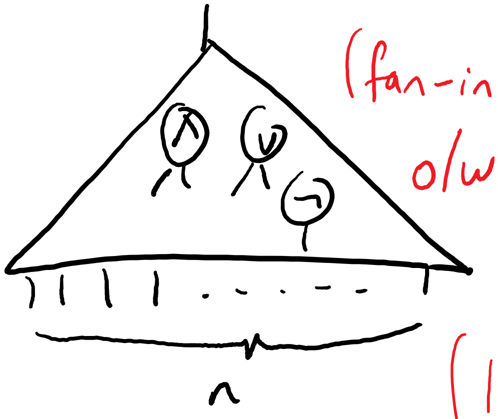
$\subseteq$ SPACE($p^2 \log^2 p$) by Savitch's Thm!

↑

$\tilde{O}(v^2) = \tilde{O}(T^{2/3})$ ☺.

□

Lecture 4 - Circuits



(fan-in 2 unless o/w specified)

(actual rep. is list of gates)

gate #	type	incoming #'s
1	input	
2	input	
⋮	⋮	
n	input	
n+1	∧	1, 3
n+2	¬	n+1
	⋮	
s	∧	99, 107

(let's say last line is output)

(always < gate #, so dag.)

$C: \{0,1\}^n \to \{0,1\}$

(identify ckt & fcn)

Cxty measures

Size: #gates (≈ sequential time)

depth: longest input/output path (≈ parallel time)

(As mentioned in lec 1, the notable fact abt. circuits is they have fixed input len.)

Fixed input lens = "non-uniform" model of comp.

def: Circuit family = $(C_n)_{n \in \mathbb{N}}$, C_n has n inputs. Computes a lang. $\subseteq \{0,1\}^*$.

def: $\mathrm{SIZE}(s(n)) = \{L \subseteq \{0,1\}^* :$ computable by a ckt family (C_n), $\mathrm{size}(C_n) \xrightarrow{} O(s(n))\}$

(baked in again)

def: $\underline{P/poly} = \mathrm{SIZE}(\mathrm{poly}(n))$. (we'll explain bizarre not^n later)

(cf. P)

def: $\underline{NC}$ = langs comp'ble by poly(n)-size, polylog(n)-depth ckt fams (cf. eff. parallel algs)

(Nick [Pipp]'s class, believe it or not)

def: AC^0 = " " " poly(n)-size, O(1)-depth ckt fams (constant(?) parallel time)

(alt. class; O = O(1)-depth)

↳ assume $\wedge$, $\vee$ allowed <u>unbounded</u> fan-in; size = n + # "wires"

(Back to weirdness of non-uniformity.)

Prop: Every unary lang $L \subseteq \{1\}^*$ in P/poly (AC^0).

(<u>Proof?</u>: For each n, output $0 = x_1 \wedge \neg x_1$ or )

Rem: ∃ undecidable unary langs.

(e.g.: $L = \{1^n : [\mathrm{bin}(n)]_{DTM}$ halts on empty tape$\}$)

(The "trouble"/non-realism: complexity of gen'ing the ckt family is ignored.)

def: A ckt fam (C_n) is <u>P-uniform</u> if ∃ poly(n)-time alg. computing $1^n \mapsto \langle C_n \rangle$.

(artificial input of len. n) ("std" encoding of C_n)

later: P-uniform P/poly is... P. (⊆ is easy)

P-uniform NC? (It makes sense: langs w/ a fast parallel chip findable sequentially. Still, seems a little weird to give uniformity alg "more power" than "actual" alg. Motivs weak unif'ty...)

def: <u>L-uniform</u>: $1^n \to \langle C_n \rangle$ comp'ble in $O(\log n)$ <u>space</u>.

(Pretty std. Think: given n, gate # in binary, must output gate line.)

def: <u>DLOGTIME-unif.</u> [hard-core notion: but "usually" achievable, of some theoretical importance]

↓ (sketch)

Can test "gate i of type t?" & "gate i feeds into j?" in O(log(size)) time (given random access to input tape!)

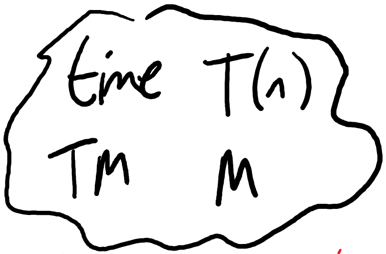
(usually O(log n))

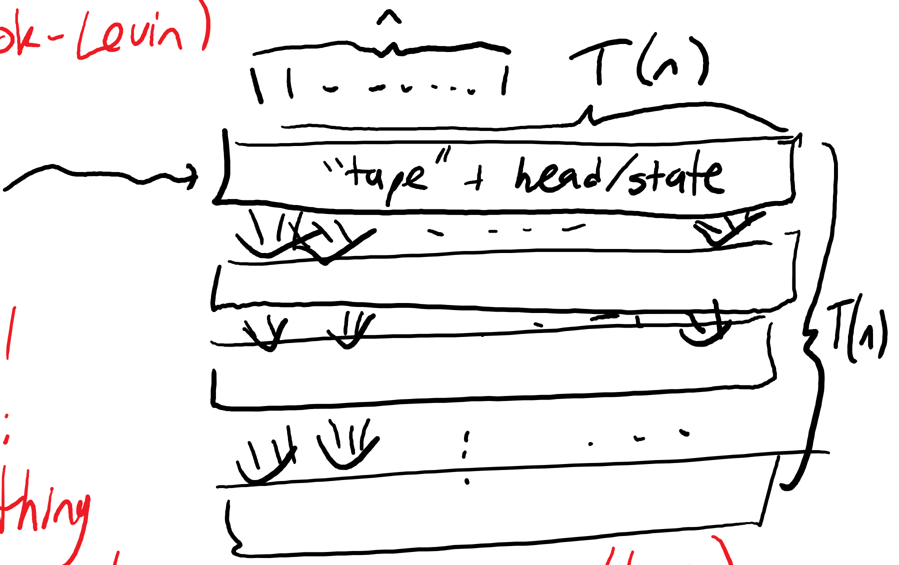
(Don't worry much abt. uniformity distinctions.)

thm (you've seen) : $P \subseteq P/poly$ (& P-uniform)

(either explicitly, or in Cook-Levin)

Sketch:





("comp. tableau": local checks that either: head not here & nothing happens, or head here and expected thing given M happens)

(k tapes: no problem)

Size: $O(T(n)^2)$

P-unif? ✓ [It's very regular/simple] (assuming constr'bility)

in fact...: • L-unif? ✓

• DLOGTIME-unif? ✓ (!)

• can get it to $O(T \log T)$ size.

How? def: TM M oblivious if head movements dep only on input size, (not on "x" itself, just "n")

easy facts: • time T & oblivious → ckts of size $O(T)$

(ex: don't need tableau; just connect time-t gadgetry to snapshot from known preceding time)

• time T M ⇝ equiv time $O(T^2)$ M'

(hint: [redacted] ....)

Hard fact: [HS'64] k-tape $\to$ 2-tape

$T \to O(T \log T)$

can also get oblivious! [Pip., M. Fischer '79]

(End of most boring technicalities!)

(Let's go back to nonuniformity.)

thm [Shannon '49]: ⊛ $\forall n$: $\exists f:\{0,1\}^n \to \{0,1\}$ req'ing ckt size $\geq \frac{2^n}{n}$,

(counting arg: almost all req. such)

$\forall f$, size $4 \cdot \frac{2^n}{n}$ suffices

[Getting $n \cdot 2^n$ easy: can use a DNF. Lupanov got $4 \to 1+o(1)$.]

cor (SIZE Hier Thm.): $\forall\, n \leq s(n) = o(S(n))$, $S(n) \leq \frac{2^n}{n}$,

$\mathrm{SIZE}(s(n)) \subsetneq \mathrm{SIZE}(S(n))$.

(indeed, just need factor-4 gap; even $s(n) < S(n) - O(n)$ known)

Pf: ␣ Take k just large enough so $\frac{2^k}{k} \geq s(n)$

Apply ⊛ fcn to first k inputs. □

ex: $\exists L \in \mathrm{EXPSPACE}$, L reqs size $\Omega(2^n/n)$. [Brute force all ckts!]

Open: $\exists$? such an L in NP?

"Non uniform SETH": $\forall k\ \exists \varepsilon > 0$ s.t. k-SAT ($\in$NP) reqs. size $2^{(1-\varepsilon)n}$.

"Advice": (We saw that, given a non-unif. ckt class like NC, could produce a unif. class by adding "uniformity". Other dir.? Making unif classes dep. on input len.?)

NP (by certs)

$|x| = n$

V (poly-time)

Merlin (wizard) → "certif." y, $|y| = \text{poly}(n)$ ← untrust-worthy.

$|x| = n$

V

Advisor: "advice" y, $|y| = \text{poly}(n)$, deps only on n, trust-worthy

(trust-worthy, short on time)

(gives all 1st-year grad students same advice, 2nd-year, ..., seniors...)

def: a(n)-advice-taking TM: on input x, also gets read-only tape with an "advice" string of len $a(|x|)$.

def: $L \in \text{TIME}(t(n))/a(n)$ if $\exists$ time-$O(t(n))$ a(n)-advice-taking TM M & $\exists$ advice strings $y_0, y_1, y_2, \ldots$ with $|y_n| \le a(n)$ s.t. $\forall x$, M(x) w/ advice $y_{|x|}$ acc. $\iff x \in L$.

(n, $O(\cdot)$)

(Don't have to "check" advice. L in TIME(T)/a if $\underline{\exists}$ some magic advice making it work.)

eg.: Any $L \subseteq \{1\}^*$ in TIME(n)/1.

(Advice tells you whether or not to acc. Have to check, if adv = 1, if all input bits 1.)

notn: · Can also define NTIME(T)/a(n), etc.

· $\underline{\mathcal{C}/\text{poly}} = \bigcup_k \mathcal{C}/kn^k$.

<u>Thm</u>: P/poly accurately named.

pf: ① poly-size ckt fam $(C_n) \Rightarrow$ poly-time/adv-taking TM? ✓

$\rightarrow$ length-n adv. is $\langle C_n \rangle$.

(CIRCUIT-EVAL $\in$ P.) (Notice use of trust.)

② Poly-time/adv. TM $\Rightarrow$ poly-size ckt fam?

✓ "Hardwire" advice string into C_n, use $P \subseteq P/\text{poly}$. $\square$

$BPP \subseteq P/poly$: (feasible randomized $comp^n$)

Rec: $L \in BPP$ iff $\exists$ randomized poly-time TM M s.t.:
- $x \in L \Rightarrow Pr[M(x) \text{ acc.}] \geq \frac{3}{4}$
- $x \notin L \Rightarrow Pr[M(x) \text{ acc}] \leq \frac{1}{4}$.

Rem: $\frac{3}{4}$ a bit arbitrary.

So long as $\geq \frac{1}{2} + \frac{1}{poly(n)}$, can do "success amplification" → trade time for error.

Fact ("Chernoff"): Given $\geq \frac{1}{2}+\varepsilon$ vs $\leq \frac{1}{2}-\varepsilon$ alg., (easier, actually) by repeating $O(\frac{1}{\varepsilon^2}\log(1/\delta))$ times & taking maj. answer, boost to $\geq 1-\delta$ vs. $\leq \delta$.

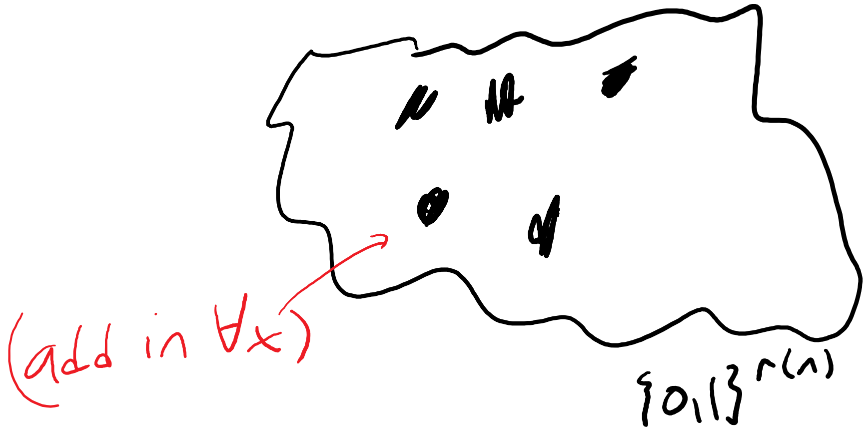
e.g.: $\varepsilon = \frac{1}{4}$, $\delta = 4^{-n}$: By $O(n)$-factor slowdown on M, can assume:

$$\forall x, \quad Pr[M(x) \neq L(x)] \leq 4^{-|x|}.$$

Say $M(x)$ uses $r(|x|) \leq poly(|x|)$ random coins

small even if you are union-bounding over all $|x|=n$

●: $\leq 4^{-|x|}$ frac. where M errs on x



▨ : $\le 2^{-n}$ frac. where M errs on any string of len. n.

before: $\forall |x|=n$, most rand strings in $\{0,1\}^{r(n)}$ good

↓ (after amplif.)

after: for most rand strings in $\{0,1\}^{r(n)}$, simultaneously good $\underline{\forall |x|=n}$

⊛ If M had any one such "magic" string, could decide $L \cap \{0,1\}^n$ perfectly, deterministically.

∴ can do L in $\underline{P/r(n)}$.

∴ thm: $\underline{BPP \subseteq P/poly}$

(Can trade randomness for non-unif'ty. To get ckts: C_n has great random # table built in!)

Lecture 5 - Probabilistic complexity classes

[[Randomness is the 3rd major component of the course, and as we'll see, nondeterminism can be viewed as randomness.]]

Probabilistic TM:

- "flips a fair coin each time step"
  (formally: 2 TM transition fcns, δ_0, δ_1, random one followed)
- equiv: M is deterministic but gets a read-once random tape, filled w/ indep 0/1.

  $\longrightarrow$ write $M(\underset{\uparrow \text{input}}{x}, \underbrace{r}_{\text{rand tape}})$.

- running time on x = <u>max</u> time over all coin flips
  (actually doesn't matter much... expected time, bdd time whp, ... this is most convenient defn)

Probabilistic Classes

[[On x, machine accepts w/ some prob, rejects w/ some prob. How does this define a lang? Several possibilities!]]

def: $L \in$ **BPP** if $\exists$ poly-time prob. TM M s.t.:

- $x \in L \Rightarrow \Pr_{\text{coins}}[M(x) \text{ acc.}] \geq \frac{2}{3}$
- $x \notin L \Rightarrow \Pr_{\text{Coins}}[M(x) \text{ acc.}] \leq \frac{1}{3}$.

(Worst-case over input: gives correct answer w/ "high" prob.)

rem: more generally, <u>$\text{BPTIME}(t(n))$</u> (BP = bounded-err prob.)

Changing $\geq \frac{2}{3}$, $\leq \frac{1}{3}$ to...	yields	
$\geq 3/4$, $\leq \frac{1}{4}$	BPP	
$\geq 1-\varepsilon$, $\leq \varepsilon$, $0<\varepsilon<\frac{1}{2}$,	BPP	"error reduction" (we'll see soon)
$= 1$, $= 0$	$\underline{P}$	
$\geq \frac{2}{3}$, $= 0$	RP	"one-sided error"
$\geq \varepsilon$, $= 0$, $0<\varepsilon<1$	RP	
> 0 , $= 0$	$\underline{NP}$	(! can be thought of as a randomized class)
$> \frac{1}{2}$, $\leq \frac{1}{2}$	"PP"	← ("Prob. PTIME") (a powerful, quirky class)

facts: $P \subseteq RP \subseteq BPP \subseteq PP$
$\quad\quad\quad\quad \subseteq NP \subseteq$

proof: "Let M be an NP machine."
(= "Let M be a PPTIME machine, interpret it as defining a language via the NP criterion.")
The following "PP machine" defines the same lang:
M'(x) : "Flip coin. If heads acc.
Else simulate M(x)."

fact: $PP \subseteq PSPACE$ (Sim. M(x,r) over all r, reusing space.)

rec: $\underline{co\mathcal{C}} = \{L : \overline{L} \in \mathcal{C}\}$

$\therefore$ $coP = P$, $coBPP = BPP$, $coPP = PP$? ✓

ex: "$> \frac{1}{2}$, $< \frac{1}{2}$" yields PP, too

(hint: n^2-time machine has all its probs $\frac{\text{int}}{2^{n^2}}$. Modify machine so it first halts by acc. w. prob $2^{-n^{10}}$) (takes time n^{10}) →

$coRP \subseteq BPP$, $coNP \subseteq PP$

"Sidedness"

rem: RP (like NP) has no "false positives"

- coRP " " " negatives.

(BPP may have both)

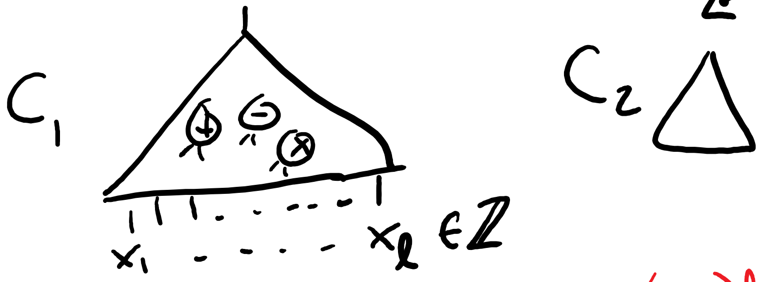
- $ZPP = RP \cap coRP$: "zero-error" → (ex) ≡ always correct, expected poly-time
- (almost) all natural probs we know in BPP are in RP or coRP

(BPP is for if you're too lazy to disting. false +ve from -ve)

eg.: Polynomial Identity Testing (PIT):

$L = \{ \langle C_1, C_2 \rangle : C_1, C_2$ are arithmetic circuits over $\mathbb{Z}$ computing the same function$\}$

fact: $PIT \overset{?}{\in} P$ unknown.



Rand. alg.

- Let $m = size(C_1) + size(C_2)$

Pick $x_1, \dots, x_\ell$ unif., indep. from $\{1, 2, \dots 2^{m+1}\}$

(just $(m+1)\ell$ bits, poly in input size)

- Compute$_{(?)}$ $C_1(x_1, \dots, x_\ell)$, $C_2(x_1, \dots, x_\ell)$, accept iff

- Compute (?) $C_1(x_1,\ldots,x_\ell)$, $C_2(x_1,\ldots,x_\ell)$, accept iff same ans.

Compute mod 2^{2m}. (Mults can blow up # size, this keeps computations poly time.)

Fact 1: Poly-time alg.

Fact 2: $L \in PIT \Rightarrow Pr[\text{Alg. acc}] = \underline{1}$

Fact 3: (A bit nontriv. – read [AB, Ch. 7.2.3]) "Schwartz-Zippel"

$$L \notin PIT \Rightarrow Pr[\text{Alg. acc.}] \leq 1 - \frac{1}{4m} \quad (< 1)$$

What can we *immediately* conclude?

∴ PIT ∈ $\underline{\text{coNP}}$, (There's always a witness for non-membership, when $C_1 \neq C_2$.)

In fact, PIT ∈ coRP (⊆ BPP) by *error reduction*:

Repeat alg. $8m$ (still poly-time) times,
overall acc. iff all tries acc.

Now $L \notin PIT \Rightarrow Pr[\text{Alg. acc}] \leq \left(1 - \frac{1}{4m}\right)^{8m} \leq \left(e^{-\frac{1}{4m}}\right)^{8m}$
$= e^{-2} < 1/3$. ✓

Generally: 1 vs. $1-\varepsilon$ $\overset{\#}{\widetilde{\text{repeated}}}$ $\frac{1}{\varepsilon} \cdot \ln(1/\delta)$ times,
yields 1 vs. $\leq \left(e^{-\varepsilon}\right)^{\frac{1}{\varepsilon}\ln\frac{1}{\delta}} = \delta$. For any $\varepsilon = \frac{1}{poly(n)}$,
$\delta = 2^{-poly(n)}$, just poly(n) repetitions.

∴ For RP, $x \in L \Rightarrow Pr[M(x) \text{ acc.}] \geq \frac{1}{2}$ ("trading time for error")

(sim for coRP) $x \notin L$ $= 0$,

it's equiv. to change ½ to anything in $\left[\frac{1}{poly(|x|)}, 1 - 2^{-poly(|x|)}\right]$.

(Can't make $\frac{1}{poly} \to 2^{-poly}$ tho; would need expon. many reps to see a success. Why $NP \neq RP$, presumably!)

BPP is similar: (need some weak form of Chernoff)

Given a 0/1 rand var that is 1 w.p. $\frac{1}{2}+\varepsilon$, 0 w.p. $\frac{1}{2}-\varepsilon$, majority vote of $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ reps. is 1 w.p. $\geq 1-\delta$.

$$\Rightarrow \quad x \in L \rightsquigarrow \geq \frac{1}{2} + \frac{1}{poly(|x|)} \rightsquigarrow \geq 1 - 2^{-poly(|x|)}$$

$$x \notin L \qquad \leq \frac{1}{2} - \frac{1}{poly}(|x|) \rightsquigarrow \leq 2^{-poly(|x|)}.$$

(again, $\frac{1}{2} \pm 2^{-n}$ not good enough: why $PP \neq BPP$. Need a gap.)

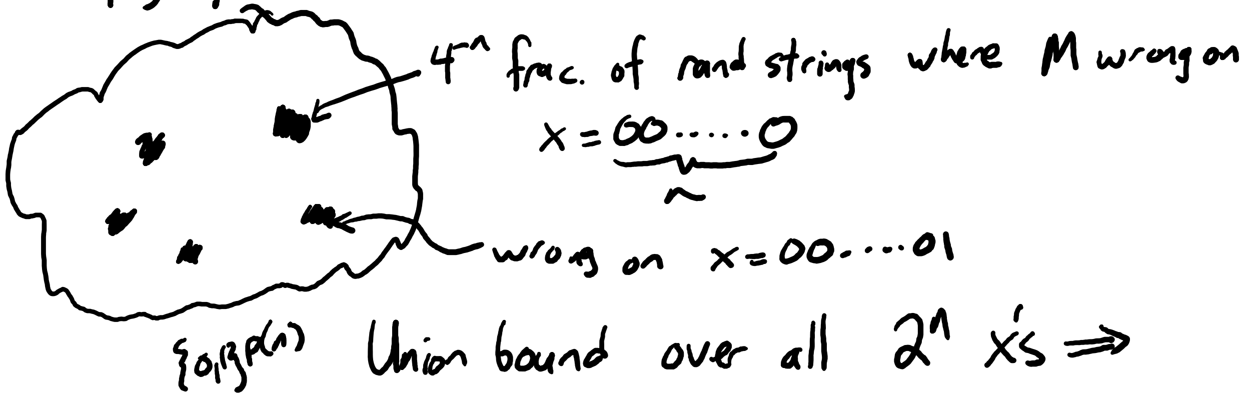
e.g.: can get $\Pr_{r\in\{0,1\}^{p(|x|)}}[\underset{\text{det'ic}}{\underset{\uparrow}{M}}(x,\underset{\text{coins}}{\underset{\uparrow}{r}}) \text{ wrong}] \leq \underset{\uparrow}{4}^{-|x|}$ ⊛

(big enough for union bound over all $|x|=n$!)

thm: $BPP \subseteq P/poly$ (don't know if in P, but anything in BPP has poly-size circuit fams. We'll use advice viewpoint.)

pf: Let $L \in BPP$. (with a linear factor slowdown...)

$\exists$ poly $p(n)$, det'ic M st. $\forall x \in \{0,1\}^n$, ⊛ holds (re $x \overset{?}{\in} L$).



Union bound over all 2^n x's $\Rightarrow$

At most $2^n \cdot 4^{-n} = 2^{-n}$ frac. of r's cause M to be wrong about <u>any</u> length-n string.

(For almost any fixed block of rand bits, M will be right about all strings $x \in \{0,1\}^n$ simultaneously. Note crucial reliance on err. reduc below 2^{-n}.)

$\therefore \forall n,\ \exists$ (many) strings $r_n^* \in \{0,1\}^{p(n)}$ s.t.:

$\forall x \in \{0,1\}^n,\quad M(x, r_n^*)$ acc. $\Longleftrightarrow x \in L$.

det'c. poly(n)-len. advice string

$\therefore L \in P/poly$. $\square$

* * *

(As we saw, what distinguishes RP from NP, and BPP from PP, is the gap.)

Terminology: P, NP, PP are "syntactic" cxty classes.
RP, BPP (& ZPP, NP∩coNP) are "semantic" classes

Informal definition:

Syntactic: every poly-time det'c/nondet'c/randomized TM M defines a lang in P/NP/PP.

e.g.: for prob. ptime M, $L = \{x : \Pr[M(x)\,\text{acc}] > \frac{1}{2}\} \in PP$.

Also: can add a cn^c-time "alarm clock" to any TM M to ensure it halts in $O(n^c)$ time.

⇒ can computably enumerate all P/NP/PP machines/langs.

Semantic: Given a poly-time clocked prob. TM M, it does not automatically define a lang. in BPP (or RP). L = ...? May exist x s.t. $\Pr_{\text{coins}}[M(x)\ \text{acc.}]$ is neither $\geq \frac{3}{4}$ nor $\leq \frac{1}{4}$.

Some x's are "yes" inputs, some are "no" inputs, but some are neither.

And given M, whether $\exists\, x$ with neither is undecidable (ex.)

$\Rightarrow$ no obvious way to computably enumerate BPP machines / langs

Conseq 1: No known BPP-complete lang.

(Syntactic classes have "generic" complete langs:

e.g. · $L = \{\langle x, M, 1^m \rangle : M$ is a DTM, $M(x) = 1$ in m steps$\}$ is P-complete,

· $L = \{\langle x, M, 1^m \rangle : M$ is NTM, $M(x) = 1$ in m steps$\}$ is NP-complete.

(Can't really do same for BPP: $M(x)$ may not be "1" or "0" for a given PTM M.)

Conseq 2: Good BPTIME Hierarchy Theorem is *unknown*.

$$\mathrm{BPTIME}(n) \overset{?}{=} \mathrm{BPTIME}(n^{\log n})$$

(Enumerating the machines you're trying to diag. against is a key component of Hier. Thms. Now actually, we believe BPP = P, so BPP may be "syntactic" and have decent hiers, but can't prove...)

(These thoughts lead to a somewhat important cxty thy def^n. When $L \in$ BPP b/c of machine M, you're "promised" no x falls into the "gap" betw. $\frac{1}{3}$, $\frac{2}{3}$.)

def: A *decision prob.* ($\equiv$ lang.) is a pair $Y, N \subseteq \{0,1\}^*$, disjoint, s.t. $Y \cup N = \{0,1\}^*$. ("Yes & No" instances.)

A *promise prob.*: same, except not. May be $x \notin Y \cup N$.

- Often promise probs are more natural; eg.
  $Y = \{\langle G\rangle : G \text{ 3-col'ble}\}$, $N = \{\langle G\rangle : G \text{ not 3-col'ble}\}$,
  $\overline{Y \cup N} = \{x : x \neq \langle G\rangle \text{ for any graph } G\}$
- Reductions usually map decision probs to promise probs:
  e.g.: $x \in SAT \Rightarrow R(x) \in CLIQUE$
  $x \notin SAT \Rightarrow R(x) \notin CLIQUE$
  But range(R) probably not $\{0,1\}^*$.
- Often $\overline{Y \cup N}$ poly-time recognizable. Can then artificially add to N (or Y) to get a (poly-time equiv.) dec. prob.

* * *

def: $prP = \{(Y,N) :$ ∃ poly-time TM M s.t.
$x \in Y \Rightarrow M(x)$ acc.
$x \in N \Rightarrow M(x)$ rej. $\}$

fact: $P \subseteq prP$.

def: $prBPP = \{(Y,N) :$ ∃ prob Poly-time TM M s.t.
$x \in Y \Rightarrow \Pr_{coins}[M(x) \text{ acc.}] \geq \frac{2}{3}$
$x \in N \Rightarrow \quad \ldots\ldots \leq \frac{1}{3} \}$

(& sim. prRP, ...)

obs: Every prob ptime TM M does define a promise problem in prBPP.

(Sometimes promise problems, promise classes nicer to work with. Some cxty theorists really advocate them...)

Lecture 6 – Quasilinear Cook-Levin

(a technical, preparatory lecture)

Cook-Levin: SAT is NP-hard. I.e.,
let $L \in \text{NTIME}(T(n))$, $T(n) = n^{O(1)}$. Then ①
$\exists$ poly(n)-time ② $R : x \longmapsto \varphi_x$ s.t.

$$x \in L \iff R(x) \in \text{SAT}.$$

(This is good... but we should make it better!)

3 Questions: ① NTIME in what model? (TM? RAM?)
(These all poly-sim. each other, so why does it matter? B/c we'll care about qlin.)

leave up!

② Can R be quasilinear-time? (What does it even mean?) ~~$n \cdot \text{polylog}(n)$~~ $\rightarrow$ $T(n)\,\text{polylog}\,T(n)$ (we assume $T \geq n$)

(In life, actual efficiency more closely assoc'd to quasilin time than to poly time. $O(n^2)$ is very slow!)

②a Can we at least get $|\varphi_x| = \text{quasilin}(T(|x|))$
(Nec. but not suff. But often the main important thing.)

②b If so, can R be super-efficient – e.g., polylog(T)-time? (Given ⟨i⟩, output i^{th} 3SAT clause.)

Answers: Everything you might want is true.

Question: Why care? (Why not? Better is better.)

Answers: ② Arguably, "truly efficient" ≡ QLIN, not P
"complete"

Answers : • Arguably, "truly efficient" $\equiv$ QLIN, not P

② $\Rightarrow$ SAT is "NQLIN-hard" - - → complete

SAT $\in$ NQLIN? ✓ (Obvious on RAMs, on multitape TM? Exercise — hint, sorting.)

- Almost all natural NP probs are in NQLIN:

(ex) SAT, 3COL, INDEP-SET, GISO...

↳ in QLIN $\iff$ SAT in QLIN

(not so interesting, since prob. not. More interesting...)

↳ $\notin \mathrm{TIME}(2^{n^{1-\varepsilon}})$ for any $\varepsilon > 0$

$\iff$ SAT $\notin \mathrm{TIME}(2^{n^{1-\varepsilon}})$ " " "

(? ETH variant)

NB: (2a) is what's really important here:

$|x|$ should go to $\leq |T(x)|/\mathrm{polylog}\, T$ (R running in any poly, or even subexp time would be ok)

(2b) motiv.: SUCCINCT-SAT is NEXP-complete (will prove today)

↓

Given ckt C_n, is truthtable$(C_n) \in$ SAT?

$\in \{0,1\}^{2^n}$

(natch NEXP-complete prob., needs (2b), not (2a))

[Fortnow.... Williams '07] Any SAT alg. using $n^{o(1)}$ space needs $n^{2\cos(\pi/7)-o(1)} \approx n^{1.8}$ time.

(will prove later)

Needs (2a) + (2b).

[DG'84] " PALINDROMES " " " " "

$n^{2-o(1)}$ time!! → multitape TM only

Any model, e.g. RAMs → needs ①.

(We'll talk about ①, (2a), (2b) now, tho won't get into many details.)

① RAMs vs. TMs.

time T ⟶ time $O(T^2)$. Difficulty: dictionary data struct.

RAM: $A[1047] := 2593$

⋮

$x := A[1047]$

⋮

$A[x] := 42$

⋮

⟶ need [dictionary] for (addr, val) pairs, polylog time ops. Bal'd BST? How to implem. on TM??

[GS'89]: nondet time T ⇝ nondet time $O(T \log T)$!

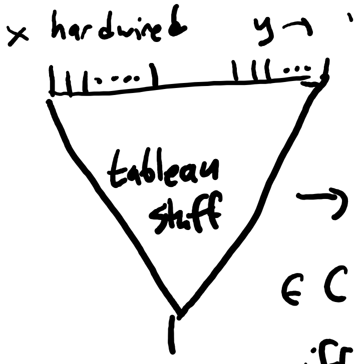
idea: Guess seq. of all mem writes & read-results init'ly. Check that seq. is logically consistent by sorting (qlin. time on TM) primarily by loc., secondarily by time.

fact: NQLIN is highly model-insensitive (Not true, sadly, for det'ic QLIN)

---

② Review Cook-Levin: Let M be NTIME(T(n)) machine. WLOG on input x, $|x|=n$, uses $T(n)$ "guess bits" y, from sep. tape.

$$M(\underset{\overset{\uparrow}{x}}{\cdot}, \underset{\overset{\uparrow}{y}}{\cdot}) \xrightarrow[P \subseteq P/poly \text{ idea}]{R(x)}$$



→ $C_x(y)$: ∈ CIRCUIT-SAT iff $M(x)$ "overall accs." ($\exists y \ldots$)

Good news: [HS, PF, …] k-tape → 2-tape, obliv. stuff

implies $|C_x|$ can be $O(T(n) \log T(n))$ ☺

(2a) ✓   ① + (2a) ✓, $T \rightsquigarrow T \log^2 T$.

[Robson '91]: $\rightsquigarrow T \log T$.

(Good to know, but will be last time we get excited about $T \log T$ vs. T polylog T.)

(2b)? P/poly stuff is DLOGTIME, so (2a) + (2b) ✓

[JMV'14]: ≈ R computable in $\underline{O(1)}$ parallel time (Rel. useful for Williams's NEXP $\not\subseteq$ ACC^0)

① + (2a) + (2b)? RAM ⇝ TM reduction not obviously oblivious-able.

But… can do it [Tourlakis '01, vM '07 simple-ish]

(Sat next to me in my U of T "455")   (Direct proof of 1+2a+2b, avoiding "obliv.")

* * *

finally: CIRCUIT-SAT ≤ 3-SAT reduction is super-simple, & $O(1)$ size blowup. ∴ can reduce all the way to 3SAT.

* * *

Application (of (2b)): SUCCINCT-3SAT is NEXP-complete.

(key for Williams ACC^0)

def: Given language L, SUCCINCT-$L = \{ \langle C \rangle : tt(C) \in L \}$.

→ like "$x \overset{?}{\in} L$", but x "succinctly given": can get i^{th} bit of x via $C(\langle i \rangle)$.

eg: $L = \underset{(\in P)}{OR} = \{x \in \{0,1\}^n : x \text{ contains a } 1\}$. SUCCINCT-$L = \underbrace{\quad\quad}_{}$ CKT-SAT!

Boring: • for natural L, assume "appropriate encoding",

so, e.g., $x \equiv x'$ with len. a power of 2

(e.g., allow padding after doubling symbols)

• ex: SUCCINCT-3SAT $\underset{\geq^p_m}{\leq^p_m}$ variant where $C(\langle i \rangle) =$ description of i^{th} clause

(main work: arithmetic on bit positions: poly(n)-bit #'s)

prop: SUCCINCT-3SAT $\in$ NEXP

pf: Indeed, $L \in NP$ (or P) $\Rightarrow$ SUCC-$L \in$ NEXP (EXP)

- Given input C_n: in exp(n) time, write out $tt(C_n)$, interp. as a 3CNF.
- Using nondet., guess & check if it's sat'ble. □

Thm: SUCC-3SAT is NEXP-hard.

("More generally", if $A \leq B$ in polylog time, succ-$A \leq_P$ succ-B. True of all (?) natch, NP-complete probs.)

Pf: WTS: $\forall L \in \mathsf{NEXP}$, $L \le_m^p$ SUCCNCT-3SAT.

By assump. $\exists$ nondet M running in time $T(n) = 2^{n^c}$ deciding L. Need to design

$$\underset{|x|=n}{x} \xrightarrow[\text{poly}(n)\text{ time}]{R(x)} C_x$$

s.t.

$$\begin{tikzcd} x \in L \arrow[r, Leftrightarrow] \arrow[d] & C_x \in \text{SUCCINCT-3SAT} \arrow[d] \\ \exists y,\ |y| \le 2^{n^c} \text{ s.t. } M(x,y) \text{ acc.} & \Phi_x := tt(C_x) \in \text{3SAT}. \end{tikzcd}$$

As we've seen, $\exists R$ running in time polylog(2^{n^c}) (2b)

"producing" ~~Circuit~~ 3SAT formula Φ_x of size $2^{n^c}\,\text{polylog}(2^{n^c}) = 2^{O(n^c)}$ s.t. $x \in L \Leftrightarrow \Phi_x \in \text{3SAT}$.

((2a) not needed)

means that i^{th} bit of Φ_x outputtable in poly$(\langle i \rangle)$ time.

R is poly(n)-time, $\therefore$ has (P-uniform) poly-size ckt families $\mathcal{R} \longrightarrow$ $\square$

Lecture 7 - The Polynomial Time Hierarchy

(A family of complexity classes between P & PSPACE. Not considered feasible, but useful for studying many aspects of cxty theory.)

(we'll see ≥ 3 equiv. def^{n}s of PH today.)

Q: Say NP = P. Then what else is in P?

A: All langs in PH. (Poly Hier., a large class we'll define.)

eg. NP = P $\Rightarrow$ SAT $\in$ P $\Rightarrow$ UNSAT $\in$ P

⌞ not known in NP

$\Downarrow$

coNP = coP = P (UNSAT is coNP-complete).

(is that it? Nope)

def: MIN-CKT = $\{\langle C\rangle$: C is smallest circuit computing tt(C)$\}$

$\in$ NP? (not known; how would a prover certify it?)

$\in$ coNP? (given C, how could you certify it's non-minimal? could try giving smaller equiv. C', but how could poly verifier check equivalence?)

def: EQ-CKT = $\{\langle C, C'\rangle$: tt(C) = tt(C')$\}$

$\in \underline{\text{coNP}}$, (can witness non-membership: prover gives x s.t. $C(x) \neq C'(x)$)

If NP = P: , NONEQ-CKT $\in$ NP = P ⊛

- Now NONMIN-CKT $\in$ NP:

  Given C, nondet. ptime machine guesses smaller inequiv. C', det'ically verifies it (using $\circledast$).

- $\therefore$ NONMIN-CKT $\in$ P. $\therefore$ MIN-CKT $\in$ P.

(What happened? Alt. viewpoint)

$$C \in \text{MIN-CKT}$$

$$\iff \forall C': (\text{size}(C') < \text{size}(C)) \to \exists x: C'(x) \neq C(x)$$

$$\iff \forall C': \neg(\text{size}(C') < \text{size}(C)) \vee \exists x: C'(x) \neq C(x)$$

$$\iff \forall C': \exists x: \underbrace{\neg(\text{size}(C') < \text{size}(C)) \vee C'(x) \neq C(x)}_{\text{ptime-checkable}} \quad (\ast)$$

(I ignored a bit the issue of encoding, poly-size bounds.)

not^n: Given x, $\exists^p y$ means $\exists y$ with $|y| \leq p(|x|)$, p a polynomial.

def: Let $\mathcal{C}$ be a cxty class. Define new classes $\underline{\exists\mathcal{C}}$: $L \in \exists\mathcal{C}$ if $\exists R \in \mathcal{C}$ s.t.

$$x \in L \iff \exists^p y : (x,y) \in R.$$

$\underline{\forall\mathcal{C}}$: $L \in \forall\mathcal{C}$ if $\exists R \in \mathcal{C}$ s.t.

$$x \in L \iff \forall^p y : (x,y) \in R.$$

e.g.: $\exists P$ is $\underline{NP}$. $\forall P$ is $\underline{coNP}$.

$\exists\exists P$ is $\underline{NP}$ $\forall\forall P$ is $coNP$. $\forall\exists P$?

fact: MIN-CKT $\in \forall\exists P$, by ($\ddagger$).

not^n: $\underline{\Sigma_i P}$ is $\underbrace{\exists\forall\exists\forall\cdots}_{i \text{ quantifs}} P$, $\underline{\Pi_i P}$ is $\underbrace{\forall\exists\forall\cdots}_{i} P$

obs: If $NP = P$: $\exists P = P$. $\forall P = P$.

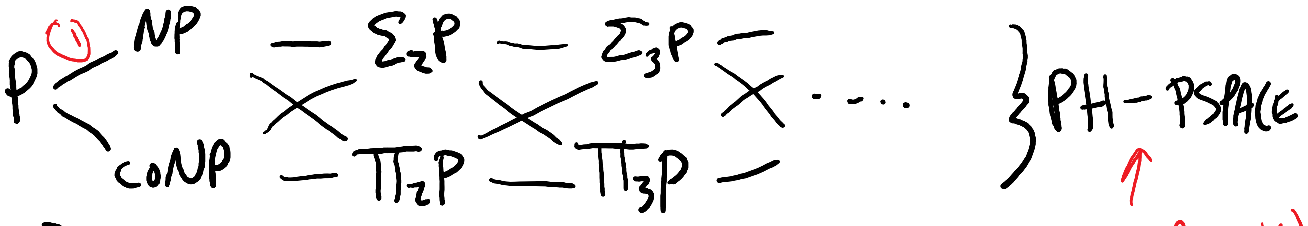
$\Sigma_2 P = \exists\forall P = \exists P$ (by $\circlearrowleft$)

$= P$

(consider def^n: the "R" now in P)

$\forall_2 P = \forall\exists P = \forall P = P$.

$\Sigma_3 P = \exists\forall_2 P = \exists P = P$, etc.



(easy: $\Sigma_i P \subseteq PSPACE\ \forall i$)

def: $\underline{PH} = \bigcup_{i \geq 0} \Sigma_i P = \bigcup_{i \geq 0} \Pi_i P$

thm: $NP = P \Rightarrow PH = P$ ①

What if $NP = coNP$? $(\Sigma_1 = \Pi_1)$

Might not equal P, but... $\Sigma_2 P = \exists\forall P$

$= \exists\exists P = \exists P = \forall P$

& $\Pi_2 P = \exists\forall P = \exists\exists P = \exists P$, $\Sigma_3 P = \exists\Pi_2 P = \exists P$

$\dots$ $PH = \Sigma_1 P = \Pi_1 P$ "PH collapses to the 1st level"

What if $\Pi_7 P \subseteq \Sigma_3 P$? $\Pi_3 P \subseteq \Pi_7 P$, so $\Pi_3 = \Sigma_3$.

Then $\Pi_4 = \forall \Sigma_3 = \forall \Pi_3 = \Pi_3$,

$\Sigma_4 = \exists \Pi_3 = \exists \Sigma_3 = \Sigma_3$,

$\dots$ $PH = \Sigma_3 P$ "PH collapses to 3rd level"

Popular hypothesis (stronger than $P \neq NP$)

PH does not collapse. (I.e., all levels distinct. Why believed? Dunno, like why it's believed MIN-CKT $\notin$ NP or coNP or $\Sigma_2 P$, even.)

Q: Does $\Sigma_i P$ have a complete prob?

A: Yes, $\Sigma_i SAT = \{$ true QBFs of form $\underbrace{\exists x^{(1)} \forall x^{(2)} \exists x^{(3)} \dots}_{i \text{ blocks}} \underset{\substack{\uparrow \\ \text{formula (or ckt)}}}{\phi}(x^{(1)}, \dots, x^{(i)})\}$

Pf: Ex (Cook-Levin)

Ex: $\Sigma_i P$ closed under $\leq_m^P$: $A \leq_m^P B$, $B \in \Sigma_i P \Rightarrow A \in \Sigma_i P$.

Q: Does PH have a complete prob?

A: No! (Probably.) Suppose L is PH-complete. Then

$L \in PH$ & $\forall A \in PH$, $A \leq_m^P L$

$L \in PH$ & $\forall A \in PH,\ A \le_m^p L$

$\Downarrow$

$\exists_j\ L \in \Sigma_j \longrightarrow A \in \Sigma_j$

i.e., PH collapses to j^{th} level.

Q: What about TQBF = { true QBFs of form

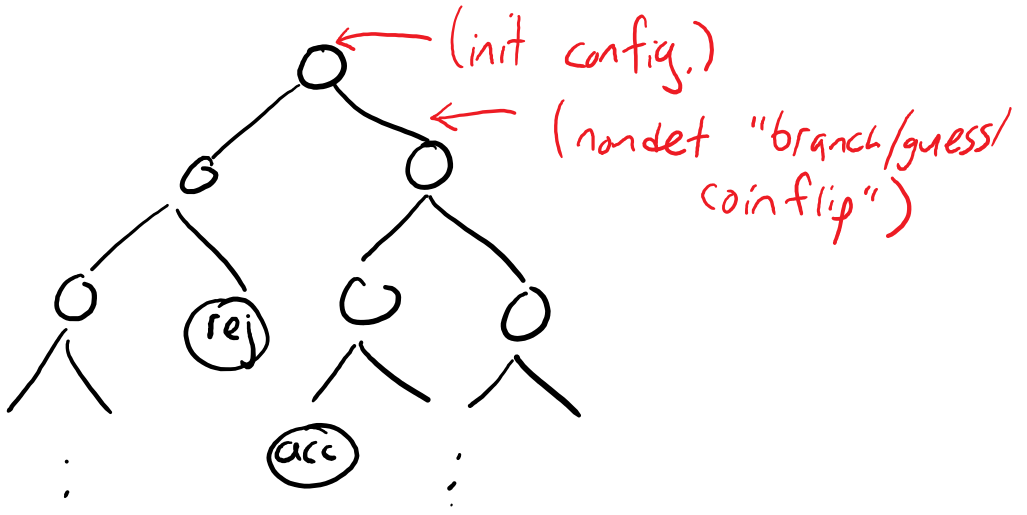
$$\underbrace{\exists x^{(1)} \forall x^{(2)} \exists \dots}_{\text{no restriction on \#}} \phi(\dots) \}$$

?

A: That's PSPACE-complete. (I assume you learned that in ugrad cxty.)

(Alternative definition...)

Alternation: Recall nondet. TM has a "computation tree":



Def: Comp. "overall accepts" iff ≥ 1 leaf is "acc."

Co-nondet machine: same, but iff all leaves "acc".

Nondet: like putting $\vee$ (OR) at each branch pt.

Conondet.: " " $\textcircled{\wedge}$ (AND) " " " "

def: <u>Alternating TM</u>: a nondet. TM that chooses whether each nondet. branch is $\textcircled{\vee}$ or $\textcircled{\wedge}$.

(Affects interp. of what "overall accept" means)

"existential guessing" ↗ "universal guessing" ↑

eg: An alternating TM deciding MIN-CKT:

" On input C:

Universally guess C' of smaller size. $\circledast$

Existentially guess x.

Acc. iff $C(x) \neq C'(x)$."

def: An ATM is of "type Σ_3" (eg.) if on every comp. branch, the guesses are $\vee, \vee, \dots, \vee, \wedge, \wedge, \dots \wedge, \vee, \dots, \vee$.

eg: $\circledast$ is of type Π_2.

ex: $\Sigma_i P = \{ L :$ decided by poly-time Σ_i-type ATM$\}$

(hint: it's like assuming a ckt is "levelled")

def: $\Sigma_i TIME(T(n)) = \{$ " " " " $O(T(n))$-time " - - - - $\}$

thm: If $t(n+1) = o(T(n))$ (& constructible), $\forall i \geq 0$,

$\Sigma_i TIME(t(n)) \subsetneq \Sigma_i TIME(T(n))$ & Π_i too.

ex: Same as nondet. T.H.T.

Actually shows: MIN-CKT $\in \mathrm{coNP}^{\overline{\text{EQ-CKT}}} \subseteq \mathrm{coNP}^{\mathrm{NP}}$.

thm [Celia Wrathall '77]: $\mathrm{coNP}^{\mathrm{NP}} = \Pi_2 P$

[LHS: Stockmeyer's orig. def. of $\Pi_2 P$ in '76]

Indeed: $\Sigma_2 P = \mathrm{NP}^{\mathrm{NP}}$,

$\Pi_3 P = \mathrm{coNP}^{\Sigma_2 P} = \mathrm{coNP}^{\Pi_2 P}$,

$\Sigma_3 P = \mathrm{NP}^{\Sigma_2 P} = \mathrm{NP}^{\Pi_2 P}$, etc...

rem: $\Sigma_i \subseteq P^{\Sigma_i} \subseteq \mathrm{NP}^{\Sigma_i} = \Sigma_{i+1}$

Thm.

$$\begin{array}{lcccl} \Sigma_i & \subseteq & P^{\Sigma_i} & \subseteq & NP^{\Sigma_i} = \Sigma_{i+1} \\ \Pi_i & \subseteq & & \subseteq & coNP^{\Sigma_i} = \Pi_{i+1} \end{array}$$

$$\begin{tikzcd}
 & NP \arrow[dr, dash] & & \Sigma_2 \arrow[dr, dash] & & \Sigma_3 \arrow[dr, dash] & \\
P \arrow[ur, dash] \arrow[dr, dash] & & P^{NP} \arrow[ur, dash] \arrow[dr, dash] & & P^{\Sigma_2} \arrow[ur, dash] \arrow[dr, dash] & & P^{\Sigma_3} \cdots \\
 & coNP \arrow[ur, dash] & & \Pi_2 \arrow[ur, dash] & & \Pi_3 \arrow[ur, dash] &
\end{tikzcd}$$

Lecture 8 - Oracles, and circuits vs. the hierarchy

(Say Alice proves NP=P. And she codes up an efficient SAT-solver. Or... if you like, sps Alice is in Verification, and has a SAT-solver that is great. Cool. So she solves SAT probs. & UNSAT probs. And Ckt-Equivalence probs. And MIN-CKT probs. And any stuff in PH!

Bob wants to get in on action, so Alice sells him a "black box" that solves SAT. Same input/output behavior as Alice's alg.

So Bob is happy, and efficiently solves SAT probs, & UNSAT probs, & CKT-EQUIV probs... Then he wants to solve MIN-CKT probs... What can he do? Given C... what? . . . He can't? But... his box has same i/o behavior as Alice's alg. So...?!)

$$C \in \text{MIN-CKT} \Leftrightarrow \forall C', |C'|<|C|, \neg\text{CKT-EQUIV}(C,C').$$

(black box for SAT can solve this, but only Alice, who "knows the code", can convert it into a poly-size formula/ckt ϕ, allowing $\forall C' \phi$ to be fed into a TAUT solver)

(So: code >> black box. How to formalize "black box"?)

def/rec: TM M with oracle for language L:

M has an "oracle tape" & "oracle state". When it enters oracle state with x on oracle tape, tape replaced with "0" or "1" dep. on whether $x \in L$. (& oracle tape erased)

not^n: oracle machine: $M^?$. With L: M^L.

def: P^{SAT} $\quad$ $P^{NP} = \bigcup_{L \in NP} P^L$. (Bob's power, in the table.)

rm: If $A \leq_m^p B$ then $P^A \subseteq P^B$. cor: $P^{NP} = P^{SAT}$

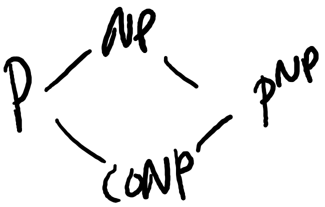
What is P^{NP}? $\quad$ P^{SAT}: (probs solvable w/ "SAT oracle") all L efficiently Turing-red. to SAT, $L \leq_T^p SAT$.

So $NP \subseteq P^{NP}$.

e.g.: $UNSAT \in P^{SAT}$: Given ϕ, query oracle, reverse answer.

Indeed $coNP \subseteq P^{NP}$ (since, ex, P^{SAT} closed under $\leq_m^p$ & UNSAT coNP-compl.

Or, given $L \in coNP$, use



$=_m$ 4 UNSAT coNP-compl.
Or, given $L \in coNP$, use $\bar{L} \in NP$ as oracle, query, ↓ answer)

But seemingly: $P^{NP} \neq PH$, indeed MIN-CKT $\notin P^{NP}$

fact: NON-MIN-CKT $\in NP^{EQUIV\text{-}CKT}$ ← (nondet poly-time, w/ an EQUIV-CKT oracle)

$= NP^{coNP}$ ∵ EQ-CKT coNP-compl

$= NP^{NP}$ (having an oracle closed under co-)

$= NP^{SAT}$ ∵ SAT NP-compl

What is it? [W'76] It's $\Sigma_2 P$.

((Celia Wrathall) (oracle def. orig one. Wrathall showed $\equiv \exists\forall P$.)

thm: $\Sigma_2 P = NP^{\Sigma_1 P}$, $\Pi_2 P = coNP^{\Sigma_1 P}$, $\Sigma_3 P = NP^{\Sigma_2 P}$, $\Sigma_4 = NP^{\Sigma_3 P}$, etc.

pf (sketch): We'll show, rest is induction,

① $\Sigma_2 P \subseteq NP^{SAT} = NP^{TAUT}$ easy:

To decide if "$\exists x \forall y \ \phi(x,y)$" true: ← $\Sigma_2 P$-complete prob,

Nondet'ly guess x,

use TAUT oracle to verify that $\forall y \ \phi(x,y)$ is true. □

② $NP^{SAT} \subseteq \Sigma_2 P$

We'll show $NP^{SAT} \subseteq NP^{UNSAT[1]} \underset{\text{clear}}{\subseteq} \Sigma_2 P$ (think ATM) (difficulty: in NP^{SAT}, nondet machine can make many queries, base its next actions on answers...)

("nondet poly-time machine, ends by making 1 UNSAT query, outputting it)

Let M^{SAT} be nondet., poly-time.

Have it first nondet'ly <u>guess</u> answers to all its SAT queries

At end:
- for "YES" answers $\phi_1, \phi_2, \ldots$ confirm all by guessing sat. asgments
- for "NO" answers $\psi_1, \psi_2, \ldots$ confirm all by forming $\Psi = \psi_1 \vee \psi_2 \vee \ldots$, using 1 UNSAT oracle call. $\square$

* * *

(One major "use" of PH is it helps us understand the difference between uniform & nonuniform computation — i.e., circuits.)

$NP \subseteq P$: disbelieved.

$NP \subseteq P/poly$? " (Doesn't seem like it helps, in solving SAT, to get diff. poly-time alg. for each input len. But... other evidence?)

Karp-Lipton Thm: $NP \subseteq P/poly \Rightarrow PH$ collapses.

To 2nd level: $PH = \Sigma_2 P = \Pi_2 P$.

rec: • SAT is "downward self-reducible": (cf. HW#3.1)

$SAT \in P^{SAT^{<n}}$ (← just made up notation: can solve SAT on insts of strictly less encoding len., can solve it)

(try $\varphi|_{x_1=0}$, $\varphi|_{x_1=1}$).

• SAT has search-to-decision: Given ϕ, $|\phi|=n$,

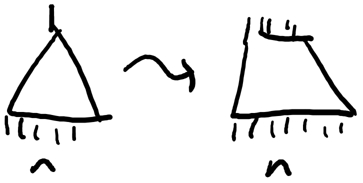
want: ⊛ {
- output NO if ϕ unsat'ble
- output a sat'ing asgnment (cert. of SAT) if sat'ble.

Can do this in poly(n) time, given oracle for $SAT^{<n}$.

• SAT, under approp. encoding, is paddable: given ϕ, $|\phi|=n$, can make equiv. ϕ', $|\phi'|=n'>n$.

∴ ∃ poly-time transformation Self Certifying:

$C \longmapsto C'$ s.t. if C decides $SAT \cap \{0,1\}^n$,



then C' does ⊛ for len.-n ϕ.

$\underbrace{\quad}_{n} \quad \underbrace{\quad\quad}_{n}$

* * *

Proof of Karp-Lipton:

Suppose $NP \subseteq P/poly$. So $\exists$ poly-size $(C_n)_n$ deciding SAT. $\Rightarrow$ $\exists$ poly-size self-certifying $(C'_n)_n$.

Want to show $PH = \Sigma_2 P$. Suffices to show $\Pi_2 SAT \in \Sigma_2 P$. (Rec. $\Pi_2 SAT$ is Π_2-complete. Will think of $\Sigma_2 P$ via ATMs.)

Given "$\forall x \underbrace{\exists y\, \psi(x,y)}$"...

- Say $\smile$ has len. m $(<n)$.
- Existentially guess C'_m. // poly-many $\vee$-nondet. guesses
- Universally guess over all x // poly $\wedge$-nondet
- Det'ically use C'_m on each "$\exists y\, \psi(x,y)$" to check it's true. $\square$

(You don't have to trust/check that you guessed (C'_m) correctly, because answers are self-certifying!).

* * *

[illegible] 99)

THT $\Rightarrow \exists\, L \in \mathrm{TIME}(n^{100})$ s.t. $L \notin \mathrm{TIME}(n^{99})$.

OPEN: $\exists ?\; L \in \mathrm{TIME}(n^{100})$ s.t. $L \notin \mathrm{SIZE}(n^{99})$

(ckt size)

$\in$ P??

$\in$ NP??

$\in$ PSPACE ✓ (not hard)

Anything smaller?

Kannan's Theorem '81: $\exists\, L \in \Sigma_4 P$ s.t. $L \notin \mathrm{SIZE}(n^{100})$.

(indeed: $\forall c \quad \exists L \in \underset{\cap \Pi_4}{\Sigma_4}\mathrm{TIME}(n^{c+1})$ s.t. $L \notin \mathrm{SIZE}(n^c)$ a.e.)

(or: $\exists L \in \Sigma_4 \mathrm{TIME}(\text{quasipoly})$ s.t. $L \notin P/\text{poly}$)

Will get $L \in \Sigma_4 \mathrm{TIME}(n^{101})$ (say) s.t. (expand def. of $L \notin \mathrm{SIZE}(n^{99})$)

($\neq$) $\forall C$ $\exists$ inf. many n s.t. $L \cap \{0,1\}^n$ has no $C n^{100}$-size ckt.

$\forall$ suff large n

Lec. 4 fact: (ckt-size hier thm)

$\forall n \geq 1000, \; \exists C_n : \{0,1\}^n \to \{0,1\}$ s.t.

⊛ { · C_n has size $\leq 4n^{101}$

• C_n has no equiv. ckt of size $< n^{\cdot\cdot}$

def: For $n \geq 1000$, let C^*_n be lexicographically <u>least</u> ckt satisfying $\circledast$. (Define $C^*_n \equiv 0$ for $n < 1000$.)

def: Let L be lang. computed by (C^*_n).

fact: (‡) holds for L. ($L \notin$ io-SIZE(n^{100}))

Remains to show: $L \in \Sigma_4 P$. (<u>ex</u>: you check Σ_4TIME(n^{101}))

Lemma 1: Given C, can check if it sats $\circledast$ in $\Pi_2 P$.

Pf: (Just like MIN-CKT!)

Check size$(C) \leq 4n^{101}$ (def'ic)

· $\forall$ over C' of size $< n^{101}$

· $\exists$ x s.t. $C'(x) \neq C(x)$, $\square$

Lemma 2: Given n-input C, can check if it's C^*_n in $\Pi_3 P$. $\leftarrow$ $\text{coNP}^{\Pi_2 P}$

Pf: ① one oracle call (& lemma 1) to check C sats $\circledast$

② universally guess over all C' lex-less than C, use oracle to check they <u>don't</u> sat $\circledast$.

Proof of $L \in \Sigma_4 P$: Given x, $|x| = n$:

① existentially guess C^*_n

② use $\Pi_3 P$-oracle to check guess

③ output $C^*_n(x)$. □

* * *

Plot twist: We showed $\exists L^* \in \Sigma_4 P$, $L^* \notin SIZE(n^{100})$

Thm: $\exists L \in \Sigma_2 (\cap \Pi_2)$ s.t. $L \notin SIZE(n^{100})$.

Pf: (<u>non</u>-constructive!)

<u>Case 1</u>: $NP \not\subseteq P/poly$.

Then $SAT \Leftarrow \in NP \subseteq \Sigma_2$ has no ckts of <u>any</u> poly-size, so none of $O(n^{100})$ size.

<u>Case 2</u>: $NP \subseteq P/poly$.

Then $\underset{\| \, \Sigma_4 P}{PH} = \underset{\|}{\Sigma_2 P}$ by Karp-Lipton

So $L^* \in \Sigma_2 P$!

$\boxed{\text{LOL!}}$

So $L^{\neg} \in \Sigma_2 P$!

LOL!

Kannan Thm: $\exists L \in \Sigma_4 P$ s.t. $L \notin SIZE(n^{99})$.

$\hookrightarrow \Sigma_2 P$

Proof: <u>Case 1:</u> $SAT \notin P/poly$

$\Rightarrow SAT \in NP \subseteq \Sigma_2 P$, $SAT \notin SIZE(n^{99})$.

<u>Case 2:</u> $SAT \in P/poly \Rightarrow NP \subseteq P/poly$

$\therefore$ Karp-Lipton $\Rightarrow PH = \Sigma_2 P$.

$\Rightarrow \Sigma_4 P = \Sigma_2 P$.

We're done.

$\boxed{\text{LoL}}$

Lecture 9 - Time/Space Tradeoffs for SAT

NP vs. L — SAT $\in$ SPACE(log n)

thm: [Williams '07]

A SAT algorithm using $O(\log n)$ space needs $\geq n^{1.8}$ time.

$\uparrow$ $n^{o(1)}$, $n^{.0000\cdots01}$

$\uparrow$ any # $< 2\cos(\pi/7)$

$\uparrow$ root of $c^3 - c^2 - 2c + 1 = 0$

· even for RAMs.

not^n: TISP $(t(n), s(n)) = \{$L decidable by RAM running in time $O(t(n))$, space $O(s(n))$, simultan'ly$\}$

Wil'07: SAT $\notin$ TISP$(n^{1.8}, n^{o(1)})$

[Kan'84, For'97, LV'99, FvM'00, Wil'05, DvM'06, Wil'07]

SAT $\notin$ TISP$(n^{1.414}, n^{o(1)})$ $\quad \notin$ TISP$(n^{1.618}, n^{o(1)})$ $\quad n^{1.66}$

$< \sqrt{2}$ $\quad < \phi$ $\quad 2^{\frac{1}{4}} \cdot 3^{\frac{1}{8}} \cdot 4^{\frac{1}{16}} \cdot 5^{\frac{1}{32}} \cdots$

[Cob'66, San'01]: PAL (& SAT) $\notin$ TISP$(n^{2-o(1)}, n^{o(1)})$ for multitape TMs.

n^2 time $\quad$ access mem cell # 2^{n^2} ??

We'll prove stmts like

⊛ $NTIME(n) \not\subseteq TISP(n^{1.41}, n^{.021})$ ✓

rec (Lec.6): SAT is NQL-complete

↑ (& "most" natural NP-c probs also NQL-complete)

$\Rightarrow$ $SAT \notin TISP(n^{1.41}/polylog(n), n^{.021}/polylog(n))$

↳ $\notin (n^{1.4}, n^{.02})$.

Ingredients

① "No complementary speedup"

$\Sigma_k TIME(t(n)) \not\subseteq \Pi_k TIME(o(t(n)))$, $k \geq 1$ ✓

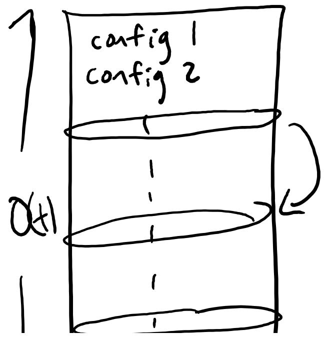
② Padding (cf. HW 1.3) ✓

③ "Trading time for alternations" [Nep'70] ✓

e.g.: thm: $TISP(t,s) \subseteq \Sigma_2 TIME(\sqrt{t}\sqrt{s})$

Π_2 -

pf: $TISP(t,s)$ work tape tableau:

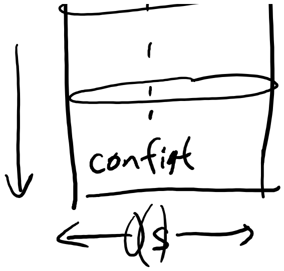


Simulating by $\Sigma_2 TIME$-machine.

∃-guesses configs ✓ t/r configs

$C_r, C_{2r}, C_{3r}, \ldots$ $O(\frac{t}{r} \cdot s)$ time ✓

∀-guesses $j \in \{1, \ldots, t/r\}$



∀-guesses $j \in \{1, \dots, t/r\}$

$\left\{ \text{Deterministically verify } C_{(j-1)r} \overset{}{\underset{\text{in } r \text{ steps}}{\longmapsto}} C_{jr} \right\}$ $O(r)$ time

total time: $O(\frac{t}{r} s + r)$

↑ balance params: choose $r = \sqrt{t}\sqrt{s}$

$O(\sqrt{t}\sqrt{s})$.

$* \quad TISP(t,s) \subseteq \underset{\Pi_2\text{-}}{\Sigma_2} TIME(\sqrt{t}\sqrt{s})$ □

rem: Yes, e.g., $TISP(n, n^{.4}) \subseteq \Sigma_2 TIME(\underset{\uparrow \text{ sublinear!}}{n^{.7}})$.

Makes sense in RAM model.

* * *

thm: $NTIME(n) \not\subseteq TISP(n^{1.4}, n^{.02})$.

AFSOC $NTIME(n) \subseteq TISP(n^{1.4}, n^{.02})$ ⊛ ✓

$\Pi_2 TIME(n) = \underbrace{\forall\forall\forall \cdots \forall \overbrace{\exists\exists\exists \cdots \exists \text{ det.}}^{}}_{O(n)}$ NTIME(n)

$\subseteq \Pi_1 TIME(n^{1.4})$. (†) $\subseteq TIME(n^{1.4})$

co-⊛: $\Pi_1 TIME(n) \subseteq TISP(n^{1.4}, n^{.02})$

padding $\Longrightarrow$ $\Pi_1 TIME(n^{1.4}) \subseteq TISP(n^{1.96}, n^{.028})$

padding $\Longrightarrow$ $\Pi_1 TIME(n^{1.4}) \subseteq TISP(n^{1.96}, n^{.028})$

(Alternation trading) $\subseteq \Sigma_2 TIME(\sqrt{n^{1.96}}\sqrt{n^{.028}})$

$= \Sigma_2 TIME(n^{.994})$.

$\Pi_2 TIME(n) \subseteq \Sigma_2 TIME(n^{.994}) \not\Longleftarrow\!\!\!\Longrightarrow$ "no complex. speed" □

Recap:

AFSOC $NTIME(n) \subseteq TISP(n^c, n^{o(1)})$ (*) ✓

$$\left\{\begin{array}{ll} \Pi_2 TIME(n) \subseteq \Pi_1 TIME(n^c) & (\text{"alternation elim."}) \\ \subseteq TISP(n^{c^2}, n^{o(1)}) & (\text{co-assump., padding}) \\ \subseteq \Sigma_2 TIME(n^{c^2/2 + o(1)}) & (\text{"alt. trading"}) \end{array}\right\}$$

$NTIME(n) \subseteq TIME(n^c)$

If $c^2/2 < 1$, get contrad. to "no complex. speedup".

$\Longleftrightarrow c < \sqrt{2}$.

□

Q: If, say, $c = 1.5$, would we be dead?

A: Well, at least we have a new "fact",

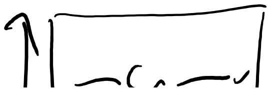
$$\Pi_2 TIME(n) \subseteq \Sigma_2 TIME(n^{1.125 + o(1)}).$$

Alternation Elim: Let $k \in \mathbb{Z}^+$, $c \geq 1$.

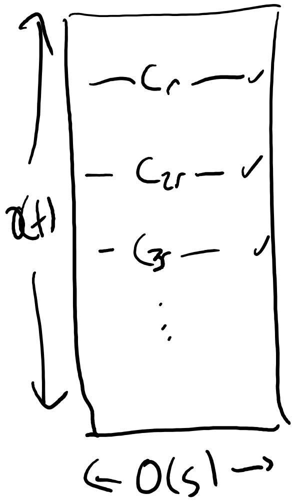
Suppose $\Sigma_{k-1} TIME(n) \subseteq \Pi_{k-1} TIME(n^c)$ ✓

Then $\Pi_k TIME(t) \subseteq \Pi_{k-1} TIME(t^c)$, for constr'ble $t(n) \geq n$,

Alternation Trading $TISP(t, s)$



✓ $\exists$-guess $C_r, C_{2r}, C_{3r}, \ldots$ time $O(\frac{t}{r} \cdot s)$ ✓



✓ $\exists$-guess $C_r, C_{2r}, C_{3r}, \ldots$ time $O\left(\frac{t}{r}\cdot s\right)$ ✓

✓ $\forall$-guess j

det'ic: check $C_{(j-1)r} \xrightarrow{r \text{ steps}} C_{jr}$ ⊛ -

$O(r)$

↳ $\exists\forall\text{TIME}\left(\frac{t}{r}\cdot s + \underline{r}\right)$

or $\forall\exists\text{TIME}\left(\frac{t}{r}\cdot s + r\right)$ ✓

Idea: recursively do ⊛.

It's a $\underline{\text{TISP}(r, s)}$ computation.

Make this inner comp. → $\subseteq \forall\exists\text{TIME}$ $(\sqrt{r}\sqrt{s})$

$\text{TISP}(t,s)$ $\underline{\forall\exists}\,\underline{\text{TIME}(\sqrt{r}\sqrt{s})}$ ⊛

$\subseteq$

$$\exists\underline{\forall}\forall\exists\text{TIME}\left(\frac{t}{r}\cdot s + \sqrt{r}\sqrt{s}\right)$$

$$\Sigma_3\text{TIME}\left(\frac{t}{r}\cdot s + \sqrt{r}\sqrt{s}\right)$$

Balance r: $r = t^{2/3} s^{1/3}$

$$\boxed{\text{TISP}(t,s) \subseteq \underset{(\Pi_3)}{\Sigma_3}\text{TIME}\left(t^{1/3} s^{2/3}\right)}$$

↳ if $s = t^{o(1)}$, $\subseteq \Sigma_3\text{TIME}\left(t^{1/3+o(1)}\right)$

By more iters:

thm: $\text{TISP}(t,s) \subseteq \underset{(\Pi_k)}{\Sigma_k}\text{TIME}\left(t^{1/k} \cdot s^{\frac{k-1}{k}}\right)$

cor: $\text{TISP}(t, t^{o(1)}) \subseteq \underset{\Pi_k.}{\Sigma_k}\text{TIME}\left(t^{1/k + o(1)}\right)$

$\underline{\text{Rec:}}$ AFSOC $\text{N}\underline{\text{TIME}(n)} \subseteq \text{TISP}(n^c, n^{o(1)})$. ✓

$\Pi_2\text{TIME}(n) \subseteq \Pi_1\text{TIME}(n^c)$ (Alt. elim, assump.)

☺ $\subseteq \text{TISP}(n^{c^2}, n^{o(1)})$ (" " ", padding)

$\subseteq \Sigma_2\text{TIME}(n^{c^2/2+o(1)})$ (Alt. trading)

If $c^2/2 > 1$, call above "Lemma".

$\Sigma_3\text{TIME}(n) \subseteq \Sigma_2\text{TIME}(n^{c^2/2+o(1)})$ (Lemma, Alt. Elim.)

$\subseteq \text{TISP}((n^{c^2})^{c^2/2+o(1)}, n^{o(1)})$ (co-☺, padding)

$\subseteq \Pi_3\text{TIME}(n^{c^4/6+o(1)})$ (k=3 Alt. trading)

If $c^4/6<1$, we get contrad. to "no omplem. speedup"!

$c < 6^{1/4} = 1.56\ldots$ ↝ $\text{SAT} \notin \text{TISP}(n^{1.56}, n^{o(1)})$

$2^{1/4}\cdot 3^{1/4}$

If $c^4/6 \geq 1$, we do get a new "Lemma"!

$\Sigma_4\text{TIME}(n) \subseteq \Sigma_3\text{TIME}(n^{c^4/6+o(1)})$ (new lemma, Alt. Elim.)

$\subseteq \Sigma_2\text{TIME}(n^{c^6/12+o(1)})$ (Alt Elim padding)

(co-☺) $\subseteq \text{TISP}(n^{c^8/12+o(1)}, n^{o(1)})$

$\subseteq \Pi_4\text{TIME}(n^{c^8/48+o(1)})$ (Alt. trading)

If $c^8/48<1$, get a contrad.

$c < 48^{1/8} = 2^{\frac{1}{4}}3^{\frac{1}{8}}4^{\frac{1}{8}} = 1.62\ldots$

If $c^8/48 \geq 1$,

$\Sigma_5\text{TIME}(n) \subseteq \Sigma_4\text{TIME}(n^{c^8/48+o(1)})$ ...

$$\subseteq \Sigma_3 TIME\left(n^{c^{12}/6\cdot 48 + o(1)}\right)$$

$$\subseteq \Sigma_2 TIME\left(n^{c^{14}/6\cdot 48\cdot 2 + o(1)}\right)$$

$$\subseteq TISP\left(n^{c^{16}/6\cdot 48\cdot 2 + o(1)}, n^{o(1)}\right)$$

$$\subseteq \Pi_5 TIME\left(n^{c^{16}/6\cdot 48\cdot 2\cdot 5 + o(1)}\right)$$

Contrad. if $c < 2^{1/4}\cdot 3^{1/8}\cdot 4^{1/16}\cdot 5^{1/16} = 1.645\ldots$

Etc.: limit is $2^{1/4}\cdot 3^{1/8}\cdot 4^{1/16}\cdot 5^{1/32}\cdot 6^{1/64}\ldots$

$= 1.66\ldots$

$SAT \notin TIME(n^{1.66}, n^{o(1)})$.

[Wil'07]: Codified all poss. proof strats using

- assump.
- alt elimin.
- alt trading ←
- contrad. to no-complem-speedup

Found $NTIME(n) \not\subseteq TISP(n^c, n^{o(1)})$

$\forall c < 2\cos(\pi/7)$.

Conj'd optimal among proof system:

Buss-Williams'11 proved it.

$TISP(n, n^{o(1)}) \subseteq \Sigma_k TIME(n^{\frac{1}{k}+o(1)})$

Not improvable!

$PARITY = \{x : |x| \text{ is odd}\} \in TISP(n, 1)$

ex: $\Sigma_k TIME(n^{\alpha})$ has depth $k+1$,

<u>ex</u>: $\Sigma_k \mathrm{TIME}(n^\alpha)$ has depth $k+1$, size $2^{O(kn^\alpha)}$ circuits.

[Håstad '86]:

Any depth $k+1$ circuit computing PARITY needs size $2^{\Omega(n^{1/k})}$,

Lecture 9 - Time/Space tradeoffs for SAT

(NP vs. P is of course very hard, but what about...)

NP vs. L: SAT $\overset{?}{\in}$ SPACE(log n)

(Super-unlikely, but can we prove anything?)

thm: [Williams '07] (CMU PhD thesis)

A SAT alg. using O(log n) space needs $\geq n^{1.8}$ time.

- even $n^{o(1)}$, or $n^{.000\cdots01}$ ↗
- 1.8: any # $< 2\cos(\frac{\pi}{7})$ ↗
- root of $c^3 - c^2 - 2c + 1$. ↑
- even on RAMs.

not^n: $TISP(t(n), s(n)) = \{$ L decidable by RAM running in time $O(t(n))$, space $O(s(n))$ (simul.) $\}$

Williams: SAT $\notin TISP(n^{1.8}, n^{o(1)})$

History: [Kan'84, For'97, LV'99, FvM'00, Wil'05, DvM'06, Wil'07]

SAT $\notin TISP(n^{1.414}, n^{o(1)})$ ↑ $<\sqrt{2}$

↓ $\notin TISP(n^{1.618}, n^{o(1)})$ ↗ $<\phi$

↓ $n^{1.66}$ ↗ $2^{\frac{1}{4}}3^{\frac{1}{8}}4^{\frac{1}{16}}5^{\frac{1}{32}}\cdots$

Today: ↗ & sketch →

rem: [Cob'66, San'01]: PAL (& SAT) $\notin TISP(n^{2-o(1)}, n^{o(1)})$

for multitape TMs (Exploit TM locality)

Of course, $PAL \in TISP(n, \log n)$ on RAMs.

(So it's important that these lower bounds hold for RAMs.)

(Brief remark on RAM space: Say you run in n^2 time. Could access memory cell 2^{n^2}. Not cool? But... can change all algs to use a dict. data structure, so they access only contiguous memory. Log. overhead in time, which is neglig. for today's results. Convenient, b/c now a space S machine has a "config." that just involves the first O(S) tape cells.)

rem: We'll prove things like

$$\circledast \quad NTIME(n) \not\subseteq TISP(n^{1.41}, n^{.021}).$$

(What does this have to do with SAT?)

rec (Lec. 6): SAT is NQL-complete.
(& "most" NP-c langs)

$$\overset{}{\underset{\circledast}{\Rightarrow}} \quad SAT \notin TISP(n^{1.41}$$

$$\notin TISP(n^{1.4}, n^{.02}).$$

//

Ingredients:

① "No complementary speedups":

$$\Sigma_k TIME(t(n)) \not\subseteq \Pi_k TIME(o(t(n))), \quad k \geq 1$$

(As mentioned on Piazza... you might have (assuming constructibility) expected Σ_k on RHS. You could get that; that's THT for Σ_k, which has same proof as for NTHT. But in fact we'll "only" need the above ver., with complem. alt. types. And proving it is much <u>easier</u> than NTHT. Recall that in THTs you want the bigger time class to sim. the smaller & negate. Tough for Σ_k on both sides, but <u>easy</u> for opp. types!)

② Padding (cf. HW1.3).

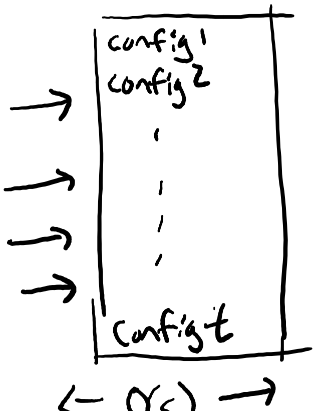
③ "Trading time for alternations" [Nep'70]

eg thm: $TISP(t,s) \subseteq \underset{\text{or } \Pi_2}{\underset{\uparrow}{\Sigma_2}} TIME(\sqrt{t}\sqrt{s})$

(assuming constructibility) (∵ TISP closed under co-)

(think of $s = n^{o(1)}$: says $TISP(t, n^{o(1)}) \subseteq \Sigma_2 TIME(\sqrt{t}^{+o(1)})$)

pf: $TISP(t,s)$ work tape tableau:



With Σ_2-ATM...

(∃) Existentially guess configs $C_r, C_{2r}, C_{3r}, \ldots$

(∀) Universally guess $j \in 1 \ldots t/r$

Deterministically check

$\leftarrow \alpha(s) \rightarrow$

Deterministically check

$$C_{(j-1)r} \underset{r \text{ steps,}}{\vdash} C_{jr}$$

Time: $O(\frac{t}{r} \cdot s)$ (for guessing $\frac{t}{r}$ configs)
(∀ time neglig.)

$+ \; O(r)$ (for det. sim'ing r steps)

Balance: $\frac{t}{r} s = r \iff r = \sqrt{ts}$. □

Rem: Yes, e.g., $\mathrm{TISP}(n, n^{.4}) \subseteq \Sigma_2\mathrm{TIME}(n^{.7})$
↖ sublinear!

Makes sense in RAM model! (If it's weird to you, pad everything.)

[The most basic "$\sqrt{2}$" TISP tradeoff for SAT.]

Thm: $\mathrm{NTIME}(n) \not\subseteq \mathrm{TISP}(n^{1.4}, n^{.02})$.

(recall, implies same for SAT, up to polylog factors)

Pf: AFSOC $\underline{\mathrm{NTIME}(n) \subseteq \mathrm{TISP}(n^{1.4}, n^{.02})}$. ⊛

$$\Pi_2\mathrm{TIME}(n) = \forall\forall\cdots\forall \underbrace{\exists\exists\cdots\cdots\exists \text{ determin., time } O(n)}_{\mathrm{NTIME}(n) \subseteq \mathrm{TIME}(n^{1.4}), \text{ by } ⊛}$$

$\therefore \subseteq \Pi_1\mathrm{TIME}(n^{1.4})$. (†)

co-⊛: $\Pi_1\mathrm{TIME}(n) \subseteq \mathrm{TISP}(n^{1.4}, n^{.02})$

$\underset{\text{padding}}{\Rightarrow} \; \Pi_1\mathrm{TIME}(n^{1.4}) \subseteq \mathrm{TISP}(n^{1.96}, n^{.028})$

$$(\text{Alt. trading}) \subseteq \Sigma_2\mathrm{TIME}\left(\sqrt{n^{1.96}\, n^{.028}}\right)$$
$$= \Sigma_2\mathrm{TIME}(n^{.994}).$$

$\therefore\ \Pi_2\mathrm{TIME}(n) \subseteq \Sigma_2\mathrm{TIME}(n^{.994})$, contradicting "no complem. speedup"! $\square$

(Say we limit to $n^{o(1)}$ space, try to get really high time bound.)

Recap: AFSOC $\mathrm{NTIME}(n) \subseteq \mathrm{TISP}(n^c, n^{o(1)})$.

$$\Pi_2\mathrm{TIME}(n) \subseteq \Pi_1\mathrm{TIME}(n^c) \quad (\text{"alternation elim!"})$$
$$\subseteq \mathrm{TISP}(n^{c^2}, n^{o(1)}) \quad (\text{assump., + padding})$$
$$\subseteq \Sigma_2\mathrm{TIME}(n^{c^2/2+o(1)}) \quad (\text{alt. trade})$$

contradic. if $c^2/2 < 1 \Leftrightarrow c < \sqrt{2}$.

Q: If, say, $c = 1.5$, are we dead?

A: Well, at least we have a new "fact",

$$\Pi_2\mathrm{TIME}(n) \subseteq \Sigma_2\mathrm{TIME}(n^{\overset{\uparrow}{1.125}+o(1)})$$
$$(1.5)^2/2.$$

(We'll see if that helps soon. But first let's sharpen tools.)

<u>Alt. Elim. Thm:</u> Let $k \in \mathbb{Z}^+$, let $c > 1$, let $t(n) \geq n$.

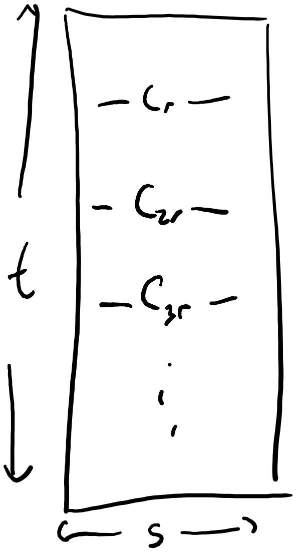
Suppose $\Sigma_{k-1}\mathrm{TIME}(n) \subseteq \Pi_{k-1}\mathrm{TIME}(n^c)$ $\uparrow$ (& const.)

Then $\underset{(\Sigma_k)}{\Pi_k\mathrm{TIME}(t)} \subseteq \underset{(\Sigma_{k-1})}{\Pi_{k-1}\mathrm{TIME}(t^c)}$.

(Pad assump. from $n \to T$, then flip one quantif.

Can also get $\Sigma_k \to \underbrace{\Sigma_{k-1}}$ ver by co-ing both sides)

Alt. Trading: TISP(t,s)



$\exists$-guess $C_r, C_{2r}, \ldots$ $\underline{\text{time } \frac{t}{r} \cdot s}$

$\forall$-guess j

det'ic: check $C_{(j-1)r} \overset{r \text{ steps}}{\underset{\text{time } r}{\vdash}} C_{jr}$ ⊛

↳ $\exists\forall\text{TIME}(\frac{t}{r}s + r)$. (‡)

$\underline{\text{OR}}$ $\forall\exists\text{TIME}($ " $)$ (∵ co-TISP = TISP)

Idea: recursively do ⊛!

It's a TISP(r,s) computation

Can make it $\forall\exists\text{TIME}(\sqrt{rs})$.

Now (‡) becomes $\exists\underbrace{\forall\forall}_{\text{merge}}\exists\text{TIME}(\frac{t}{r}s + \sqrt{rs})$.

↓

Balance: $r = t^{2/3}s^{1/3}$

$\therefore$ TISP$(t,s) \subseteq \underset{(\Pi_3)}{\Sigma_3}\text{TIME}(t^{1/3}s^{2/3})$.

By more iters:

Thm: TISP$(t,s) \subseteq \underset{(\Pi_k)}{\Sigma_k}\text{TIME}(t^{\frac{1}{k}}s^{\frac{k-1}{k}})$.

cor: TISP$(t, t^{o(1)}) \subseteq \Sigma_k\text{TIME}(t^{\frac{1}{k}+o(1)})$.

Rec: AFSOC NTIME$(n) \subseteq$ TISP$(n^c, n^{o(1)})$.

$\Pi_2\text{TIME}(n) \subseteq \Pi_1\text{TIME}(n^c)$ (Alt. elim., assump.)

☺ $\subseteq$ TISP$(n^{c^2}, n^{o(1)})$ (" " " , padding)

$\subseteq \Sigma_2 TIME(n^{c^2/2 + o(1)})$ (Alt. trad.)

If $c^2/2 > 1$, call it "Lemma" & keep going...

$\Sigma_3 TIME(n) \subseteq \Sigma_2 TIME(n^{c^2/2 + o(1)})$ (Lemma, Alt. elim) (♡)

$\subseteq TISP(n^{c^4/2 + o(1)}, n^{o(1)})$ (as in ☺)

$\subseteq \Pi_3(n^{c^4/6 + o(1)})$ (Alt trading, $k=3$)

If $c^4/6 < 1$, get contradic!

↳ $c < 6^{\frac{1}{4}} = 2^{\frac{1}{4}} 3^{\frac{1}{4}} = 1.56...$

Else: have a new Lemma!

$\Sigma_4 TIME(n) \subseteq \Sigma_3 TIME(n^{c^4/6 + o(1)})$ (Lemma, Alt elim)

(there's actually a few ways to proceed now; I'll show a good one)

use co-♡: & padding $\subseteq \Sigma_2 TIME(n^{c^6/12 + o(1)})$

$\subseteq TISP(n^{c^8/12 + o(1)}, n^{o(1)})$ (as in ☺)

$\subseteq \Pi_4(n^{c^8/48 + o(1)})$ (Alt. trad.)

Get contra. if $c^8/48 < 1 \rightarrow c < 48^{\frac{1}{8}} = 2^{\frac{1}{4}} 3^{\frac{1}{8}} 4^{\frac{1}{8}}$

↳ $= 1.62...$

Else: $\Sigma_5 TIME(n) \subseteq \Sigma_4 TIME(n^{c^8/48})$

$\subseteq \Sigma_3 TIME(n^{c^{12}/6 \cdot 48})$

$\subseteq \Sigma_2 TIME(n^{c^{14}/6 \cdot 48 \cdot 2})$

$\subseteq TISP(n^{c^{16}/6 \cdot 48 \cdot 2}$ ☺

$$\subseteq \Pi_5 \mathrm{TIME}\left(n^{c^{16}/6\cdot 48\cdot 2\cdot 5}\right)$$

Contra if $c < 2^{1/4} 3^{1/8} 4^{1/16} 5^{1/16} = 1.645\ldots$

Etc.: limit is $2^{1/4} 3^{1/8} 4^{1/16} 5^{1/32} \ldots = 1.66\ldots$

$$\not\leadsto$$

Scope for improvement?

Arg. assumes $\mathrm{NTIME}(n) \subseteq \mathrm{TISP}(n^c, n^{o(1)})$, ⊛,

repeatedly uses it via

$$\Sigma_2\mathrm{TIME}(n) \subseteq \mathrm{NTIME}(n^c)$$
$$\nearrow \quad \subseteq \mathrm{TISP}(n^{c^2}, n^{o(1)})$$

only used

$$\mathrm{NTIME}(n) \subseteq \mathrm{TIME}(n^c).$$

Next step: exploit ⊛ more fully.

[Wil'07]: Codified all poss. proof strats using

(+)
- assump
- Alt Trades (& padding)
- Alt elims
- contrad. to no-complem-speedup

Found the $\mathrm{NTIME}(n) \not\subseteq \mathrm{TISP}(n^c, n^{o(1)})$

$\forall c < 2\cos(\pi/7)$.

Conj'd optimal among all proofs using (†)?
Proved true by Buss-Will'll.

(Most crucial ingredient:) $\mathrm{TISP}(n, n^{o(1)}) \subseteq \Sigma_k \mathrm{TIME}(n^{\frac{1}{k}+o(1)})$.

Improvable?

No! PARITY $\in \mathrm{TISP}(n, 1)$.
($\{x : |x| \text{ odd}\}$)

ex : $\Sigma_k \mathrm{TIME}(n^{\alpha})$ has depth $k+1$, size $2^{O(kn^{\alpha})}$
(like HW) circuits.

[Håstad '86]: Any depth $k+1$ ckt computing PARITY
(later) needs size $2^{\Omega(n^{\frac{1}{k}})}$.

Lecture 10 – Arthur-Merlin classes

(One motivation for this lecture, & next couple:)

What if "efficient" ≡ BPP? Then what is "NP"?

① "MA": prover gives verifier a certif.

("Merlin") (Max Avg) randomized. ("Arthur")

$x \in L \Rightarrow \exists y :$ ~~$V(x,y)=1$ whp~~ for most z : $R(x,y,z)=1$

$x \notin L \Rightarrow \forall y :$ for most z : $R(x,y,z)=0$

"ref"

("publishable proofs")

today's not^n: "$\exists y$" → $|y| \le \text{poly}(|x|)$, R: poly-time always assumed

② "AM": rec: $L \in NP$ iff $L \le_m^P SAT$

let it be randomized

$L \in AM$: $x \in L \Rightarrow$ ~~$R(x) \in SAT$ whp~~

~~for most y, $R(x,y) \in SAT$~~

for most y, $\exists z$ $R(x,y,z)=1$

(∃ more usual)

$x \notin L \Rightarrow$ for most y, $\forall z$, $R(x,y,z)=0$.

$x \notin L \Rightarrow$ for most y, $\forall z$, $R(x,y,z) = 0$.

Shorthands: "Я y" = "for $\geq \frac{2}{3}$ of all y's" (of fixed len.)

MA = "∃Я / ∀Я"

$M(x,y,z)=1$ $\quad$ $M(x,y,z)=0$

AM = "Я∃ / Я∀"

(We'll see: "$\frac{2}{3}$" doesn't matter, as usual.)

rec: BPP = "Я/Я"

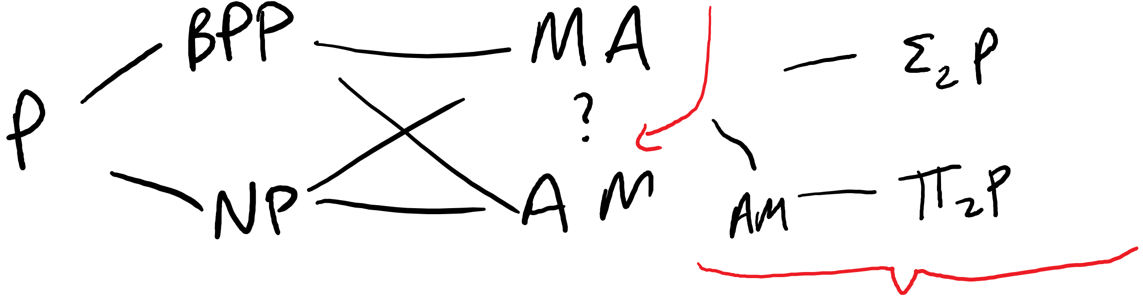
NP = "∃/∀"

$\Sigma_2 P$ = "∃∀/∀∃"

ex: co-(Q/Q') = (Q'/Q)

(test your intuition...)

Clearly:



(later addition)

today's thms:

① MA ⊆ AM

② MA, AM equivalent to 1-sided-error vers.

I.e.: • MA = "∃∀/∀Я"

$x \in L \Rightarrow$ ∃ proof y : Arthur *always* acc.

$\notin$ $\quad$ ∀ "proofs" y : Arthur rej. whp.

$\notin$ $\forall$ "proofs" y : Arthur rej. whp.

- $\mathsf{AM} = ``\forall\exists/ꓘ\forall"$

Cor: $\mathsf{MA} \subseteq \Sigma_2\mathsf{P}$, $\underset{\substack{\rotatebox{90}{$\subseteq$}\\ \mathsf{MA}}}{\mathsf{AM}} \subseteq \Pi_2\mathsf{P}$

cor: $\mathsf{BPP} \subseteq \Sigma_2\mathsf{P} \cap \Pi_2\mathsf{P}$

Belief: $\mathsf{MA} = \mathsf{NP}$,

Belief: $\mathsf{AM} = \mathsf{NP}$.

(All evidence for $\mathsf{BPP} = \mathsf{P}$ (existence of hard fcns) also $\Rightarrow \mathsf{MA} = \mathsf{NP}$. Existence of "very" hard fcns $\Rightarrow \mathsf{AM} = \mathsf{NP}$.)

(rem: No (?) problems known in MA, not known in $\mathsf{NP} \cup \mathsf{BPP}$. (Couple very obscure ones in prMA.)

rem: Several natural probs in AM not known in $\mathsf{NP} \cup \mathsf{BPP}$. E.g.:

- given some multivar polys w/ int coeffs, $\exists$? common root over $\mathbb{C}$
- $\overline{\mathsf{GISO}} \in \mathsf{AM}$

(rec: has 2-round interactive pfs where Arthur has "private coins")

(G_0, G_1): · A. picks $j \in \{0,1\}$ rand, tells M. rand. relabeling of G_j

· M must guess j.

later: Can make this "public coin"!

today thm ③: $\underbrace{AMAM\cdots A}_{\text{finite round, public coin I.P.}} = AM.$ (!) (the "hierarchy" collapses!)

---

thm①: $\underline{MA \subseteq AM}$

Say $L \in MA$. $x \in L \Rightarrow \exists y \; Pr[\overset{\text{rand.}}{A}(x,y)=1] \geq \frac{2}{3}$

$x \notin L \Rightarrow \forall y \; Pr[A(x,y)=0] \geq \frac{2}{3}$

("Publishable proof")

Let V run many times, take maj. ans.

Boost $\frac{2}{3} \rightarrow 1 - 4^{-p(n)}$, where

$n = |x|$, $p(n) = |y|$.

$n = |x|$, $p(n) = |y|$.

$$\underset{\text{Merl.}}{\exists} y \;\; \underset{\text{Art.}}{\mathcal{S}|_{1-4^{-p(n)}}} z \;\; \underset{\text{det'ic referee}}{R}(x,y,z) = 1$$

Now just let Art announce his rand coins first!

When $x \in L$, Merlin just chooses "the good" y, (and everything's OK.)

When $x \notin L$: (Now Merlin can deviously try to send a sneaky "cert" dep'ing on coins. But...) For all $2^{p(n)}$ y, at most $4^{-p(n)}$ frac. of z choices yield (wrongly) $R(x,y,z) = 1$. $\therefore$ (union bound) except for $\leq 2^{-p(n)}$ frac. of z's, $R(x,y,z) = 0$ (rightly). $\square$

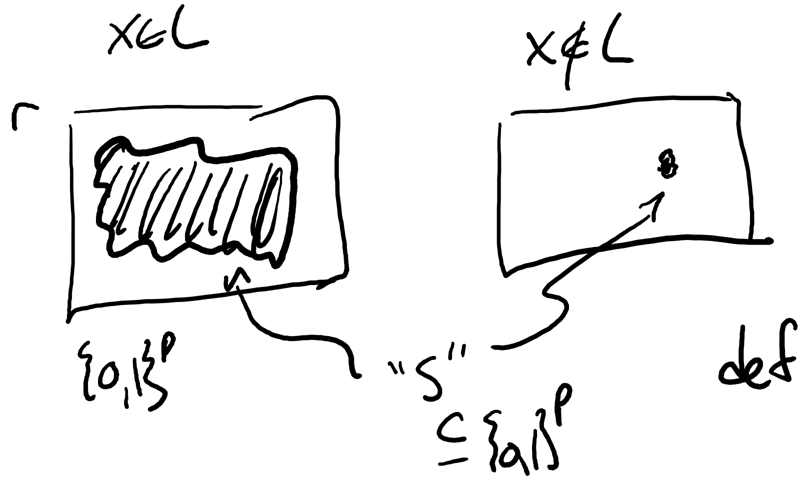
thm ②: one-sided error .... $\Rightarrow$ BPP $\subseteq \Sigma_2 P \cap \Pi_2 P$ [Sip-Gács-Lautemann]

prove first

Suff to show BPP $\subseteq \Sigma_2 P$. (Why? co-)

Say $L \in BPP$. $x \in L \Rightarrow \Pr_{r \in \{0,1\}^p}[R(x,r)=1] \geq \frac{2}{3} \rightarrow 1-2^{-n}$

$\notin$ $\Pr_r[R(x,r)=1] \leq 2^{-n}$.



▨: where $R(x,r)=1$

S

def: "Translate of S by w" $\leftarrow \in \{0,1\}^p$

$w \oplus S := \{w \oplus r : r \in S\}$.

rem: $|w \oplus S| = |S|$.

Idea: consider $U := (w_1 \oplus S) \cup (w_2 \oplus S) \cup \dots$
$\dots \cup (w_{p(n)} \oplus S)$

- If S has "density" $\leq 2^{-n}$, density of U is $\leq p(n) 2^{-n}$ = tiny $\forall w_1, \dots, w_p$
- OTOH, if S has density $\geq 1-2^{-n}$, & $w_1, \dots, w_p$ are random, U is very prob. everything

pf: Fix r. $\Pr_{w's}[U \not\ni r] \leq (2^{-n})^{p(n)}$.

Union-bound over $r \in \{0,1\}^p$:

$$\Pr[U \text{ misses anything}] \le 2^p \cdot 2^{-np}$$

$$= 2^{(1-n)p}$$

$$= \text{tiny}.$$

$\therefore$ $x \in L \Rightarrow \text{Я} w_1, \dots, w_p \ \forall r : r \in U$

$x \notin L \Rightarrow \forall w_1 \dots w_p \ \text{Я} r : r \notin U$

"$r \in U$" is poly-time det'ic:

$$\Longleftrightarrow w_1 \oplus r \in S \text{ or } w_2 \oplus r \in S \text{ or } \cdots w_p \oplus r \in S$$

$$\Longleftrightarrow \bigvee_{i=1}^{p} R(x, w_i \oplus r) = 1$$

$\therefore$ $\mathrm{BPP} \subseteq \text{Я}\forall / \forall\text{Я} \subseteq \exists\forall / \forall\exists = \Sigma_2 P$. $\square$

cor: $\subseteq \forall\text{Я} / \text{Я}A$ (by co-)

rec: $\mathrm{BPP} = \text{Я} / \text{Я}$.

cor: $\mathrm{MA} = \exists\text{Я} / \forall\text{Я} \subseteq \underbrace{\exists\text{Я}}\forall / \forall\forall\text{Я}$
(ex)

$\subseteq \exists\forall / \forall\text{Я}$

$= \mathrm{MA}$ with 1-sided err.

cor: $\mathrm{AM} = \underline{\text{Я}}\exists / \underline{\text{Я}}\forall \subseteq \underline{\forall\text{Я}}\exists / \underline{\text{Я}\forall}\forall$ (?)

(?)

$\subseteq \forall\exists / \text{Я}\forall \subseteq$

(END LECTURE)

$L \in \mathrm{AM}$: $x \in L \Rightarrow \Pr_y[\exists z \; V(x,y,z) = 1] \geq \frac{2}{3}$

$x \notin L \Rightarrow \qquad \leq \frac{1}{3}$

$\geq 1 - 2^{-n}$, $\leq 2^{-n}$?

Yes: AM has efficient error reduction:

"Arthur" chooses $y_1, \dots, y_T$ indep.,

"Merlin" sends $z_1, \dots, z_T$,

"Ref" outputs $\mathrm{Maj}_{i=1}^{T} \{V(x, y_i, z_i)\}$.

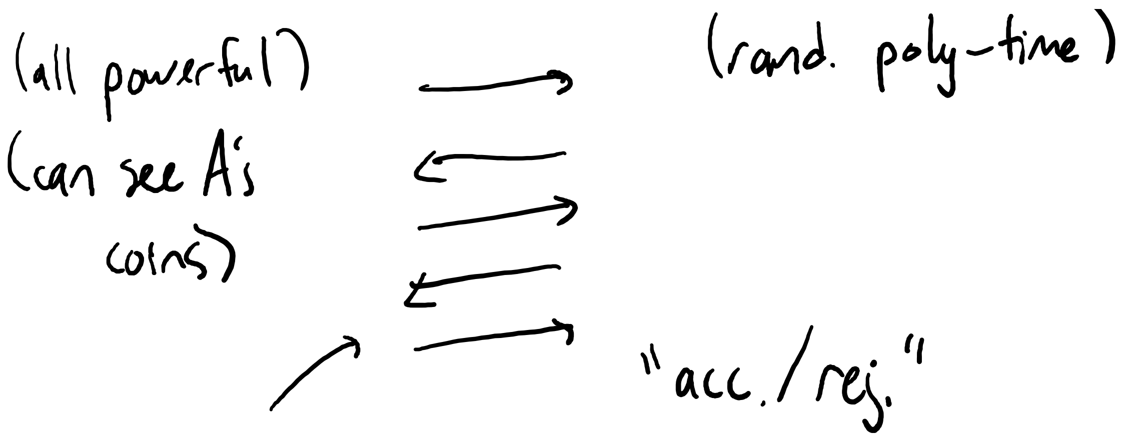
"Parallel Rep."

(Doesn't help Merlin to see all $y_1, \dots, y_T$; may as well play opt'ly on each game.)

("Public-coin") Interactive proofs:

$x \overset{?}{\in} L$

$\boxed{M}$ $\boxed{A}$



"MAMAM protocol"

L ∈ MAMAM if: $x \in L \Rightarrow \exists M$ s.t. $Pr[acc.] \geq \frac{2}{3}$

$x \notin L \Rightarrow \forall M \quad Pr[acc.] \leq \frac{1}{3}$

Claim: MAMAM (or any sim.) has efficient error reduction, by "parallel repetition"

(T parallel convos, Art acting indep, ⇒ error $2^{-\Theta(T)}$)

WLOG: All of Arthur's deterministic comp. can be moved to the end.

→ Art. just flips coins & sends them

→ after all msgs, det'ic ref V takes transcript, outputs acc./rej.

claim: "MA" = MA, "AM" = AM

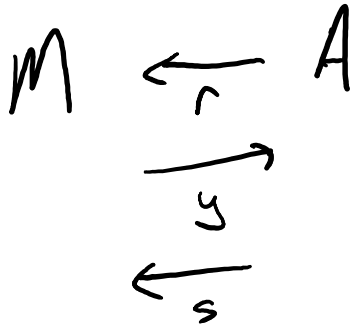
thm: [Babai '85] MAMA⋯$\underline{M}$ = AM

$\underbrace{\qquad\qquad\qquad}_{\text{any fixed len}}$

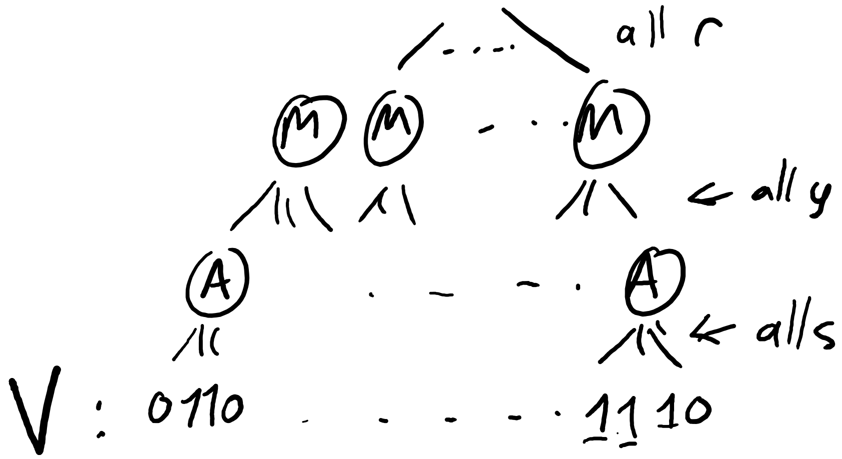
(later: poly-len ⇒ PSPACE!)

AMA e.g.:

$x \overset{?}{\in} L$



$V(x,r,y,s) = 0/1$



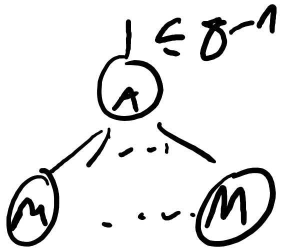
Pr[x acc.] = A → Average kids

M → Max kids

Obs: WLOG (error reduc.), final Pr[x acc.] (Ⓐ)

is, e.g. $\geq 1-8^{-n}$ or $\leq 8^{-n}$.

Say $x \notin L$, so



Markov: at most 4^{-n} frac. of Ⓜ's are outputting $\geq 4^{-n}$

Lecture 11 - Constant Round Interactive Protocols

Last time:

MA: "$\exists \text{Я} / \forall \text{Я}$" ← "for most"

i.e. $L \in MA$ if $x \in L \Rightarrow \exists y\ \text{Я} z\ V(x,y,z)=1$

$x \notin L \Rightarrow \forall y\ \text{Я} z\ V(x,y,z)=0$

(poly-len bounded) (poly time det)

AM: "$\text{Я}\exists / \text{Я}\forall$".

thm: $MA \subseteq AM$

thm: $BPP = \text{"Я/Я"} \subseteq \text{"Я}\forall / \forall\text{Я"}$ ⓐ

& "$\forall\text{Я} / \text{Я}\forall$" ⓑ (by co-ing)

idea: $AM = \text{"Я} \cdot NP\text{"} = \underline{\text{Я}}\exists / \underline{\text{Я}}\forall \underset{?ⓐ?}{\subseteq} \underline{\text{Я}\forall}\exists / \forall\underline{\text{Я}\forall}$ ☹

(true, but not useful...)

$?ⓑ? \subseteq \underline{\forall\text{Я}}\underline{\text{Я}\exists} / \underline{\text{Я}\forall}\underline{\forall\forall}$ ☺

$\subseteq \forall\exists / \text{Я}\forall$ (1-sided err ver.)

(write on board ahead of time)

↓

idea: error amplif:

$x \in L \Rightarrow$ Я r "yes"

$x \in L \Rightarrow Я_{\geq 1-2^{-n}}\, r$ "$r \in S$"

$x \notin L \Rightarrow Я_{\geq 1-2^{-n}}\, r$ "$r \notin S$"

$S = \{r : V(x,r) = 1\}$

Considered $U = w_1 \oplus S \cup \dots \cup w_p \oplus S$

S huge $\Rightarrow$ Я shift seqs $w_1 \dots w_p$ $\forall z$ "$z \in U$"

S tiny $\Rightarrow$ $\forall$ " " " " $Я z$ "$z \notin U$"

"$z \in U$": $\underbrace{w_1 \oplus z \in S \text{ OR } \dots \text{ OR } w_p \oplus z \in S}_{\text{checkable in PTIME}}$ $(\ddagger)$

☹/☺? [Let's do ☹ even tho not useful; psychologically easier....]

Q1: In "Я∃/Я∀", can we get expon. small err in Я part?

A: Yes... later.

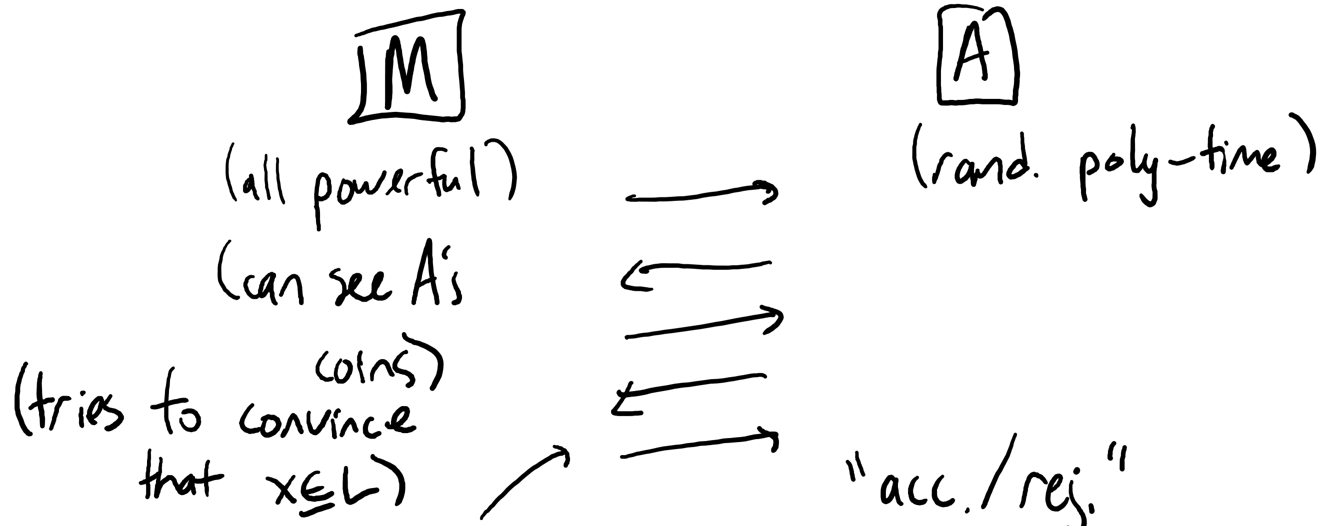
Q2: Now $S = \{r : \langle x, r \rangle \in SAT\}$

Is $(\ddagger)$ ("$z \in U$") in NP?

A: Yes. (NP closed under poly-union & also poly-intersec, for ☺.)

Const-round, "public coin", interactive pfs:

$x \overset{?}{\in} L$



"MAMAM protocol"

$L \in$ MAMAM if: $x \in L \Rightarrow \exists M$ s.t. $\Pr[\text{acc.}] \geq \frac{2}{3}$

$x \notin L \Rightarrow \forall M \quad \Pr[\text{acc.}] \leq \frac{1}{3}$

Claim: MAMAM (or any sim.) has efficient error reduction, by "parallel repetition"

(T parallel convos, Art acting indep, $\Rightarrow$ error $2^{-\Theta(T)}$)

WLOG: All of Arthur's deterministic comp. can be moved to the end.

→ Art. just flips coins & sends them

→ after all msgs, det'ic ref V takes transcript, outputs acc./rej.

claim: "MA" = MA, "AM" = AM

thm: [Babai '85] $\underbrace{MAMA\cdots M}_{\text{any fixed len}} = AM$

(later: poly-len ⇒ PSPACE!)

idea (?): $MA \subseteq AM$. $\therefore$

$$A\underline{MA} \subseteq AAM = AM$$

$$\underline{MA}M \subseteq AMM = AM$$

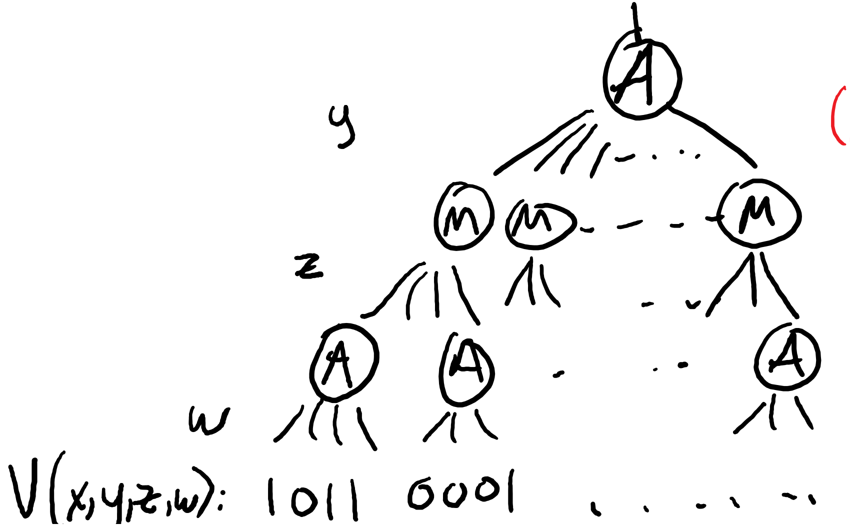
$$A\underline{MA}M \subseteq AAMM = AM$$

etc. ?

(It's more complicated than that.)

sketch of correct pf:

Say $L \in AMA$. Given x...



(rand strings A might send)

(strings M might send)

To compute overall acc. prob.: Ⓐ ≡ average
Ⓜ ≡ max.

After error reduction, can assume
$x \notin L \Rightarrow$ Ⓐ on top is $\leq 16^{-n}$, say.
(and sim., $\geq 1-8^{-n}$ when $x \in L$)

$\therefore$ (Markov-esque): At most 4^{-n} of the Ⓜ's computing $\geq 4^{-n}$.

(So it's not just A catches cheating M whp; it's better; whp over his first move, he gets into a sitch where he catches M whp.
Compare, e.g., 50% of 1st moves get him a 99% win rate, 50% get him 40% win rate. Overall win is $70\% \geq \frac{2}{3}$, but he doesn't frequently have a compelling adv.)

Can repeat arg. at all stages, conclude:
AMA = "Я∃Я/Я∀Я";
AMAMA = "Я∃Я∃Я/Я∀Я∀Я" ⊛ etc.
(not obv!)

Now: MA ⊆ AM actually showed can switch

$\exists\text{Я} / \forall\text{Я}$ with $\text{Я}\exists / \text{Я}\forall$.

(good!)

More precisely, $\exists \underset{\substack{\uparrow \\ \text{len } p}}{y}\ \text{Я}_{\geq 1-4^{-p}}\, z \rightsquigarrow \text{Я} z \exists y$

(Daniel's Q: why not $\text{Я} \underset{\nwarrow p}{y}\ \exists \underset{\uparrow q}{z} \rightsquigarrow \exists z \text{Я} y$?

Need $2^q \ll \text{Я}$'s error!)

Can appropriately amplif. errors to do all swaps in ⊛.

(Each swap causes polynomial blowups in param. lens & running time; hence only constant-round AM prots compressible to AM.)

* * *

Starting motiv.: what if "efficient" = BPP?

PH motiv.: all langs solvable eff. if
NP 〃 eff.

thm: $NP \subseteq P \Rightarrow PH \subseteq P$.

Q: What if $NP \subseteq BPP$?

A: [Zachos] thm: $\Rightarrow PH \subseteq BPP$.

Pf: Assume $NP \subseteq BPP$. $\Rightarrow$ $\underset{(\forall P)}{coNP} \subseteq BPP$

$\Sigma_2 P = \exists \forall P$

$L \in$

$$x \in L \iff \exists y \ \ \underbrace{(x,y) \in UNSAT}_{\downarrow \text{ now in } BPP}$$

$\therefore$ visibly, $L \in MA$.

$\subseteq AM$. (Recall orig. def as langs randomly reducible to SAT.)

$L \in AM \Rightarrow \exists$ rand poly-time R s.t. . . .

$$\underbrace{\begin{array}{lll} x \in L & \Rightarrow & R(x) \in SAT \text{ whp} \\ x \notin L & \Rightarrow & R(x) \notin SAT \text{ whp} \end{array}}_{\text{both checkable in } BPP}$$

(using a little error red. here.)

$\therefore L \in BPP$.

So: $\Sigma_2 P \subseteq BPP \underbrace{\subseteq \Pi_2 P}$.

(Our BPP in hierarchy thm. Or, just $AM \subseteq \Pi_2$)

$\therefore$ PH collapses to $\Sigma_2 P = \Pi_2 P$, & both are BPP. $\square$

Obs. (Saved for Toda's Thm, later.)

This proof "relativizes". I.e.,

∀ langs A, $NP^A \subseteq BPP^A \Rightarrow PH^A \subseteq BPP^A$.

(what/why? What this means in terms of classes like Я∃Я/Я∀Я is that the "V" at the end gets an A-oracle. Basically, just check this changes nothing: error amplif., closure under OR, it's all the same.)

* * *

(One more similar theorem. Motivation...)

rec: $\overline{GISO} \in AM$ (we'll show; we saw it for private coins, which we'll show equiv to public)

$\Downarrow$

$GISO \in coAM$.

clearly: $GISO \in NP$.

Q: could GISO be NP-complete?

(Anecdote: apparently Levin delayed publishing his paper b/c he couldn't prove it. We now...

[Bab'16] know GISO in $2^{\text{polylog}(n)}$ time, so if NP-complete, then NP $\subseteq \text{TIME}(\text{Qpoly})$ which is very unlikely, would contradict ETH. But even back in 1987 we hat good evidence against it.)

<u>If</u> you believe AM = NP, ⇒ coAM = coNP, so GISO ∈ NP ⇒ NP = coNP, PH collapses.

(In fact, can get uncondit. PH collapse.)

[BHZ'87] thm: If a lang. in coAM is NP-complete ($\Leftrightarrow$ NP $\subseteq$ coAM) then $\Sigma_2 P = \Pi_2 P$ (= AM).

pf: Suppose NP $\subseteq$ coAM, $\Leftrightarrow$ coNP $\subseteq$ AM.

Then
$$\begin{aligned}\Sigma_2 P &= \exists \cdot \text{coNP} \\ &\subseteq \exists \cdot \text{AM} \\ &= \text{MAM} \\ &= \text{AM} \\ &\subseteq \Pi_2 P. \qquad \square\end{aligned}$$

Lecture 12: Approximate Counting

(Start with a useful digression from elt'ry prob.)

Suppose $A_1, \ldots, A_m$ are events, $\Pr[A_i] = p$ $\forall i$.

Assume A_i's independent.

Let $T =$ # of A_is that occur.

Let $\mu = E[T] = \underline{mp}$. Then:

① If $\mu \le \frac{1}{4}$, $\Pr[T=0] \ge \frac{3}{4}$

② If $\mu \ge 8$, $\Pr[T \ge 1] \ge 3/4$.

③ If $\frac{1}{4} \le \mu \le \frac{1}{2}$, $\Pr[T=1] \ge 1/8$.

(leave up)

Proofs: $T \sim \text{Binomial}(m,p)$, just calculate.

Something "softer"?

① Markov: $\Pr[T \ge 4\mu] \le \frac{1}{4}$. Didn't need independence.

(Do ② & ③ "need" independence?)

(How do you bound $\Pr[T \ll \mu]$?) Chebyshev:

② $\Pr[T \notin [\mu - 2\sigma, \mu + 2\sigma]] \le \frac{1}{2^2}$, $\sigma = \sqrt{Var[T]}$.

$Var[T] = \underline{mp(1-p)} \le mp = \mu$.

If $\mu \geqslant 8$, $\mu - 2\sigma \geqslant \mu - 2\sqrt{\mu} > 1$.

$\Rightarrow$ ②

LPT: "Chebyshev only needs <u>pairwise indep.</u>"

$T = \sum_{j=1}^{m} I_j$. $\mathrm{Var}[T] = E[T^2] - E[T]$ $\searrow mp$

{ 0/1 indic. for A_j.

$$E[T^2] = E\Big[\sum_{j,j'} I_j I_{j'}\Big] = \sum_j E[I_j^2] + \sum_{j \neq j'} E[I_j I_{j'}]$$

$$p^2 = E[I_j]E[I_{j'}] \overset{\uparrow}{=}$$

indep.

<u>OR</u> just pairwise indep.

∴ Variance calc. same in pairwise indep & fully indep (Binomial) case.

∴ pairwise indep. of A_is suffices for ②.

<u>ex</u>: " " " " " " ③.

def: Given ckt C, <u>$\#C$</u> $:= \#\{x : C(x) = 1\}$

def: <u>Approx-#CKT-SAT</u> task: given ckt C, output α s.t. $\alpha \overset{2}{\approx} \#C$

$$\#C \le \alpha < 2\#C \quad \nearrow$$

(NB: not a decision prob., so doesn't quite make sense to ask if it's in P, BPP, etc.)

def: (Promise) decision version: Given C, s, say YES if $\#C > 2s$, NO if $\#C \le s$.

Remarks:
- Solving Approx. ver $\Rightarrow$ solving decision ver.
- Decision ver. is at least as hard as CKT-SAT: take $s=0$.
- $\boxed{\text{Today:}}$ not "much" harder: it's in (pr)AM.
- Decision ver + binary search $\Rightarrow$ approx. ver.

  $\Rightarrow$ Approx-#CKT-SAT "$\in$" $\cancel{P^{(pr)AM}}$

  "$\in BPP^{NP}$"

  (Given NP oracle, can solve prob. whp in poly time.)
- Factor "2" unimportant:

  e.g.: can do factor 64 $\Rightarrow$ can do factor 2

(Stockm'85) trick: Say you can do factor 64.

Given (C, s)... Form circuit "$C^{\otimes 6}$":

Has $6n$ inputs: $C^{\otimes 6}(x^{(1)}, \dots, x^{(6)}) =$

$$C(x^{(1)}) \wedge C(x^{(2)}) \wedge \dots \wedge C(x^{(6)}).$$

$\#C^{\otimes 6} = \underbrace{(\#C)^6}_{}$. So "do $(C^{\otimes 5}, s^6)$" to factor 64; i.e., decide

$$\#C^{\otimes 6} \le s^6 \quad \text{OR} \quad \#C^{\otimes 6} > 64 s^6$$

$$\iff \quad \#C \le s \quad \text{OR} \quad \#C > \underset{2}{\cancel{64^{1/6}}} \cdot s$$

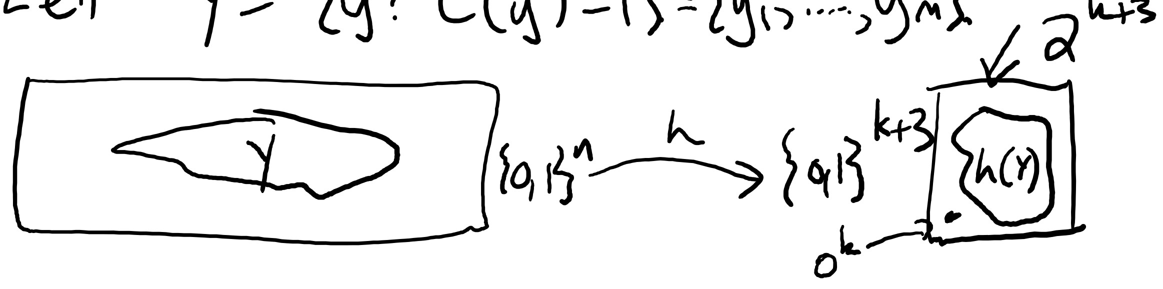
$\therefore$ any factor $K \to K^{\frac{1}{\text{poly}(n)}} = 1 + \frac{1}{\text{poly}(n)}$

Thm [GS'86]: There's an AM protocol for factor-$\underset{(64)}{\cancel{2}}$ approx-#CKT-SAT decision

Given C, s, want Arthur to acc. w.p. $\geq \frac{2}{3}$ if $\#C > 64s$, rej. w.p. $\geq \frac{2}{3}$ if $\#C \le s$. (If neither, don't care.)

idea Suppose $s = 2^k$. Arthur picks a "random hash fcn" $h : \{0,1\}^n \to \{0,1\}^{k+3}$, challenges Merlin: "tell me x s.t. $C(x) = 1$ & $h(x) = 0^k$". Merlin responds, Arthur checks both.

Let $Y = \{y : C(y) = 1\} = \{y_1, \dots, y_m\}$.



Let $A_i = \underset{\text{(choice of } h)}{\text{event}}$ that $h(y_i) = 0^k$.

Let T = # A_i's that occur.

Merlin can win iff $T \geq 1$.

Assuming h fully random,

$$\Pr_h[A_i] = \underbrace{2^{-k-3}} \quad \& \ A_i\text{'s indep.}$$

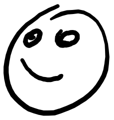
Case ①: $m = \#C > 64s = 2^{k+6}$:

$$E[T] \geq 8 \Rightarrow \Pr[T \geq 1] \geq \frac{3}{4}.$$

② $m = \#C \leq s = 2^k$:

$$E[T] \le 2^{-3} \le \frac{1}{4} \Rightarrow \Pr[T=0] \ge \frac{3}{4}.$$

(slack compensates if s not power of 2)



($\underline{s=0 \text{ case?}}$,)

$\boxed{?}$

Problem: How can Arthur choose/send a fully rand. $h \in \mathcal{H} = \{\text{all fcns } \{0,1\}^n \to \{0,1\}^k\}$??

Needs 2^{k2^n} bits to specify!

Sol$^{\underline{n}}$: Choose from a much smaller, "2-universal hash family $\mathcal{H}$" with 2 properties:

① Specifying $h \in \mathcal{H}$ requires only $O(kn)$ bits. (& h's efficiently computable)

② For $h \sim \mathcal{H}$, the 2^n events A_y = "$h(y) = 0^k$", $y \in \{0,1\}^n$, are pairwise indep.; (All we $\underline{\text{needed}}$.)

i.e., ⓐ $\forall y$, $\Pr_{h\sim\mathcal{H}}[h(y)=0^k] = 2^{-k}$,

ⓑ $\forall y \neq y'$, $\Pr_{h}[h(y)=h(y')=0^k] = 2^{-2k}$.

* * *

How to choose h:

- pick $r^{(1)},\ldots,r^{(k)} \in \{0,1\}^n$ unif., indep.
- pick $b_1,\ldots,b_k \in \{0,1\}$ " , " .
- $h(y) = (r^{(1)} \underset{\substack{\nearrow\\ \text{dot prod.}}}{\cdot} y \underset{\substack{\uparrow\\ \text{mod 2}}}{+} b_1, \ldots, r^{(k)}\cdot y + b_k) \in \{0,1\}^k$

Ppty ①? ✓

Ppty ②a? Even fixing $r^{(i)}$'s, randomness of b_i's means $h(y)$ is unif'ly rand. on $\{0,1\}^k$.

②b? First consider when $h(y)=h(y')$.

$\Rightarrow h(y)-h(y') = (0,\ldots,0) \mod 2$

$\Rightarrow (r^{(1)}\cdot(y-y'), \ldots, r^{(k)}\cdot(y-y')) = (0,\ldots,0)$ ⊛

Note: indep. of b_i's. So suffices to show $\Pr[h(y)=h(y')] = 2^{-k}$

when $y \neq y'$. Let $w = y - y' \in \{0,1\}^n$.
$\neq 0$.

By ⊛, $h(y) = h(y') \iff r^{(i)} \cdot w = 0 \ \forall i = 1 \ldots k$.

Since $r^{(i)}$'s indep., need to show
$w \neq 0 \Rightarrow \Pr_r[r \cdot w = 0] = 1/2$.

$\boxed{\text{Lemma}}$ ($\vee$ randomly reduces to $\oplus$)

$$w = (0, \ldots, 0) \Rightarrow \Pr_r[r \cdot w = 0] = 1,$$
$$\neq \qquad \Rightarrow \qquad 1/2.$$

Pf: w has some nonzero coord, w_j.
Imagine picking r_j last. □

* * *

Application 1 (hmwk or next class):
private-coin I.P. $\subseteq$ public-coin I.P.

Application 2: Valiant-Vazirani Theorem

(Could it be that SAT is hard because of potential multiplicity of solutions?)

potential multiplicity of solutions!)

Unique-(Ckt-)SAT: You're promised #C is 0 or 1; decide which.

(Could that help? Is that solitary sat'ing assignment a beacon guiding you?)

[W]: Not really easier than SAT.

∃ rand. reduction SAT → UniqueSAT (1-sided err)

(So if you could solve Unique case, could whp solve gen. case.)

Given C, can randomly produce

n inputs

$C^{(1)}, \dots, C^{(t)}$,

$t = O(n)$ s.t.

- $C \notin$ CKT-SAT $\Rightarrow C^{(j)} \notin$ CKT-SAT $\forall j$.
- $C \in$ CKT-SAT $\Rightarrow$ whp $\exists j$ s.t. $\# C^{(j)} = 1$.

Pf: #C is either $0, 1, 2 \dots 3, 4 \dots 7, 8 \dots 15, \dots, 2^{n-1} \dots 2^n$.

For each range $[2^k, \dots, 2^{k+1}-1]$, choose random "2-universal hash" $h_k : \{0,1\}^n \to \{0,1\}^{k+2}$.

For "correct" k, "$\underbrace{C(x)=1 \wedge h_k(x)=0^{k+2}}_{C^{(k)}(x)}$"

has $\geqslant \frac{1}{8}$ chance

of being uniquely sat'ble ("T=1")

by ③.

Use, say, 10 h's for each $k=1\dots n$;

for correct k, prob none unique-sat

$\leq \left(\frac{7}{8}\right)^{10}$ = small.

$\square$

Lecture 13 – Valiant-Vazirani & Exact Counting

Approx-#CKT-SAT decision task: Given (C, s), accept if $\#C \geq 2s$, rej. if $\#C \leq s$.

Last time: it's "in" AM.

App #1: Private-coins IP $\subseteq$ Public-coins IP.

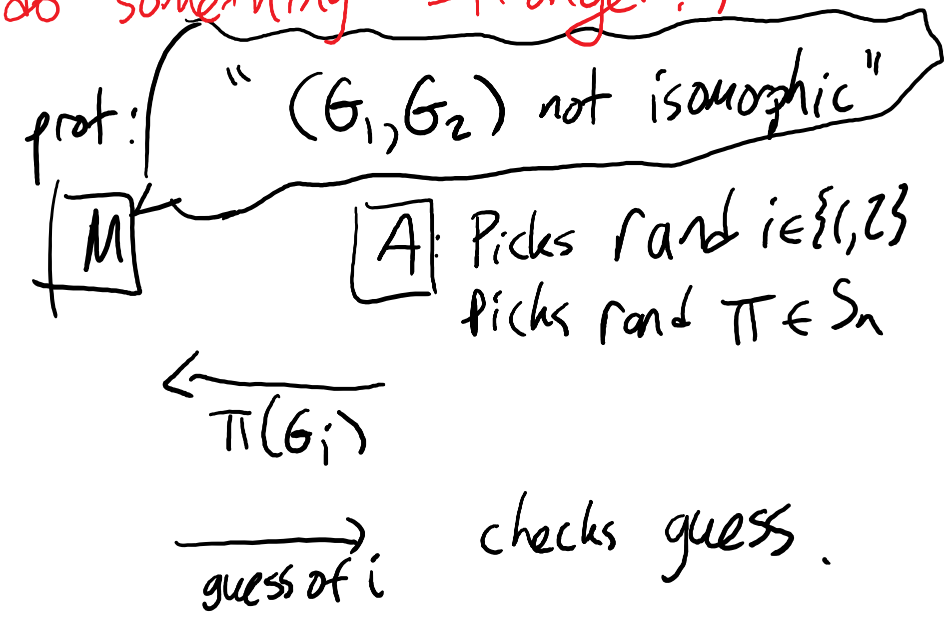
App#2: Complexity of Unique-SAT.

App#1: See homework.

Today: sketch of GNISO $\in$ AM.

(We won't even give a proper proof! But [AB] has one, & hmwk will do something stronger.)

Private coins prot:



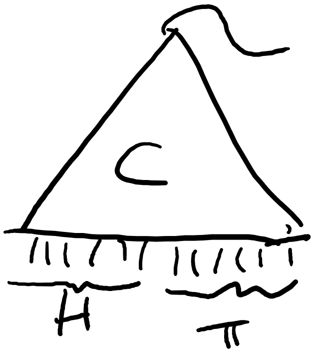
(skip!!)

Public coins?? Assume, with l.o.g., that G_1, G_2

Public coins?? Assume, with l.o.g., that G_1, G_2 have no self-isomorphisms ⊛

("Rigid": no auts, $\pi(G)=G$.)

A&M think about



$$\begin{cases} 1 & \text{if } \pi(H)=G_1 \text{ or } G_2 \\ 0 & \text{else} \end{cases}$$

$\boxed{\#C = ?}$

Case1: $\underline{G_1 \cong G_2}$: Then $\#C = \underline{n!}$, (Using ⊛)

Case2: $G_1 \not\cong G_2$: $\#C = \underline{2n!}$,

$\therefore$ distinguishing is in AM. "□"

* * *

App #2: Unique-(Ckt-)SAT: You're promised #C is 0 or 1; decide which.

(Could that help? Is that solitary sat'ing assignment a beacon guiding you?)

<u>Valiant-Vazirani Thm:</u>

Not really easier than SAT.

∃ rand. reduction SAT → UniqueSAT

(1-sided err)

(So if you could solve Unique case, could whp solve gen. case.)

Given C, can randomly produce $C^{(1)}, \ldots, C^{(t)}$,
$\nearrow$ n inputs

$t = O(n)$ s.t.

- $C \notin$ CKT-SAT $\Rightarrow C^{(j)} \notin$ CKT-SAT $\forall j$.
- $C \in$ CKT-SAT $\Rightarrow$ whp $\exists j$ s.t. $\# C^{(j)} = 1$.

Pf: $\# C$ is either $0, 1, \underline{2 \ldots 3}, \underline{4 \ldots 7}, 8 \ldots 15, \ldots$ $2^{n-1} \ldots 2^n$.

For each range $[2^k, \ldots, 2^{k+1} - 1]$, choose random "2-universal hash" $h_k : \{0,1\}^n \to \{0,1\}^{k+2}$

For "correct" k, "$C(x) = 1 \wedge h_k(x) = 0^{k+2}$" ⊛ has $\geq \frac{1}{8}$ chance of having a unique sol^n.
("③" about $T=1$, last time)

$\because$ h_k easy to compute, $\therefore$ ⊛ expressible as a poly-size ckt $C^{(k)}$.

Use 10 h's for each $k = 1 \ldots n$ to reduce failure prob. from $\frac{7}{8} \to \left(\frac{7}{8}\right)^{10}$ = small.

$\square$

Exact Counting

Q: What is the cxty of computing #C exactly?

(This is a fcn prob., not a decision prob. Let's define a cxty class for it...)

Consider NTM M that, given $C : \{0,1\}^n \to \{0,1\}$, uses nondet. to guess $y \in \{0,1\}^n$, accepts iff $C(y) = 1$. $\therefore$ # acc. "branches" is #C.

def: **#P** $= \{ f : \{0,1\}^* \to \mathbb{N} : \exists$ poly-time NTM M s.t. $\underbrace{\#M(x)}_{\text{\# of acc. branches of } M(x)} = f(x) \}$

technicality: WLOG, can assume makes exactly $p(|x|)$ nondet. guesses on input x. (ex).

$\therefore$ #CKT-SAT $\in$ #P. So is #CNF-SAT, #HAM-CYCLE, #3COL, etc. (Most NP probs have a natural counting ver.)

rem: These probs all seem <u>very</u> hard.
"$\in$" PSPACE; not clear if "$\in$" PH

(we'll see they're prob. not, in fact!)

Also in #P: #CYCLE, #DNF-SAT.

Interesting: These <u>also</u> seem hard, even tho decision probs – "does G have a cycle?", "does this DNF have a sat'ing asgn?" – <u>easy</u>

(Alg. for 1st: try to topo-sort,
2nd: <u>Say "YES" (unless empty)</u>)

(Why hard?) [AB]: HAM-CYCLE "$\leq_T^P$" #CYCLE

$\in P^{\#CYCLE}$

- #DNF-SAT $\in P^{\#CNF\text{-}SAT}$

Given DNF φ, form $\neg\varphi$, a CNF. (of same size)

$$\#(\neg\varphi) = 2^n - \#\varphi. \quad \square$$

Another sim. e.g.: #PERF-MATCHINGS (in bip. G)

(decision in P)

Most important prob. in #P...

* * *

(Issues re) <u>Fcn. vs. decision</u> (Probs.)

(Issues re) Fcn. vs. decision (Probs.)

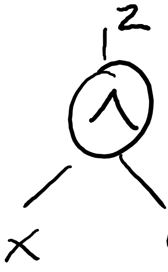
Reductions: Analogue of $\le_m^p$ is "parsimonious reductions": "preserve answer"

$f \le_m^p g$ if $\exists_{\text{poly-time}}\, R : f(x) = g(R(x))$.

↑ not std not^n

e.g.: #CKT-SAT $\le_m^p$ #3SAT.

why?



⇝ little 3CNF encoding $z = x \wedge y$

(no "extraneous sol^ns" introduced; each setting of input vbls 'forces' auxiliary vbls)

fact: Cook-Levin is parsim.:

#NTM-ACC-PATH $\le_m^p$ #CKT-SAT

(i.e., given NTM M, & x, what is #M(x)?)

(pf sim. to above; just think)

cor: #CKT-SAT, #3SAT are "#P-complete"

empirical fact: ∃ parsim reducs between "all" classic probs: #HAM-CYCLE, #3COL, etc.

All #P-complete.

* * *

Also $f \text{ "}\leq_T^P\text{" } g$: $f \text{ "}\in\text{" } P^g$. Eg: #DNF-SAT $\leq_T^P$ #CNF-SAT

$\uparrow$(technically, FP)

(the negation, $2^n -$ thing)

def: <u>#P-complete</u> is trad. defined using $\leq_T^P$.

So, e.g., #DNF-SAT is #P-complete.

* * *

(Fitting languages into the picture.)

def: <u>$P^{\#P}$</u>: <u>langs</u> decidable in poly-time w/ a #P oracle ($\Longleftrightarrow$ w/ a #SAT oracle)

(Recall, you can turn a fcn. prob. into a decision prob. by asking for a single bit.)

- Asking for MSB of #C is asking if $\geq \frac{1}{2}$ of all assignments sat. C.

"Maj-SAT" problem, complete for <u>PP</u>,

(Given NTM M, input x, do $\geq \frac{1}{2}$ of nondet choices/

coin flips give acc.?)

- Asking for LSB of #C (!):
  asking if #C odd/even. "Odd-SAT"!
  Complete for class "$\oplus P$"

def: $L \in \oplus P$ iff $\exists$ ptime NTM M
s.t. $x \in L \iff \#M(x)$ odd.

(Crazy class? Yes! But...)

thm: $SAT \in RP^{\oplus P}$

pf: Valiant-Vazirani! $C \xrightarrow{\text{rand}} C^{(1)}, C^{(2)}, \ldots$

unsat → all unsat ⇒ all $\underline{\text{even}}$

sat → at least one 1 ⇒ ≥ 1 $\underline{\text{odd}}$.

$\square$

(More weirdness...)

$\oplus P$, $PP \subseteq P^{\#P}$. (Obvious)

$P^{\oplus P}$, $P^{PP} \subseteq P^{\#P}$. (Obvious.)

thm 1: $P^{PP} = P^{\#P}$.

thm 2: $P^{\oplus P} = \oplus P$. $= \oplus P^{\oplus P}$ (!)

thm 2: $P^{\oplus P} = \oplus P. \quad = \oplus P^{\oplus P}$ (!)

(You may think I've gone off the deep end. Like, we're really supposed to care about $\oplus P^{\oplus P}$?!? Insanity! But actually, key for Toda's Thm.)

Toda's 1st Theorem: $PH \subseteq BPP^{\oplus P} \subseteq BPP^{\#P}$.

(Message: (allowing randomness) ability to solve #SAT is more powerful than all of PH.)

(Implies #SAT "$\notin$" PH unless it collapses.)

Pf sketch: (We'll clean up next time.)

$$\begin{array}{lcccc} \text{V.V.} \Rightarrow & \text{(up to randomness)} & NP & \text{"}\subseteq\text{"} & \oplus P \\ \Rightarrow & \text{"} \quad \text{"} & NP^{NP} & \text{"}\subseteq\text{"} & \oplus P^{\oplus P} \\ & & \| & & \| \\ & & \Sigma_2 P & & \oplus P \end{array}$$

$\rightsquigarrow NP^{\Sigma_2 P} = \Sigma_3 P$ "$\subseteq$" $\oplus P$, etc.

Toda's 2nd Theorem: (much later)

$BPP^{\oplus P} \subseteq PP^{\oplus P} \subseteq P^{\#P}$.

"Derandomizes" Toda 1.

rem: $P^{\#P}$ is huge. $\underline{\underline{\text{No}}}$ evidence against $P^{\#P} = \text{PSPACE}$.

$\underline{\text{pf of } P^{PP} = P^{\#P}:}$ Suffices to show $\supseteq$.

" " " $\#\text{CKT-SAT} \in P^{\text{MAJ-SAT}}$.

Given C (n inputs) & fixed number N,

$0 \le N < 2^n$, let C' be

$C'(x, \underset{\overset{\uparrow}{1 \text{ bit}}}{b})$ = "If $b = 1$ then $C(x)$

else output 1 iff $x \geq N$ as n-bit nums."

$\#C' = \underbrace{\#C + (2^n - N)}$. $\therefore$ $\#C' \in$ MAJ-SAT iff

$$\#C + (2^n - N) \geq \frac{1}{2} \cdot 2^{n+1}$$

$$\Longleftrightarrow \quad \#C \geq N.$$

$\therefore$ can use MAJ-SAT queries to bin. search for $\#C$! $\square$

Lecture 14 - Toda's 1st Theorem & The Permanent

Today: Toda 1: $PH \subseteq BPP^{\oplus P}$

(later) 2: $\subseteq P^{\#P}$

(Recall message: #SAT-power >> PH.)

Ingreds: ① Valiant-Vazirani:

$SAT \xrightarrow{rand} USAT, \Rightarrow NP \subseteq RP^{USAT}$
$\subseteq BPP^{\oplus P}$

② $\oplus P^{\oplus P} = \oplus P$ (mucking around w/ NTMs)

③ $NP \subseteq BPP \Rightarrow PH \subseteq BPP$ (mucking around w/ $MA \subseteq AM$)

& it "relativizes"!

Toda Proof [Fortnow]:

"VV relativizes" → $$\left.\begin{aligned} NP^{\oplus P} &\subseteq BPP^{\oplus P^{\oplus P}} \\ &= BPP^{\oplus P} \end{aligned}\right\} (?)$$

∴ by rel'd ③: $$\begin{array}{c} PH^{\oplus P} \subseteq BPP^{\oplus P} \\ \cup\! \mid \\ PH. \end{array}$$

□

PH.

(?) (Let's spell this out!!)

$NP^{\oplus P} \subseteq BPP^{\oplus P}$? Say $L \in NP^{\text{OddSAT}}$, so

L dec'd by ptime NTM M with O-oracle ← OddSAT

To decide L in $BPP^{\oplus P}$:

Given x: Apply Cook-Levin, get a circuit-with-O-gates C_x s.t.

$x \in L \iff C_x$ sat'ble.

Apply V.V. to C_x: $\xrightarrow{\text{rand}} C^{(1)}, C^{(2)}, \ldots$ ckts w/ O-gates,

need to decide if ≥ 1 has an $\oplus$ odd # of sat'ing asgns.

- "Is $\#C^{(i)}$ odd?" is a $\oplus P^{\oplus P}$ problem. $\rightsquigarrow \oplus P$.

$\therefore$ $BPP^{\oplus P}$ can do $\oplus$. □

* * *

Course: Now $\frac{1}{2}$-complete!

Congrats: With #P (& $\oplus$P), you now know all "std." cxty classes.

(And more! S_2P, BPP_{path}, ...)

(Now if you ever encounter a class you don't know – rest assured it's obscure!!)

(BQP....)

#P, $P^{\#P}$: (lovely class, b/c) has a complete problem w/ several beautiful properties: Permanent

(cf. how great NP's complete prob, SAT is)

PERM is:
- "downward self-reducible"
- "randomly " " ", i.e., has worst-case → avg.-case hardness reduction
- has an "instance checker"

(we'll see today)

complete for "AlgNP"

(we'll see classes def'd by arithm. circuits)

...

(Cxty theorists love the perm! So what is it?)

(leave

$$\boxed{\text{perm}(A) = \sum_{\sigma \in S_n} \underbrace{A_{1,\sigma(1)} \cdot A_{2,\sigma(2)} \cdots A_{n,\sigma(n)}}_{\prod_i A_{i,\sigma_i}.}}$$

$n \times n$ mtx

permutations

(leave up!)

("Gen'd diag.")

cf: $\det(A) =$ mult. "$\text{sgn}(\sigma)$" $\in \{\pm 1\}$ here

(Which looks more complic'd? Naively: det!)

For A with entries from $\mathbb{N}$:

det $\in$ FP

perm is #P-complete! [Valiant '79]

e.g.

$$\text{perm}\begin{pmatrix} 3 & 1 & 4 \\ 1 & 5 & 9 \\ 2 & 6 & 5 \end{pmatrix} = 3 \cdot \text{perm}\begin{pmatrix} 5 & 9 \\ 6 & 5 \end{pmatrix} + 1 \cdot \text{perm}\begin{pmatrix} 3 & 4 \\ 2 & 5 \end{pmatrix} + 4 \cdot \text{perm}\begin{pmatrix} 1 & 5 \\ 2 & 6 \end{pmatrix}$$

"cofactor expansion"

(correctness follows from def.)

(det same, but w/ alt'ing ±1's)

- n recursive calls to perm_{n-1}

(so still expon. time, but...)

(so still expon. time, but....)

→ "downward self-reducible"

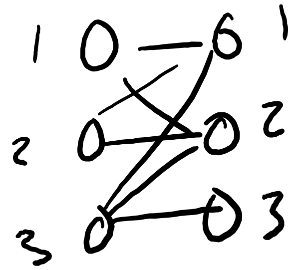
(will use later in a thm about "derand ⇒ ckt lower bounds")

- Swapping 2 rows (or cols): perm unchanged

(det swaps sign)

(det has another prop: adding one row to another leaves unchg'd. Lets one do G.E. → poly-time alg. No such for perm.)

Meaning? Let A be 0/1 "adj Mtx" for bip graph G.



$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

(put $A_{ij}=1$ if i on left touches j on right)

Then perm(A) = # perfect matchings in A.

(Literally straight from defining formula.)

∴ $PERM_{0,1} \in \#P$. (Guess/check p.m.)

[Val]: PERM is #P-hard (even 0/1 case)

[Val]: $\text{PERM}_{0,1}$ is #P-hard (even 0/1 case)

① (later): #3-SAT $\varphi \rightsquigarrow$ A with entries $\{-1,0,1\}$

m clauses $\qquad \text{perm}(A) = C^m \cdot (\#\varphi)$

↑ (a one-off, very careful gadget pf)

(↑ 64 in [AB], sthg else in our proof)

② : $\text{perm}_{-1,0,1} \leq^p_T \text{perm}_{\mathbb{N}}$ (today)

③ : $\text{perm}_{\mathbb{N}} \leq^p_m \text{perm}_{0,1}$. (later)

rem: ③ $\Rightarrow \text{perm}_{\mathbb{N}} \in \#P$. (Not obv. $\text{Perm}_{-1,0,1} \in \#P$ imposs. ∵ neg. But in $FP^{\#P}$.)

Two proofs of ②.

Pf 1: Mod. arith.

Suppose $A \in \mathbb{Z}^{n\times n}$, entries are L-bit ints (input size: n^2L). $|\text{perm}(A)| \leq \underbrace{\ ?\ }$

$\downarrow$

$n!$ perms σ $\cdot \left|\prod_{i=1}^{n} A_{i,\sigma(i)}\right| \leq (2^L)^n \qquad n!\,2^{Ln} = Q$

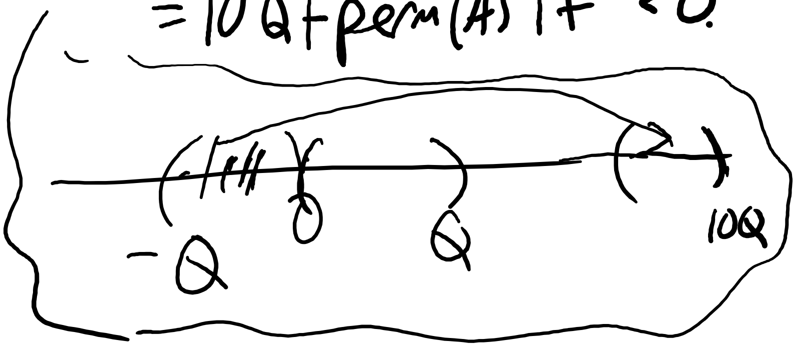
☺ Ans. is $\lesssim n\log n + Ln$ bits.

☺ Ans. is $\lesssim n \log n + 2n$ bits. Q

perm(A) mod 10Q : $= \text{perm}(A)$ if it's ≥ 0

$= 10Q + \text{perm}(A)$ if < 0

$\therefore$ suffices to compute



- Replace all neg. entries of A with equiv-mod-10Q positive ones $\rightsquigarrow A'$
- Compute $\text{perm}_{\mathbb{N}}(A')$ (Q has poly bits, still poly time)
- Reduce answer mod 10Q. ☐ (and do diagram thing)

Proof 2: Interpolation

Given $A \in \{-1,0,1\}^{n \times n}$. (indet.)

Replace all −1's with "x", forming A_x.

$\text{perm}(A_x) = \ldots$ a (univariate) poly in x of degree $\leq n$.

Call it $p(x)$.

Given $\text{PERM}_{\mathbb{N}}$ oracle, can compute p(x)

on [illegible] #'s say 0, 1, 2, ..., n

on nonneg. #'s, say $\underbrace{0, 1, 2, \ldots, n}_{n+1 \text{ pts, } \deg n.}$

Interpolate! (Lagrange) to get p(x)'s coeffs!

Now plug in $x = -1$. □

Perm is "randomly self-reducible"

Focus on $\mathrm{PERM}^{n \times n}_{\mathbb{Z}_p}$, p prime, $p > n+1$

(PERM is very natural over finite fields) [BF90]

thm: (Lipton '91) Suppose ∃ alg $\mathcal{O}$ that computes $\mathrm{PERM}_{\mathbb{Z}_p}$ on "most" inputs:

$$\Pr_{A \sim \mathbb{Z}_p^{n \times n}} [\mathcal{O}(A) = \mathrm{perm}(A)] > 1 - \frac{1}{3n}.$$

(not worst-case good, but good on rand input)

(natural, why we went to fin. field)

Then $\mathrm{PERM}_{\mathbb{Z}_p} \in \mathrm{BPP}^{\mathcal{O}}$.

↓

ALL inputs computed correctly (whp over "coins").

(whp over coins).

rem 1: PERM "hard on avg" ⇒ "worst-case hard"

(this is very rare; don't know such a thing for, say, SAT)

rem 2: Can improve $1-\frac{1}{3n}$ to $\frac{1}{2}+\varepsilon$

Pf: Given any $A \in \mathbb{Z}_p^{n \times n}$.

Pick $R \in \mathbb{Z}_p^{n \times n}$ uniformly.

Let $B_x = A + xR$.

Note: $B_1, B_2, \dots, B_{p-1}$ all unif. rand (not indep!)

Let $p(x) = \mathrm{perm}(B_x)$, poly of deg. $\leq n$.

$$\Rightarrow \Pr_{x \sim \{1,2,\dots,p-1\}} [\mathcal{O}(B_x) \neq p(x)] < \frac{1}{3n}.$$

Pick $a_1, \dots, a_{n+1} \in \{1, \dots, p-1\}$ unif & distinct.

Union bound: $\Pr[\mathcal{O}(B_{a_i})$ wrong for any $a_i]$ $< \frac{n+1}{3n} \approx 1/3$.

∴ whp we have

$n+1$ correct pairs $(a_i, p(a_i))$.

Interp., get $p(x)$.

Output $p(0)$. $\square$

Lec 15 – Algebraic Complexity Theory

(In 1 lecture! It's a huge topic. Std. ref. is 600page book by...) Bürgisser-Clausen-Shokrollahi

(last time we punted on showing PERM is #P-complete; I'll show later. Today we'll intro. an "alg'ic cxty class for which PERM is complete – and again punt on proof ☺.)

(Alg. cxty sort of a mathy topic, predating "Boolean" cxty theory in some ways. About...)

How many "arithmetic ops" needed to...

- eval. a polynomial?

$$a_0 + a_1x + a_2x^2 + \cdots + a_nx^n.$$

(put q's on board to start)

Naive: · mults: $n-1$ (for x^i's)

$+ n$ (for coeffs)

· adds: n

"Horner's Method":

$$a_0 + x(a_1 + x(a_2 + x(a_3 + \cdots (a_{n-1} + xa_n))\cdots)$$

mults: n, adds: n.

Ostrowski '54 conj: $\geq n$ mults/divs nec.
(even if ± free)
(Kickstarted area, took ≈10yrs to fully prove.)

- compute coeffs of
$(a_0 + a_1 X + \dots + a_n X^n)(b_0 + b_1 X + \dots + b_n X^n)$?

Naive: · $O(n^2)$ ops
· Kolmogorov '60 conj.: $\Omega(n^2)$ (implic.)
· $O(n \log n)$ doable! (Strassen-Schönhage)
How? $\mathbb{D}$

- compute D.F.T.? (Multiplying a vec. by a certain $n \times n$ mtx.)
  - FFT (Cooley-Tukey, Gauss)
- mult. two $n \times n$ mtcs?
  - Strassen: $n^{\log_2 7}$ ..... Le Gall '14: $n^{2.3728\dots}$
- compute symm. polys like $\sum_{|S|=n/2} \prod_{i \in S} X_i$?
- det? perm?

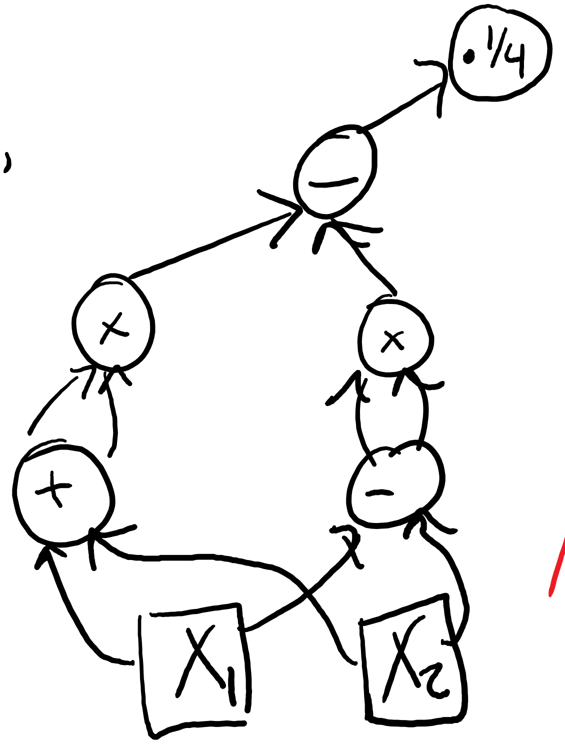
(Model for these mathy questions will be alg. circuits / formulas, ignoring bit cxty of arithmetic. Realistic, no. But natural.)

Alg. Circuits:

Dag. Nodes $+, -, \times, \div, \cdot c$, for any "scalar" c.

Inputs: "vbls" $X_1, \ldots, X_n$

Scalars come from some fixed field.



(Put up before.)

Computes... $\underbrace{X_1X_2}$

(Over const.-size fields, alg. cxty ≈ Bool. cxty; think $\cdot = \wedge$, $+ = $ XOR in $\mathbb{F}_2$. So we focus on ∞te fields. Alg.-closed is also convenient, so let's, for simplicity...)

Assume $\mathbb{C}$.

Division by 0?? (Don't stress about it.)

Think of ckt as not being #'s ⟼ #, but indets. ⟼ formal polynom. (or rational fcn).

(All the example probs were actually abt.

computing 'polys', or collections thereof.)

e.g. M.M.: On inputs $A_{11}, \dots, A_{nn}, B_{11}, \dots, B_{nn}$, compute all the n^2 deg-2 polys

$$C_{ij}(A,B) := \sum_k A_{ik} B_{kj}.$$

def: Formula: circuit is a tree (all fanouts $= 1$).

Cost Model

(two popular choices)

1. "Total" size: each gate costs 1.
2. "Nonscalar" size: $\times, \div$ cost 1, $+, -, \cdot c$ all free. (nonscalar mults. cost.)

(Bit cxty ignored. #2 seems a little unnatural! But relevant: cf Horner, controls asymps of ω in M.M. And most L.B.'s we can prove are for it, & that's stronger.)

* * *

e.g.: • Compute $X_1 X_2 \cdots X_n$.

Size $n-1$ suff. Necessary?! (We saw a

Size $n-1$ suff. Necessary?! (We saw a crazy $n=2$ case.)

Yes, nonscalar size $\geq n-1$ nec.

(Not overly easy!)

- Compute $\underline{X_1^{31}}$. (Univ. is special case, still interesting.)

$^{24}, X^{28}, X^{30}, X^{31}$. ⑧

Generally: X^m doable w/ $\leq 2 \log m$ mults.

(repeated sq'ing, combine)

(Not always optimal. Consider...)

$X^2, X^3, X^6, X^{12}, X^{24}, X^{30}, X^{31}$ ⑦

"Addition chains..."

(Here we rely on ability to reuse exprs.)

<u>Formula</u> size? $\underline{=30}$ (Each step can incr. deg by ≤ 1.)

$\leq$ deg.

(We really like formulas, so we tend to insist on poly-deg.)

(Analogue of a "language"...)

def: <u>Poly-degree family</u>: $(f_n)_{n \in \mathbb{N}^+}$, $f_n \in \mathbb{C}[X_1, \ldots, X_n]$, $\deg(f_n) \le p(n)$, p a poly.

e.g.: $\mathrm{Det}_{n \times n}$: degree n in n^2 vbls

(technically, not as above. $N = n^2$, $\deg = \sqrt{N}$, artificial convention if N not square)

$\mathrm{Perm}_{n \times n}$: deg n, n^2 vbls.

Alg. ckt size of these?? (Big Q.)

X^{31}: $X^2, X^4, X^8, X^{16}, X^{32}, X^{31}$ using ÷ ! ⑥

(so yes, to compute a poly, can help to use ÷! But not by too much...)

thm [Strassen '73]: (Infinite fields only.)
For polynoms of deg. d, can elim. ÷ from ckts at expense of $\le d^2$

multip've factor in total/nonscalar size.

(∴ basically, don't worry about ÷. Can use it in u.b.'s, ignore it in l.b.'s.)

pf sketch: To elim. "$\div(1-X)$", repl. w/ "$\times(1+X+X^2+X^3+\cdots\cdots)$".

Can trunc. @ deg d (∵ computing deg-d thing)

In gen.: can efficiently compute coeffs of Taylor series of denoms

"□"

* * *

Cor: $\mathrm{Det}_{n\times n}$ has alg. circuits with $+,-,\times$ of cost poly(n)

Pf: Gaussian elim. translates directly to a $+,-,\times,\div$ circuit of size $O(n^3)$. Use Strassen to kill $\div$.

(Rem: direct proofs exist, not too hard, via mtx inversion or char poly.

Size is even $O(n^{\omega})$.)

Formula size?

thm [Hyafil '79, Val++] (Not terribly hard.)

Circuits of size S, degree d

$\Rightarrow$ Formulas of size $S^{O(\log d)}$.

$\therefore$ Det_n has quasipoly-size formulas, $n^{\Theta(\log n)}$

$\therefore$ for poly-deg. families, qpoly-size ckts $\equiv$ qpoly-size formulas

(So it's same if you're chill about qpoly. Det generally believed to need $n^{\log n}$ size formulas!)

$\mathrm{Perm}_{n \times n}$: Naive: $\approx n!$-size formula.

Ryser '92: $O(n^2 \cdot 2^n)$-size formula,

(smaller ckts not known, I think)

(Q: Could it have small ckts? Huge open prob.)

def: AlgP/poly (AKA VP): poly-deg families computable by poly-size alg. ckts.

(AlgQP/poly : quasi-p. ≡ formulas)

def AlgNP/poly (AKA VNP):

poly-deg families (f_n) expressible as

$$f_n(X_1,\ldots,X_n) = \sum_{\substack{e_1,\ldots,e_m \\ \in \{0,1\}}} g_{n+m}(X_1,\ldots,X_n,e_1,\ldots,e_m),$$

for some $m = \text{poly}(n)$, $(g) \in$ AlgP/poly.

Why?: · $\sum_{e\in\{0,1\}^m}$ kinda like $\text{OR}_{e\in\{0,1\}^m}$?

· Shoulda been called Alg#P/poly.

fact: Def^n of AlgNP/poly unchanged if (g) must be poly-size formulas.

(Somehow "nondet'ism" helps in ckt→formula.)

prop: $\text{Perm}_{n\times n}$, #HamCycle, ... in AlgNP/poly.

pf: ↳ $\text{Perm}(X) = \sum_{\sigma\in S_n} \prod_i X_{i,\sigma(i)}$

$$= \sum_{E\in\{0,1\}^{n^2}} \text{IsPermMtx}(E)\cdot \underbrace{\prod_{i,j} E_{ij}X_{ij}}_{\in \text{AlgP/poly}}$$

claim $\in$ AlgP/poly

e.g. $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \mapsto 1$ ✓

$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \mapsto 0$ ×

✓ proof:

$$= \hat{\prod_{i=1}} \text{ExactlyOne}(E_{i1},\dots,E_{in}) \cdot \text{ExactlyOne}(E_{1i},\dots,E_{ni})$$

(all visibly in AlgP/poly)

$$\text{ExOne}(Y_1,\dots,Y_n) = \sum_i Y_i \prod_{j\neq i}(1-Y_j). \square$$

* * *

Completeness:

def: $f(X_1,\dots,X_n)$ is a *projection* of $g(Y_1,\dots,Y_m)$

if $\exists \sigma : \{Y_1,\dots,Y_n\} \to (\{X_1,\dots,X_n\} \cup \mathbb{C})$ s.t.

$f(X_1,\dots,X_n) = g(\sigma(Y_1),\dots,\sigma(Y_m))$.

e.g.: $A^2 - AB$ is proj. of $\det\begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22}\end{pmatrix}$:

$\sigma(X_{11}) = A$, $\sigma(X_{22}) = A$, $\sigma(X_{12}) = B$,

$\sigma(X_{21}) = 1.$

(Very special, restrictive notion.)

def: $(f_n) \leq_{(q)proj} (g_n)$ if $\exists\, m = (q)poly(n)$
s.t. f_n is proj. of $g_{m(n)}$ $\forall n$.

(Why so strict? Well, we can prove cool results even with it.)

hmwk: If f has formula size S,
it's a proj. of $\det_{(3S-1)\times(3S-1)}$

(Explains ubiquity of dets in math!!
Jacobians, Alexander polys, char polys...
whenever there's any kind of small formula,
there's a same-size "determinental formula!")

$\therefore$ $(\det_{n\times n})$ is complete for AlgQP/poly.

thm (Valiant '79): $(\mathrm{Perm}_{n\times n})$ complete for
AlgNP/poly. (Assuming char of field not 2! Bürgisser:
#HAM-CYCLE complete for any char.)

pf: Gadgets...

$\therefore \quad \text{AlgNP/poly} \overset{?}{=} \text{Alg}_{\overset{\uparrow}{\mathbb{Q}}}\text{P/poly}$

$\iff$ $\text{perm}_{n\times n}$ is proj of $\det_{m\times m}$ for $m = \text{quasipoly}(n)$.

Val. conj: Not so.

Open since Pólya, Szegö 1913.

(They couldn't even rule out $m=n$. Great problem for mathematicians to hear about: P vs. NP analogue w/ no mention of TMs, or computation.)

VzG'87: $m \geq 1.06n$ necessary

Mignon-Ressayre'04: $m \geq \frac{1}{2}n^2$ nec. (!)

$\not\sim$

(Relationship to P vs. NP? Easier!! So maybe you should start with it!)

thm: NP/poly $\Rightarrow$ AlgNP/poly $\neq$ AlgP/oly.

$\#P/poly$ (over $\mathbb{C}$ or $\mathbb{Q}$, under GRH)

pf sketch: By contrapos. Suppose $AlgNP/poly \subseteq AlgP/poly$,

So $PERM_{0,1}$ comptble by poly-size alg. ckts. $\overset{?}{\Longrightarrow} PERM_{0,1} \in P/poly$?

Difficulty: scalar constants.

Sps for simplicity we're over $\mathbb{Q}$, all consts. have nums & denoms of poly(n) bits. (Under GRH, can get this for $\mathbb{Q}$, $\mathbb{C}$.)

Now compute exactly, mod a rand. prime of Poly(n) bits. (Findable in BPP.)

Whp., no num. or denom. becomes 0, get correct ans. in $\mathbb{Z}$.

$$\therefore\ PERM_{0,1} \in BPP/poly$$
$$= P/poly \quad (\because BPP \subseteq P/poly)$$

$$\Rightarrow\ P^{\#P} \subseteq P/poly \quad \text{(Valiant)}$$

$\Rightarrow \quad NP \subseteq P/poly$ (Toda

$(\Rightarrow \quad PH = \Sigma_2 P$ (Karp-Lipton)).

$\square$

thm: $NP \not\subseteq P/poly \Rightarrow AlgNP/poly \neq AlgP/poly$

(over any inf. field
$\mathbb{C}$ or $\mathbb{Q}$,
assume GRH)

contrapos.:

$AlgNP/poly = AlgP/poly \Rightarrow NP \subseteq P/poly$

$\Rightarrow PH = \Sigma_2 P$

(Karp-Lipton)

$Perm_{n \times n} \subseteq AlgP/poly$

$\to P/poly$ (?)

$\Rightarrow P^{\#P} \subseteq P/poly$

$\Rightarrow PH \subseteq P/poly$ (Toda)

$\Rightarrow NP \subseteq P/poly.$

① Bit complexity. Over rationals. Assume for simplicity, all the scalar multiple consts. representable w/ n^{10} bits.

↝ work modulo n^{20}-bit

prime #.

② Constants.

$\mathbb{C}$ - irrational consts.

$\mathbb{Q}$ - $2^{2^{2^n}}$

Think of consts as variables,
imagine q.: "∃? consts to
fill in so that ckt
computes perm on "lots of
integers"

thm: $\underline{NP \not\subseteq P/poly} \Rightarrow \underline{AlgNP/poly \neq AlgP/poly}$

($\mathbb{C}$, $\mathbb{Q}$ - assume GRH)

contrapos: $\underline{AlgNP/poly = AlgP/poly \Rightarrow NP \subseteq P/poly}$

$\Rightarrow PH = \Sigma_2 P$

(Karp-Lipton)

$$\text{Perm}_{n \times n} \in \underline{AlgP/poly}$$

$$\overset{?}{\rightsquigarrow} \underline{FP/poly}$$

over $\mathbb{Q}$: assume also $\underline{\text{scalars}}$ used in circuit all poly(n)-bits $\Big\}$ $2^{2^{2^{\cdot^{\cdot}}}}$?!

intermediate bit cxty?

$\rightarrow$ suffices to work mod p, p is Poly(n) bits.

$\text{Perm}_{0,1} \in FP/poly$

$P^{\#P} \subseteq P/poly$

$\Rightarrow PH \subseteq P/poly$ (Toda)

$\Rightarrow NP \subseteq P/poly.$

Elim. constants

- $\because$ $\mathrm{PERM}_{n\times n}$ has deg n

  $\therefore$ if C computes PERM correctly on "enough" ints.

- imagine replace scalar consts in circuit by "variables",

  ask yourself:

  "$\exists$? settings for vbls making the ckt compute

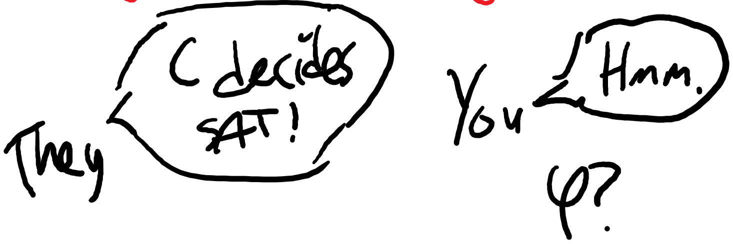
  ↗ Perm correctly on integers?"

  $\exists$ sol^n to some poly equations w/ int coeffs, & bounded deg.

Lecture 16 - In which somebody claims to be able to compute Permanent but you're not sure whether to believe them

(Let's start w/ SAT. Say some SAT-solving person claims to have an alg — a ckt, say — solving SAT.)

(Not sure how to check, but say you just care abt a specific input φ.)



$C(\varphi) = $ "yes" → use DSR

→ either get sat. asgn, or det. "C is bogus!"

$C(\varphi) = $ "no" → ?

((critically used in...))

Karp-Lipton: $SAT \in P/poly \Rightarrow PH = \Sigma_2 P$

(hmwk) $(= S_2 P)$

rec: [Kan'82] (Lec. 8) $\Sigma_4 P \not\subseteq SIZE(n^{100})$.

(or any n^c)

LOL: $\Sigma_2 P$ neither.

pf: If $\Sigma_2 P \not\subseteq P/poly$. → done.

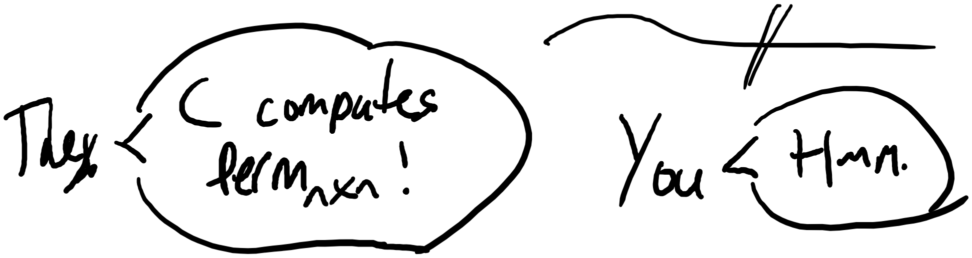
Else $SAT \in \Sigma_2 P/poly \Rightarrow PH = \Sigma_2 P$

$$\begin{tikzcd} PH \arrow[d, phantom, "\parallel"] & \Sigma_2 P \arrow[dl, phantom, "\parallel"] \\ \Sigma_4 P & \end{tikzcd}$$

$SIZE(n^{100}) \not\supseteq \Sigma_4 P$ □

cor: S_2P neither. (Same proof.)

(Today: We'll eventually see sim./better Karp-Lipton collapses, hence ckt lower bds. Will follow b/c Perm is DSR & RSR.)



Case 1: C an alg. ckt, ±, ×, over $\mathbb{Z}$

(for simplic; assume also consts are poly(n) bits)

Perm is DSR:

$$\boxed{ID:n} \quad Perm_{n\times n}(X) = \sum_{i=1}^{n} X_{1,i}\, Perm_{n-1}(X^{(i)}),$$

where $X^{(i)} = X$ w/ row 1, col. i del'd.

ID:(n-1), ..., ID:2: same, about $k \to k-1$.

ID:1: $\mathrm{Perm}_{1\times 1}(X) = X$.

(Satisfying all these ≡ <u>being</u> Perm.)

If C sats. all ID:1, ..., ID:n, it indeed computes $\mathrm{Perm}_{n\times n}$.

(Wait: a little catch. C only takes n×n inputs, How do you plug in a k×k mtx?)

For $A \in \mathbb{Z}^{k\times k}$, define "C(A)" $= C\left(\begin{array}{c|c} A & 0 \\ \hline 0 & I_{n-k} \end{array}\right)$.

(Take that as def^{n}. Correct if C is correct.)

Each ID:k is an instance of "PIT"

(Poly Ident Testing: Given 2 alg. ckts over $\mathbb{Z}$, do they compute ident. polys?)

Rec Lec 5, [AB ch 7.2.3]: PIT $\in$ coRP.

(eff'ly solvable!)

Pf used "Schwarz-Zippel Lemma":

If degree-d polys p, q agree on rand inputs from $\{1, ..., M\}^n$ with prob $> \frac{d}{M}$, then $p \equiv q$.

[So given C, $\deg(C) \le 2^{size(C)}$, take $M \gg$ that (poly(n) bits), check if $C = C'$ on rand input → can do mod a big #.)

∴ can check all ID:k in coRP (error amplif.)

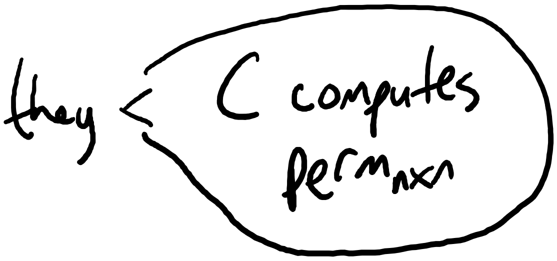
⇒ decide if C correct! (efficient!)

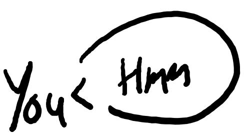
later cor: if BPP ⊆ P

⇒ ckt lower bounds for NEXP

And later: converse too! ("Hardness vs. Randomness").

Case 2: C is Boolean ckt $(\wedge, \vee, \neg)$





((Can you do the same checking trick? Well, checking if 2 Bool. ckts same is coNP-complete! :(

$n \times n$ mtx of
n^2-bit ints (say)

(But... could use randomness to check identities are "usually" true.)

0. Find n^2-bit prime p. (Doable in BPP.)

1. For $k = 1 \ldots n$:
   - Pick poly(n) random $k \times k$ mtcs mod p. Check ID:k on them, with C.

     (Note that minor of random mtx is rand.)
   - Output "C is bogus" if a failure — else become confident C correct (mod p) on $\geq 1 - \frac{1}{n^3}$ frac. of $k \times k$ mtcs.

End: (If you didn't catch a mistake, you believe in correctness on almost all $n \times n$ mtcs)

Can compute Perm(A) (mod p) for <u>any</u> worst-case A, whp, using Perm's R.S.R. on C.

(Don't output $C(A)$, do the degn interp thing from Lec. 14)

[M. Blum,

We got an <u>Instance Checker</u> for Perm (mod p): [M. Blum, Luby, Rubinf; Kannan]

def: Instance Checker for a fcn (or lang) f: a rand. poly-time alg $\mathcal{A}$ taking "oracle" C (supposedly computing f) & input x, s.t.:

① $\mathcal{A}^C(x)$ outputs an answer or "C is bogus"

② If C indeed comps. f, $\mathcal{A}^C(x) = f(x)$ w. prob. 1. (can relax)

③ If $C \neq f$, $\mathcal{A}^C(x)$ is "correct" whp

output is f(x) or "C is bogus".

(Imagine, e.g., C is correct except on <u>one</u> input. Say it's not x. Couldn't hope to notice, but OK: just output C(x). If x <u>is</u> the wrong one, better use RSR!)

Thm (proved): Perm (mod p) has an inst. checker.

(Actually, a little hackery b/c of ,

(Actually, a little hackery b/c of fcn/prime/mod thing. Langs nicer.)

* * *

thm: GISO has an instance checker. (We'll see.)

(Cool. Recall Babai '16 gave a very complic. quasipoly time alg. for GISO; 80pp., bug, etc. Imagine he coded it up, gave you some crazy code. Inst. checker is great! Can use on your fave (G,H).)

If Babai(G,H) = "Yes" → can get isomor.

= "No" → $\underbrace{\text{with confidence}}_{\text{or: find a (proof of) bug.}}$

pf/

rem: Checkers for L ≈ I.P. for L (We'll see more later.)

where:
- prover only needs power of L
- prover commits to strat.

eg: TQBF (PSPACE-complete) has checker.

(follows from IP=PSPACE...)

(follows from IP = PSPACE ⊆ ...)

thm: Any L that's DSR & RSR has a checker. (Rubinf. Proof is basically same as proof for Perm. Can be applied for PSPACE, EXP.)

(Does SAT have a checker? Don't think so...)

* * *

(Cool thing about checkers...)

thm: If L has checker:

$$L \in P/poly \Rightarrow L \in MA.$$

(Like Karp-Lipton, but better collapse, as $MA \subseteq S_2P \subseteq \Sigma_2P \cap \Pi_2P$.)

Pf: Assume $L \in P/poly$. On x, $|x| = n$, Merlin sends ckt C for L_n to Art, Art uses inst. check alg. (Rand.)

$x \in L$: Merlin honest... Art gets "$x \in L$" w/prob. 1

$x \notin L$: no matter Merlin, Art gets "$x \notin L$" or "C is bogus" whp. □

Cor: $D^{\#P} \subseteq P/poly \to D^{\#P} \subseteq MA$ [LFKN'90]

Cor: $P^{\#P} \subseteq P/poly \Rightarrow P^{\#P} = MA$. [LFKN'90]

Pf: (Basically a cor.; must be lil careful.)

$P^{\#P} \subseteq P/poly \Rightarrow$ poly-size ckt for PERM (each bit of PERM has ckt)

M sends. Art picks $p \gg 2^{n^2}$, uses inst. check to answer all #P-queries... □

rem: Doesn't "relativize". (Think abt why...)

LOL conseq: [Vereshchagin'92]

thm: $PP \not\subseteq SIZE(n^{100})$.

(Is that good?)

$$\underbrace{MA^{?} \subseteq S_2P \overset{\checkmark}{\subseteq} \Sigma_2P \overset{\checkmark}{\subseteq} \Sigma_4P \overset{\checkmark}{\subseteq} PH}_{\text{test}\ \subseteq} \overset{\text{TODA}}{\subseteq} P^{PP} = P^{\#P}$$

$$PP \overset{\checkmark}{\subseteq} P^{PP}$$

pf: $PP \not\subseteq P/poly \rightarrow$ done.

Else $PP \subseteq P/poly \Rightarrow P^{PP} \subseteq P/poly$

$P^{PP} = P^{\#P}$.

$\Rightarrow P^{\#P} = MA. = PP = \Sigma_4P$ ∴ done by

$= PP = \Sigma_4 P$. ∴ done by Kannan

These ideas + IP stuff + more:

Santhanam '09:

- $\underline{pr}MA \not\subseteq SIZE(n^{100})$
- (or $MA/1$)
- (just using today's lecture...)

At least one of:

(i) $MA \not\subseteq SIZE(n^{100})$

(ii) "$MA \not\subseteq AlgSize(n^{100})$"

$\hookrightarrow$ $\exists$ p-fam (p_n) s.t.

$L = \{(x, p_n(x)) : x \in \mathbb{N}^n\} \in MA$,

p_n doesn't have size $O(n^{100})$ $\mathbb{Z}$-alg-ckts.

Lecture 17 - IP = PSPACE

(This is a famous thm in cxty theory from '90. Kicked off many other famous thms, like MIP = NEXP, PCP Theorem... "Nonrelativizing." Illustrates surprising power of interaction & randomness in proof verification.)

def: $\underline{\text{IP}} = \text{AM}[\text{poly}]$ (Languages w/ poly-round AM prots.)

ex: $\text{IP} \subseteq \text{PSPACE}$. (Most reasonable classes in PSPACE. Can evaluate poly-depth, expon. fan-in "game tree".)

[Sha'90]: $\text{TQBF} \in \text{IP}$

($\therefore$ $\text{PSPACE} \subseteq \text{IP}$.)

Immed. followed... [LFKN'90]: $\#3\text{SAT} \in \text{IP}$

($\Rightarrow$ $\text{P}^{\#\text{P}} \subseteq \text{IP}$)

(Prior, people believed maybe IP = AM. Oracles against $\text{coNP} \subseteq \text{IP}$.)

1st: proof of $\#3\text{SAT} \in \text{IP}$.

More generally, let $f(x_1,\dots,x_n)$ be a polynom. of deg $d \le \text{poly}(n)$ over $\mathbb{Z}_p$, where: $\cdot\ p \approx 2^{2n}$ a prime, known to A & M

- Arthur can efficiently compute f

(May not know its coeffs — prob. doesn't in fact, ∵ expon. many. Merlin knows all...)

Consider a claim like

$$\circledast \quad \text{“} \sum_{x_1\in\{0,1\}} \sum_{x_2\in\{0,1\}} \cdots \sum_{x_n\in\{0,1\}} f(x_1,\dots,x_n) = \underset{\overset{\uparrow}{\in \mathbb{Z}_p}}{c} \text{”} \quad (\text{mod } p)$$

There is an I.P. for such claims:

- if true, M convinces A w. prob. 1

if false, A rejects whp.

Relevance? For #3SAT: Input is some 3CNF $\phi = (x_1 \vee \bar{x}_2 \vee \bar{x}_5) \wedge (\bar{x}_2 \vee x_6) \wedge \dots$

($m = O(n^3)$ clauses.)

Let $f = \left(1-(1-x_1)x_2x_5\right)\cdot\left(1-x_2(1-x_6)\right)\ldots$

For $x\in\{0,1\}^n$, $f(x) = \begin{cases} 1 & \text{if } x \text{ sats } \phi \\ 0 & \text{else.}\end{cases}$

So $\circledast \equiv$ "#sat asgns for $\phi = c \pmod{p}$".

Merlin's claim. $\because$ $p \gg 2^n$, same as <u>not</u> mod p.

(Merlin can send p, Arthur checks.)

Arthur can eval f: ✓

<u>$d = \deg(f) \le 3m = \text{poly}(n)$: ✓</u>

Description of IP for $\circledast$:

A: "Consider $g(Y) := \sum_{x_2\in\{0,1\}} \cdots \sum_{x_n\in\{0,1\}} f(Y, x_2, \ldots, x_n)$.

That's a $\deg \le d$ <u>univariate</u> poly in Y over $\mathbb{Z}_p$. Tell me, Merl: what are its coeffs?"

M: Sends some $g'(Y)$ explicitly
(d coeffs, $\log p$ bits each, $= \mathrm{poly}(n)$ bits)

A: Checks $g'(0) + g'(1) = c$.
Suspicious about $g' \stackrel{?}{=} g$.

(How do you test 2 low-deg. polys are same? Pick a rand. input!)

Picks $r \in \mathbb{Z}_p$ at random. Computes $c_2 := g'(r)$. Needs convinced that

$$c_2 = g(r) = \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} \underbrace{f(r, x_2, \ldots, x_n)}_{f_2(x_2, \ldots, x_n)}$$

* * *

Analysis: If orig claim ⊛ correct, Merlin answers honestly, $\Pr[A \text{ accepts}] = 1$.

Sps a given claim is false.

We'll show: either A catches M, or else except w. prob. $\le \frac{d}{p} < \frac{1}{2n}$, M forced into false inductive claim. (Now can union-bound over all n rounds. A can know truth by self once $n=1$.)

Why? Say $c \neq \sum\sum\cdots\sum f(x)$.
M can't send real $g(Y)$ b/c then $g(0)+g(1) \neq c \rightarrow$ A catches. So $g'(Y) \neq g(Y)$. Distinct deg-d polys agree in $\le d$ places. $\therefore$ except w. prob. $\le \frac{d}{p}$, $g'(r) \neq g(r)$, so next claim is false. □

* * *

(Interlude: power of the prover.)

Q: How powerful does Merlin need to be?

A: Ability to compute #P fcns suffices.

(An honest Merlin could just be a #P oracle.)

Also for "TQBF ∈ IP", we'll see it suffices for Merlin to be PSPACE.

(these facts ⇒ #SAT, TQBF have inst. checkers)

prop: Suppose $L, \bar{L} \in IP$ with Merlin implementable in P^L. Then L has inst. checker.

pf: Hmwk!

(not a long, but proof still ok)

cor: TQBF, GISO, #SAT have inst. checkers. (As promised last time.)

Rem: Reverse is not nec. true.

Being in IP harder than having checker.

↓ Arthur must "beat" "adaptive" Merlin.

↓ "Merlin strat" committed to in advance.

* * *

TQBF ∈ IP

TQBF ∈ IP

$\Phi(x) := \forall x_1 \exists x_2 \cdots \exists x_n \, \phi(x)$ ← 3CNF ← (OK, by theorem)

Again, repl. by $f(x_1, \ldots, x_n)$, poly computing ϕ on $\{0,1\}^n$.

Idea 1: $\forall x_i \rightsquigarrow \prod_{x_i \in \{0,1\}}$? Sorta works, but

$d = \deg(\Phi(x))$ becomes $\approx m^n$ → too big for Merl to send univ. g's.

Idea 2: If only care about poly $f(x)$ on $x \in \{0,1\}^n$, no point in it ever having x_i^2 (or higher): $x_i^2 \equiv x_i$.

$$f \underset{\text{on } \{0,1\}^n}{=} \text{Multilinearization}(f)$$

$$= L_1 L_2 \cdots L_n f, \text{ where}$$

- L_i "multilinizes" x_i:

eg: $L_1 f(x) = x_1 \, f(1, x_2, \ldots, x_n) +$

eg: $L_1 f(x) = x_1 f(1, x_2, \dots x_n) +$
$(1-x_1) f(0, x_2 \dots x_n)$.

(leave up)

(Exact claim: L_i reduces x_i-deg to 1, even formally, while pres. f's value on 0-1 inputs)

- $\forall x_1 f \to \prod_1 f = f(1, x_2 \dots x_n) \cdot f(0, x_2 \dots x_n)$
- $\exists x_1 f \to$ "$\coprod_1 f$" $= f(1, x_2 \dots) + f(0, x_2 \dots) - f(1, x_2 \dots) \cdot f(0, x_2 \dots)$.

$$\therefore \Phi(x) \underset{\text{on } \{0,1\}^n}{\equiv} \prod_1 L_1 \coprod_2 L_1 L_2 \prod_3 \dots \dots L_{n-1} \coprod_n L_1 \dots L_n \phi(x).$$

(Rem: L_i's don't get rid of any vbl.)

(Incidentally, ∵ answer exactly 0/1, p just needs to be large enough to beat the d/p.)

NB: ϕ has deg $\leq 3m$, L_i's bring it down to n,
$\coprod_n$ raises to $\leq 2n$, L_i's " " " " "
$\prod_{n-1}$ " " " - - - ∴ Always $\leq$ poly(n).

IP now much like before: (Maybe ex: it's in

– text...)

Init., Merlin claims

$$\circledast \quad \underbrace{\prod_1 L_1 \coprod_2 \cdots\cdots \phi(x)}_{\text{a lin., univ poly, } g(x_1)=ax_1+b.} = 1 \mod p.$$

Merlin sends some $g'(x_1) = a'x_1 + b'$.

A checks $g'(0) \cdot g'(1) = 1 \mod p$.

→ If A gets convinced $g' = g$, → convinced of $\circledast$.

Picks $r_1 \in \mathbb{Z}_p$ at rand, tries to check $g(r_1) = g'(r_1) \mod p$ (If $g' \neq g$, true only w. prob. $\leq \frac{1}{p}$.)

$$L_1 \underbrace{\coprod_2 L_1 L_2 \forall_3 \cdots L_n \phi(x)}_{?} \Big|_{x_1 = r_1} = c_1$$

$\coprod_2$: a lin poly in x_1, x_2; e.g. $a + bx_1 + cx_2 + dx_1x_2$

$\coprod_2$ of it is: plug in $x_2 = 0, 1$, add, subtract prod.

L_1: a quadratic in x_1.

$g(x_1) = ax_1^2 + bx_1 + c$.

M sends a quadratic $g'(x_1)$

A checks $\left(x_1 g'(1) + (1-x_1)g'(0)\right)\Big|_{x_1=r_1} = c_1$.

Then picks $s_1 \in \mathbb{Z}_p$ at rand, tries to check $g(s_1) = g'(s_1) \mod p$

$\swarrow \qquad \searrow$

$$\coprod_2 \underbrace{L_1 L_2 \forall_3 \cdots L_n \phi(x)}_{\text{multilin poly in } x_1, x_2.}\Big|_{x_1=s_1} = c_2$$

$x_1 := s_1$ what's left is some linear $g(x_2)$. M esl sends g'.

A checks $g'(0) + g'(1) - g'(0)\cdot g'(1) = c_2$.

Then picks $r_2 \in \mathbb{Z}_p$ at rand, tries to check $g(r_2) = g'(r_2)$

$\swarrow \qquad \searrow \#$

$$\underbrace{L_1 L_2 \forall_3 \cdots L_n \phi(x)}_{\text{lin in } x_2, \text{ quadrat in } x_1}\Big|_{\substack{x_1=s_1 \\ x_2=r_2}} =$$

$\rightarrow$ plug in $x_2 = r_2$ - - . . . - - . "

- - - - . $\square$"

Lecture 18 - Random Restrictions & AC^0 lower bounds

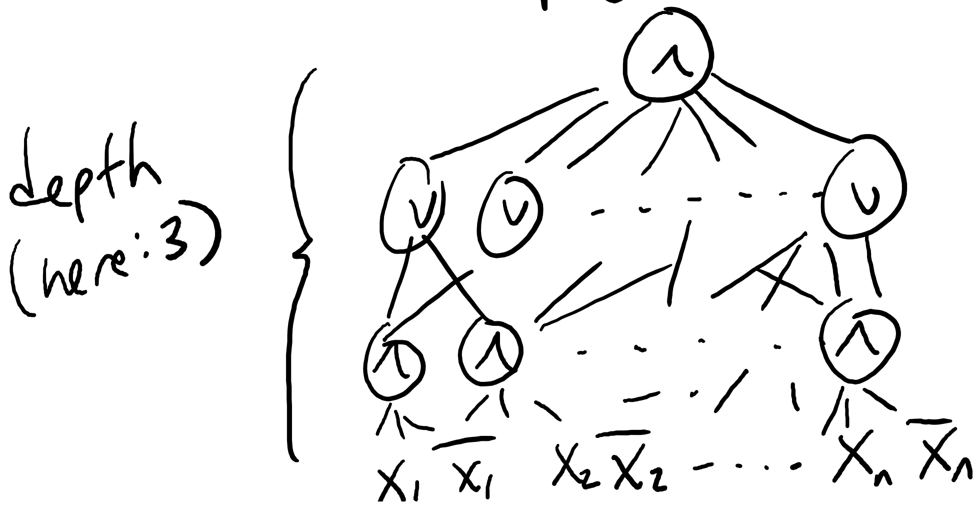
(Will spend a few lectures now on "Concrete Complexity" - a branch of cxty theory devoted to "combinatorial" lower bounds for concrete, simple models of computation. Specifically, circuit lower bounds.)

Dream: $NP \not\subset P/poly$. (Super hard. Try to build up to it from weaker models?)

$\swarrow$ P :(

$\downarrow$ $\rightarrow$ weak circuit models (So... maybe we just showed models very weak, as opposed to NP hard.)

today: "AC^0": poly-size constant depth ckts



(Can assume ¬ pushed to bottom by de Morgan; only doubles size. In fact, we don't

...

charge for them)

(Consec. $\textcircled{\scriptsize$\wedge$}$ or $\textcircled{\scriptsize$\vee$}$ gates mergeable so we can assume "layered" into d alt'ing levels.)

(Unbdd fan-in $\wedge, \vee$ of course.)

Size S: # of $\vee, \wedge$ gates

Width w = max bottom fan-in (A key param.)

(Why care? Gotta start somewhere! A model of "constant parallel time" computation.
Depth-2 = old friends CNFs, DNFs.
Depth-3: open problems of fund. cxty import!)

def. Parity$_n$: $\{0,1\}^n \to \{0,1\}$, $x \longmapsto x_1 + \cdots + x_n \bmod 2$.

($\in$P. $\in$L. $\in$SPACE(1). Size-2 DFA.)

[Håstad'86] thm: depth d circuits for Parity$_n$ require size $2^{\Omega(n^{\frac{1}{d-1}})}$.
(FSS, Ajtai, Yao)

depth 2: $2^{\Omega(n)}$ (easy) (we'll see)

depth 3: $2^{\Omega(\sqrt{n})}$

4: $2^{\Omega(n^{1/3})}$

.

$\Omega(n^{\frac{1}{d-1}})$

4: 2

⋮

rems: • Sharp: can be done in $2^{O(n^{\frac{1}{d-1}})}$. (ex.)

- Shows poly size ⇒ $\Omega\left(\frac{\log n}{\log\log n}\right)$ depth.
- Don't know any $L \in NP$ requiring depth-3 ckts of size $2^{\omega(\sqrt{n})}$.

HW2.3: Would imply not in $O(n)$-size $O(\log n)$-depth.

Rec. Lec. 9 (TISP tradeoffs for SAT)

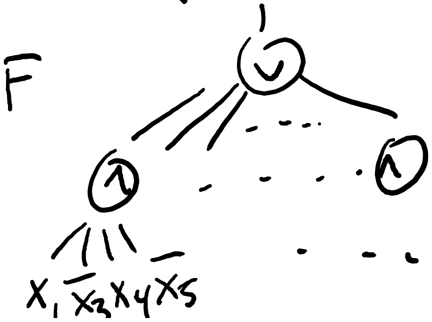
used $TISP(n, n^{o(1)}) \subseteq \Sigma_d TIME(n^{\frac{1}{d}+o(1)})$

- Håstad ⇒ can't put n^{α} for $\alpha < \frac{1}{d}$
- "Easily" ⇒ ∃ oracle L s.t. $PSPACE^L \not\subseteq PH^L$.

Sim techniques ⇒ " " " " $(\Sigma_{k+1}P)^L \not\subseteq (\Sigma_k P)^L$.

warmup: depth-2 ckt for Parity needs size 2^{n-1}.

pf: WLOG DNF



(Assume no ∧ has same vbl twice, WLOG)

Claim: every term ($\wedge$) must have width n.

Pf: If term T is missing some $\overset{(-)}{x_i}$:

- Can make $T = 1$ → output 1.
- Flip x_i → output still 1. $\Rightarrow\Leftarrow$

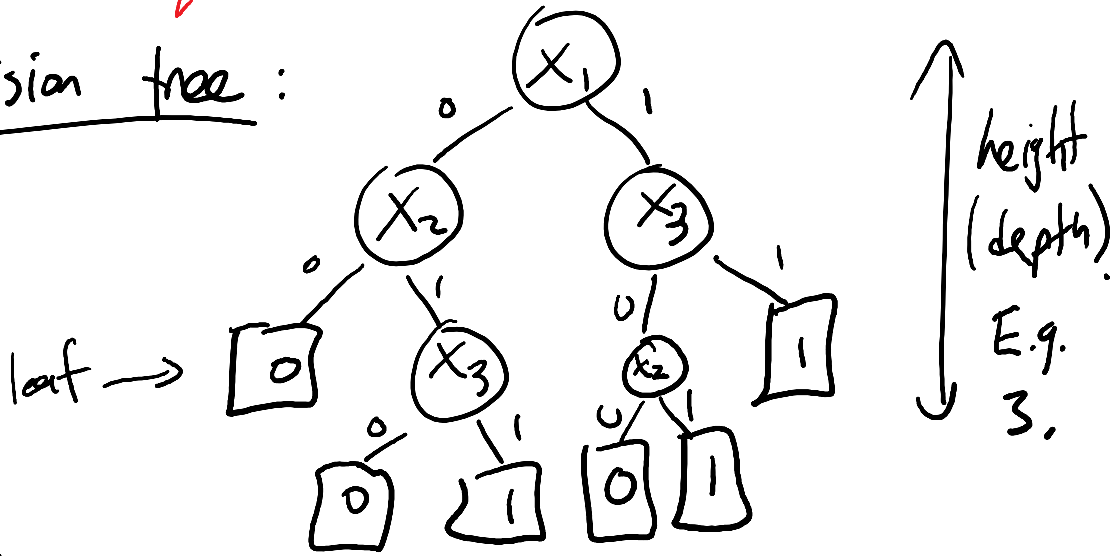
(Not poss. for Parity: flipping bit always flips output.)

∴ each term only 1 on one $x \in \{0,1\}^n$.

But Parity_n has 2^{n-1} 1-inputs. □

(Håstad's proof requires one more key object.)

def: <u>Decision tree</u>:



Computes a fcn $\{0,1\}^n \to \{0,1\}$.

(WLOG: no repeated vbl. on any path.)

def: $DT(f)$ = min height of a DT computing f.

(Least # queries you need to x's bits to for sure know f's val.)

rem: $DT(f) \le n$ always. $DT(f) = 0 \iff f$ constant

fact: $DT(f) \le h \Rightarrow f$ has a width-h DNF
& " " " CNF

pf: DNF: Take OR of ANDs of all DT paths to $\boxed{1}$-leaves.

CNF: $DT(\neg f) \le h$: use de Morgan!

* * *

Key technique : <u>Random restrictions</u>

Restriction: $\rho : \{x_1, \dots, x_n\} \to \{\underbrace{0, 1}_{\text{"fixed"}}, \underset{\uparrow\ \text{"free/unset"}}{*}\}$ [Subbotovskaya '61]

$f|_\rho$: plug in fixed vals, star vbls free.

<u>Obs</u>: $(\text{Parity}_n)|_\rho$ = Parity on $*$-vbls!

<u>Random restriction</u> with $*$-prob. ε : "$\rho \sim R_\varepsilon$"

$$\forall i \text{ indep., } \rho(x_i) = \begin{cases} * & \text{w. prob } \varepsilon \\ 0 \\ 1 & \text{w. prob. } \frac{1-\varepsilon}{2} \text{ each} \end{cases}$$

$\boxed{\text{Idea:}}$ For C a shallow ckt or DT, $C|_\rho$ is drastically simplified, but

''

$PAR|_\rho$ is still PAR on $\approx \varepsilon n$ vbls.

(Perhaps $C|_\rho$ is now "clearly" too weak to compute PAR)

Håstad's "Switching" "Lemma"

(It's hard! Should be "thm")

Let ϕ be a CNF (or DNF) of width $\le w$.

Then $\forall h \in \mathbb{N}$, $\Pr_{\rho \sim R_\varepsilon}[DT(\phi|_\rho) \geq h] \le (5\varepsilon w)^h$.

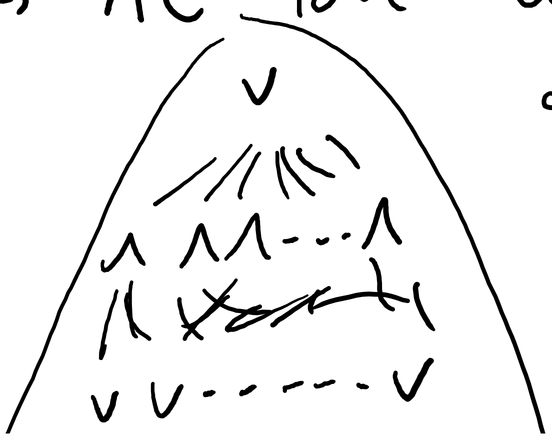
(Narrow CNFs/DNFs get crushed, to tiny DTs.)

Rem:
- No dep. on size(ϕ)!!!
- Typically: $\varepsilon = \frac{1}{10w}$: $\Rightarrow E[\#\ *\text{'s in a clause}] \le \frac{1}{10}$.

  $\Pr[DT(\phi|_\rho) \geq h] \le 2^{-h}$.
- "Switching": width w $\underline{\text{CNF}} \xrightarrow[\rho]{}$ height h DT $\Downarrow$ width h $\underline{\text{DNF}}$

Switching $\Rightarrow$ AC^0 lower bounds.

Suppose

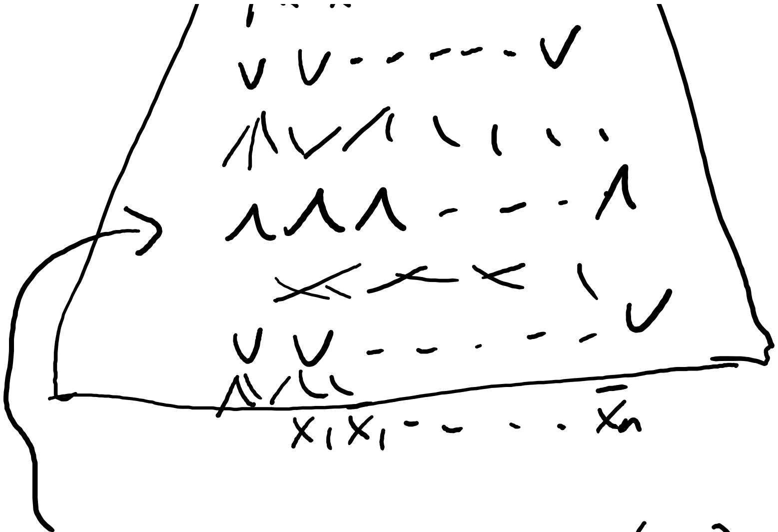


depth $d=5$

size S,

computes PAR_n.

Assume (we'll fix



Assume (we'll fix later) width = bottom fan-in is $\leq w := 20 \log S$.

At layer 2: many ($\leq S$) CNFs of width w.

Do rand restr. $\rho_1 \sim R_\varepsilon$, $\varepsilon := \frac{1}{10w}$.

For any partic CNF ϕ, $\Pr_{\rho_1}[DT(\phi|_{\rho_1}) \geq w] \leq 2^{-w} = \frac{1}{S^{20}}$

by Switching Lemma.

$\therefore$ wvhp (union bound),

<u>all</u> level-2 CNFs become height-w DTs,

hence width-w DNFs.

Collapse $\vee$'s at levels 2&3.

Now depth-4, bottom fan-in $\leq w$.

Do a further rand restr. $\rho_2 \sim R_\varepsilon$ on still-free vbls (stars(ρ_1)).

NB: $\rho_2\rho_1 \sim R_{\varepsilon^2}$ on $\{x_1, \dots, x_n\}$.

(In gen.: ε-rand restr. followed by δ-rand $\Rightarrow$ $\varepsilon\delta$-rand.)

Again: wvhp: width-w DNF $\rightarrow$ height-w DT

= width-w CNF.

Collapse $\wedge$'s: depth-3, width w.

Once more: $\rho_3 \sim R_\varepsilon$ ($\rho := \rho_3\rho_2\rho_1 \sim R_{\varepsilon^3}$)

wvhp get depth-2, width-w ckt.

Stop. (depth d $\longrightarrow$ depth 2: final $\rho \sim R_{\varepsilon^{d-2}}$)

It's computing PAR on $\approx \varepsilon^{d-2} n$ vbls (wvhp).

$\therefore$ $\underline{w \geqslant \varepsilon^{d-2} \cdot n}$ (by our depth-2 bound)

Recall: $\varepsilon = \frac{1}{10w}$. $\Rightarrow$ $(10w)^{d-1} \geqslant n$

$\Rightarrow$ $10w \geqslant n^{\frac{1}{d-1}}$

$\Rightarrow$ $\underset{\underset{20\log S}{\uparrow}}{w} \geqslant \Omega(n^{\frac{1}{d-1}})$.

$\Rightarrow$ $\log S \geqslant \Omega(n^{\frac{1}{d-1}})$. $\boxed{?}$

Fixing initial assump. that init. bottom fan-in $\leq 20 \log S$: Just start with $\rho_0 \sim R_{\frac{1}{100}}$.

(only affects Ω in above derivation.)

Bottom level has $\leq S$ $\vee$ gates (say).

If any gets a 1 → killed.

$\Pr[\text{no } 1\text{'s}] = \left(\frac{1+.01}{2}\right)^{\text{width}} \ll \frac{1}{S}$ if width $\geq 20\log S$

∴ whp, all bott-level gates of width $\geq w$ killed. □

Proof of Switching Lemma [AB Ch. 14.1].

(Probably a video....)

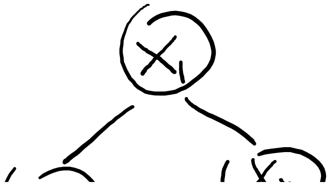
Warmup: Let ϕ be a D.T. of depth $\leq w$.

Then $\Pr_{\rho \sim R_\varepsilon}[DT(\phi|_\rho) \geq h] \leq (2\varepsilon w)^h$.

(Sim to Håstad (2, not 5), but for much simpler case of DT→DT.)

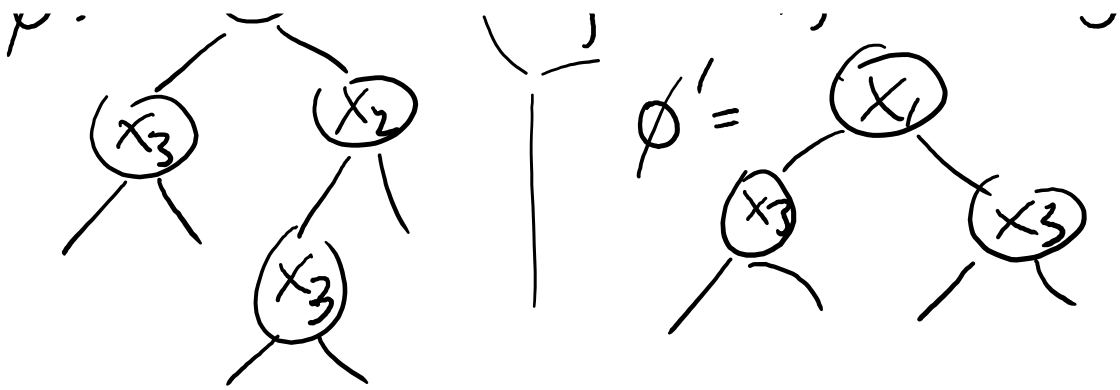
Pf: Say $\phi' = \phi|_\rho$.

ϕ:



$\rho(x_1)=*$, $\rho(x_2)=0$, $\rho(x_3)=*$

$\phi' =$



Let P be a random root-leaf path in ϕ'.

(start @ root, make all L/R choices unif.)

Then $\text{height}(\phi') \geq h \Rightarrow \Pr_{P}[\text{len}(P) \geq h] \geq 2^{-h}$.

($\exists$ at least one len-h path; prob it's chosen as P is 2^{-h}.)

$$\therefore \Pr_{\rho \sim R_\varepsilon}[DT(\phi|_\rho) \geq h]$$

$$\leq \Pr_{\rho \sim R_\varepsilon}\left[\Pr_{P \text{ in } \phi|_\rho}[\text{len}(P) \geq h] \geq 2^{-h} \right]$$

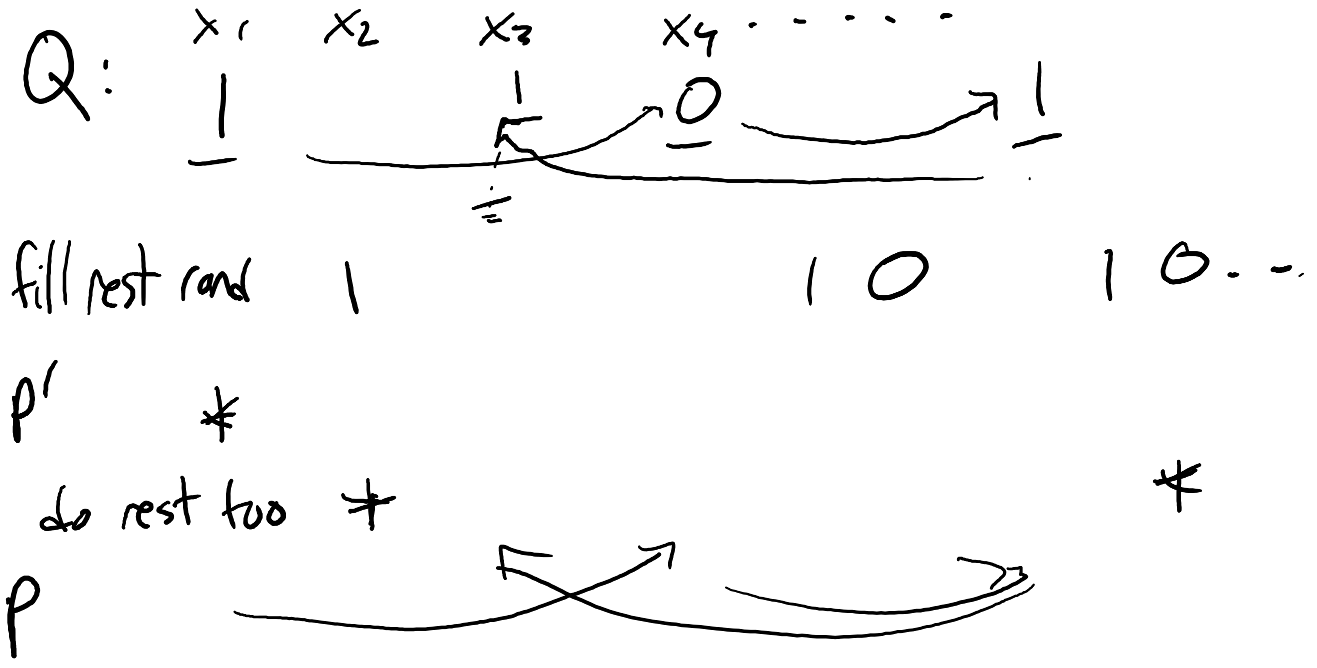
$$\text{(Markov)} \quad \leq 2^h \cdot \underbrace{\mathop{E}_{\rho \sim R_\varepsilon}\left[\Pr_{P \text{ in } \phi|_\rho}[\text{len}(P) \geq h] \right]}$$

exper.1 { ·Randomly restr. ϕ by $\rho \sim R_\varepsilon$
·P is path from root-leaf

exper.2 { ·Take Q = rand path in orig ϕ
·Let P' be an ε-rand subset of P

(Let ρ' be an ε-rand subset of ρ

Claim: ρ & ρ' have same distrib.



$$\le 2^h \Pr_{Q}\left[\mathrm{Bin}(|Q|,\varepsilon) \ge h\right]$$

$$\le 2^h \Pr\left[\mathrm{Bin}(w,\varepsilon) \ge h\right]$$

$$= 2^h \sum_{t \ge h} \binom{w}{t} \varepsilon^t \cancel{(1-\varepsilon)^{w-t}}$$

$$\le 2^h \underbrace{\left(\frac{w^h \varepsilon^h}{h!} + \frac{w^{h+1} \varepsilon^{h+1}}{(h+1)!} + \dots\right)}_{\le w^h \varepsilon^h} \quad (\text{for } h \ge 2)$$

$$= (2\varepsilon w)^h. \qquad \square$$

Lecture 19 - The Switching Lemma.

Let ϕ be a DNF (or a CNF) of width $\leq w$. Then $\forall h \in \mathbb{N}$:

$$\boxed{\Pr_{\rho \sim R_\varepsilon} \left[DT(\phi|_\rho) \geq h \right] \leq (5\varepsilon w)^h .}$$

1. Håstad's orig. proof ✓
2. Easier/weaker : [PRST '16]

$$\cancel{(5\varepsilon w)^h} \rightarrow O(\varepsilon \cdot w 2^w)^h$$

↳ Parity_n reqs depth-d circuits of size $n^{\Omega\left(\frac{\log n}{d^2}\right)}$

3. Warmup [PRST]

thm: Let ϕ be a DT of depth $\leq w$.
Then $\forall h \in \mathbb{N}$,

$$\boxed{\Pr_{\rho \sim R_\varepsilon} \left[DT(\phi|_\rho) \geq h \right] \leq \frac{(2\varepsilon w)^h}{h!} .}$$

pf: If $DT(\phi|_\rho) \geq h$, then

$$\Pr_{\substack{\text{rand. path} \\ p \text{ in } \phi|_\rho}} \left[\mathrm{len}(p) \geq h \right] \geq 2^{-h} . \quad ✓$$

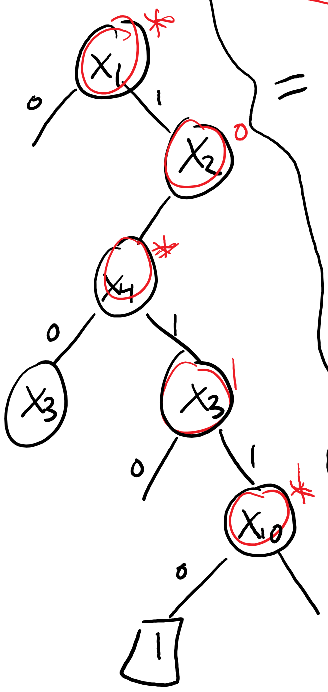
$$\therefore \Pr_{\rho \sim R_\varepsilon} \left[DT(\phi|_\rho) \geq h \right]$$

$$\leq \Pr \left[\Pr \left[\mathrm{len}(p) \geq h \right] \geq 2^{-h} \right]$$

$$\leq \Pr_{\rho \sim R_\varepsilon}\left[\Pr_{\substack{P \text{ a path} \\ \text{in } \phi|_\rho}} [\text{len}(P) \geq h] \geq 2^{-h} \right]$$

$$(\text{Markov}) \quad \leq 2^h \mathop{E}_{\rho \sim R_\varepsilon}\left[\Pr_{\substack{P \text{ path} \\ \text{in } \phi|_\rho}} [\text{len}(P) \geq h] \right]$$

$$= 2^h \cdot \Pr_{\substack{\rho \sim R_\varepsilon,\ P \text{ path} \\ \text{in } \phi|_\rho}} [\underline{\text{len}(P)} \geq h]$$



$$= 2^h \cdot \Pr_{\substack{Q \text{ rand path} \\ \text{in orig. } \phi, \\ \text{then } *\text{'s put} \\ \text{in w. prob } \varepsilon}} [\#\text{ stars in } Q \geq h]$$

$$= 2^h \cdot \Pr_{\substack{Q \text{ rand path} \\ \text{in orig } \phi}} [\text{Bin}(|Q|, \varepsilon) \geq h] \quad (*)$$

$$\leq 2^h \cdot \Pr[\text{Bin}(w, \varepsilon) \geq h]$$

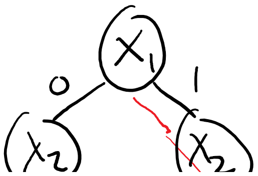
$$= 2^h \cdot \binom{w}{h} \cdot \varepsilon^h \leq 2^h \cdot \frac{w^h}{h!} \cdot \varepsilon^h$$

$$= (2\varepsilon w)^h / h! \quad \square$$

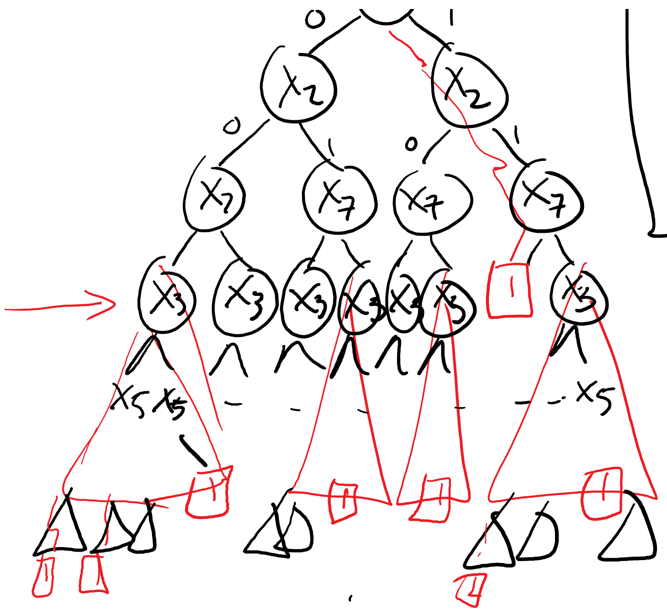
PRST Switching Lemma

ϕ: width-w DNF

e.g. $w=3$ $(x_1 \wedge x_2 \wedge \bar{x}_7) \vee (x_3 \wedge \bar{x}_2 \wedge x_5) \vee (x_4 \wedge x_{10}) \vee \dots$



This is a <u>w-clipped</u>:
at every internal node



at every internal node there is a leaf at dist. $\leq w$.

Every width $\leq w$ DNF (or CNF) has a w-clipped DT.

thm: Let ϕ be a w-clipped DT. Then $\forall h \in \mathbb{N}$, $\Pr_{\rho \sim R_\varepsilon}\left[DT(\phi|_\rho) \geq h\right] \leq \left(O(1) \cdot \varepsilon \cdot w \cdot 2^w\right)^h$.

$3 \cdot (\ 2e\ \ \ \ )$

Pf: As before, prob. is at most

$$2^h \cdot \Pr_{\substack{Q \text{ rand path} \\ \text{in } \phi}}\left[\mathrm{Bin}(|Q|, \varepsilon) \geq h\right]$$

$$\leq 2^h \cdot \underset{\uparrow}{E}\left[\binom{|Q|}{h} \cdot \varepsilon^h\right]$$

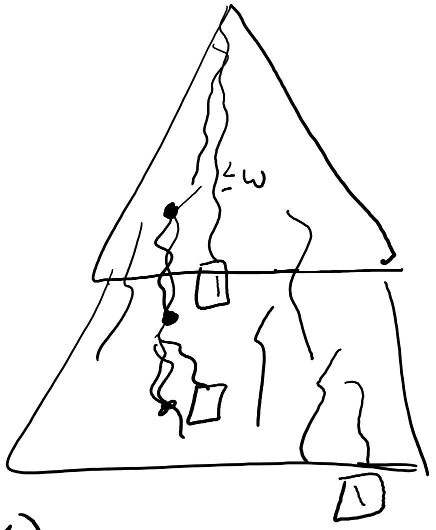
Q rand path in ϕ

$$\leq \frac{(2\varepsilon)^h}{h!} \cdot \underline{E\left[|Q|^h\right]} \quad ⊛$$

$|Q|$ dominated by

$\underline{w} \cdot G, \quad G \sim \mathrm{Geom}(2^{-w})$

$p = 2^{-w}$.



$$p = 2^{-w}$$

$$\le \boxed{\frac{(2\varepsilon)^h}{\cancel{h!}} \cdot w^h} \cdot \underbrace{E[G^h]}_{\text{careful: } \le \frac{\cancel{h!}}{p^h}}$$

$$\le (2\varepsilon w \cdot 2^w)^h.$$

$$E[G^h] \le ? \qquad G \sim \mathrm{Geom}(p).$$

	$1 \ldots \frac{h}{p}$	$\frac{h}{p} \ldots \frac{2h}{p}$	$\frac{2h}{p} \ldots \frac{3h}{p}$	$\frac{3h}{p} \ldots \frac{4h}{p}$	$\ldots$
$\Pr[G \in \text{range}]$	≤ 1	$\le (1-p)^{h/p}$ $\le (\exp(-p))^{h/p}$ $= \exp(-h)$	$\le (1-p)^{\frac{2h}{p}}$ $\le e^{-2h}$	$\le (1-p)^{3h/p}$ $\le (e^{-p})^{3h/p}$ $\le e^{-3h}$	$\ldots \le e^{-4h}$
G^h	$\le \left(\frac{h}{p}\right)^h$	$\le \left(\frac{2h}{p}\right)^h$	$\left(\frac{3h}{p}\right)^h$	$\left(\frac{4h}{p}\right)^h$	

$$\therefore E[G^h] \le \left(\frac{h}{p}\right)^h + e^{-h} \cdot \left(\frac{2h}{p}\right)^h + e^{-2h}\left(\frac{3h}{p}\right)^h + e^{-3h}\left(\frac{4h}{p}\right)^h + \ldots$$

$$= \left(\frac{h}{p}\right)^h \left[1 + \left(\frac{2}{e}\right)^h + \left(\frac{3}{e^2}\right)^h + \left(\frac{4}{e^3}\right)^h + \cdots\right]$$

$$\le 1 + \frac{2}{e} + \frac{3}{e^2} + \frac{4}{e^3} + \cdots$$

$$= \left(\frac{e}{e-1}\right)^2 \le 2.51 \le 3$$

$$\boxed{\le 3 \cdot \left(\frac{h}{p}\right)^h}.$$

$$\therefore \text{final prob. bounds} \quad (2\varepsilon w)^h \cdot 3 \cdot \left(\frac{h}{2^{-w}}\right)^h$$

$\therefore$ final prob. bounds $\dfrac{(2\varepsilon w)^h}{h!} \cdot 3 \cdot \left(\dfrac{h}{2^{-w}}\right)^h$

$h! \geq \left(\dfrac{h}{e}\right)^h$

$$= 3\left(2\varepsilon w 2^w\right)^h \cdot \frac{h^h}{h!}$$

$$\leq 3 \cdot \left(2e \cdot \varepsilon \cdot w \cdot 2^w\right)^h .$$

$\square$

Lecture 21 - Monotone Circuit Lower Bounds

def: $f:\{0,1\}^n \to \{0,1\}$ is <u>monotone</u> if

$$x \le y \Rightarrow f(x) \le f(y);$$

(coord-wise)

i.e., changing a 0→1 in input cannot cause a 1→0 chg. in output.

e.g.:
- Majority: $\{0,1\}^n \to \{0,1\}$
- And_n
- Or_n

non-e.g.: Parity_n

e.g.: $n = \binom{v}{2}$, $x \in \{0,1\}^n$ encodes edges/non-edges of v-vtx graph, $\text{Clique}_{v,k} : \{0,1\}^n \to \{0,1\}$ indics. if input graph has a k-clique.

def: <u>Monotone circuit</u>: $\wedge, \vee$ gates, no $\neg$ gates.

(today: can allow unbdd fan-in, and only charge # of <u>gates</u> in size; OK because we'll be proving lower, bds)

<u>fact</u>: Mono. ckts compute mono. fcns (obvs.)

<u>ex</u>: Every mono fcn. computable by a mono ckt. Eg., as DNF:

$$\text{Clique}_{v,3}(x) = \bigvee_{i<j<k} \underbrace{(x_{ij} \wedge x_{jk} \wedge x_{ik})}_{\text{"minterms"}}$$

$$\left(x_{ij} = \begin{cases} 1 & \text{if edge ij present} \\ 0 & \text{else} \end{cases} \right)$$

("1-witness") minimal set of vbls which, if all 1, force f=1.

Or CNF/maxterms

Q: Is there a smaller ckt if you allow ¬?

(You might say: why would ¬ help? It's a mono. fcn! But....)

A: Yes, $\text{Clique}_{v,3} \in \text{SIZE}(v^{2.38})$!

Q: A smaller <u>mono</u> ckt?

A: [Razb. 85] • No smaller than $\frac{v^3}{\text{polylog}(v)}$.

(Homework! ↑)

& • $\text{Clique}_{v,.25\ln v}$ needs mono ckts of size $v^{\Omega(\log v)}$.

↑ ↑

in NP ("explicit") super-poly!

(Is that cool? Is what it is: a superpoly mono lower bd. for an explicit mono fcn.)

[AB'87]: $\text{Clique}_{v,\tilde{\theta}(v^{2/3})}$ needs $2^{\tilde{\Omega}(v^{1/3})} = 2^{\tilde{\Omega}(n^{1/6})}$

[And'85,'87]: ∃ mono lang in NP needing mono ckt size $2^{\Omega(n^{1/3}/\log n)}$

$\downarrow$

[HR'00] $2^{\Omega(n^{1/3}/\log^{1/3} n)}$

[É.Tardos'87]: ∃ mono lang in P needing mono ckt size $2^{\tilde{\Omega}(n^{1/8})}$.

(So once again: maybe not telling us NP is hard, just mono ckts stink.)

Today: Andreev's fcn needs mono size $2^{\tilde{\Omega}(n^{1/4})}$

~~[N.Blum'17]: " " " ~~mono~~ " " !~~

* * *

"Mono Switching Lemma" [Gyárfás, Jukna, Berg-Ulfberg'86]

Let C be a k-CNF. (width ≤ k)

Then ∃ a ℓ-DNF D

Then $\exists$ an ℓ-DNF D

& an exact-$\ell+1$-DNF E, $|E| \le k^{\ell+1}$

(width of all terms exactly $\ell+1$)

(#terms)

s.t. $D \le C \le D \vee E$ (ptwise)

→ "CNF approx'ly switched to DNF"

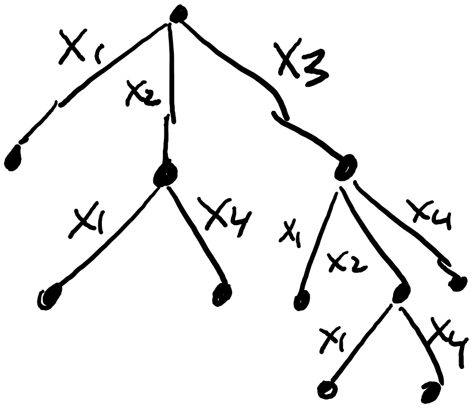
→ k, ℓ arbitrary. No dep. on "n".

And dual stmt for DNF → CNF.

Pf: E.g. $C = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_4)$,

$k = 3$.

Build a "witness tree" for C:

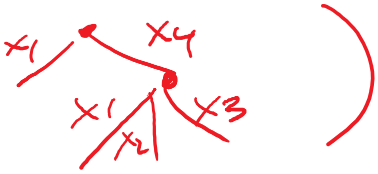


- Branch on 1st clause.
- Path = "imagine all these vbls set to 1"
- Terminate path when $C \equiv 1$
-

(Tree does dep. on order

E.g.

of clauses; it's OK. backwards order →



Claim: $\forall x \in \{0,1\}^n$,
$C(x) = 1$ iff
x consistent with some path.

How to make D: OR together all paths of length $\leq \ell$.

ℓ-DNF? ✓ $D \leq C$? ✓

rem: D may be $\emptyset$ → $D \equiv 0$.

How to make E: OR together all partial paths of len. $\ell+1$.

exact-$\ell+1$-DNF? ✓

$C \leq D \vee E$? ✓

$|E| \leq k^{\ell+1}$ ✓ (branch factor $\leq k$).

$\square$

Approximation Theorem. Fix $k, \ell \in \mathbb{N}^+$.

Let f be computable by size-S mono ckt.

Then $\exists$ k-CNF C, ℓ-DNF D,

exact-k+1-CNF E_C, exact-ℓ+1-DNF E_D

$|E_C| \leq S \cdot \ell^{k+1}$ $|E_D| \leq S \cdot k^{\ell+1}$

$|E_C| \le S \cdot \ell^{k+1}$ $|E_D| \le S \cdot k^{\ell+1}$,

$C \wedge E_C \le f \le D \vee E_D$ & $D \le C$.

("$C \lesssim f \lesssim D$") (So, sorta $C \approx f \approx D$, & f smushed to "narrow CNF, DNF")

Pf: Structural induction on f's circuit's gates.

For each gate g, will create "approx'ing" k-CNF C_g, ℓ-DNF D_g, with $D_g \le C_g$.

Base case: $g = x_i$. → $D_g = C_g = x_i$,

∧ case:

. Given $D_{h_i} \le C_{h_i}$, $i = 1 \dots t$:

① Let $C_g = C_{h_1} \wedge \cdots \wedge C_{h_t}$, a k-CNF since C_{h_i}'s are.

② Let D_g = SwitchingLemmaD(C_g).
∴ D_g an ℓ-DNF, $D_g \le C_h$.

Perhaps some "errors" introduced:

$x \in \{0,1\}^n \xrightarrow{} $ s.t. $D_{h_i}(x) = C_{h_i}(x) = h_i(x)$ $\forall i$

but $D_g(x) \neq C_g(x)$ $\Downarrow$ $C_g(x) = g(x)$

(error) (no error)

But: any such "error" x makes "E_g" $\leftarrow$ exact-$\ell+1$-DNF, $|E_g| \le k^{\ell+1}$ true, $\therefore$

$$\underset{0}{\underset{\shortparallel}{D_g(x)}} \le \underset{1}{\underset{\shortparallel}{C_g(x)}} \le (D_g \vee E_g)(x).$$

$\vee$ case: dual.

At end have $D_f \le C_f$.

For some inputs $x \in \{0,1\}^n$, no error introd at any gate; then $C_f(x) = D_f(x) = f(x)$.

For remaining x's "charge" error to first gate where approx. errs

If that's some $\wedge$ gate g, we know $E_g(x) = 1$.

" " " $\vee$ " " " " $E_g(x) = 0$.

$$\therefore \underbrace{C_f \wedge \bigwedge_g E_g}_{} \le f \le D_f \vee \underbrace{\bigvee_g E_g}_{}$$

$\ell+1$ DNF

$\underbrace{\qquad\qquad}_{\text{sim.}}$ $\underbrace{\qquad\qquad}_{\text{exact-}\ell+1\text{-DNF,}}$

size $\leq S \cdot k^{\ell+1}$.

$\square$

Typical use:

Case 1: C nonempty.

∴ contains clause (OR) on vbl-set R, $|R| \leq k$.

Now $D \leq C \leq OR_R$.

∴ $f \leq OR_R \vee E_D$

(say in words?)

∴ every $x \in f^{-1}(1)$ either hits R (but $|R| \leq k$) or one of E_D's terms

each has width $\ell+1$, can't cover too many x's, so $|E_D| \leq S \cdot k^{\ell+1}$ must be big.

Case 2: C empty $\Rightarrow$ $C \equiv 1$ (for CNFs)

∴ $E_C \leq f$.

(say in words?)

∴ for every $x \in f^{-1}(0)$, the 0-coords of x must hit every clause in E_C ← narrow, so size must be in $E_C \leftarrow$ narrow, so size must be large.

E.g.: Andreev's Fcn. f.

Let q be prime, $\boxed{d \sim \frac{1}{10}\sqrt{\frac{q}{\log q}}}$.

Inputs are $A \in \{0,1\}^{q \times q}$. $\boxed{(n = q^2)}$.

Minterms: one for each poly $P: \mathbb{Z}_q \to \mathbb{Z}_q$ of deg $< d$.

Minterm $A^{(P)}$ has $A^{(P)}_{i,P(i)} = 1 \ \forall i \in \mathbb{Z}_q$, 0 else.

(one 1 per row).

Fact: $(f_n) \in NP$ (guess the poly P)

- \# polys P is q^d, $\therefore$ exists mono ckts of size $\leq q^d$ computing f

thm: Mono circuit size $q^{\Omega(d)}$ required.

$$2^{\Omega(\underbrace{d \log q}_{\sqrt{q \log q} \approx n^{1/4}\sqrt{\log n}})}$$

Pf: Take $k \sim d \log q$, $\ell = d-1$.

Case 1: $\exists R \subseteq [q]\times[q]$, $|R| \le k \sim d\log q$,

& DNF E_D of exact-width-d, size $\le S\cdot k^d$

s.t. $\forall$ minterms $A^{(p)}$ makes true OR_R or E_D.

q^d of these

how many can this make true?

\# deg $\le d-1$ polys taking one fixed val is q^{d-1}. $\therefore$ OR_R covers $\le k\cdot q^{d-1}$

each term of width d covers ≤ 1 $A^{(p)}$

$\therefore$ E_d covers $\le S\cdot k^d$ minterms.

$$\therefore \quad q^d - \underset{\substack{d\log q \\ \approx \sqrt{q}}}{k}\, q^{d-1} \le S\cdot k^d$$

$$\therefore \quad \Omega(q^d) \le S\cdot k^d$$

$$\Rightarrow S \geq \Omega\left(\left(\frac{q}{k}\right)^d\right)$$

$$= q^{\Omega(d)}. \quad \checkmark$$

Case 2: $f \geq E_C \leftarrow$ an exact-$k+1$-CNF of

$$\text{size} \leq S \cdot \ell^{k+1}$$

Pick $A \in \{0,1\}^{q \times q}$ at <u>random</u>, each entry 1 w. prob $1 - \frac{2d \ln q}{q}$,

$$\Pr[f(A)=1] \underset{\substack{\text{union}\\\text{bound}}}{\leq} \underset{\substack{\uparrow\\\text{all } A^{(p)}}}{q^d} \left(1 - \frac{2d \ln q}{q}\right)^q \leq q^d \exp(-2d \ln q)$$

$$\leq q^{-d}.$$

$$\Pr[E_c(A)=0] \underset{\text{u.B.}}{\leq} \underbrace{\left(S \cdot \ell^{k+1}\right)\left(\frac{2d \ln q}{q}\right)^{k+1}}$$

$\therefore$ must be $\geq \frac{1}{2}$, $\Rightarrow S \geq \Omega\left(\left(\ell \cdot \frac{2d \ln q}{q}\right)^{k+1}\right)$

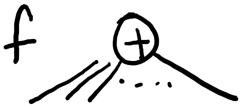
$$\left(\ell = d \cdot \frac{1}{10} \sqrt{\frac{q}{\log q}}\right)$$

$$\geq \Omega(1) \cdot \left(\frac{1}{10}\right)^{\overset{k+1}{\nwarrow \, d \log q.}}$$

$$\geq q^{\Omega(d)}. \quad \square$$

Lecture 22 - Razborov-Smolensky lower bounds for $AC^0[p]$

Lec. 18, [Håstad]: Depth-d ckts for Parity_n req. size $\geqslant 2^{\Omega(n^{\frac{1}{d-1}})}$

Q1: What if  ($\text{Parity}/\text{Mod}_2$) gates allowed?

(Obvs., can now compute Parity_n in size 1!)

[Razb'87]: Maj_n reqs. depth-d "$\wedge, \vee, \neg, \text{Mod}_2$" ckts of size $2^{\Omega(n^{\frac{1}{2d}})}$

Q2: What about computing Parity_n (Mod_2) using Mod_3 gates (and $\wedge, \vee, \neg$)?

def: $\text{Mod}_m : \{0,1\}^w \to \{0,1\} : x \mapsto \begin{cases} 0 & \text{if } \sum x_i = 0 \bmod m \\ 1 & \text{if } \sum x_i \neq 0 \text{ " "} \end{cases}$

[Smol'87], <u>today</u>: For depth d, reqs. size $2^{\Omega(n^{1/2d})}$

$\Longrightarrow$ New (easier) proof of Håstad. (Slightly weaker quant'ly; cf $\frac{1}{d-1}$)

def: $\underline{AC^0[m]}$: Const-depth, poly-size ckts with $\wedge, \vee, \neg$, Mod_m gates.

[Smol'87]: $\forall$ primes $p \neq q$, n-bit Mod_p fcn not in $AC^0[q]$.

(All Razb-Smol. results have sim. proofs; as mentioned, we'll do $\mathrm{Mod}_2 \notin AC^0[3]$.)

(Why restriction to primes? Actually, prime powers OK, but what one needs is a field of that size. For $\mathrm{Mod}_6 \to$ got nothing)

Rem: No known $L \in NP$, $L \notin AC^0[6] = AC^0[2,3]$, tho $(\mathrm{Maj}_n)_n$ conjectured so.

(So maybe SAT computable eff'ly in <u>constant</u> parallel time using $\wedge, \vee, \neg, \mathrm{Mod}_6$ gates....)

Idea: depth-d, size-S Mod_3-ckt

$\rightsquigarrow$ approx'ly computable by $O(\log S)^d$-deg. $\mathbb{F}_3$-polynomial, (And it's implausible that a poly of deg. $\ll n$ can well-approx. parity.)

def: A polynom $p : \mathbb{F}_q^n \to \mathbb{F}_q$ is <u>proper</u> if output in $\{0,1\}$ whenever inputs are.

def: A <u>randomized poly</u> $p(x,r)$: we think of $x \in \{0,1\}^n$ as the "real" input, $r \in \{0,1\}^\ell$ as "random bits".

thm 1: Let $f:\{0,1\}^n \to \{0,1\}$ be computable by depth-d, size-S $\wedge, \vee, \neg, \text{mod}_3$ circuit. Let $t > 1$. Then $\exists$ proper rand. poly $p(x,r)$ over $\mathbb{F}_{\boxed{3}}$, $\deg \le (\boxed{2}t)^d$, and $\forall x \in \{0,1\}^n$,

$$\Pr_{r \in \{0,1\}^n}[p(x,r) \neq f(x)] \le S \cdot \boxed{2}^{-t}.$$

More generally…

q

$q-1$

(can get q here if allow $r \in \mathbb{F}_q$)

$$\Rightarrow \Pr_{x,r}[\neq] \le S \cdot 2^{-t}$$

$$\Rightarrow \mathop{E}_{r} \Pr_{x}[\neq) \le S 2^{-t}$$

$$\Rightarrow \exists r \text{ s.t. } \Pr_{x}[p_r(x) \neq f(x)] \le S \cdot 2^{-t}.$$

cor: $t = O(\log S)$

$\exists$ proper poly $p: \mathbb{F}_3^n \to \mathbb{F}_3$ of deg $\le O(\log S)^d$ s.t. $\Pr_{x \sim \{0,1\}^n}[p(x) = f(x)] \geqslant .9999$.

thm 2: $\nexists$ poly $p: \mathbb{F}_3^n \to \mathbb{F}_3$ of deg $\le \sqrt{n}$ s.t.

$$\Pr_{x}[p(x) = \widehat{\text{PARITY}}(x)] \geqslant .999.$$

<u>Conclusion</u>: If a depth-d, size-S mod_3-

ckt computes Mod_2, then

$$O(\log s)^d \geq \sqrt{n}$$

$$\Rightarrow \quad s \geq 2^{\Omega(n^{\frac{1}{2d}})} \; . \; ✓$$

(OK, now must prove Thms 1, 2. Thm 1 is most interesting to us. Thm 2 is a counting thing.)

Thm 1 proof: Gate-by-gate polynom. replacement.

Input gate: $x_i \longrightarrow x_i$. Proper ✓
Deg 1 .
0 error.

¬ gate:

$\longrightarrow 1-p(x,r)$.
Proper, deg & error unchanged.

Mod₃ gate

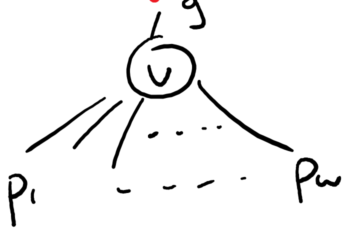
: $(p_1 + \cdots + p_w)^{2 \leftarrow q-1 \text{ if } \text{Mod}_q}$

Proper ✓ $\begin{cases} a^{q-1}=1, & a\neq 0 \\ a^{q-1}=0, & a=0 \end{cases}$

(FLT: where we use q prime!) $\longrightarrow$

(We'll analyze deg, error later.)

OR:



$1-(1-p_1)\cdots(1-p_w)$?
Terrible degree.

Instead: Let $r_1, \dots, r_w$ be random $\{0,1\}$ bits. Let $g = (p_1 r_1 + \dots + p_w r_w)^2$

- Proper ✓ · If all p_i's $0 \rightarrow 0$ ✓
  - If at least one 1, say WLOG $p_1 \dots$

    Choose $r_2 \dots r_w$ first.

$$\underbrace{p_1 r_1 + \underbrace{p_2 r_2 + \dots + p_w r_w}_{\theta}}$$

$$= \begin{cases} \theta + 1 & \text{w.p. } 1/2 \\ \theta & \text{w.p. } 1/2 \end{cases} = 0 \text{ w.prob} \le \tfrac{1}{2}.$$

$\therefore \Pr_r[g \text{ wrong}] \le \frac{1}{2}.$

(Reminiscent of Valiant-Vazirani)

- Actually, choose t indep. such r's, multiply together all g's.

AND: Use OR construction, & de Morgan.

Analysis: Proper? ✓

Deg: Goes up by $\le 2t$ at each gate $\Rightarrow \le (2t)^d$ at end. ✓

Error: For fixed x, only AND/OR gates can intro. error.

Prob. they do so $\le 2^{-t}$.

$\therefore$ overall error $\leq S \cdot 2^{-t}$. $\square$

Rem: # rand bits used $\leq S \cdot S \cdot t$ $\leftarrow O(\log S)$.

Can reduce. (Don't care, for these results, but care for some other apps...)

ex: For 1 gate, instead of $r_1, \ldots, r_w$ being unif., if 1's chosen using 2-wise universal hashing (as in V.V. Thm), can spend $O(\log w) \leq O(\log S)$ rand bits, get success w. prob. $\geq \Omega\left(\frac{1}{\log S}\right)$

$\Rightarrow O(t \log^2 S)$ rand bits for failure prob. $\leq 2^{-t}$.

$O(\log^3 S)$ - - - - - - . $\ll \frac{1}{S}$.

Now reuse these bits at every OR/AND gate! Union bound still OK.

$\longrightarrow$ only $O(\log^3 S)$ rand bits needed for final rand poly p.

Thm 2: $\nexists$ poly $p : \mathbb{F}_3^n \to \mathbb{F}_3$ of deg $\leq \sqrt{n}$ s.t. $\Pr_x[p(x) = \text{PARITY}(x)] \geq .999$.

Pf by contra:

Suppose $p(x) = \sum x_i \bmod 2$ for $\geq .999 \cdot 2^n$
$\deg(p) \leq \sqrt{n}$ of $x \in \{0,1\}^n$

Suppose $p(x) = \sum x_i \bmod 2$ for $\geq .999 \cdot 2^n$
& $\deg(p) \leq \sqrt{n}$. of $x \in \{0,1\}^n$.

Switch to ±1 notation:

$$\text{Let } \tilde{p}(x) = p(\underbrace{x_1-1}_{\substack{-1 \mapsto 1 \\ +1 \mapsto 0}}, \dots, x_n - 1) \underset{\substack{\downarrow \\ 0 \mapsto 1 \\ 1 \mapsto -1}}{+1}$$

So $\deg(\tilde{p}) \leq \sqrt{n}$, & now

$\tilde{p}(x) = \prod_{i=1}^{n} x_i$ for all $x \in G$,
where $G \subseteq \{\pm 1\}^n \subseteq \mathbb{F}_3^n$
has $|G| \geq .999 \cdot 2^n$.

Consider $\mathcal{F} = \{ f : G \to \mathbb{F}_3 \}$, so $|\mathcal{F}| = 3^{|G|}$.

Every $f : G \to \mathbb{F}_3$ representable as <u>some</u> n-variate poly.
(cf. HW8, #1)
(Interpolate over all $|G|$ inputs.)

Can assume poly never has $x_i^2, x_i^3, x_i^4, \dots$
repl. w/ $1 \quad x_i \quad 1 \quad x_i \dots$
$\because$ only care abt. $G \subseteq \{\pm 1\}^n$.

So f rep'ble by <u>multilin</u> poly,

$$f(x) = \sum_{S \subseteq [n]} c_S \cdot \underbrace{\prod_{i \in S} x_i}.$$

$$\text{On } \{\pm 1\}^n \rightarrow = \prod_{i \in \bar{S}} x_i \cdot \prod_{i=1}^{n} x_i$$

$$\text{On } G \rightarrow = \underbrace{\prod_{i \in \bar{S}} x_i \cdot \tilde{p}(x)}_{\deg \leq |\bar{S}| + \sqrt{n},}$$

Make this switch if $|\bar{S}| \leq \frac{n}{2} \Leftrightarrow |S| \geq \frac{n}{2}$.

$\therefore$ f rep'ble as a deg $\leq \frac{n}{2} + \sqrt{n}$ poly on G,

$$f(x) = \sum_{|S| \leq \frac{n}{2} + \sqrt{n}} d_n \cdot \prod_{i \in S} x_i.$$

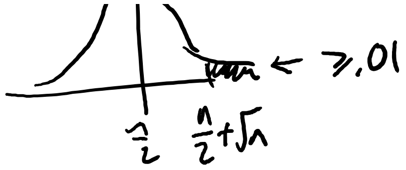
$$\#\text{ of such polys } \leq 3^{\#\{S \subseteq [n] : |S| \leq \frac{n}{2} + \sqrt{n}\}}$$

But $3^{|G|}$ diff. $f : G \to \mathbb{F}_3$.

$$\therefore \ |G| \leq \#\{S \subseteq [n] : |S| \leq \tfrac{n}{2} + \sqrt{n}\}$$
$$= \sum_{i=0}^{\frac{n}{2}+\sqrt{n}} \binom{n}{i}$$

Claim: $\overset{>}{<}$ $.99 \cdot 2^n$. $\rightarrow$ Contra to $|G| \geq .999 \cdot 2^n$.

$\Longleftrightarrow$ $\Pr[\, < \frac{n}{2} + \sqrt{n}$ H's in n coin flips$] \leq .99$.



ex. #H has mean $\frac{n}{2}$, stddev $\frac{\sqrt{n}}{2}$,

so it's $\Pr[H < \text{mean} + 2\,\text{stddevs}]$

$\approx \Pr[\text{Gaussian} < 2]$

$\leq \Pr(-2 < \text{Gaussian} < 2]$

$\approx .95.$

Lecture 23: Toda's 2nd Theorem, & Uniform ACC lower bounds

Recall Toda 1: $PH \subseteq BPP^{\oplus P}$ $(\subseteq BPP^{\#P})$

Toda 2: $BPP^{\oplus P} \subseteq P^{\#P[1]}$ $(\subseteq P^{\#P} = P^{PP})$ ("derand")

$\hookrightarrow = BPP^{\oplus P[1]}$ (using $\oplus P^{\oplus P} = \oplus P$ idea, HW6.2)

(can also assume oracle answer = final answer)

$\therefore \forall L \in BPP^{\oplus P}$ $\exists$ rand. poly-time reduction:

$$x \xrightarrow[\uparrow]{M(x,r)} \varphi_r$$

$p := poly(|x|)$ rand bits

s.t. $x \in L \Rightarrow \Pr_r[\#\varphi_r \text{ odd}] \geq \frac{3}{4}$ $\circledast$

$\quad\;\; \notin \qquad\qquad\qquad\quad$ even

Simple tricks: Given φ_1, φ_2,

$\psi = \varphi_1 \wedge \varphi_2$ (on disj. vbls)

has $\#\psi = \#\varphi_1 \cdot \#\varphi_2$

ex: a simple ψ with $\#\psi = \#\psi_1 + \#\psi_2$

Toda Trick: $\exists$ det. poly$(|\varphi|, k)$-time reduction

$$\varphi \longrightarrow \Psi \quad \text{s.t.}$$

$$\#\varphi = 0 \pmod 2 \Rightarrow \#\Psi = 0 \pmod{2^k}$$

$$= -1 \quad " \quad " \qquad\qquad = -1 \quad " \quad "$$

Apply Toda Trick to $\textcircled{\dagger}$, $k = p+1$.

$x \in L \Rightarrow \geq \frac{3}{4} \cdot 2^p$ of the φ_r have $\#\varphi_r = -1 \bmod 2$

$\longrightarrow \#\Psi_r = -1 \bmod 2 \cdot 2^p$, rest 0.

$$\Rightarrow \sum_r \#\Psi_r \in [-2^p \ldots -\tfrac{3}{4} \cdot 2^p) \bmod 2 \cdot 2^p$$

$x \notin L \Rightarrow \geq \frac{3}{4} \cdot 2^p$ of Ψ_r have $\#\Psi_r = 0$, rest $-1 \bmod 2 \cdot 2^p$

$$\Rightarrow \sum_r \#\Psi_r \in [-\tfrac{1}{4} 2^p \ldots 0] \bmod 2 \cdot 2^p$$

$\therefore$ can tell difference given $\sum_{r \in \{0,1\}^p} \#\Psi_r \; (\ddagger)$ (over $\mathbb{Z}$)

There's a nondet poly time machine N s.t.

$\#N(x) = (\ddagger)$; i.e., $x \mapsto (\dagger)$ in $\#P$.

$\therefore \quad PH \subseteq BPP^{\oplus P} \subseteq P^{\#P[1]}$. $\blacksquare$

* * *

Toda trick proof:

Using $+, \cdot$ tricks $O(1)$ times can convert $\varphi \to \varphi'$ s.t.

$$\#\varphi = a \Rightarrow \#\varphi' = \underline{3a^4 + 4a^3} =: q(a)$$

Claim: (i) $a \equiv 0 \mod m \Rightarrow q(a) \equiv 0 \mod m^2$

(ii) $\quad \equiv -1 \qquad\qquad\qquad\qquad -1$

Proof: (i) $m | a \Rightarrow m^2 | a^2 | q(a)$.

(ii) $\underline{\text{ex}}$: $3(cm-1)^4 + 4(cm-1)^3 \mod m^2$

$= -12cm + 3 + 12cm - 4 = -1$.

Repeat transformation $\log k$ times:

$\mod 2 \rightsquigarrow \mod 2^{2^{\log k}} = 2^k$

size (φ) $\nearrow_{\times O(1)}$ $\log k$ times $\Rightarrow$ $\nearrow_{\times \mathrm{poly}(k)}$. ■

* * *

(Generalized/improved by...)

Beigel-Tarui'94: For $k \in \mathbb{N}^+$, $\exists$ a deg-$2k$ **Mod-Amplifying-Poly** Q_k over $\mathbb{Z}$ s.t.

$$\forall m, \quad \begin{array}{l} a \equiv 0 \pmod m \\ \equiv 1 \end{array} \Rightarrow \begin{array}{l} Q_k(a) \equiv 0 \pmod{m^k} \\ \equiv 1 \end{array}$$

(Better deg: $2k$ instead of k^2; $2k$ is tight. More natch 0/1 vs. 0/-1. Downside is coeffs may be neg. Can get around that in Toda's Thm.)

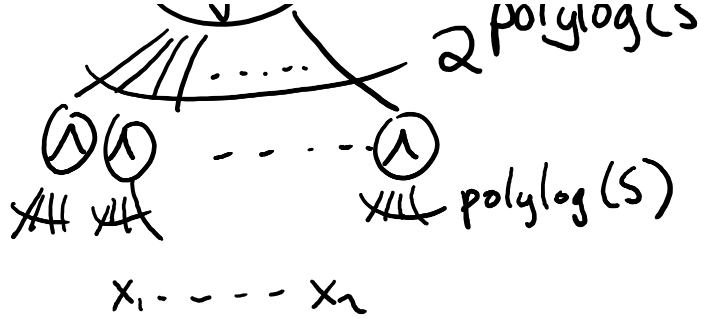
their app.:

BT Theorem: "ACC $\subseteq$ SYM+" ← (some circuit class)

Fix m (e.g., 6). Let C be an $O(1)$-depth, size-S ckt with $\vee, \wedge, \neg, \mathrm{Mod}_m$ gates. Then $\exists$ equiv. ckt D of this form:

(leave)

leave up



"Sym" is some symmetric fcn; $\mathrm{sym}(y_1,\ldots,y_w) = h(\sum_i y_i)$ for some $h : \{0,1,\ldots,w\} \to \{0,1\}$.

Equiv, $D(x) = h(p(x))$, where p is a poly of deg. polylog(s), coeffs in $\{0,1,\ldots,2^{\mathrm{polylog}\, s}\}$.

(It is by virtue of this theorem that we can prove some weak lower bds against ACC.)

[AG'94]: $C \to D$ transformation is efficient, "uniform"

If (C_n) is "DC-uniform" ACC-family,
(D_n) " " SYM^+ - " .

Pf: [AB "web addendum", Wil'11 App. A.]

Ingred #1: Getting polylogs fan-in OR/AND.

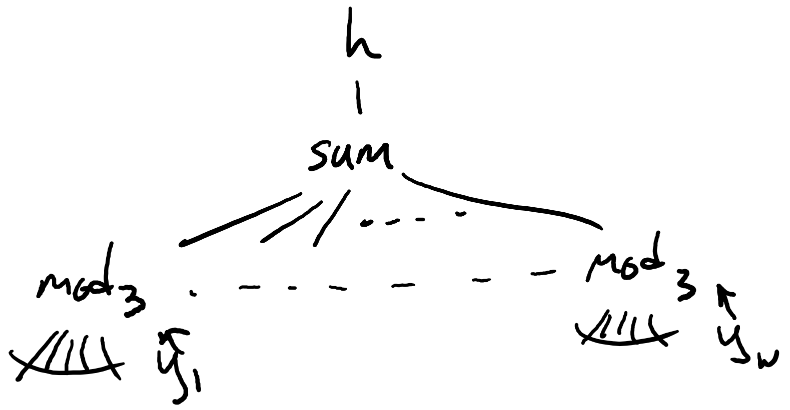
$$OR_s \to OR_{\mathrm{polylog}(s)} \circ Mod_2$$

+ polylog(s) rand bits

"Prob. poly" trick of Razb. - Smol.

Rand bits elim'd by enumeration, putting Maj on top (a symm. gate)

Ingred #2: Getting rid of mod_2, mod_3 gates



$\mathrm{mod}_3 : \mathbb{Z} \to \{0,1,2\}$

↓

not the "mod 3 gate"

$$\sum_{i=1}^{w} (y_i \bmod 3) = \sum_{i=1}^{w} \left(\underset{\uparrow}{Q}(y_i) \bmod 3^{\overset{k}{}} \right)$$

$> w$

mod-amplif-poly

$$= \Big(\sum_{i=1}^{w} Q(y_i) \Big) \underbrace{\bmod 3^k}_{\text{foldable into } h}$$

(some poly → sum & AND gates...)

"□"

//

def: Let (unif-) ACC = languages computable by def: Let **ACC** = **languages** computable by **DC-uniform** const-depth, poly-size, $\wedge, \vee, \neg, \text{Mod}_m$ ckt families.

$\hookleftarrow$ (const, eg. 6)

fact: $\overset{(unif)}{AC^0} \subseteq \overset{(unif)}{ACC} \subseteq L \subseteq P \subseteq \cdots \subseteq PSPACE$

$\neq$, by Håstad (Parity)

triv.: $ACC \neq PSPACE$, $\because L \neq PSPACE$

Conseq. of BT Theorem [AG94].

Thm: $ACC \subseteq TIME(n^2)^{\#3SAT[1]}$

$\uparrow$ even if $2^{n^{o(1)}}$ size allowed.

Rem: Like a "scaling down" of Toda 2:

$$PH \subseteq P^{\#3SAT[1]}$$

PH $\swarrow$ DC-uniform AC^0 of size $2^{poly(n)}$

P $\longrightarrow$ **fixed** p time

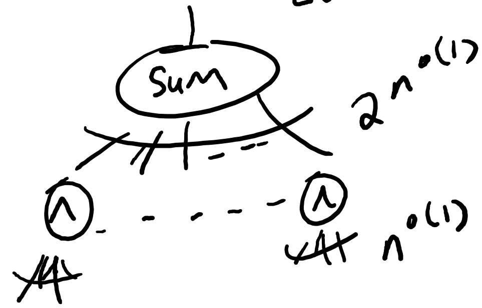
+ mod gates!

Proof

Suppose $L \in ACC$ is decided by unif. ACC ckt fam. $(C_n)_n$ of size $2^{n^{o(1)}}$.

By BT Theorem, also by (D_n):

D_n $\qquad h : [2^{n^{o(1)}}) \to \{0,1\}$



Uniformity: ckt structure, & h, computable in $n^{o(1)}$ time.

There's an NTIME(n) machine N_L s.t.

$\#N_L(x)$ = output of sum gate

→ guess ∧, acc. if it's 1.

Cook-Levin to an equiv #3SAT formula φ of size $O(n^2)$.

Get $\#\varphi$ from oracle ($n^{o(1)}$ bits). Compute h on it.

▨

cor: $ACC \subseteq TIME(n^4)^{PP}$ Pf: binary search.

cor: $\overset{\text{(unif-?)}}{ACC} \neq PP$ (With slightly more work: $PP \not\subseteq ACC$ (quasipoly size).)

Pf: Otherwise, $ACC = P = PP$

So $TIME(n^5)^{PP} = TIME(n^5)^{P} = P$.

But $P = ACC \subseteq TIME(n^4)^{PP}$.

Contradicts THT (relativized w/ PP)!

$\square$

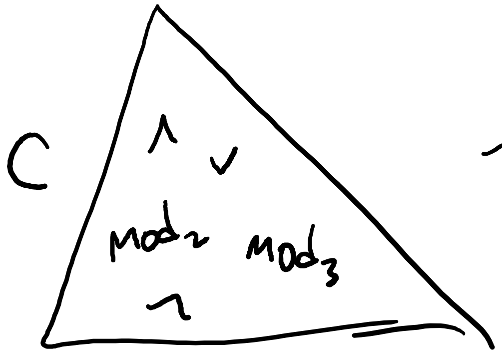
With a little more work: (Hmwk...)

thm: $PERM \notin ACC$

↳ even $2^{n^{o(1)}}$ size

Rem: Rare examples of using uniformity in ckt lower bound. Don't know how to drop it. Pre-[Wil'11], couldn't rule out PSPACE, EXP, NEXP $\subseteq$ non-unif. ACC.

(Rec: $L \notin$ non-unif ACC believed: Maj!)

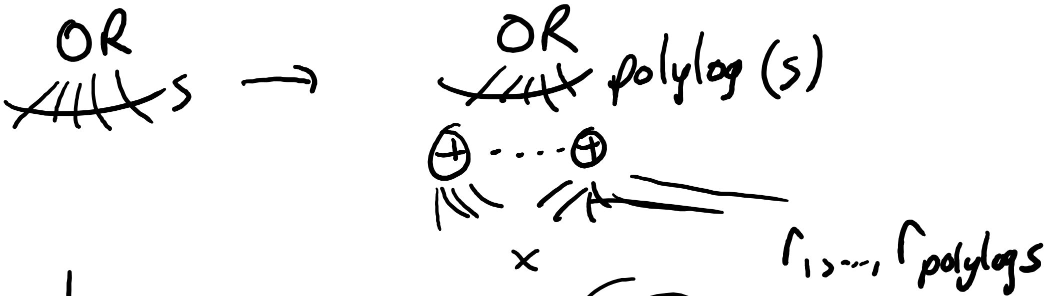


① ⇝ leveled tree

(all gates same at same level;
$d = O(1) \rightarrow d' = O(1)$
$S \rightarrow S' = poly(S)$)

② ⇝ no AND gates (de Morgan)

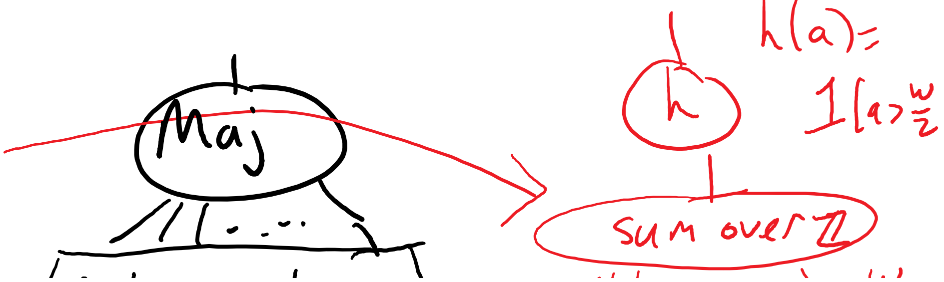
③ ⇝ OR gates repl'd by "probabilistic mod-2 trick" as in Razb-Smol:

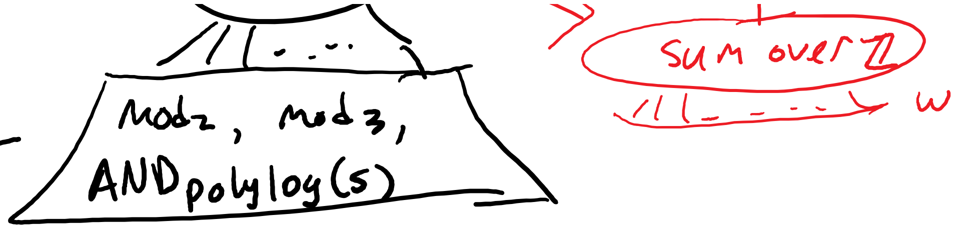


④ ⇝ Enumerate over r, put $\boxed{Maj}$ gate at top.
(Maj is symmetric. $2^{polylog s}$ blowup. Ckt is exactly computing C now.)

⑤ ⇝ · OR → AND (de Morgan)
· $\neg \rightarrow mod_2$ $(\neg y = mod_2(y, 1))$

Now:





⑥ → a circuit over $\mathbb{Z}$ with +, • (of fan-in polylog(s)), (mod 2) & (mod 3) [~~NOT~~ NOT the 0/1 ver., the "normal" ver.]

'AND → •

· "mod 2"$(y_1,\ldots,y_w) \to (y_1+\cdots+y_w)^2$ (mod 2) ← •

& mod3 (FLT).

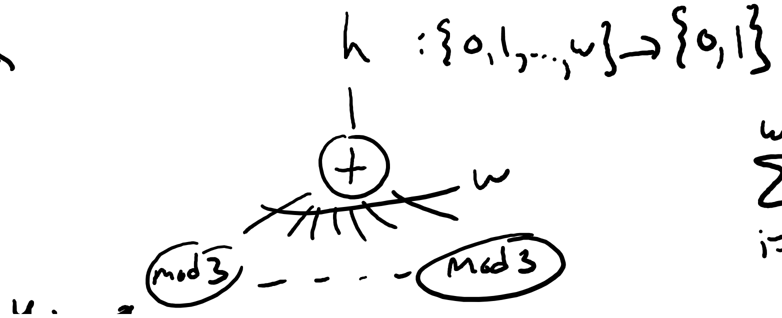
⑦ → All • gates go to the bottom.

$(a+b)\cdot(c+d) \to a\cdot c + a\cdot d + b\cdot c + b\cdot d$.

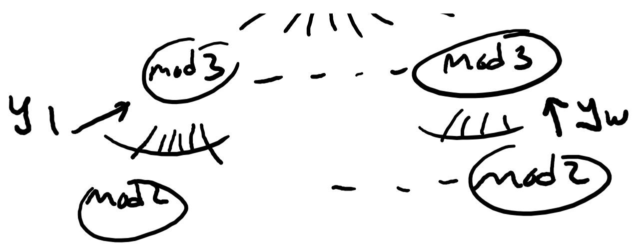
$(a \bmod p)\cdot(b \bmod p) \to (a\cdot b)^{\bmod p}$.

& + gates to top.

⑧



$$\sum_{i=1}^{w} (y_i \bmod 3)$$



$i=1$

$$= \sum_{i=1}^{w} \left(Q(y_i) \bmod 3^{l} \right),$$

↗ mod-amplif-poly ↑ $> w$

$$= \left(\sum_{i=1}^{w} Q(y_i) \right) \bmod 3^{l}$$

$\oplus$ ↗ changes h ↑

Got rid of one layer of mods,
introduced some $+, \cdot$ → push to top/bott,
repeat.

"$\square$".

Lecture 24 - Hardness vs. Randomness I

(Goal: prove - well, find evidence - that BPP = P.)

Thm [IW97]: If SAT requires ckts of size $2^{\Omega(n)}$ then BPP = P.

(basically SETH against ckts) (pretty believable)

(for all suff large n)

More generally:

(i) $\exists A \in E$ $= \mathrm{TIME}(2^{O(n)})$ s.t. A_n needs $2^{\Omega(n)}$-size ckts $\forall n \geq n_0$

$\Rightarrow$ BPP = P

(ii) $\sim\sim\sim$ $2^{n^{\varepsilon}}$ $\sim\sim\sim$

$\Rightarrow$ BPP $\in$ QuasiP

(iii) $\sim\sim\sim$ $n^{\omega(1)}$ $\sim$ $\Rightarrow$ BPP $\in$ SUBEXP (TIME($2^{n^{\varepsilon}}$) $\forall \varepsilon$)

Rem: $A \in$ EXP okay for (ii), (iii). (By padding.)

((ii) is very reasonable, and what we'll eventually show.)

(These results are actually combo of 2 totally diff results...)

[IW97]: (i) $\Rightarrow$ | $\underline{(i^*)}$: $\exists A \in E$ s.t. $\mathrm{corr}(A_n, C_n) \leq 2^{-\delta n}$

[IW97]: (i) ⇒ (i'): $\exists A \in E$ s.t. $\underbrace{\mathrm{corr}(A_n, C_n)}_{\Pr_x[A_n(x)=C_n(x)] - \Pr[\neq]} \le 2$

$\forall C_n$ of size $\le 2^{\delta n}$

(leave up)

(Cₙ is only barely better than guessing.)

"Worst-case → avg-case reduction"

(this is what we won't quite prove)

[NW94]: (i*) ⇒ BPP = P. (And slightly less hardness ⇒ BPP ⊆ QuasiP, etc.)

(today)

(What does hardness have to do with derand? PRGs.)

Let $B \in BPP$. Goal: $B \in P$.

Rem: By padding, may assume $x \overset{?}{\in} B$ decided in $O(n)$ time using n coin flips: $M(x, r)$

DTIME(n)

$|r| = |x|$.

Idea: replace r with "pseudorandom" bits.

((Could they be totally deterministic, like the digits of π? No, b/c perhaps M(x,r) acc. iff x=r. Then M "overall acc." no inputs, but it would with det'ic r. Need r "a little" rand. IRL, "true" rand bits hard to get, but often start w/ a "seed" and apply some crazy fcn to it.)

<u>Want</u>: "PRG" $G : \underbrace{\{0,1\}^\ell}_{\text{seed}} \to \{0,1\}^n$ s.t. for all $x \in \{0,1\}^n$,

$$\circledast \quad \Pr_{r \in \{0,1\}^n}[M(x,r) \text{ acc.}] \overset{\pm .1}{\approx} \Pr_{s \in \{0,1\}^\ell}[M(x,G(s)) \text{ acc.}]$$

<u>Plan</u>: • <u>Deterministically</u> try all s to <u>compute</u> RHS.
(for BPP it's OK, since LHS is $\geq \frac{2}{3}$ or $\leq \frac{1}{3}$)

• Time: $2^\ell \cdot$ (time to compute G)

• Win: if $\ell = O(\log n)$, $\underbrace{G \text{ comp'ble in } 2^{O(\ell)} \text{ time}}$

(this is unusual, helpful; only need PRG comp'le in exponential time)

Partial win (e.g. BPP $\subseteq$ QuasiP) if, e.g.
$\ell = \text{polylog}(n)$, G in $2^{\text{poly}(\ell)}$ time.

If $\circledast$ fails for ∞'ly many n, x then "M_x"(r) is a "distinguisher" beating G. [<u>ckt</u> of size $O(n^2)$

(it can "tell" diff. between rand & p-rand bits)

def: A <u>PRG</u> of seed len. $\ell = \ell(n)$: $(G_\ell)_\ell$

$G_\ell : \{0,1\}^\ell \to \{0,1\}^n$ (unif'ly) computable in time $2^{O(\ell)}$ s.t. for almost all n, all ckts $C : \{0,1\}^n \to \{0,1\}$ of size n^3 $(\geq O(n^2)$ a.a.n$)$ (¬ gates free)

$$\Pr_{s \in \{0,1\}^\ell}[C(G(s))] \overset{\pm .1}{\approx} \Pr_{r \in \{0,1\}^n}[C(r)].$$

[NW94]: Exists w/ $\ell = O(\log n)$ if (i*).

Not^n: circuits/fcns now $\{\pm 1\}^n \to \{\pm 1\}$.

def: <u>corr</u> $(f,g) = \mathbb{E}_{r \sim \{\pm 1\}^n}[\underbrace{f(r)g(r)}] \in [-1,+1]$

↑ counts +1 if same
−1 if diff

$+1$ if $f = g$, -1 if $f = \neg g = -g$;

$\text{corr} = \varepsilon \iff \Pr[f = g] = \frac{1}{2} + \frac{1}{2}\varepsilon$

Assump (i*) now: $\exists\, A \in E$ s.t. for almost all n,

$A_n : \{\pm1\}^n \to \{\pm1\}$ very hard on avg.:

$\forall\, C : \{\pm1\}^n \to \{\pm1\}$ of ckt-size $\le 2^{\delta n}$,

$\text{corr}(A_n, C) \le 2^{-\delta n}$. $\left(\delta > 0 \text{ universal}\right)$

<u>Warmup</u> for PRG w/ $\overset{\not\sim}{\ell} = O(\log n) \longrightarrow n$ "stretch":

$\ell \rightsquigarrow \ell + 1$.

Let $G(s) = (\underset{\uparrow\ \ell \text{ bits}}{s}, \underset{\uparrow\ 1 \text{ bit.}}{A_\ell(s)})$

(not .1 b/c of ±1 notation)

Claim: G indeed ".1-fools" $(\ell+1)^3$-size ckts C:

(‡) $$\mathop{E}_{|s|=\ell}\left[C(s, A_\ell(s))\right] \overset{\pm\textcircled{.2}}{\approx} \mathop{E}_{|r|=\ell+1}\left[C(r)\right]$$

(Intuition: s totally rand. Just need that $A_\ell(s)$ "looks" unif. rand. to wimpy old C, even condit. on s. Rem: $A_\ell(s)$ better be close to 50/50. Well, it is, o/w A_ℓ wouldn't be hard!)

Proof: (Has to be by contrapos.)

Suppose (‡) fails due to distinguishers

C^*_ℓ (for ∞ly many ℓ).

WLOG (repl. w/ $-C^*_\ell$):

$$\underset{|s|=\ell}{E}\left[C^*(s, A_\ell(s))\right] - \underset{|r|=\ell+1}{E}\left[C^*(r)\right] > .2$$

$$\underset{\substack{|s|=\ell \\ |b|=1}}{E}\left[C^*(s,b)\right]$$

(½ the time $b = A_\ell(s)$
½ " " $= -A_\ell(s)$)

$$= \frac{1}{2}\underset{s}{E}\left[C^*(s, A_\ell(s))\right] + \frac{1}{2}\underset{s}{E}\left[C^*(s, -A_\ell(s))\right]$$

$$\therefore \frac{1}{2}\left(\underset{|s|=\ell}{E}\left[C^*(s, A_\ell(s))\right] - \underset{|s|=\ell}{E}\left[C^*(s, -A_\ell(s))\right]\right) > .2$$ ☺

(On avg. input s, C^* more likely to output 1 when you put correct ans. in 2nd slot than wrong ans.)

Form circuit $\tilde{C}$, taking ℓ-bit input s:

$\tilde{C}(s)$: • "guess" $b \in \{\pm 1\}$ (formally: b is rand input bit)

• output $C^*(s, b)\, b$

(Should be good at computing A_ℓ.)

$$\underset{s,b}{E}\left[\tilde{C}(s,b)\, A_\ell(s)\right] = \underset{s,b}{E}\left[C^*(s,b)\, b\, A_\ell(s)\right]$$

s,b $\qquad$ s,b

$$= \underset{s}{E}\Big[\tfrac{1}{2} C^*(s, \overset{\swarrow}{A_\ell(s)}) \cancelto{1}{\cdot A_\ell(s) \cdot A_\ell(s)}$$
$$+ \tfrac{1}{2} C^*(s, -A_\ell(s)) \cdot (-A_\ell(s)) \cdot A_\ell(s) \Big]$$

(going on whether $b = \pm A_\ell(s)$)

$> .2$ by ☺

$\therefore\ \exists\, b_0 \in \{\pm 1\}$ (Prob. Meth.) s.t.

$$\underset{s}{E}\left[\tilde{C}_{b_0}(s)\, A_\ell(s) \right] > .2$$

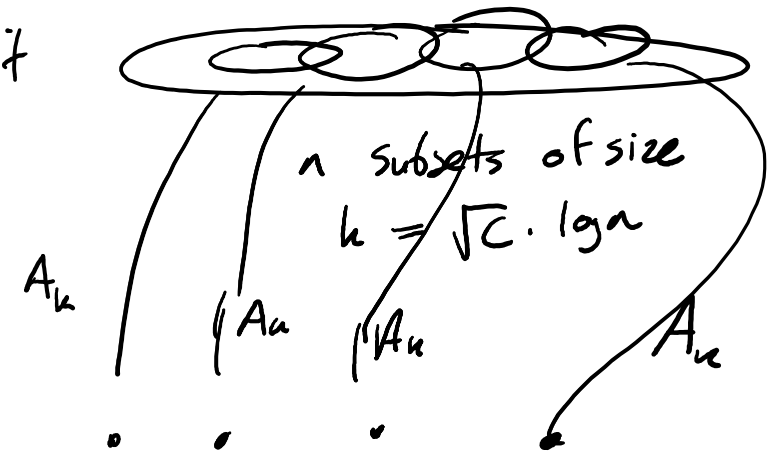
But size$(\tilde{C}_{b_0}) \le$ size$(C^*) \le (\ell+1)^3$.

Way-contradicts A_ℓ's supposed hardness (corr $< 2^{-\delta\ell}$ for size $2^{\delta\ell}$).

* * *

[NW94] PRG:

$\ell = c \cdot \lg n$ -bit seed



n output bits:

Ingredient 1: Subsets that "don't overlap much".

Ingredient 2: Yao's "Next-bit-prediction" thm.

(Ing. 1 is some elementl combinatorics thing.)

* * *

Ing 1: An (ℓ, k, d)-design is a family of n subsets of $\{1, \dots, \ell\}$, each of size k, s.t. any two subsets have intersection size $\le d$.

Thm: If $\ell \geq 10k^2/d$, can get $n \geq 2^{d/10}$, and constructible in time $2^{O(\ell)}$.

↳ Piazza/video. → greedy works.

Params: $\ell = c \log n$, $d = 10 \log n$.

$$\Rightarrow k = \sqrt{c} \cdot \log n.$$

Designs proof:

Just keep adding in allowable k-sets!

Certainly doable in $2^{O(\ell)}$ time. (Brute force checking.)

Claim: while $n < 2^{d/10}$, $\exists$ an addable set.

Pf: Prob. method: Pick a rand. k-set S.

Consider fixed S_0 in design so far.

Claim: $\Pr_S[|S \cap S_0| > d] \le 2^{-d/10}$

↳ if true, can union-bd. to complete proof.

Pf: Let I_j be 0/1 indic. that jth elt. of S falls in S_0.

$$E[I_j] = \underbrace{k/\ell} \le \frac{1}{k} \cdot \frac{d}{10}.$$

$$\therefore E[|S \cap S_0|] = E\left[\sum_{j=1}^{k} I_j\right] = \frac{d}{10}.$$

If I_j's indep., could use Chernoff bound to get claim (even with error 2^{-d}, hence $n \geq 2^d$).

Not indep., but better: negatively associated. (Chernoff still holds.) $\square$

(Other elementary fixes possible.)

(Other elementary fixes possible.)

Lecture 25 – Hardness vs. Randomness II

Recap: (ACTUALLY: DO YAO, THEN RECAP.)

def: A PRG of seed len. $\ell = \ell(n)$: $(G_\ell)_\ell$

$G_\ell : \{0,1\}^\ell \to \{0,1\}^n$ (unif'ly) computable in time $2^{O(\ell)}$ s.t. for almost all n, all ckts $C : \{0,1\}^n \to \{0,1\}$ of size n^3 ($\neg$ gates free)

$$\Pr_{s \in \{0,1\}^\ell}\left[C(G(s))\right] \overset{\pm .1}{\approx} \Pr_{r \in \{0,1\}^n}\left[C(r)\right].$$

def: An (ℓ, k, d)-design is a family of subsets $K_1, \dots, K_n$ of $\{1, \dots, \ell\}$, each $|K_i| = k$, s.t. any two subsets have intersection size $\leq d$.

thm: Given n, c can in poly(n) time construct a design with

$$\left.\begin{aligned} \ell &= c \log n \\ k &= \sqrt{c} \log n \\ d &= 10 \log n. \end{aligned}\right\} \circledast$$

$d = 10 \log n$.

Yao's Next Bit-Pred.

thm: Let $U_1, \dots, U_n$ be indep. & unif. rand ±1 bits, let $Z_1, \dots, Z_n$ be any random bits, not nec. unif. or indep.

(Think of them as output of PRG.)

Suppose ∃ "distinguisher" ckt C s.t.

$$\circledast \quad \left| E[C(U_1, \dots, U_n)] - E[C(Z_1, \dots, Z_n)] \right| > \varepsilon.$$

Then $\exists\, i \in [n]$ & $(i-1)$-input "predictor" ckt C', $\text{size}(C') \le \text{size}(C)$, s.t.

$$E[C'(Z_1, \dots, Z_{i-1}) \cdot Z_i] > \frac{\varepsilon}{n}.$$

[Given prev bits, C' decent at predicting next one.]

Pf: "Hybrid method":

Define String $H^{(i)} = (Z_1, \dots, Z_i, U_{i+1}, \dots, U_n)$.

So $H^{(0)} = U$, $H^{(n)} = Z$.

$\circledast$, triangle ineq., telescoping $\Rightarrow \exists\, i \in [n]$ s.t.

$$\left| E[C(H^{(i-1)})] - E[C(H^{(i)})] \right| > \frac{\varepsilon}{n}.$$

$$\mathop{E}_{z,u}\Big[C(z_1,\dots,z_{i-1},u_i,u_{i+1},\dots,u_n) - C(z_1,\dots,z_{i-1},z_i,u_{i+1},\dots,u_n)$$

$\therefore$ $\exists$ fixed $u_{i+1},\dots,u_n \in \{\pm 1\}$ s.t.

$$\mathop{E}_{z,u_i}\big[C'(z_1,\dots,z_{i-1},u_i)\big] - E\big[C'(z_1,\dots,z_{i-1},z_i)\big] > \varepsilon/n$$

where $C'(\dots) = C(\dots,u_{i+1},\dots,u_n)$.

$\downarrow$

Given $z_1,\dots,z_{i-1}$, good at telling if next bit is z_i or is unif. rand.

<u>Very</u> sim. to $\ell \to \ell+1$ PRG scenario.

<u>Ex</u>: $\exists b_0 \in \{\pm 1\}$ s.t. $C''(z_1,\dots,z_{i-1},b_0)\cdot b_0$ is the required next-bit-predictor. $\square$

Nisan-Wigderson Thm:

Suppose $\exists A \in E$ s.t. for almost all n, all ckts C

of size $\leq 2^{\delta m}$, $\operatorname{corr}(A_m, C) \leq 2^{-\delta m}$ $(\delta > 0)$.

$$\underset{x \sim \{\pm 1\}^m}{E}\left[A_m(x)\, C(x)\right]$$

Then $\exists$ PRG with $\ell(n) = O(\log n)$,

hence BPP $=$ P.

Pf: Use params $\circledast$, c chosen later.

Define $G : \{\pm 1\}^\ell \to \{\pm 1\}^n$, $(s[I] = \text{LTR order})$

$$G(s) = \left(A_k(s[K_1]), \ldots, A_k(s[K_n])\right).$$

Suppose <u>not</u> a PRG. Then for ∞ly many ℓ, $\exists$ distinguishing ckts C^*_n of size $\leq n^3$ &

$$\left| \underset{|s|=\ell}{E}\left[C^*(G(s))\right] - \underset{|r|=n}{E}\left[C^*(r)\right] \right| > .2.$$

By Yao, $\exists i \in [n]$, $(i-1)$-input predictor C' (size $\leq n^3$) s.t.

$$\underset{|s|=\ell}{E}\left[C'(A_k(s[K_1]), \ldots, A_k(s[K_{i-1}])) \cdot A_k(s[K_i])\right] > .2$$

$|s| = \ell$

$> .2/n.$

WLOG, $K_i = \{1, 2, \ldots, k\}$.

$$s \quad \underbrace{s_1 \,|\, s_2 \,|\, \cdots \,|\, s_k}_{s_{first}} \,|\, \underbrace{s_{k+1} \,|\, \cdots\cdots \,|\, s_\ell}_{s_{rest}}$$

so:

$$\mathop{E}_{s_{rest}} \mathop{E}_{s_{first}} \left[C'(A_k(s[K_1]), \ldots, A_k(s[K_{i-1}])) \cdot A_k(s_{first}) \right] > .2/n.$$

$\therefore \exists$ s_{rest} setting s.t. inner $E[\cdot] > .2/n$.

Once s_{rest} fixed, $s[K_j]$ just has $K_j \cap \{1, \ldots, k\}$ free bits. & by design ppty, card. $\leq d = 10 \log n$.

$\therefore \exists$ $2^d = n^{10}$-size ckt $\hat{C}_j$ computing

$$C'(A_k(s[K_j])), \quad j < i.$$

Combining those ($\leq n$ of them) with C', get $\bar{C}$ of size $\leq n^3 + n^{11}$ s.t.

$$\mathop{E}_{s_{first} \in \{\pm 1\}^k} \left[\bar{C}(s_{first}) A_k(s_{first}) \right] > .2/n.$$

$s_{\text{first}} \in \{\pm 1\}^k$

Contradiction provided $2^{\delta k} \geq n^{11} + n^3$,

$2^{-\delta k} \leq .2/n.$

i.e., $k \gtrsim \frac{11}{\delta} \log n$.

$\uparrow \sqrt{c}$, and don't mind $c = O(1/\delta^2)$. $\square$

* * *

Hardness Amplification

Want: $L \in E$ s.t. $\text{corr}(L_n, \text{SIZE}(2^{\delta n})) \leq 2^{-\delta n}$.

(Going to stop worrying about i.o. stuff.)

Assume: $\exists L \in E$ s.t. $\text{corr}(L_n, \text{SIZE}(S)) < 1$,

for, e.g., $S = 2^{\delta n}$, $S = 2^{n^\delta}$, $S = n^{\omega(1)}$

HW8#3 (BFNW'93): Assumption $\Rightarrow$

$\exists L' \in E$ s.t. $\text{corr}(L'_n, \text{SIZE}(\text{poly}(S)))$

$< 1 - \frac{1}{}$

$< 1 - \frac{1}{O(n)}$.

(Multilinear extension, polynom. interp thing.)

(Now what?)

(OK if $n\log n$, n^2...)

Yao's XOR Lemma:

Suppose $f : \{\pm 1\}^n \to \{\pm 1\}$ has $\mathrm{corr}(f, \mathrm{SIZE}(S)) < 1-\delta$.

Then $f^{\oplus k} : \{\pm 1\}^{nk} \to \{\pm 1\}$ has

$$\mathrm{corr}(f^{\oplus k}, \mathrm{SIZE}(S')) < (1-\tfrac{\delta}{2})^k + \varepsilon$$

for $S' = \Omega(1) \cdot S \cdot \frac{\varepsilon^2}{\log(1/\delta)}$,

where $f^{\oplus k}(x^{(1)}, \dots, x^{(k)}) = \underbrace{f(x^{(1)}) \cdot \ldots \cdot f(x^{(k)})}_{(\text{XOR, in } \pm 1 \text{ not}^n)}$

Rem: $S' \le S$, but close-ish.

"Intuition": To compute $f(x^{(1)}) \cdots f(x^{(k)})$ better than coin-flip, need to get all k answers right → prob. $(1-\delta)^k$.

Sample params: $\delta = \frac{1}{O(n)}$, $S = 2^{.001n}$.

$\varepsilon = 2^{-.00001n}$, $k = O(n^2)$.

Now $S' \approx 2^{.001n}$. $(1-\frac{\delta}{2})^k + \varepsilon \approx 2^{-.00001n}$.


?.

<u>Input length</u> $N := nk = n^3$.

So $\text{corr}(f^{\oplus k}, \text{SIZE}(2^{\Omega(N^{1/3})})) < 2^{-\Omega(N^{1/3})}$

$\nearrow$

If $(f_n)_n \in E$, so is $(f^{\oplus n^2})_n$.

↳ Leads to $BPP \subseteq QuasiP$.

May as well just have started w/

$L \in EXP \setminus SIZE(2^{n^\delta})$.

(IW97) "Derandomize" Yao: same hardness, but new input len. is $N = O(n)$.

Lecture 26 - Hardness Amplification

Yao's XOR Lemma:

Suppose $f : \{\pm 1\}^n \to \{\pm 1\}$ has $\mathrm{corr}(f, \mathrm{SIZE}(S)) < 1-\delta$

(leave up)

Then $f^{\oplus k} : \{\pm 1\}^{nk} \to \{\pm 1\}$ has

$$\mathrm{corr}(f^{\oplus k}, \mathrm{SIZE}(S')) < \left(1-\frac{\delta}{2}\right)^k + \varepsilon$$

for $S' = \Omega(1) \cdot S \cdot \mathrm{poly}(\varepsilon, \frac{1}{\log 1/\delta})$,

where $f^{\oplus k}(x^{(1)}, \dots, x^{(k)}) = \underbrace{f(x^{(1)}) \cdot \ldots \cdot f(x^{(k)})}_{(\text{XOR, in } \pm 1 \text{ not } 0/1)}$.

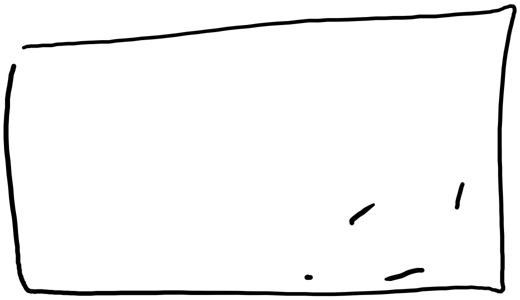
"Intuition" (?): To compute $f(x^{(1)}) \cdots f(x^{(k)})$ better than coin-flip, need to get all k answers right → prob. $(1-\delta)^k$.

Hmm: That naive alg for $f^{\oplus k}$ needs size $\simeq k \cdot S$. Hardness is for $S' < S$!

(Weird if you take $k=1$. But... not known how to get $S' \geq S$, not known how to get $> \approx 1$, perhaps not even poss....)

better intuition: "Why" is $\text{corr}(f, \text{SIZE}(s)) < 1 - \delta$?

Scenario 1



all inputs "equally hard"
(whatever that means)

Scenario 2

$|H| \approx \delta \cdot 2^n$
↳ "hard core"

For $x \in H$, computing $f(x)$ hopelessly hard for $\text{SIZE}(s)$:
"$\text{corr}_H(f, \text{SIZE}(s)) \approx 0$"

But, $f(x)$ trivial to compute when $x \notin H$,

AND, recognizing $x \overset{?}{\in} H$ is simple.

Turns out:

- f's where Scen. 2 holds are the "easiest-to-compute" f's

- [Imp'95) Scen. 2 "roughly" always holds

Q: In Scen 2, how hard is $f^{\oplus k}$?

A: When $x^{(1)}, \dots, x^{(k)}$ all $\notin H$ $\left(\text{Prob } (1-\delta)^k\right)$

$\rightarrow$ trivial. Else near impossible.

$$\longrightarrow \quad \text{corr}\left(f^{\oplus k}, \text{SIZE}(S)\right) \approx (1-\delta)^k.$$

* * *

Impagliazzo's Hard Core Set Thm:

Sps $\text{corr}(f, \text{SIZE}(S)) < 1-\delta$. (→ maybe $1-4\delta$)

Then $\exists$ $H \subseteq \{\pm 1\}^n$, $|H| \geq \frac{\delta}{2} \cdot 2^n$ (leave up)

s.t. $\text{corr}_H(f, \text{SIZE}(S')) < \varepsilon$,

$$S' := \text{poly}\left(\varepsilon, \frac{1}{\log(1/\delta)}\right) \cdot S,$$

(provided $S' \ll \text{poly}(\varepsilon, \delta) 2^{n/2}$)

Pf later.

Proof of Yao: Given f, suppose C' of size S' has $\text{corr}(f^{\oplus k}, C') > \left(1-\frac{\delta}{2}\right)^k + \varepsilon$. (*)

Want to get contradiction to Hard Core

Set Theorem.

Let H be the hard core for f, $|H| \geq \frac{\delta}{2} \cdot 2^n$.

Let $R_\ell = \{ (x^{(1)}, \dots, x^{(k)}) :$ exactly ℓ of $x^{(i)}$'s are in $H \}$.

Prob[rand input in R_0] $\leq (1-\frac{\delta}{2})^k$.

(Even if the "condit corr." of C' with f in this case is 1, still just $(1-\frac{\delta}{2})^k$.)

By $\circledast$, must exist $1 \leq \ell \leq k$ s.t.

$$\mathrm{corr}_{R_\ell}(f^{\oplus k}, C') \geq \varepsilon.$$

I.e., for this experiment...

- Let $w^{(1)}, \dots, w^{(\ell)} \sim H$ be rand, indep.
  $w^{(\ell+1)}, \dots, w^{(k)} \sim \overline{H}$ " " "
- Let $\pi \sim S_k$ be rand perm.
- Let $x^{(1)}, \dots, x^{(k)}$ be reordering of w's using π;
  $x^{(i)} = w^{(\pi(i))}$.

// now $x^{(1)}, \dots, x^{(k)} \sim R_\ell$

· Output $C'(x^{(1)},\dots,x^{(k)})\, f(x^{(1)})\cdots f(x^{(k)})$

… $E[\text{output}] \geqslant \varepsilon$.

$\therefore$ $\exists$ fixed $\underline{w}^{(2)},\dots,\underline{w}^{(\ell)} \in H$, $\underline{w}^{(\ell+1)},\dots,\underline{w}^{(k)} \in \overline{H}$, $\underline{\Pi} \in S_k$

s.t. $E[\text{output}] \geqslant \varepsilon$. WLOG $\underline{\Pi} = \text{id}$. So…

$$\underset{w^{(1)}\sim H}{E}\Big[\underbrace{C'(w^{(1)},\underline{w}^{(2)},\dots,\underline{w}^{(k)})\, f(\underline{w}^{(2)})\cdots f(\underline{w}^{(k)})\cdot f(w^{(1)})}_{\text{A ckt } C'' \text{ of size } \leq \text{size}(C') = s'}\Big] > \varepsilon.$$

with $\text{corr}_H(C'', f) > \varepsilon$, $\Rightarrow\!\Leftarrow$ to H.C. Set Thm. ∎

* * *

Pf of H.C. Set Thm:

Rem: it's basically "boosting", from M.L. [KlivServ99]

Idea: (*) Sps $\forall H$, $|H| \geqslant \frac{\delta}{2}\cdot 2^n$, $\exists$ size-s' ckt C' with $\text{corr}_H(C', f) \geqslant \varepsilon$. ("weak hypoth")

Repeatedly change H, get new C',

stitch together to get C with $\mathrm{corr}(C,f) \geq 1-\delta$.

Proof: Say $\mathcal{D}$ is a $\underline{\delta\text{-distribution}}$ on $\{\pm 1\}^n$ if it's a mixture of distribs. of the form "unif. dist. over H for some $|H| \geq \delta \cdot 2^n$".

Equiv: $\mathcal{D}(x) \leq \frac{1}{\delta} \cdot 2^{-n}$ $\forall x$.

Boring claim: Assuming (‡), we even have that $\forall \delta$-distribs $\mathcal{D}$, $\exists$ size-S' ckt C' with $\mathrm{corr}_{\mathcal{D}}(C',f) \geq \varepsilon/2$.

Pf sketch: Sps not, so $\exists$ "hard" $\mathcal{D}$. Form H by putting each $x \in H$ with prob. $\underbrace{\delta \cdot 2^n \cdot \mathcal{D}(x)}_{\leq 1}$.

$E[|H|] = \delta \cdot 2^n$, good chance $\geq \frac{\delta}{2} \cdot 2^n$

Given C', $\underset{H}{E}\left[\mathrm{corr}_H(C',f)\right] = \cdots = \underbrace{\mathrm{corr}_{\mathcal{D}}(C',f)}_{< \varepsilon/2}$.

So good chance (over H) it's $< \varepsilon$. "$\blacksquare$".

Now: Given Boring Claim, want to show $\exists$ size $S = S' \cdot O(\frac{1}{\varepsilon^2} \log(1/\delta))$ ckt C

s.t. $\text{corr}(f, C) > 1-\delta$.

Consider the following Zero-Sum Game (!)

Cindy: Picks circuit C' of size $\leq S'$.

Kory: Picks set H of size $\geq \delta \cdot 2^n$.

Payoff: $+\text{corr}_H(C', f)$ to Cindy
$-$ Kory

This game has a value v: by "Minmax Theorem": equally achieved (in expec.) by

- Cindy states a distrib. on ckts she'll use, Kory then picks an H.
- Kory states a distrib. on H's he'll use, Cindy then picks a C'.

a δ-dist. $\mathcal{D}$.

$$\downarrow \quad v = \max_{C'} \{ \text{corr}_{\mathcal{D}}(C', f) \}.$$

$\geq \frac{\varepsilon}{2}$ by Boring Claim.

I.e., Cindy can get $\geq \frac{\varepsilon}{2}$.

$\therefore$ $\exists$ distribution $\mathcal{C}$ on ckts of size $\leq S'$

s.t. $\forall H,\ |H| \geq \delta \cdot 2^n,$

$$\underset{C' \sim \mathcal{C}}{E}\left[\mathrm{corr}_H(C', f)\right] \geq \frac{\varepsilon}{2}. \quad ⊛$$

Define $\theta(x) = \underset{C' \sim \mathcal{C}}{E}\left[C(x) f(x)\right]$.

Claim: $\theta(x) \geq \frac{\varepsilon}{2}$ except for $\leq \delta \cdot 2^n$ "bad" x's,

Else: Let H be the x's where it's $< \frac{\varepsilon}{2}$; ⊛ contradicted.

Now: Say x is "good", meaning $\theta(x) \geq \frac{\varepsilon}{2}$.

If we make $t = O\left(\frac{1}{\varepsilon^2} \log \frac{1}{\delta}\right)$ draws from $\mathcal{C}$: $C_1, \ldots, C_t$, then except w/ prob. $< \delta$, $\underset{j=1..t}{\mathrm{avg}}\{C_j(x) f(x)\} \geq \frac{\varepsilon}{4}$ (Chernoff), hence

$$M(x) := \underset{C_1(x)\ \cdots\cdots\ C_t(x)}{\mathrm{Maj}} = f(x).$$

$\therefore$ for good x, $\Pr_M\left[M(x) = f(x)\right] \geq 1 - \delta$

$$\Rightarrow \mathop{E}_{M}\left[\text{frac of good } x \text{ s.t. } M(x)=f(x)\right] \geq 1-\delta$$

$\Rightarrow \exists$ outcome for M s.t. M correct for $\geq 1-\delta$ frac. of $\underbrace{\text{good } x}_{\geq 1-\delta \text{ frac of all } x.}$

Size(that M) $\approx O(t \cdot S') = S$,

& correct on $\geq 1-2\delta$ frac. of x's

$\Rightarrow \mathrm{corr}(M,f) \geq 1-4\delta$.

▨

Lecture 27 - Ironic Complexity

(Title by Santhanam & Aaronson)

Yao'82, Blum-Micali ...

Circuit lower bounds $\Rightarrow$ PRGs $\Rightarrow$ $BPP \subseteq SUBEXP$

$\vdots$

P.

Hardness $\Rightarrow$ easiness

(Irony)

HW8#2: $BPP = P \Rightarrow NEXP \not\subseteq AlgP/poly$.

Easiness $\Rightarrow$ hardness

("complicity" betw. algs & hardness)

An example [PPZ'97]:

thm: kSAT solvable in time $\dfrac{poly(n) \cdot 2^n}{2^{c_k \cdot n}}$, $c_k \sim \frac{1}{2k}$.

(fastest known for k>4?)

exponential "savings"

def: "Nontrivial" $\mathcal{C}$-SAT alg.: one with savings $n^{\omega(1)}$ (on poly(n)-size inputs)

↓ savings n (on poly(n)-size inputs)

pf sketch: They show a way to "encode" all "isolated" sat'ing asgns of ϕ

↳ (x s.t. $\phi(x)=1$, $\phi(y)=0$ when $\Delta(y,x)=1$)

using $n - \frac{n}{2k}$ bits. (Encoding deps on ϕ.)

(⇒ Unique-k-SAT solvable w/ ~$\frac{n}{2k}$ savings

↳ 1 sat asgn → isolated → try all encodings.)

(Somehow, if >1 SAT asgn, life is easier)

Cor: k-CNFs have ≤ $2^{n-n/2k}$ isolated sat. asgns.

Cor: Depth-3 circuit lower bounds for Parity!

size $\Theta(n^{1/4} 2^{\sqrt{n}})$

↑ 2^{n-1} isolated sat asgns

(tight up to Θ! Håstad just had $2^{\Omega(\sqrt{n})}$)

Irony: Truly understanding $\mathcal{C}$ = k-CNFs

⇒ identifying special notn/structure

⇒ identifying special ppty/structure

⇙ ⇗ ⇓ (limitation)

nontrivial SAT alg (learning alg...)

ckt lower bound (PRG...)

(Another e.g.: Switching Lemma-type ideas
AC^0-SAT alg.
w/ savings $2^{\Omega(n)}$
[CIP'09, ...])

Today: [Wil'11]: HW11 #3: ACC-SAT alg w/ savings $2^{n^{\Omega(1)}}$ ⇒ ...?⇒ lower bound against (non-unif) ACC.

(Brief) History (of Ckt Lower Bounds)

- early '80s:
  - $\text{Parity}^{(EL)} \notin AC^0$
  - (Kannan) $\Sigma_2 P \not\subseteq SIZE(n^{1000})$ (whatever KL gets you)
- '87ish: $\text{Majority}^{(EL)} \notin AC^0[p]$, p prime

$IP_2^{(\epsilon L)} \not\subseteq$ depth-2 Maj ckts

- '90: $P^{\#P}$, PSPACE $\subseteq$ IP $\Rightarrow$ checkers

$\Rightarrow$ PSPACE $\subseteq$ P/poly $\Rightarrow$ PSPACE $=$ MA

(HW4, Test1) KL/Meyer: EXP $\subseteq$ P/poly $\Rightarrow$ EXP $= \Sigma_2 P$

$\Rightarrow$ $=$ MA. ⊛

$$\Rightarrow \underbrace{\text{MA-EXP} \not\subseteq \text{P/poly} \quad \text{[BFT '98]}}_{\underline{\text{ex}}\text{, using ⊛ \& Kannan}}$$

(doesn't relativize)

[2008 Santhanam]: prMA $\not\subseteq$ SIZE(n^{1000}).

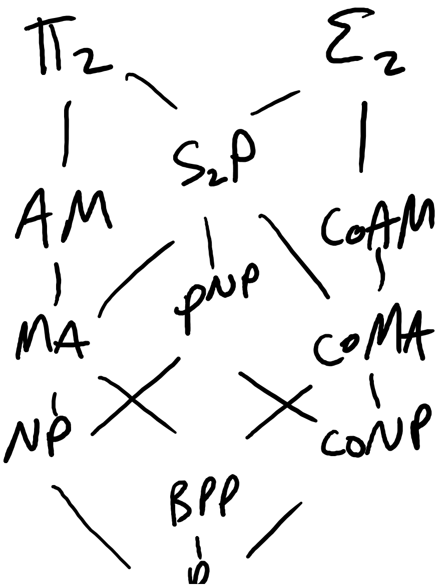
[AG '94]: PP, Perm not in Unif-ACC.

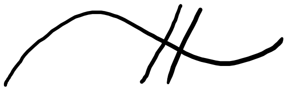
↳ Q: Is NEXP $\subseteq$ AC^0[mod 6] ???

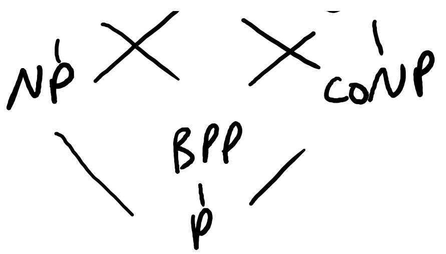
EXP^{NP} ??

(MA-EXP isn't!)

[Wil'11]: A: No.







Non-trivial SAT algs ⇒ ckt lower bounds

Warmup 1: SAT $\in$ P $\Rightarrow$ EXP $\not\subseteq$ P/poly

pf: Else: SAT $\in$ P $\Rightarrow \Sigma_2 P = \exists\forall P = \exists P = P$

$EXP \subseteq P/poly \Rightarrow EXP = \Sigma_2 P$

$\therefore EXP = P$, contra to THT

"Indirect diagonalization proof".

Warmup 2: $SAT \in TIME(2^{n^{o(1)}})$

$\Rightarrow \Sigma_2 P = \exists\forall P \subseteq \underset{(N)}{\exists} TIME(2^{n^{o(1)}})$ ⊛

repeated use → no good ☹

Can't seen to get

$EXP \not\subseteq P/poly$. ☹

What about <u>NEXP</u> $\not\subseteq$ P/poly?

AFSOC NEXP $\subseteq$ P/poly, Test2 #2 [IKW]

$\Rightarrow$ NEXP = EXP

$= \Sigma_2 P$ (b/c EXP $\subseteq$ P/poly)

Combine with $\circledast$: NEXP $\subseteq$ NTIME$(2^{n^{o(1)}})$

contra to NTHT! ☺.

Got: CKT-SAT $\in$ TIME$(2^{n^{o(1)}})$ $\Rightarrow$ NEXP $\not\subseteq$ P/poly

[Wil'10,'11]: For any circuit class $\mathcal{C}$ (with mild closure ppts) $AC^0 \subseteq \mathcal{C} \subseteq$ P/poly, it holds:

(leave up)

$\mathcal{C}$-SAT has nontriv. alg $\Rightarrow$ NEXP $\not\subseteq \mathcal{C}$.

$(2^n/n^{\omega(1)}$ time)

& <u>True</u> of $\mathcal{C}$ = ACC! (Savings $2^{n^{\Omega(1)}}$ as he/you showed, HW1#3)

(Proof deferred. Rem: getting $EXP^{NP} \not\subseteq \mathcal{C}$ easier...)

<u>Rec:</u> IKW & Test 2's proof of

NEXP $\subseteq$ P/poly $\Rightarrow$ NEXP = EXP $(\overset{\therefore}{=} \Sigma_2 P)$.

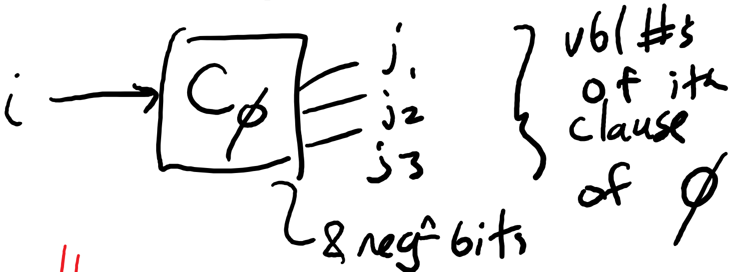
used "Easy Witness Method":

1.

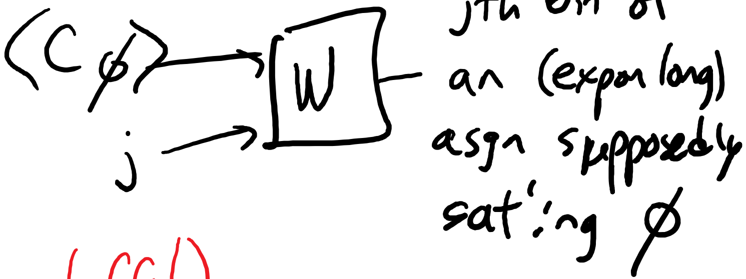
Basically showed: SUCCINCT-3SAT $\in$ P/poly $\Rightarrow$ SUCC-3SAT has succinct witnesses

↓

Given C_ϕ encoding expon-size 3CNF ϕ:



(If you don't recall IKW's proof, it's OK. We'll show an easier variation now. Doesn't require all the difficult derand, MA stuff!)



Easy to prove under $EXP^{NP} \subseteq P/poly$.

Indeed, assume $EXP^{NP} \subseteq C$.

Recall: in P^{NP}, given 3CNF ϕ, can output lex-least sat'ing asgn, or "UNSAT"

(like search-to-decision)

Sim., in EXP^{NP}, given $\langle C_\phi \rangle$, j can output j^{th} bit of ϕ's lex-least sat asgn (or "UNSAT").

Assumption ∴ implies SUCC-3SAT has succinct $\mathcal{C}$-witnesses!

Proof of Williams:

Sps $\mathcal{C}$-SAT has nontriv. alg, $EXP^{NP} \subseteq \mathcal{C}$.

(with 5 minutes extra arg, can get NEXP, using IKW + small idea. Fine-grain time...)

Let $L \in NTIME(2^n) \setminus NTIME(2^{o(n)})$

(in RAM model; OK by NTHT.)

Lec. 6: SUCC-3SAT is NEXP-hard in super-efficient way: ∃ ptime reduction

$$|x|=n \quad x \overset{?}{\in} L \longrightarrow \langle C_\phi \rangle \overset{?}{\in} \text{SUCC-3SAT}$$

where $\text{size}(\phi) = 2^n \cdot n^{O(1)}$ ($T \cdot \text{polylog}\, T$, even for RAMs)

i.e., C_ϕ takes $n + O(\log n)$ inputs.

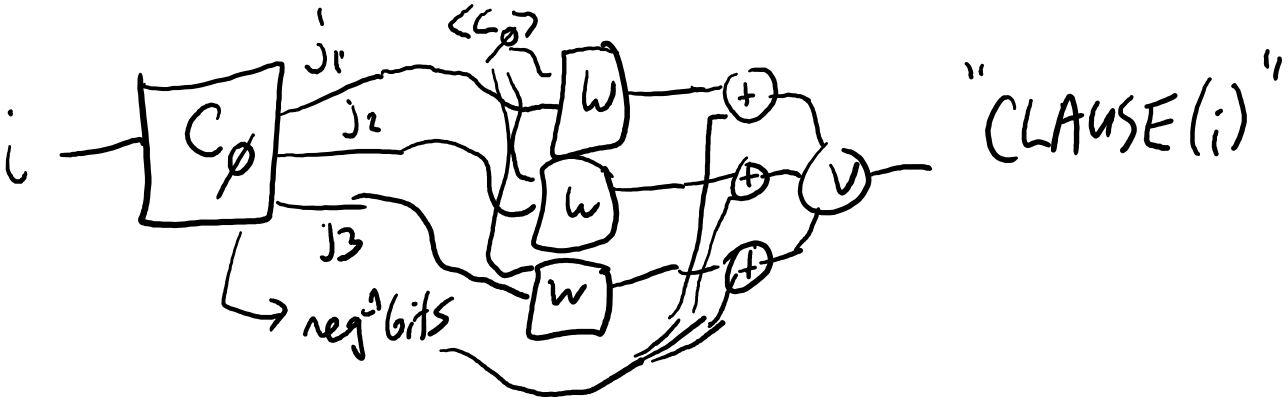
Bonus [Jahanjou - Miles - Viola '14] (also Kowalski - van Melk)

C_ϕ not just in P/poly $(TIME(\text{polylog}\, T))$

but in AC^0! (In fact, they showed AC^0 w/ fan-in 2 gates; i.e, NC^0 – each output bit deps only on $O(1)$ inputs! Rem: [Will] didn't have this – proof needs another 5 mins of trickery.)

To decide $x \overset{?}{\in} L$ in $NTIME(2^{o(n)})$ ($\Rightarrow$ contra):

- compute $C_\phi \in AC^0$ // poly(n) time
- $\because$ $EXP^{NP} \subseteq \mathcal{C}$, we know C_ϕ has succinct $\mathcal{C}$-witness W^*. Nondet guess a (supposed) $\mathcal{C}$-witness W.
- Need to check that W encodes sat asgn for ϕ



W sats $\phi \iff$ CLAUSE$(i) = 1$ $\forall i$

...

$\Leftrightarrow \neg\text{CLAUSE}(i) = 0 \quad \forall i$

$\Leftrightarrow \underbrace{\neg\text{CLAUSE}}_{}$ not sat'ble.

A $\mathcal{C}$-circuit, with input len. $\underline{n + O(\log n)}$.

(Assuming mild closure; OK for ACC.)

$\therefore$ $\mathcal{C}$-SAT alg can check in time

$$2^{n+O(\log n)} / n^{\omega(1)} = 2^{o(n)}.$$ ⨳.

∎

LECTUR

+R

pseudo

+

nondeterm

def: Nondet alg (

20

CMU Graduate Complexity Lecture 20

Valiant’s Theorem: Permanent is #P-complete

Ryan O’Donnell

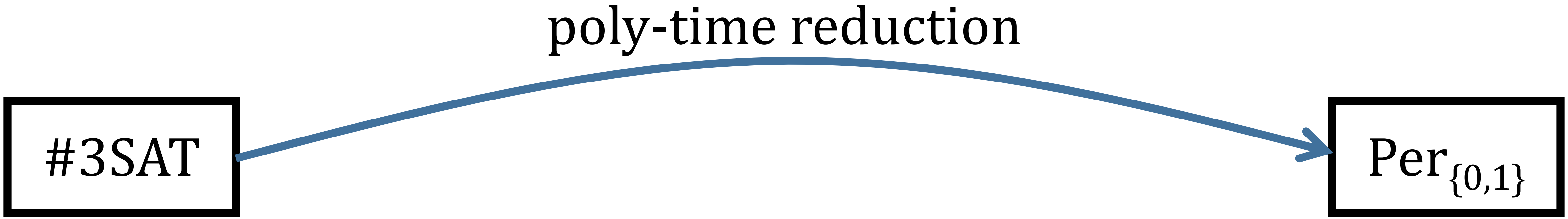
(Proof exposition from [Dell–Husfeldt–Marx–Taslaman–Wáhlen’14 & Bläser’15].)

Let $A \in \mathbb{Z}^{n \times n}$. Its **permanent** is:

$$\mathrm{Per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i\sigma(i)}$$

Valiant's Theorem:

Computing Permanent, even on 0-1 matrices, is #P-complete.



poly-time reduction

#3SAT

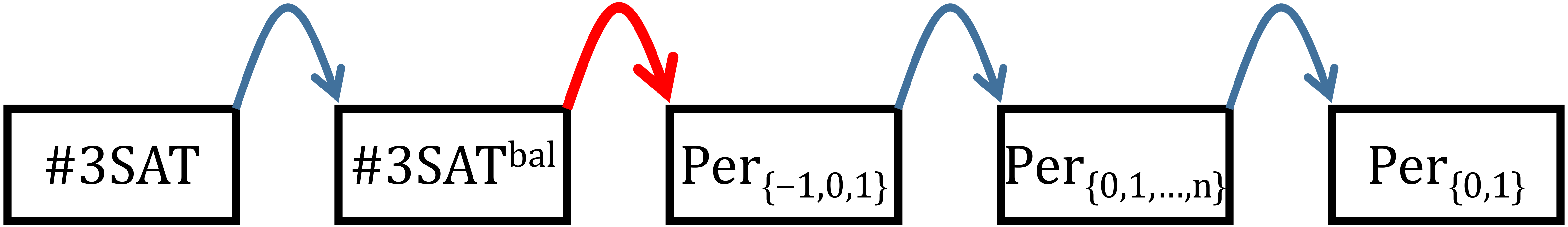
$\mathrm{Per}_{\{0,1\}}$

Let $A \in \mathbb{Z}^{n\times n}$. Its **permanent** is:

$$\mathrm{Per}(A) = \sum_{\sigma\in S_n}\prod_{i=1}^{n} A_{i\sigma(i)}$$

Valiant's Theorem:

Computing Permanent, even on 0-1 matrices, is #P-complete.



#3SAT

$\#3\mathrm{SAT}^{\mathrm{bal}}$

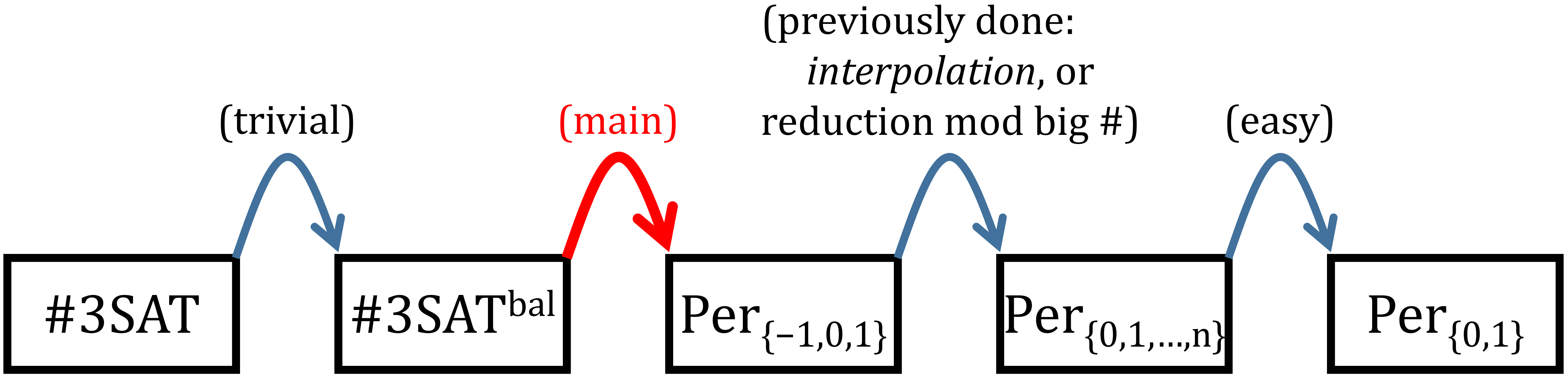
$\mathrm{Per}_{\{-1,0,1\}}$

$\mathrm{Per}_{\{0,1,\ldots,n\}}$

$\mathrm{Per}_{\{0,1\}}$

Let $A \in \mathbb{Z}^{n\times n}$. Its **permanent** is:

$$\mathrm{Per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i\sigma(i)}$$



(trivial)

(main)

(previously done:
interpolation, or
reduction mod big #)

(easy)

#3SAT

$\#3\mathrm{SAT}^{\mathrm{bal}}$

$\mathrm{Per}_{\{-1,0,1\}}$

$\mathrm{Per}_{\{0,1,\ldots,n\}}$

$\mathrm{Per}_{\{0,1\}}$

Let $A \in \mathbb{Z}^{n\times n}$. Its **permanent** is:

$$\mathrm{Per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i\sigma(i)}$$

ϕ has $\boldsymbol{m}$ clauses,
$\boldsymbol{C}$ satisfying
assignments

A_ϕ has permanent
equal to $(-\mathbf{2})^{\boldsymbol{3m/2}} \cdot \boldsymbol{C}$

(main reduction)

#3SAT$^{\mathrm{bal}}$

$\mathrm{Per}_{\{-1,0,1\}}$

Let $A \in \mathbb{Z}^{n \times n}$. Its **permanent** is:

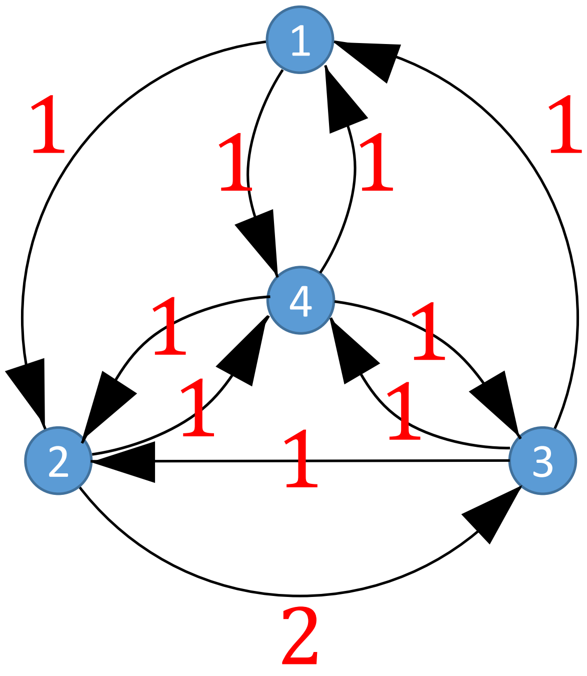
$$\mathrm{Per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i\sigma(i)}$$

Useful **reinterpretation** of $\mathrm{Per}(A)$:

Cycle Covers

Cycle Covers

$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 2 & 1 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 1 & 1 & 1 & 0 \end{array}$$



$$\mathrm{Per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i\sigma(i)}$$

$$\sigma: \begin{tikzcd} 1 \arrow[d] & 2 \arrow[d] & 3 \arrow[d] & 4 \arrow[d] \\ 2 & 3 & 4 & 1 \end{tikzcd}$$

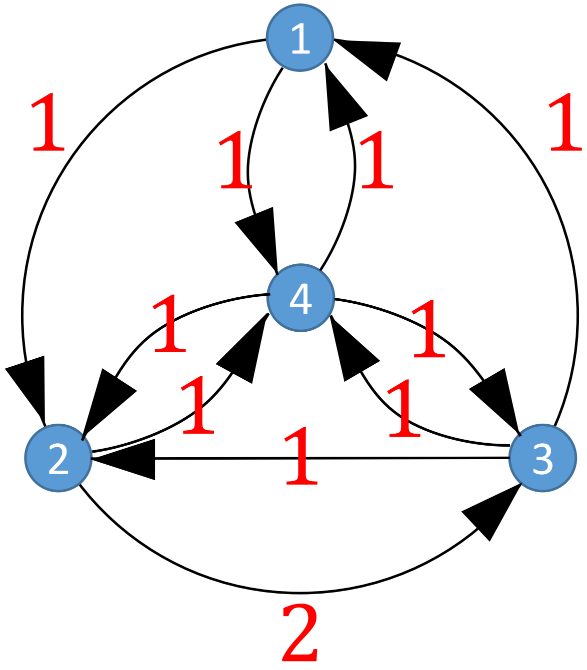
Weight:

$$1 \cdot 2 \cdot 1 \cdot 1 = \mathbf{2}$$

“Weight” = product of edge labels

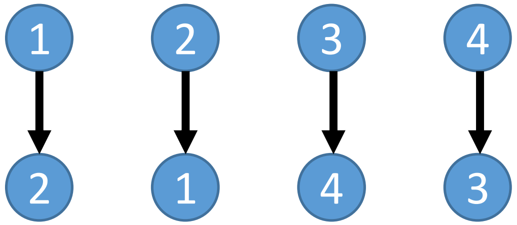
Cycle Covers

$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 2 & 1 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 1 & 1 & 1 & 0 \end{array}$$



$$\mathrm{Per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i\sigma(i)}$$

σ:



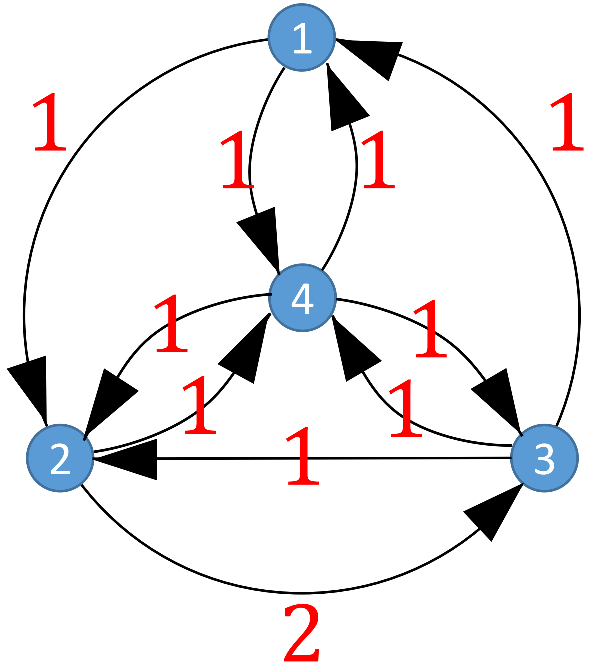
Weight:

$1 \cdot 0 \cdot 1 \cdot 1 = \mathbf{0}$

“Weight” = product of edge labels

Cycle Covers

$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 2 & 1 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 1 & 1 & 1 & 0 \end{array}$$



$$\mathrm{Per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i\sigma(i)}$$

$$\sigma: \begin{array}{cccc} 1 & 2 & 3 & 4 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 4 & 3 & 2 & 1 \end{array}$$

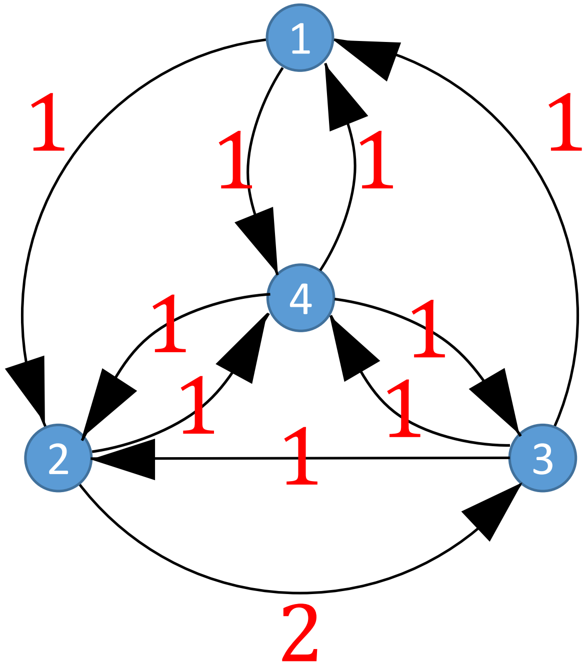
Weight:

$$1 \cdot 2 \cdot 1 \cdot 1 = \mathbf{2}$$

“Weight” = product of edge labels

Cycle Covers

$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 2 & 1 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 1 & 1 & 1 & 0 \end{array}$$



$$\mathrm{Per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i\sigma(i)}$$

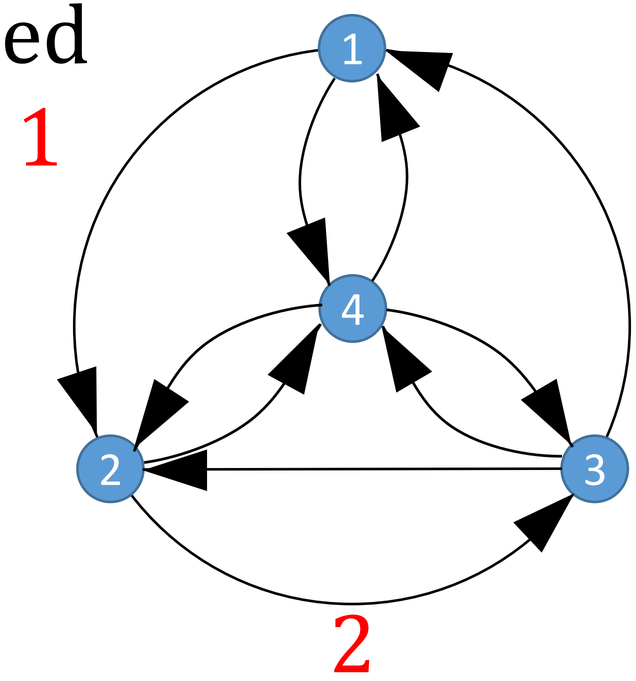
Permutations of **nonzero** weight
= Cycle Covers

$\mathrm{Per}(A)$ = sum of weights of all cycle covers

Cycle Covers

$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 2 & 1 \\ 3 & 1 & 1 & 0 & 1 \\ 4 & 1 & 1 & 1 & 0 \end{array}$$

Unlabeled edges = 1



$$\mathrm{Per}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i\sigma(i)}$$

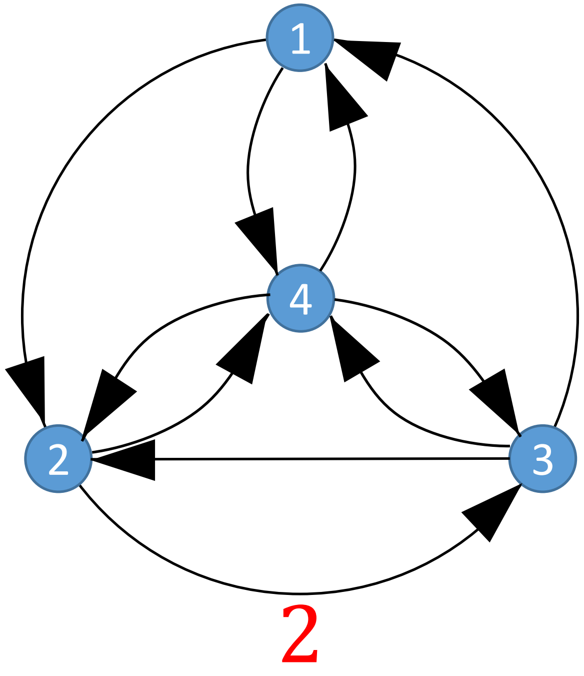
Permutations of **nonzero** weight
= Cycle Covers

$\mathrm{Per}(A)$ = sum of weights of all cycle covers

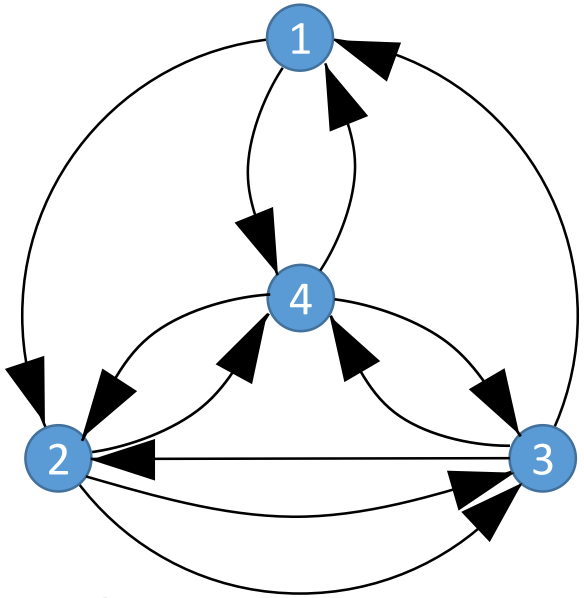
Tricks with Cycle Covers

Trick 1: Replace weights > 1 by parallel edges.

(Wait: are parallel edges "allowed"?)



Tricks with Cycle Covers



Trick 1: Replace weights > 1 by parallel edges.

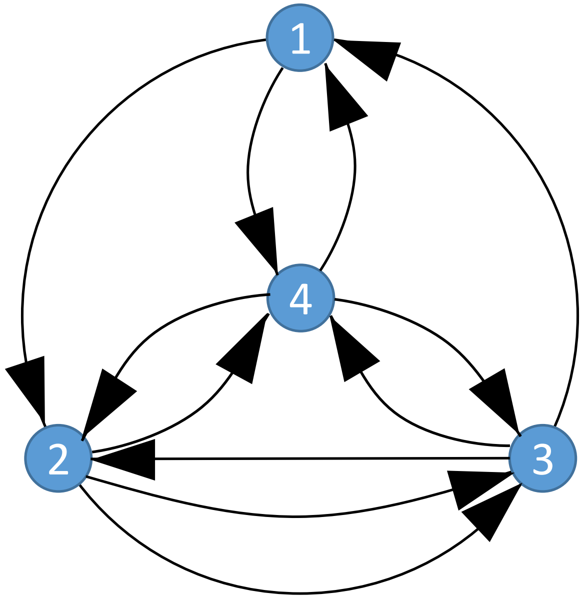
Claim: Total cycle cover weight unchanged.

Proof: What are the cycle covers in the new graph? 2

Type I: An old cycle cover that **didn't** use the replaced edge. The weight of such cycle covers is unchanged.

Type II: An old cycle cover that **did** use the replaced edge: These become new cycle covers, of 1/2 the weight, in 2 different ways.

Tricks with Cycle Covers



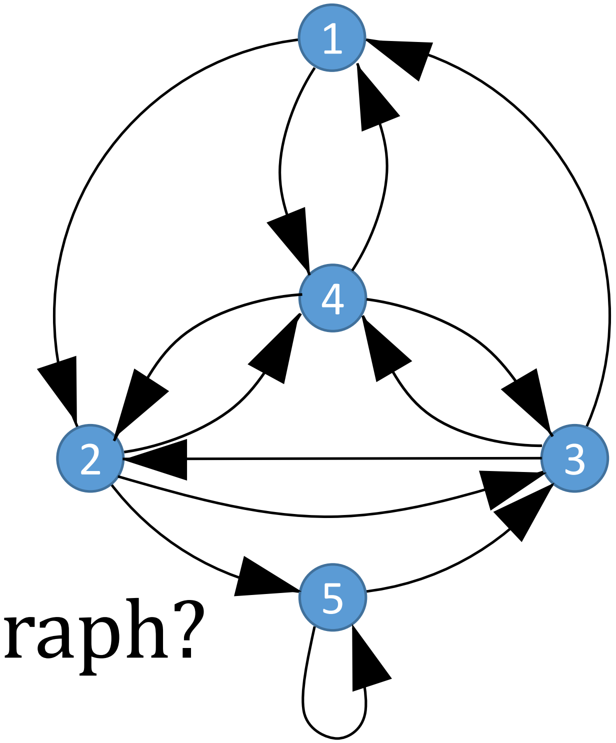
Trick 1: Replace weights > 1 by parallel edges.

Are parallel edges “allowed”?

Well, no.

But…

Tricks with Cycle Covers



Trick 2: Can always subdivide any edge, sticking in a self-loop.

Claim: Total cycle cover weight unchanged.

Proof: What are the cycle covers in the new graph?

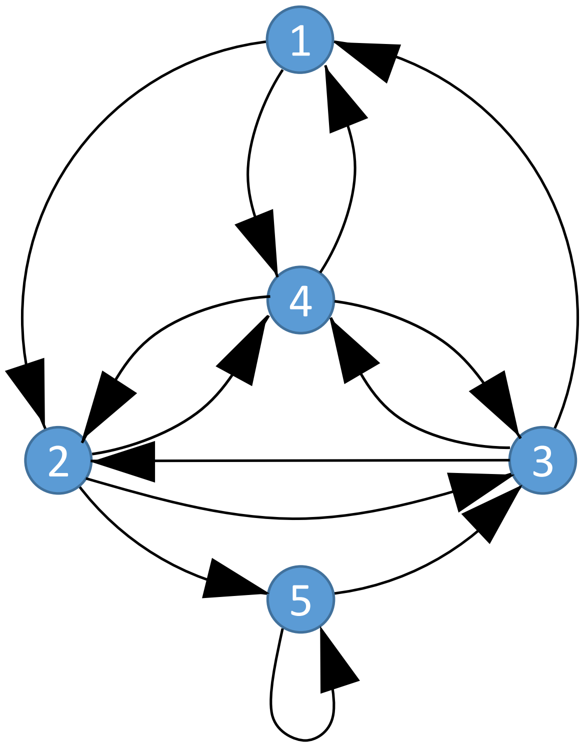
Type I: An old cycle cover that **didn't** use the subdivided edge:
→ new *same-weight* cycle cover by adding the self-loop

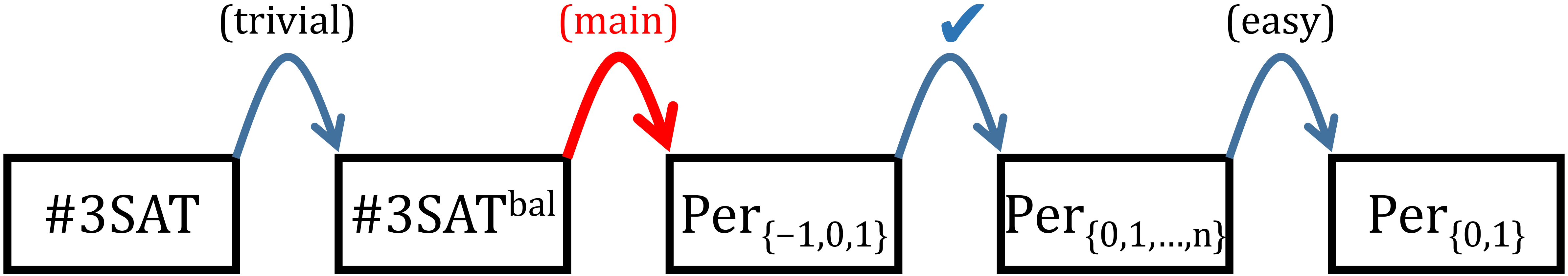
Type II: An old cycle cover that **did** use the subdivided edge:
→ new *same-weight* cycle cover that takes the length-2 path, avoiding the self-loop.

Tricks with Cycle Covers

We henceforth allow parallel edges.

We can get rid weights {2, 3, ..., poly(n)}
with poly(n)-size blowup.





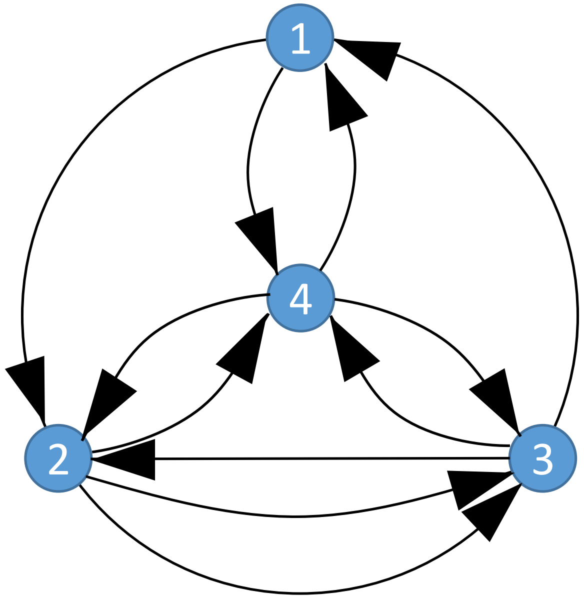
(trivial) (main) ✓ (easy)

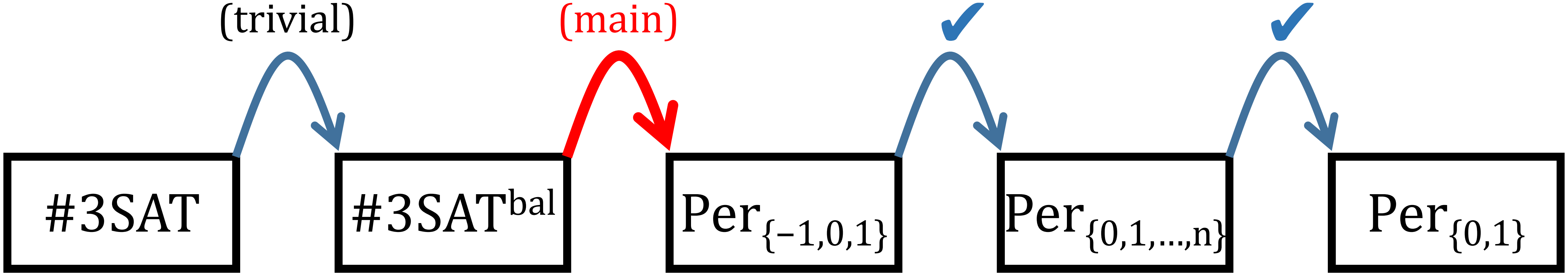
#3SAT → $\#3\text{SAT}^{\text{bal}}$ → $\text{Per}_{\{-1,0,1\}}$ → $\text{Per}_{\{0,1,\dots,n\}}$ → $\text{Per}_{\{0,1\}}$

Tricks with Cycle Covers

We henceforth allow parallel edges.

We can get rid weights {2, 3, ..., poly(n)}
with poly(n)-size blowup.





#3SAT

(trivial)

#3SAT$^{\text{bal}}$

(main)

$\text{Per}_{\{-1,0,1\}}$

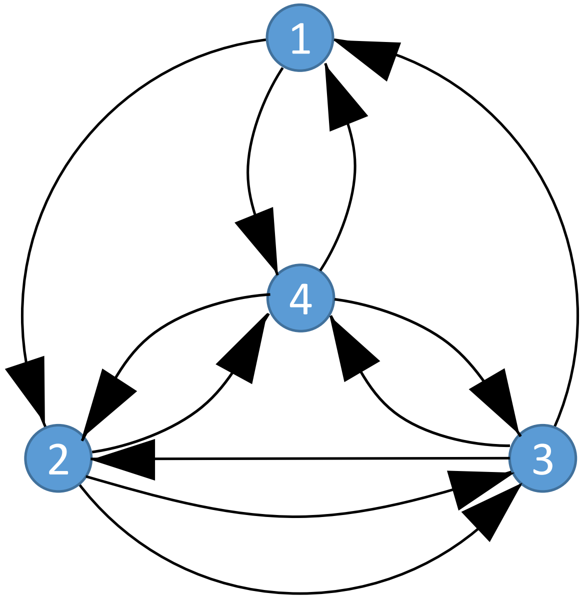
✓

$\text{Per}_{\{0,1,\ldots,n\}}$

✓

$\text{Per}_{\{0,1\}}$

Tricks with Cycle Covers

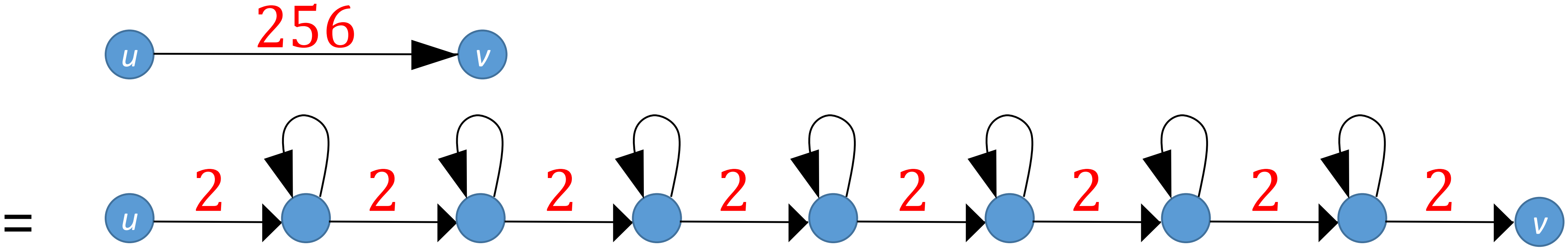


We henceforth allow parallel edges.

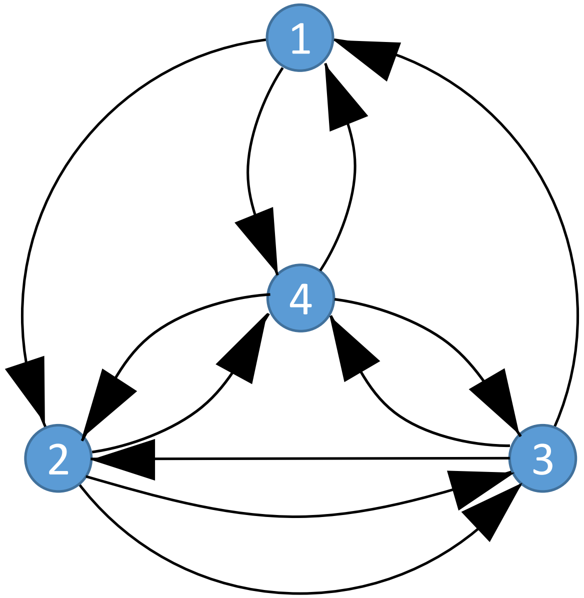
We can get rid weights {2, 3, ..., poly(n)}
with poly(n)-size blowup.

Exercise: Can get rid weights {2, 3, ..., exp(n)}
with poly(n)-size blowup.

Hint:



Tricks with Cycle Covers

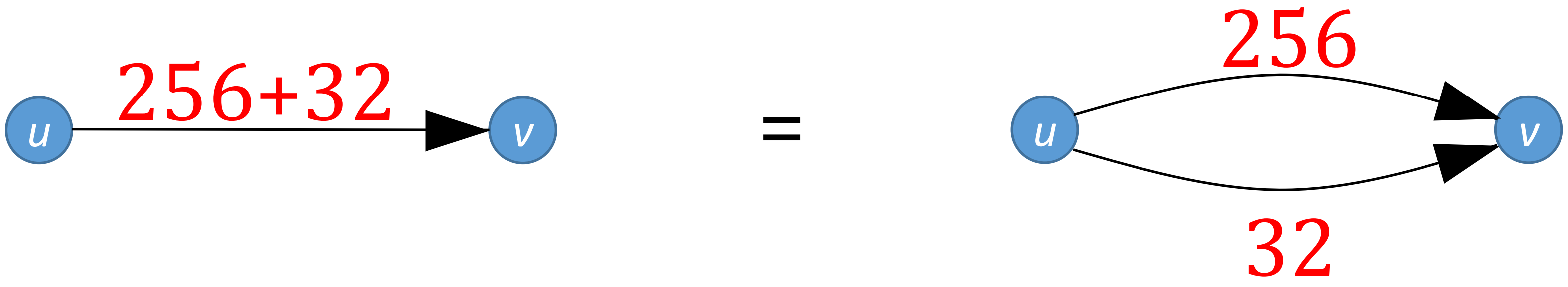


We henceforth allow parallel edges.

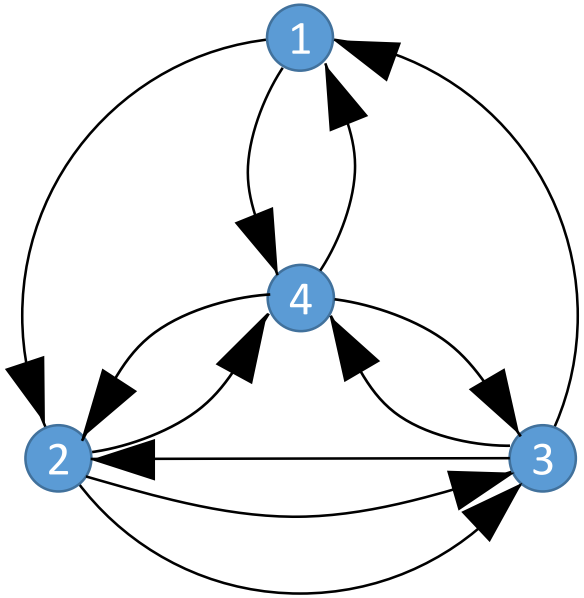
We can get rid weights {2, 3, ..., poly(n)}
with poly(n)-size blowup.

Exercise: Can get rid weights {2, 3, ..., exp(n)}
with poly(n)-size blowup.

Hint:



By the way...



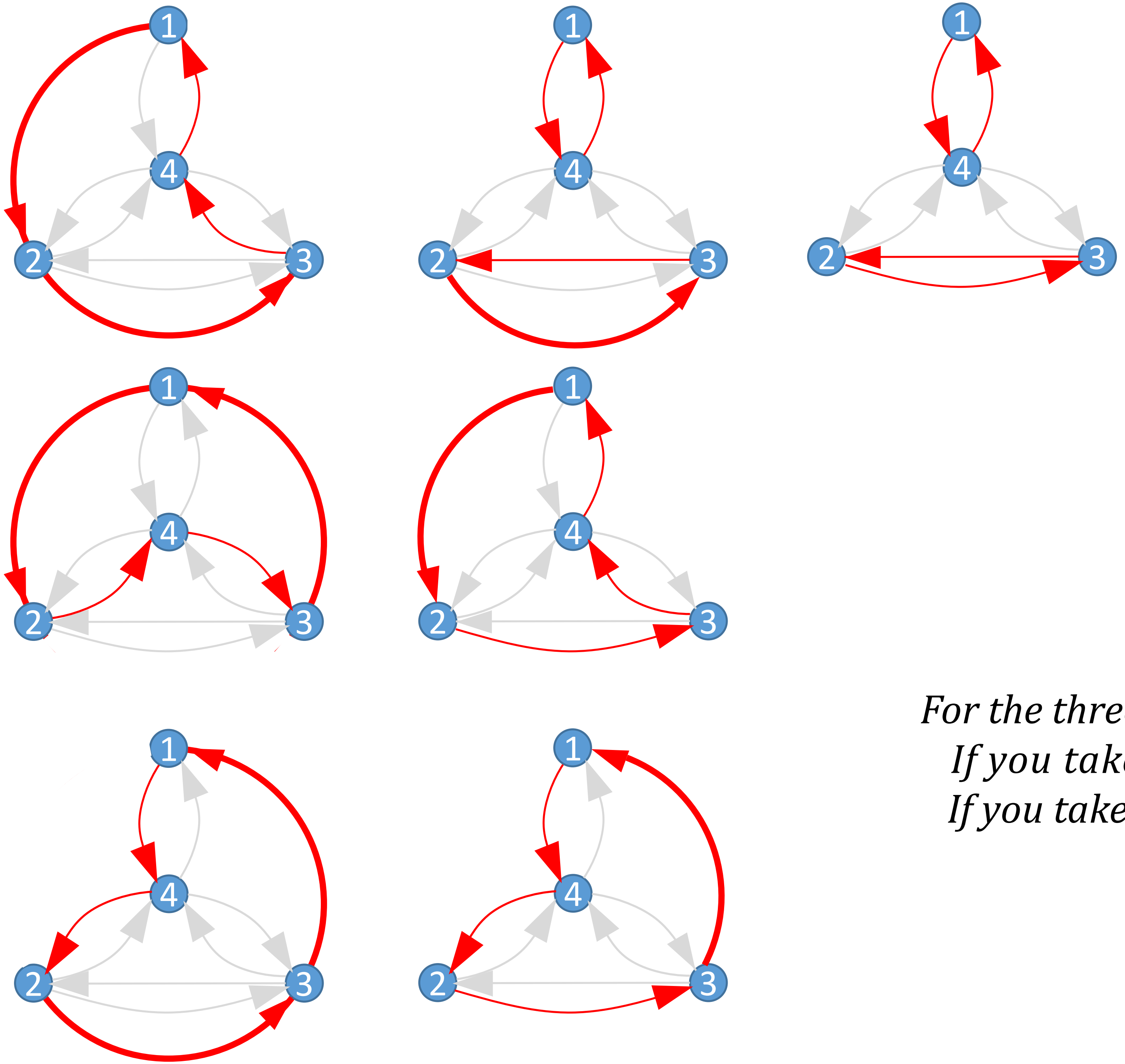
“NAND graph”

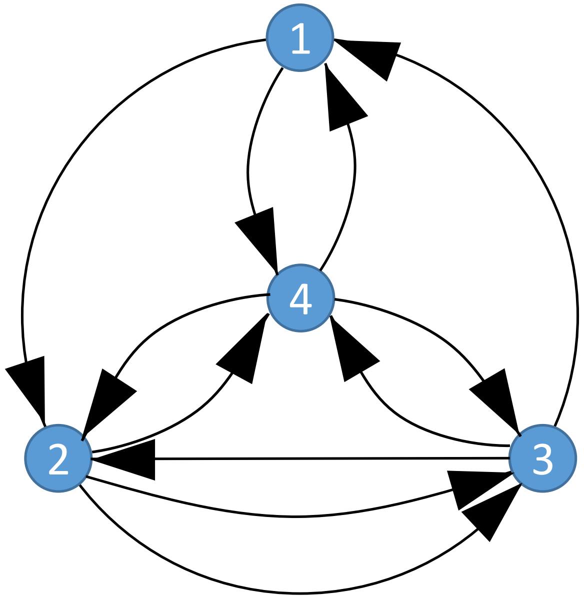
Key property:

For the three “outside” edges:

If you take all three, you ***can’t*** *get a cycle cover.*

If you take any subset, you ***can*** *get 1 cycle cover.*



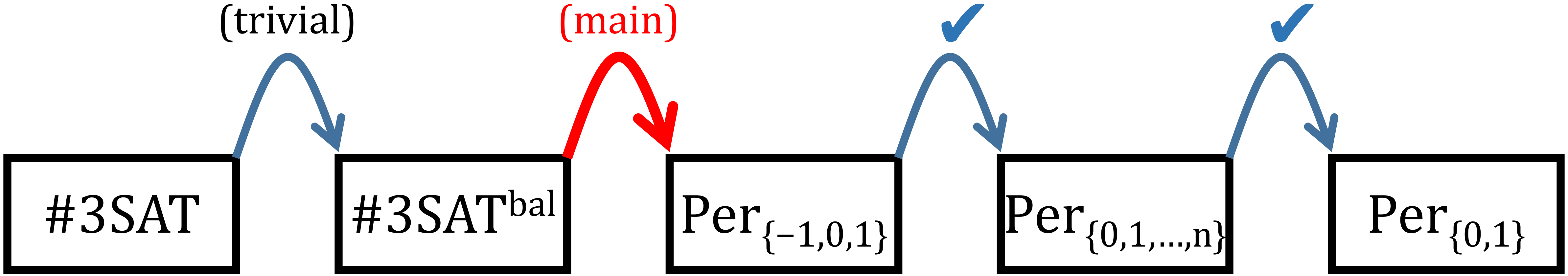


“NAND graph”

For the three “outside” edges:

If you take all three, you ***can’t*** *get a cycle cover.*

If you take any subset, you ***can*** *get 1 cycle cover.*



(trivial)

(main)

✓

✓

#3SAT

$\#3\text{SAT}^{\text{bal}}$

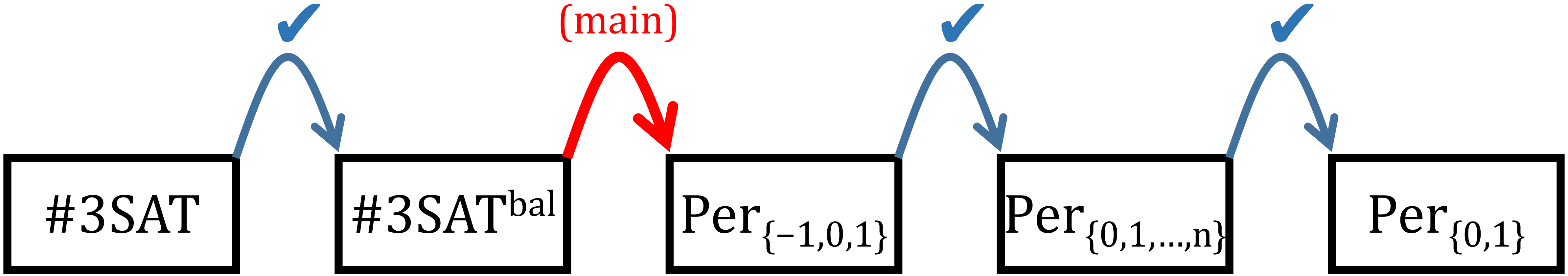
$\text{Per}_{\{-1,0,1\}}$

$\text{Per}_{\{0,1,\ldots,n\}}$

$\text{Per}_{\{0,1\}}$

$\#3\text{SAT}^{\text{bal}}$: Every variable in ϕ appears equally many times unnegated and negated.

Trick: Given ϕ adding clauses like $(x_i \vee x_i \vee \overline{x}_i)$ or $(x_i \vee \overline{x}_i \vee \overline{x}_i)$ doesn't change $\#\phi$.



✓ (main) ✓ ✓

#3SAT

$\#3\mathrm{SAT}^{\mathrm{bal}}$

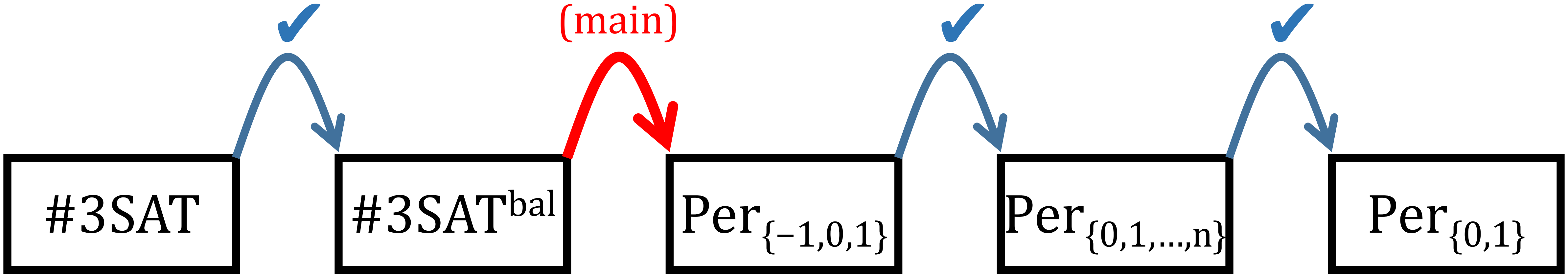
$\mathrm{Per}_{\{-1,0,1\}}$

$\mathrm{Per}_{\{0,1,\ldots,n\}}$

$\mathrm{Per}_{\{0,1\}}$

$\#3\mathrm{SAT}^{\mathrm{bal}}$: Every variable in ϕ appears equally many times unnegated and negated.

Trick: Given ϕ adding clauses like $(x_i \vee x_i \vee \overline{x}_i)$ or $(x_i \vee \overline{x}_i \vee \overline{x}_i)$ doesn't change $\#\phi$.



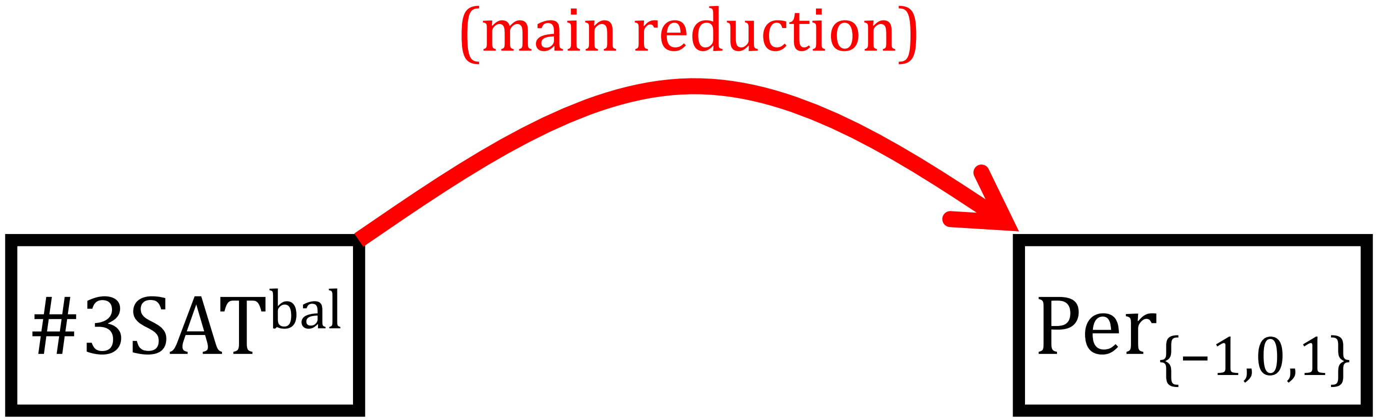
✓ (main) ✓ ✓

#3SAT → $\#3SAT^{bal}$ → $Per_{\{-1,0,1\}}$ → $Per_{\{0,1,\ldots,n\}}$ → $Per_{\{0,1\}}$

$\#3SAT^{bal}$: Every variable in ϕ appears equally many times unnegated and negated.

Every assignment makes half the literals false, half the literals true.

$\#3SAT^{bal}$ → Thus if ϕ has $\boldsymbol{m}$ clauses, then for every assignment, we get $\boldsymbol{3m/2}$ false literals and $\boldsymbol{3m/2}$ true literals.



(main reduction)

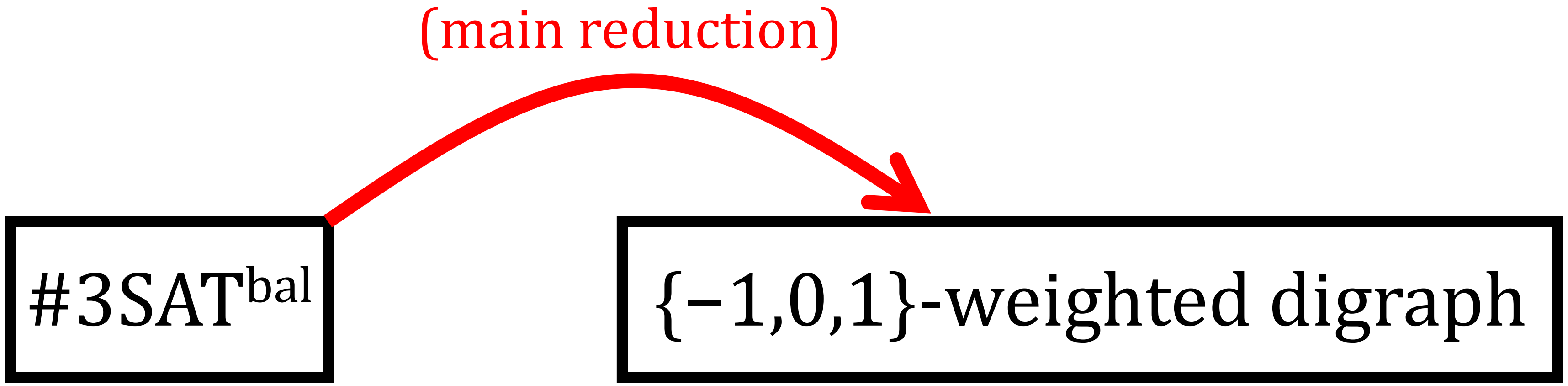
$\#3\mathrm{SAT}^{\mathrm{bal}}$

$\mathrm{Per}_{\{-1,0,1\}}$

ϕ has $\boldsymbol{m}$ clauses, $\boldsymbol{C}$ satisfying assignments

A_ϕ has permanent equal to $(-\mathbf{2})^{\mathbf{3}\boldsymbol{m}/\mathbf{2}} \cdot \boldsymbol{C}$

$\#3\mathrm{SAT}^{\mathrm{bal}} \rightarrow$ Thus if ϕ has $\boldsymbol{m}$ clauses, then for every assignment, we get $\mathbf{3}\boldsymbol{m}/\mathbf{2}$ false literals and $\mathbf{3}\boldsymbol{m}/\mathbf{2}$ true literals.



(main reduction)

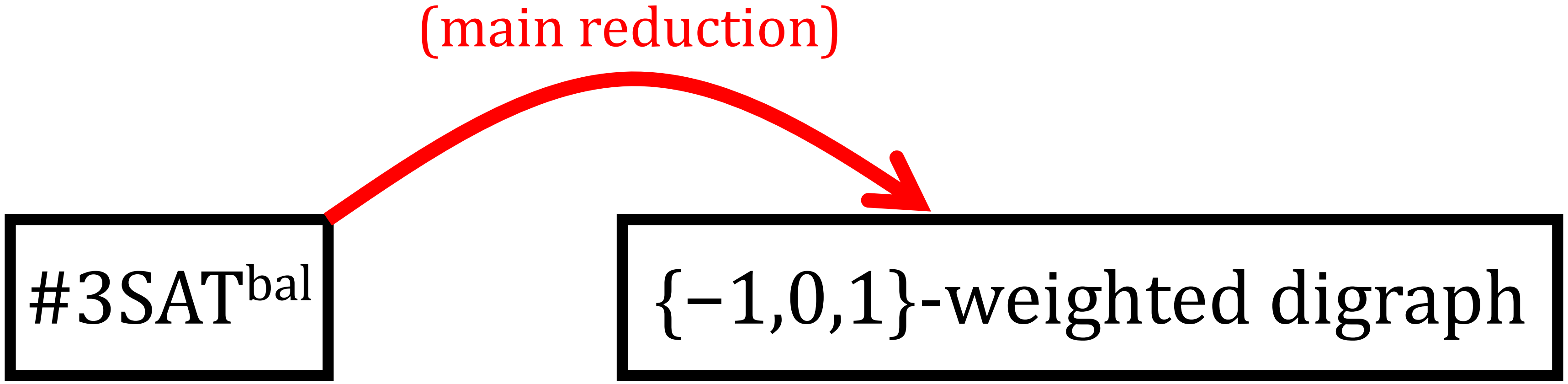
#3SAT$^{\text{bal}}$

{−1,0,1}-weighted digraph

ϕ has $\boldsymbol{m}$ clauses,
$\boldsymbol{C}$ satisfying
assignments

Total cycle cover weight: $(\boldsymbol{-2})^{\boldsymbol{3m/2}} \cdot \boldsymbol{C}$

#3SAT$^{\text{bal}}$ → Thus if ϕ has $\boldsymbol{m}$ clauses, then for every assignment, we get $\boldsymbol{3m/2}$ false literals and $\boldsymbol{3m/2}$ true literals.


(main reduction)

#3SAT$^{\text{bal}}$

{−1,0,1}-weighted digraph

ϕ has $\boldsymbol{m}$ clauses,
$\boldsymbol{C}$ satisfying
assignments

Total cycle cover weight: $(-\mathbf{2})^{\boldsymbol{3m/2}} \cdot \boldsymbol{C}$

Each satisfying assignment making
$\boldsymbol{p}$ literals false and $\boldsymbol{q}$ literals true
yields cycle covers of weight $(-\mathbf{1})^{\boldsymbol{p}}\mathbf{2}^{\boldsymbol{q}}$.

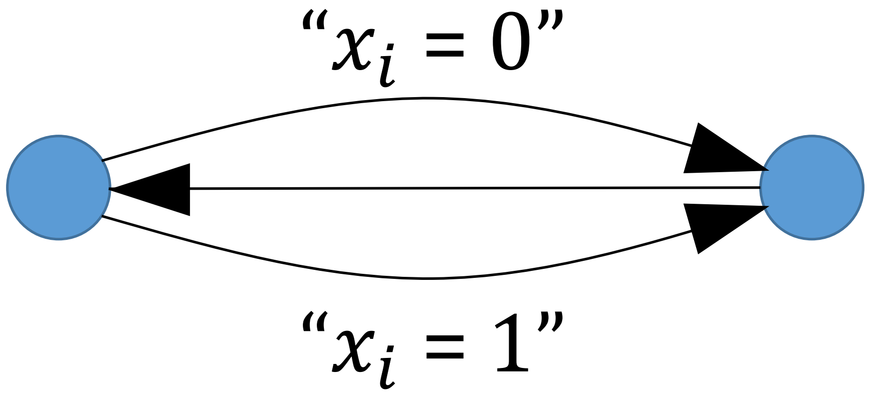
There are many more cycle covers,
but their total weight is **zero**!

#3SAT$^{\text{bal}}$ → Thus if ϕ has $\boldsymbol{m}$ clauses, then for every assignment,
we get $\boldsymbol{3m/2}$ false literals and $\boldsymbol{3m/2}$ true literals.

Main Reduction

#3SAT$^{\text{bal}}$ → {−1,0,1}-weighted digraph

Variable gadget for x_i:

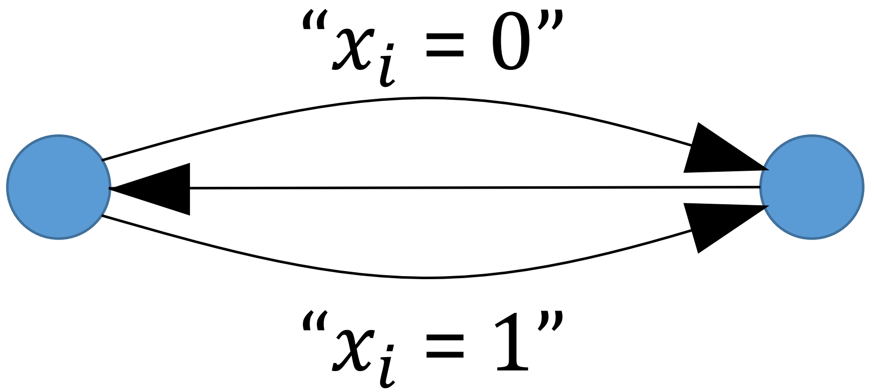


two possible cycle covers,
intended for "x_i = 0", "x_i = 1"

Main Reduction

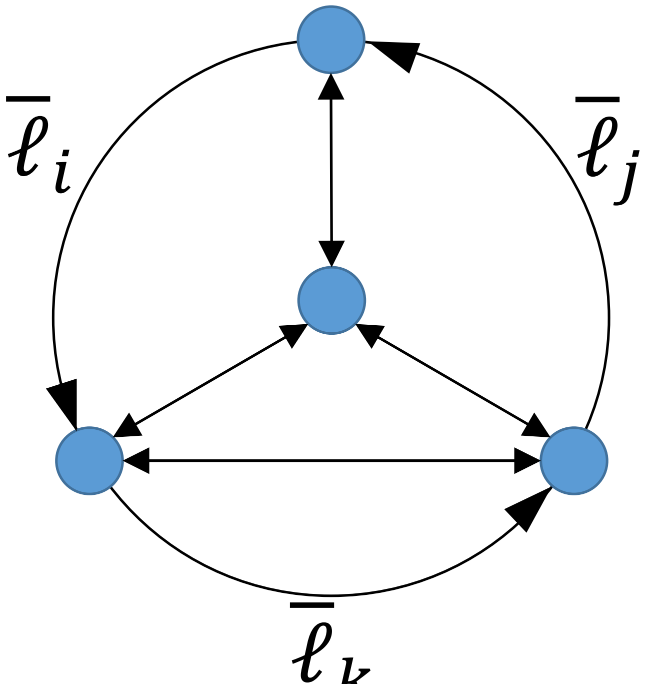
$\#3\mathrm{SAT}^{\mathrm{bal}} \rightarrow \{-1,0,1\}$-weighted digraph

Variable gadget for x_i:



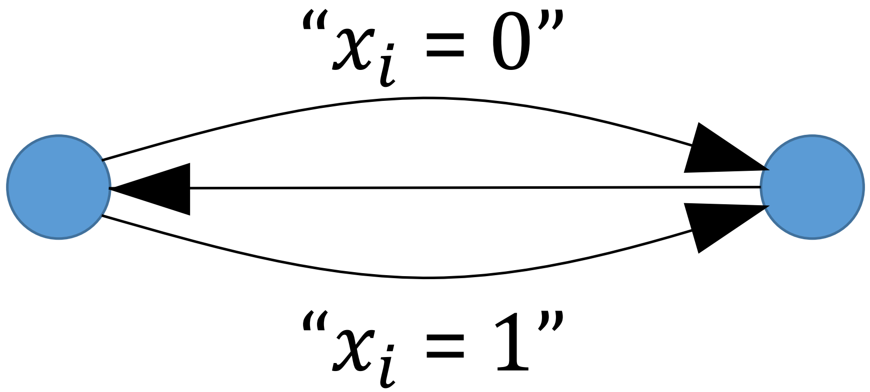
Clause gadget for
$(\ell_i \vee \ell_j \vee \ell_k)$:

$= \mathrm{NAND}(\overline{\ell}_i, \overline{\ell}_j, \overline{\ell}_k)$



Main Reduction

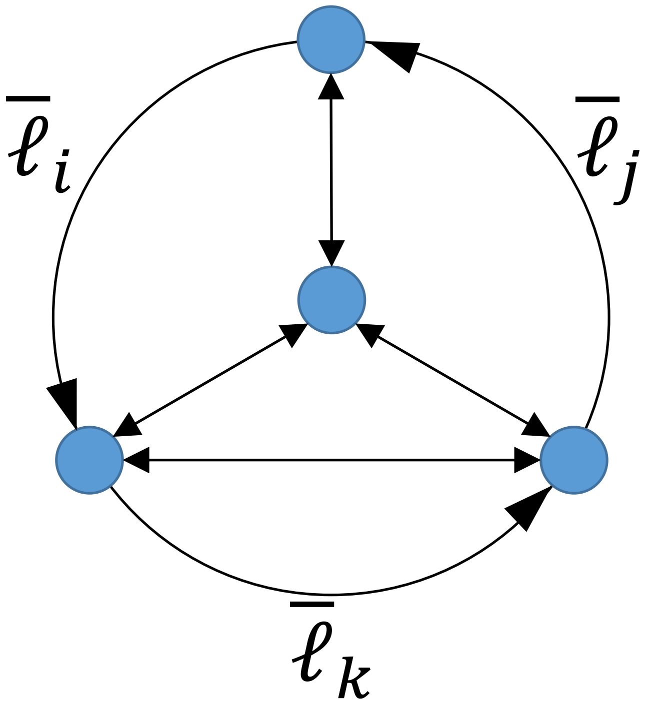
$\#3\text{SAT}^{\text{bal}} \rightarrow \{-1,0,1\}$-weighted digraph



For the three "outside" edges:
*If you take all three, you **can't** get a cycle cover.*
*If you take any subset, you **can** get 1 cycle cover.*

Clause gadget for $(\boldsymbol{\ell_i} \vee \boldsymbol{\ell_j} \vee \boldsymbol{\ell_k})$:

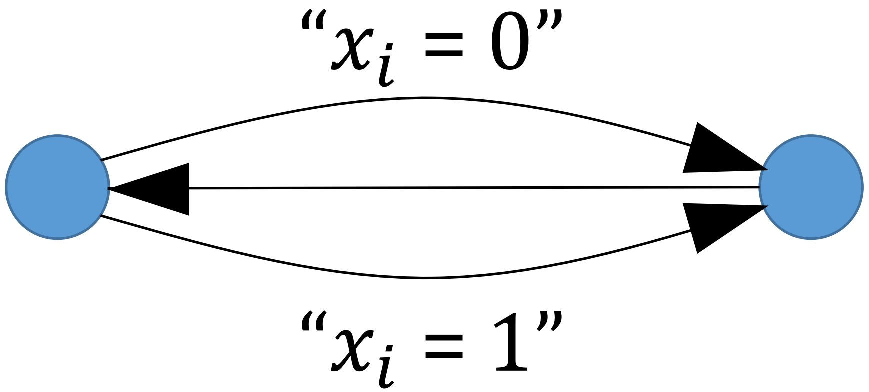
$= \text{NAND}(\overline{\ell}_i, \overline{\ell}_j, \overline{\ell}_k)$



Main Reduction

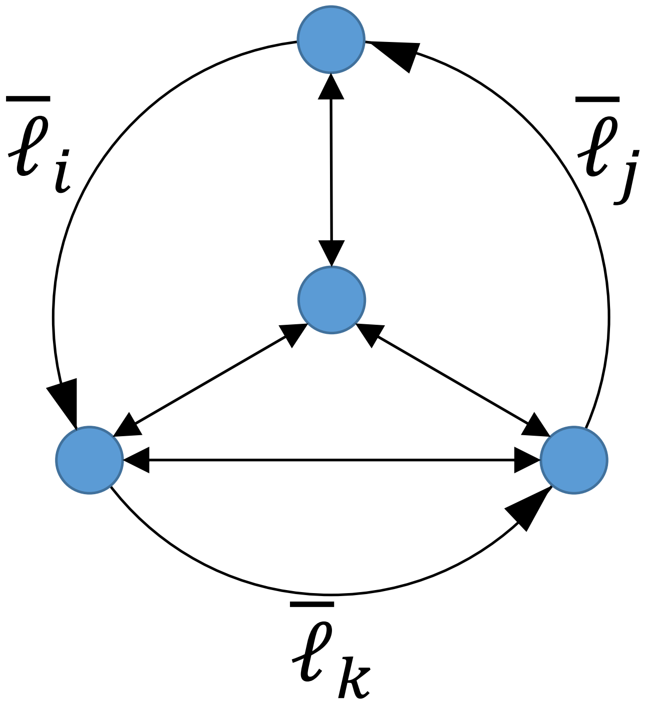
#3SAT$^{\text{bal}}$ → {−1,0,1}-weighted digraph

We now “just” need
to enforce **consistency**.



We wish to ***identify***
variable-edge “$x_i = 0$”
with all clause-edges labeled $\overline{x}_i$,

and similarly for “$x_i = 1$” and
all clause-edges labeled x_i.



Main Reduction

$\#3\mathrm{SAT}^{\mathrm{bal}}$ → $\{-1,0,1\}$-weighted digraph

We now “just” need
to enforce **consistency**.

We wish to ***identify***
variable-edge “$x_i = 0$”
with all clause-edges labeled $\overline{x}_i$,

and similarly for “$x_i = 1$” and
all clause-edges labeled x_i.

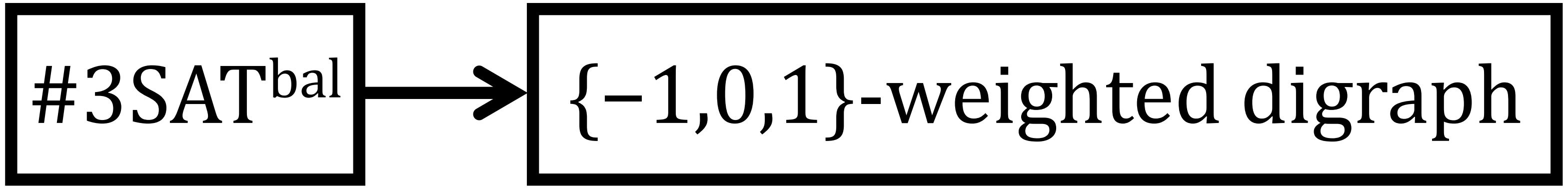
Define edges e, e' to be
perfectly identified if:

For every cycle cover:
it contains e iff it contains e'.

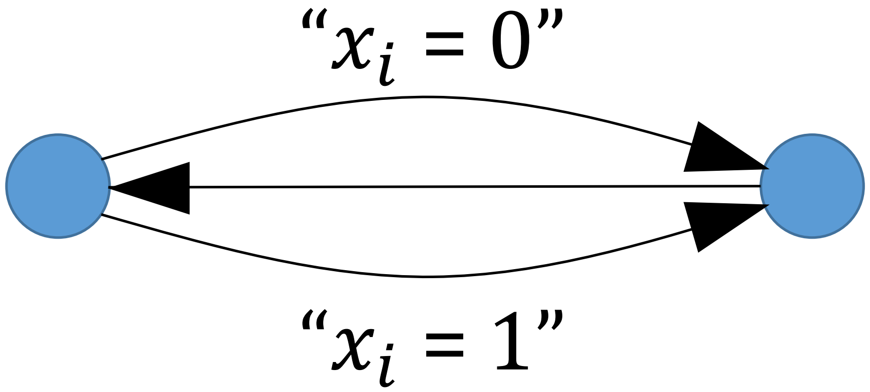
If we could perfectly identify
pairs of edges, we’d be done:

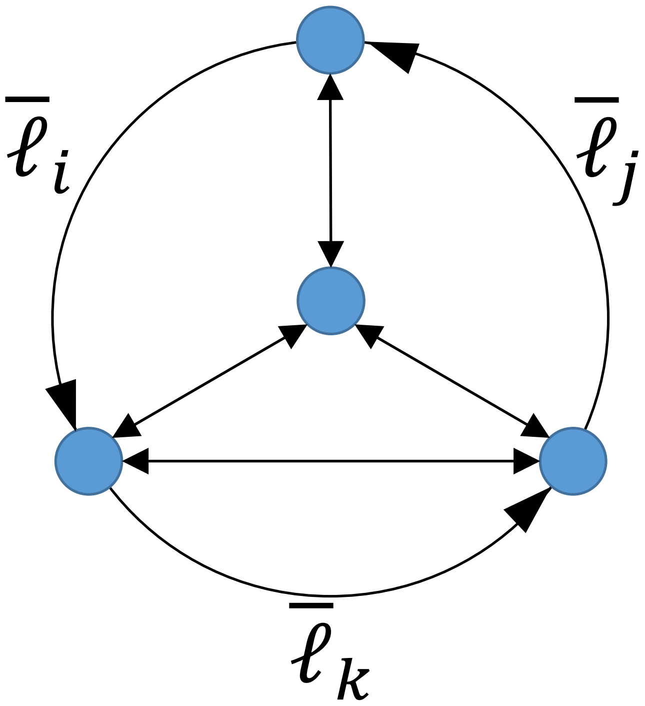
satisfying assignments
= total weight of cycle covers.

Main Reduction



#3SAT$^{\text{bal}}$ → {−1,0,1}-weighted digraph





Define edges e, e' to be **perfectly identified** if:

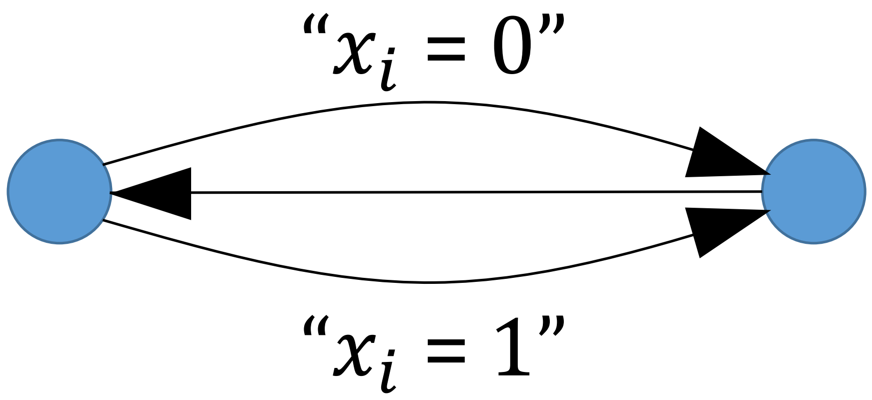
For every cycle cover: it contains e iff it contains e'.

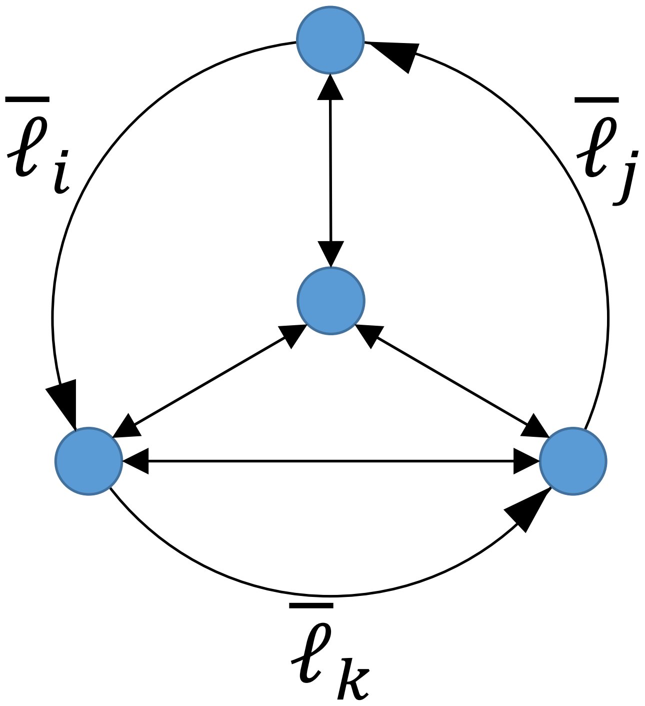
If we could perfectly identify pairs of edges, we'd be done:

\# satisfying assignments = total weight of cycle covers.

Main Reduction

$\#3\mathrm{SAT}^{\mathrm{bal}} \rightarrow \{-1,0,1\}$-weighted digraph





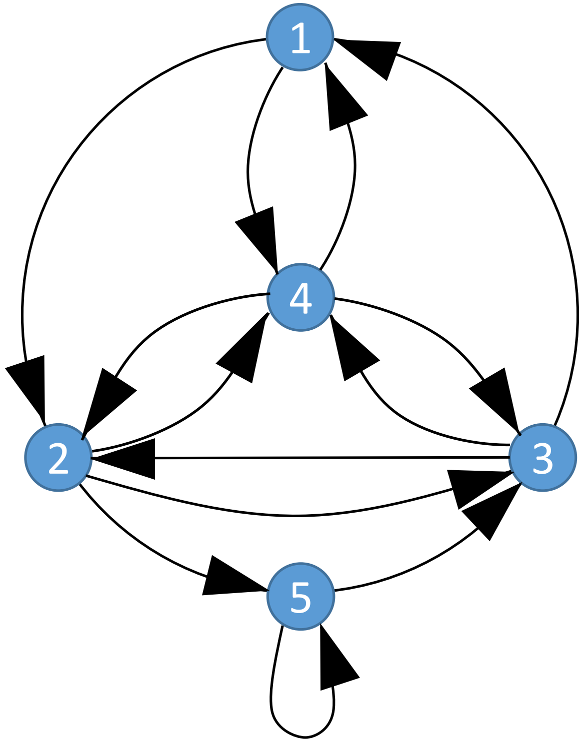
Wrinkle: Each variable appears in multiple clauses.

Idea: **Subdivide** “$x_i = b$” edges multiple times.

Tricks with Cycle Covers

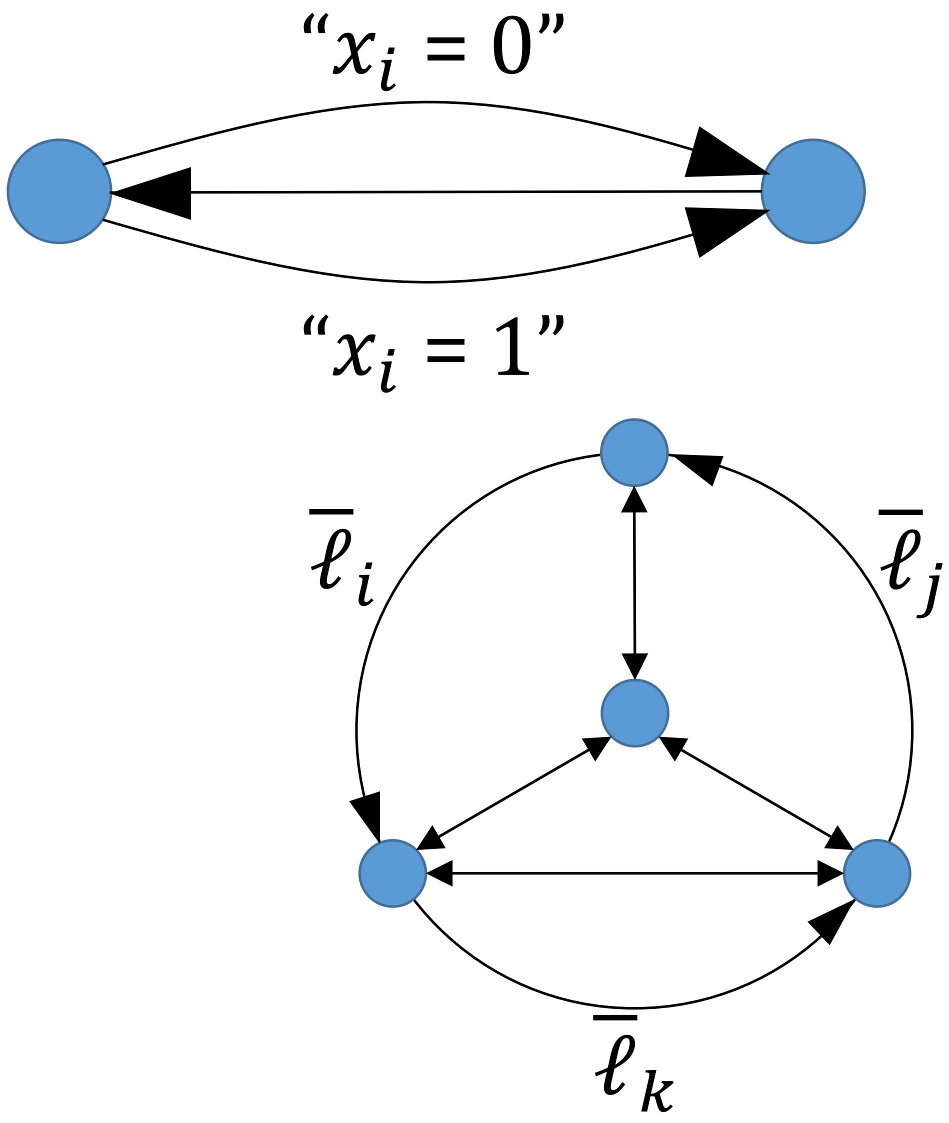
Trick 2: Can always subdivide any edge, sticking in a self-loop.

Claim: Total cycle cover weight unchanged.



Main Reduction

$\#3\mathrm{SAT}^{\mathrm{bal}} \rightarrow \{-1,0,1\}$-weighted digraph

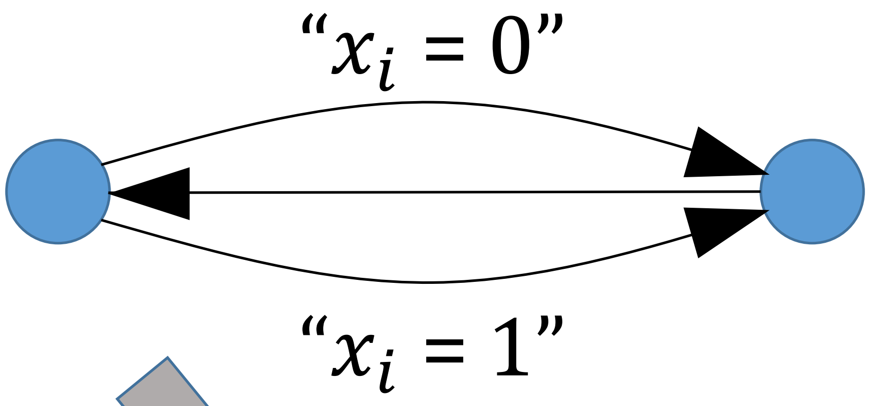


Wrinkle: Each variable appears in multiple clauses.

Idea: **Subdivide** "$x_i = b$" edges multiple times.

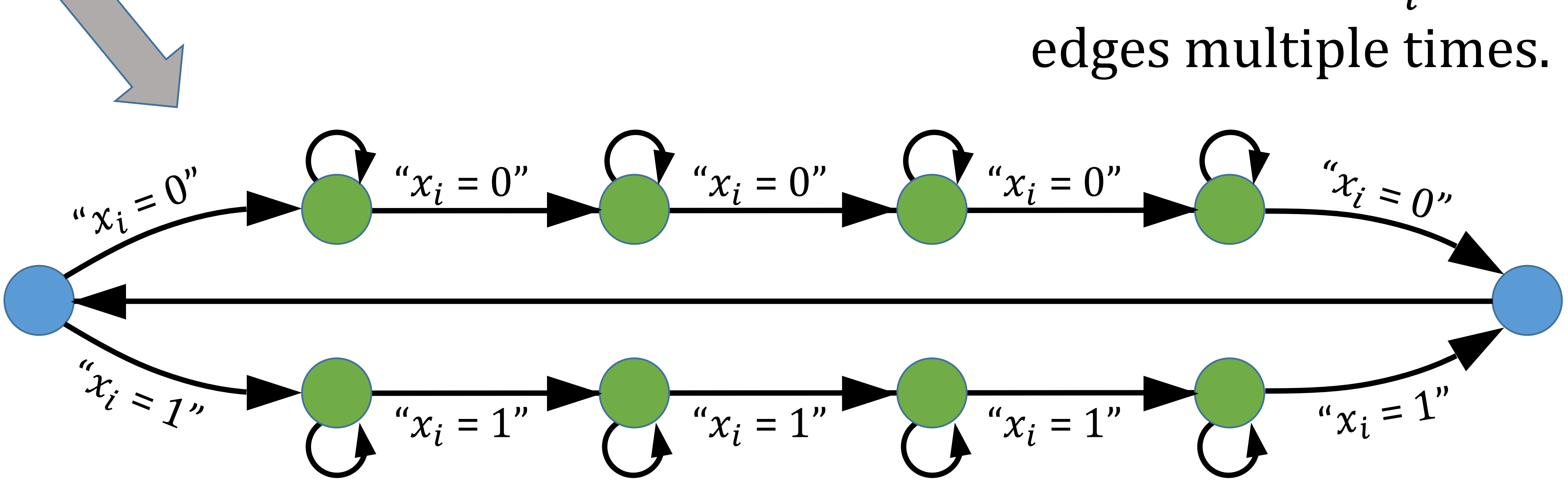
Main Reduction

#3SAT$^{\text{bal}}$ → {−1,0,1}-weighted digraph

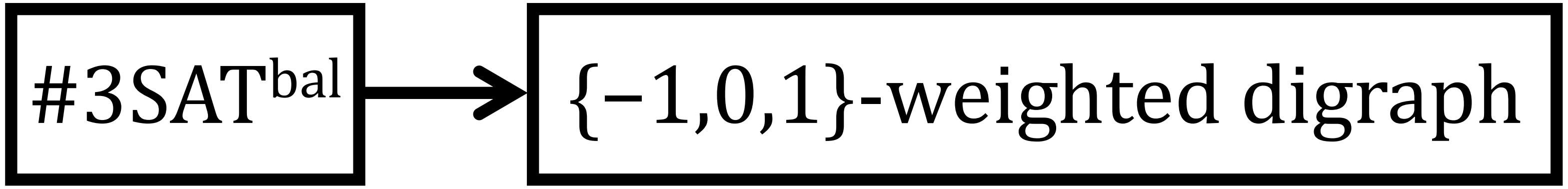


Wrinkle: Each variable appears in multiple clauses.

Idea: **Subdivide** “$x_i = b$” edges multiple times.



Main Reduction

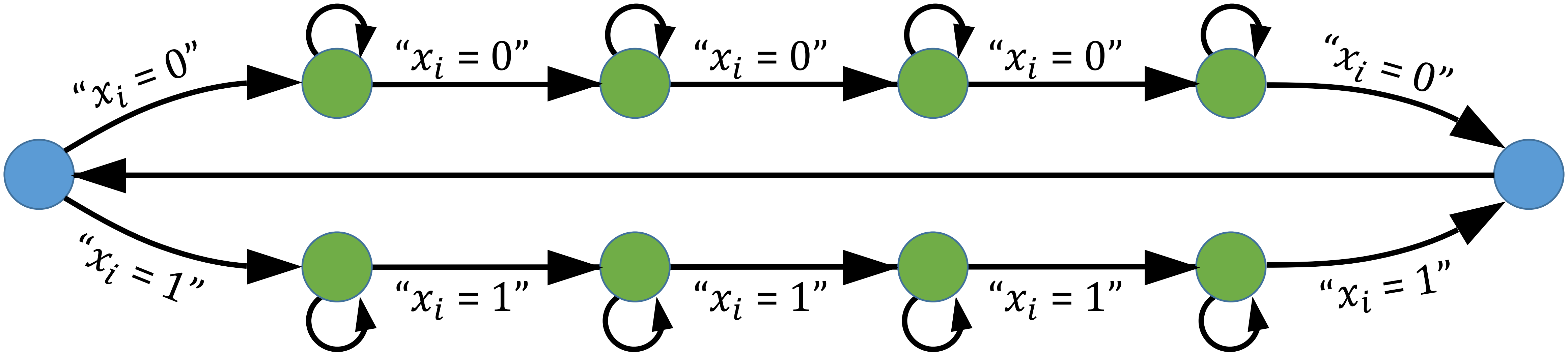


#3SAT$^{\text{bal}}$ → {−1,0,1}-weighted digraph

The “$x_i = 0$” edges are now **perfectly identified**:

Every cycle cover uses either all of them, or none of them.

Similarly for the “$x_i = 1$” edges.

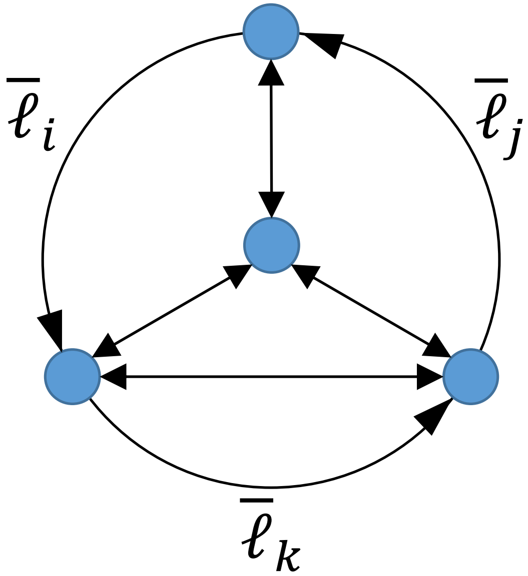


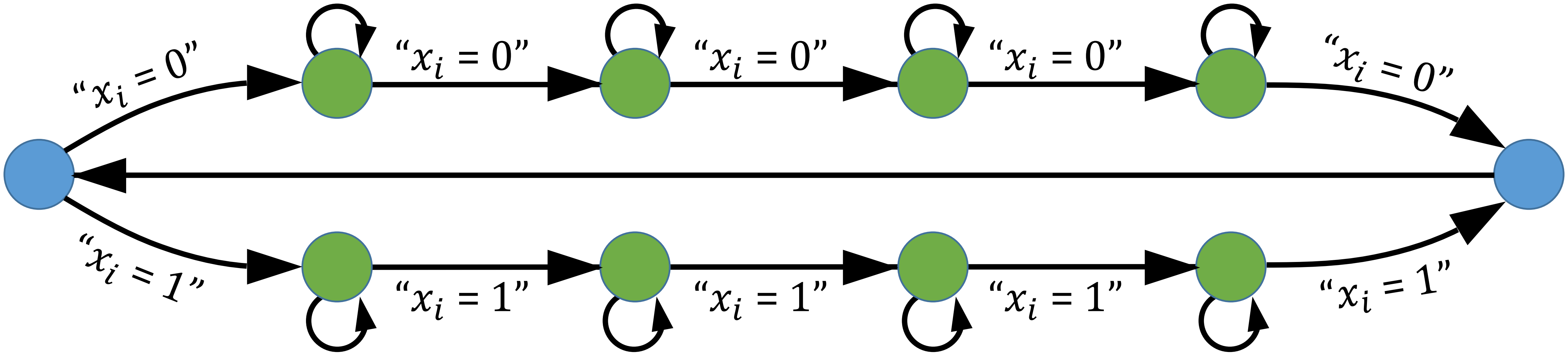
Main Reduction

$\#3\mathrm{SAT}^{\mathrm{bal}} \rightarrow \{-1,0,1\}$-weighted digraph

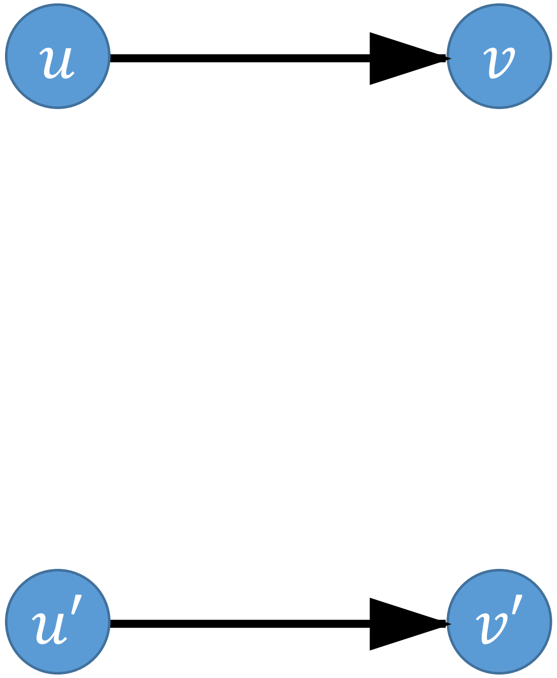
Wrinkle fixed:

For each variable/clause occurrence, we can try to identify a pair of edges.





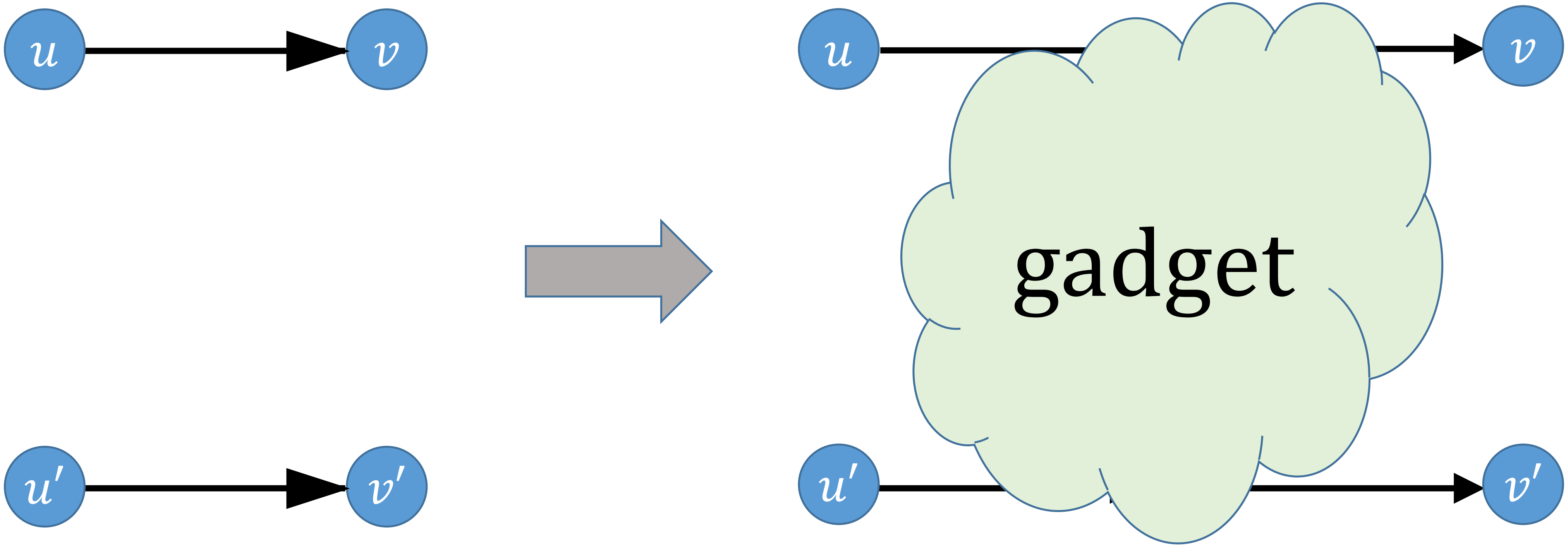
Last Step: Trying to identify a pair of edges



We will not be able to perfectly identify them. ☹

But we'll see a gadget that does something that's good enough. ☺

Last Step: Trying to identify a pair of edges



Old graph

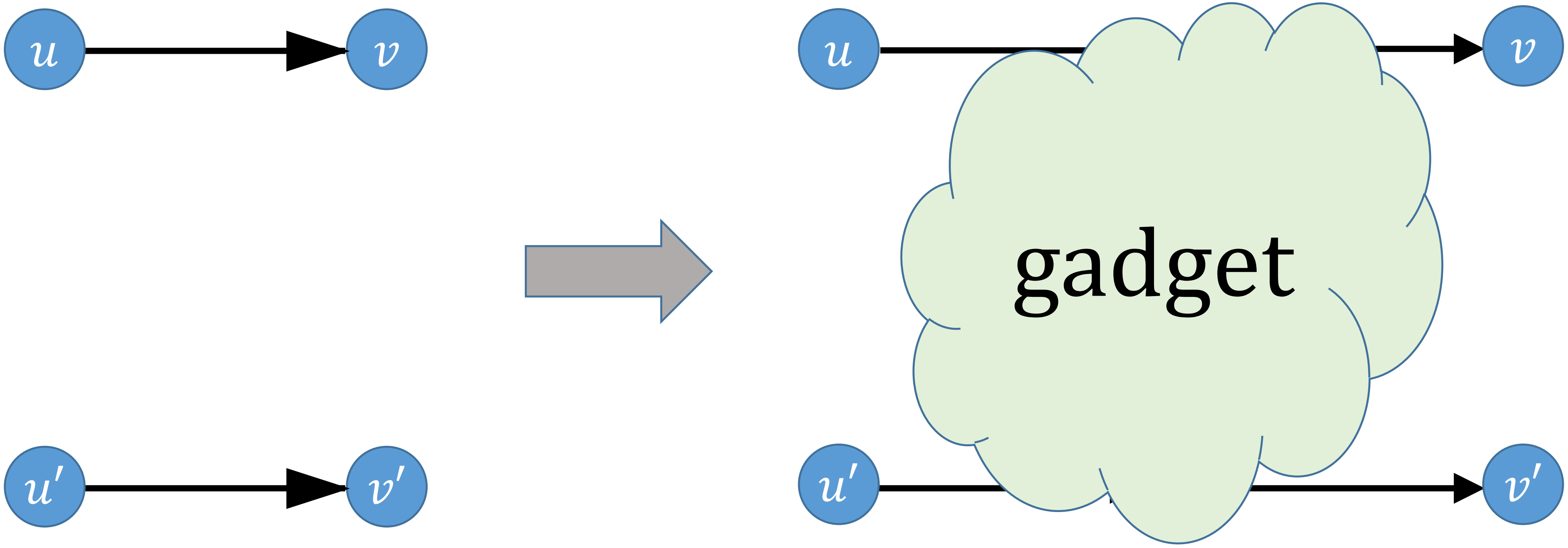
New graph

Identification gadget properties:

(a) To every "old" cycle cover of weight W using **both** edges, there corresponds a "new" cycle cover of weight $-W$.

(b) To every "old" cycle cover of weight W using **neither** edge, there correspond several "new" cycle covers of weight $2W$.

(c) All remaining other "new" cycle covers have total weight **zero**.

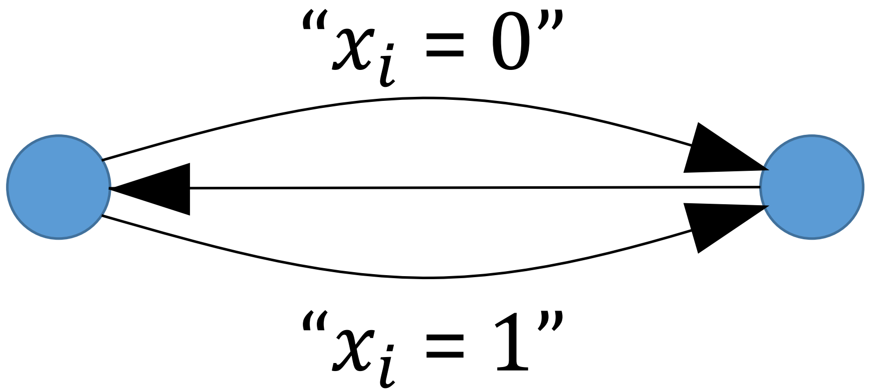


Old graph   New graph

Main Reduction

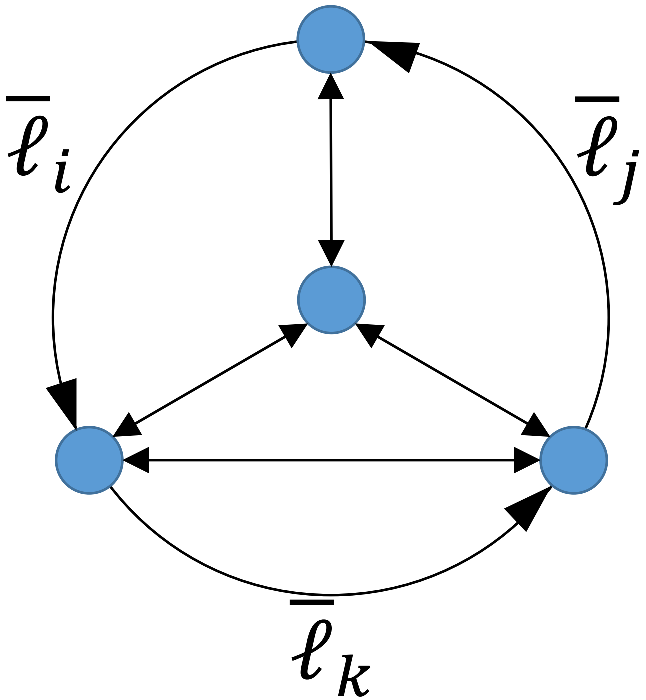
$\#3\text{SAT}^{\text{bal}} \rightarrow \{-1,0,1\}$-weighted digraph

We now “just” need
to enforce **consistency**.



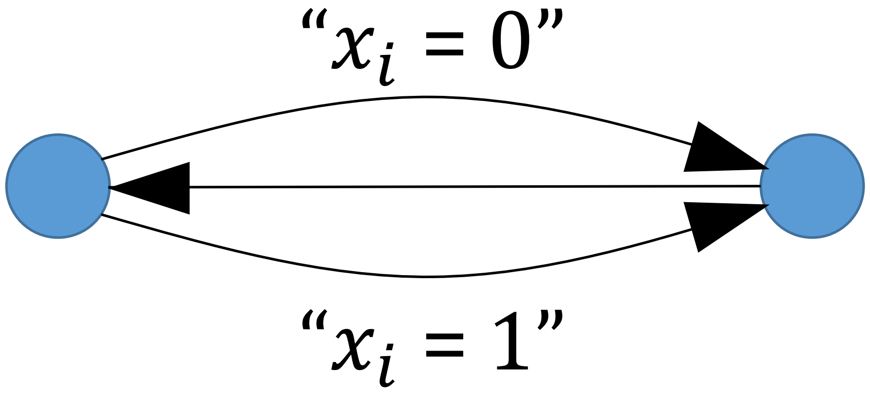
We wish to ***identify***
variable-edge “$x_i = 0$”
with all clause-edges labeled $\overline{x}_i$,

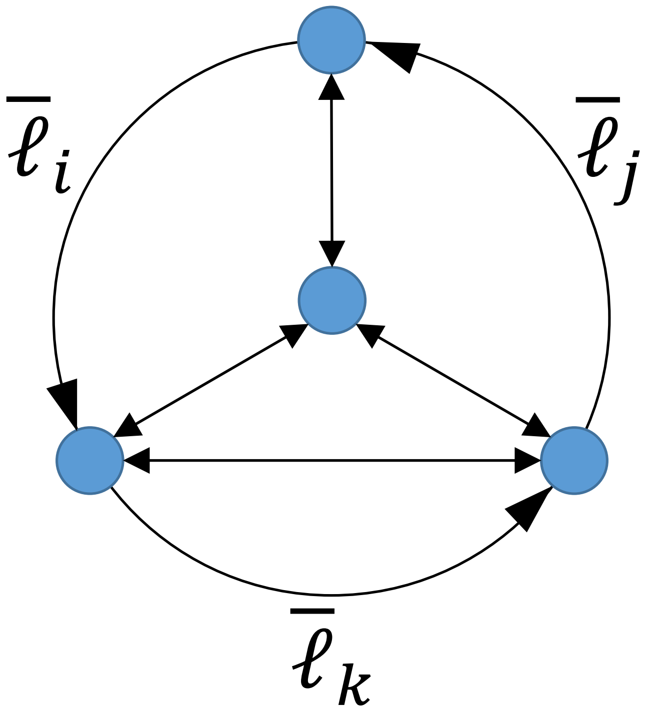
and similarly for “$x_i = 1$” and
all clause-edges labeled x_i.



Main Reduction

#3SAT$^{\text{bal}}$ → {−1,0,1}-weighted digraph





Define edges e, e' to be **perfectly identified** if:

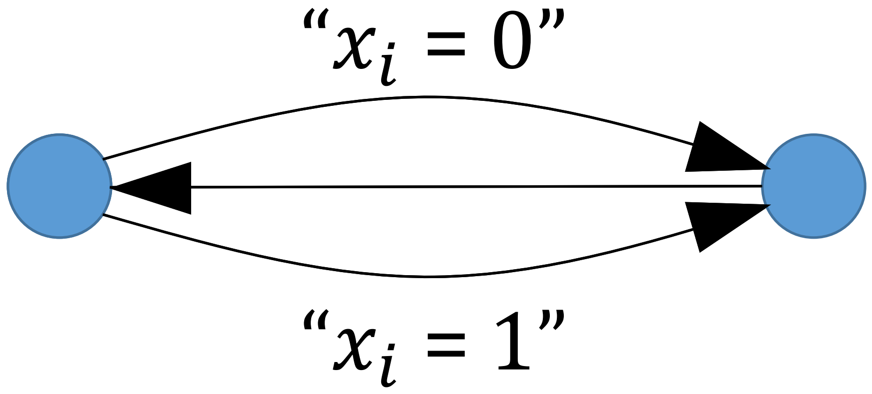
For every cycle cover:
it contains e iff it contains e'.

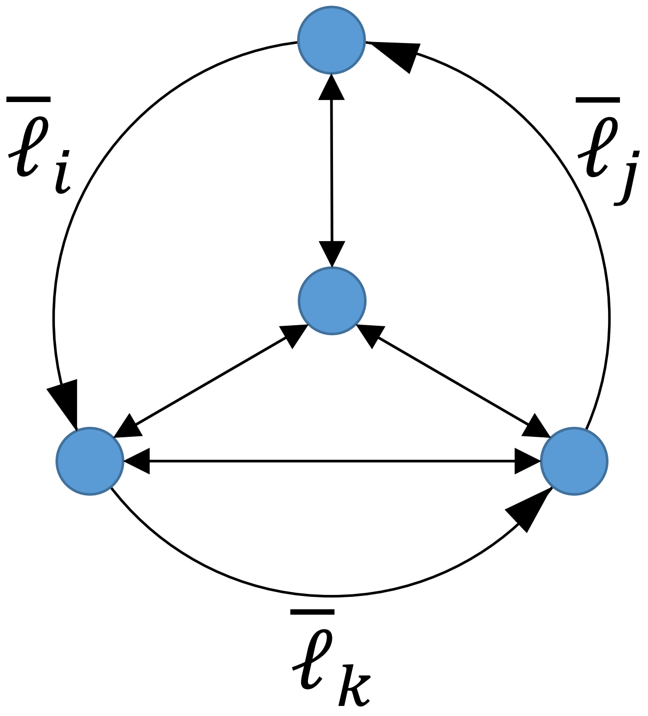
If we could perfectly identify pairs of edges, we’d be done:

\# satisfying assignments
= total weight of cycle covers.

Main Reduction

$\#3\mathrm{SAT}^{\mathrm{bal}}$ → $\{-1,0,1\}$-weighted digraph





Using Identification gadget instead…

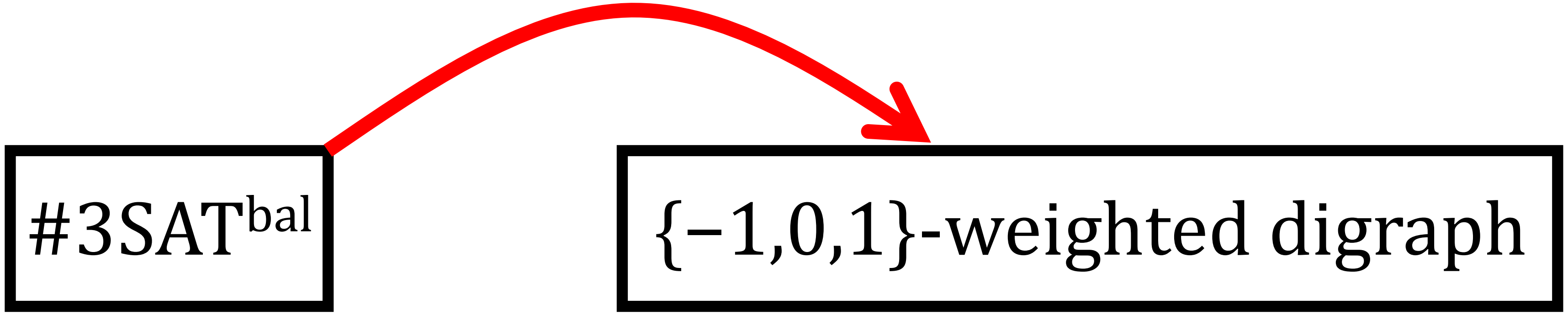
(a) To every “old” cycle cover of weight W using **both** edges, there corresponds a “new” cycle cover of weight $-W$.

(b) To every “old” cycle cover of weight W using **neither** edge, there correspond several “new” cycle covers of weight $2W$.

(c) All remaining other “new” cycle covers have total weight **zero**.

For each satisfying assignment of ϕ, we get new total cycle cover weight equal to $(-1)^{\#\text{false literals}} \cdot (2)^{\#\text{true literals}}$

Thus the Identification gadget completes the reduction!



#3SAT$^{\mathrm{bal}}$

$\{-1,0,1\}$-weighted digraph

ϕ has $\boldsymbol{m}$ clauses,
$\boldsymbol{C}$ satisfying
assignments

Total cycle cover weight: $(-\mathbf{2})^{\mathbf{3}\boldsymbol{m}/\mathbf{2}} \cdot \boldsymbol{C}$

Each satisfying assignment making
$\boldsymbol{p}$ literals false and $\boldsymbol{q}$ literals true
yields cycle covers of weight $(-\mathbf{1})^{\boldsymbol{p}}\mathbf{2}^{\boldsymbol{q}}$.

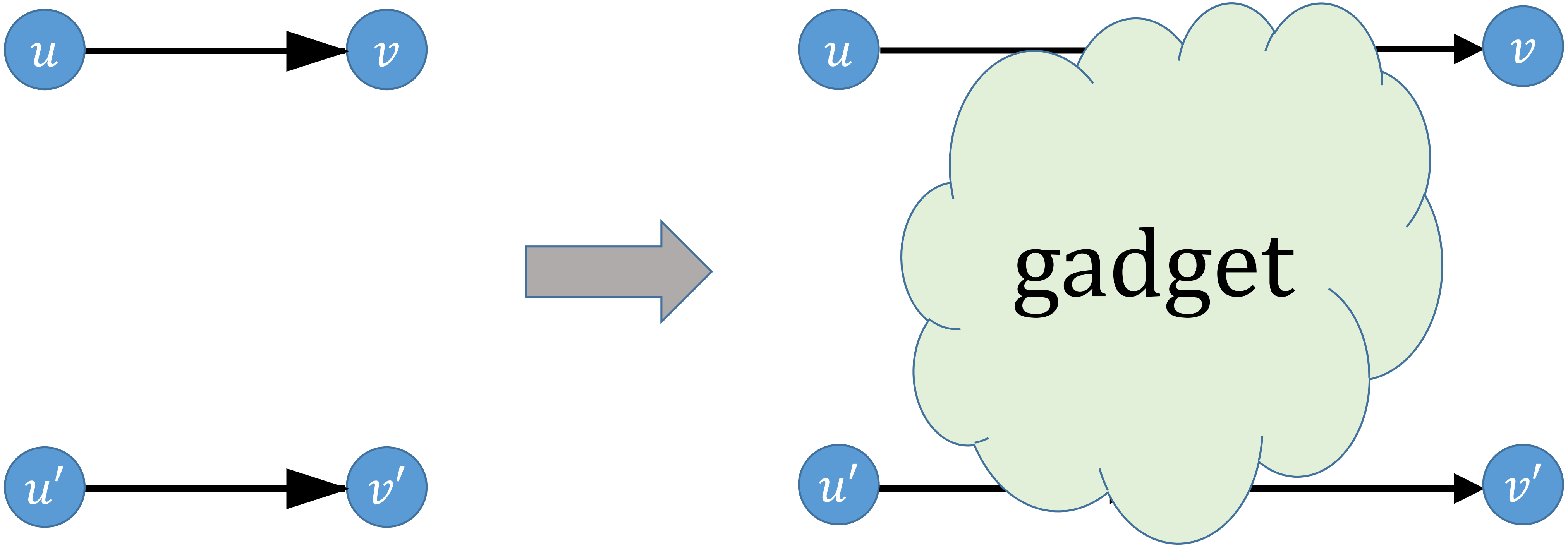
#3SAT$^{\mathrm{bal}}$ → If ϕ has $\boldsymbol{m}$ clauses, then for every assignment,
we get $\mathbf{3}\boldsymbol{m}/\mathbf{2}$ false literals and $\mathbf{3}\boldsymbol{m}/\mathbf{2}$ true literals.

Identification gadget properties:

(a) To every "old" cycle cover of weight W using **both** edges, there corresponds a "new" cycle cover of weight $-W$.

(b) To every "old" cycle cover of weight W using **neither** edge, there correspond several "new" cycle covers of weight $2W$.

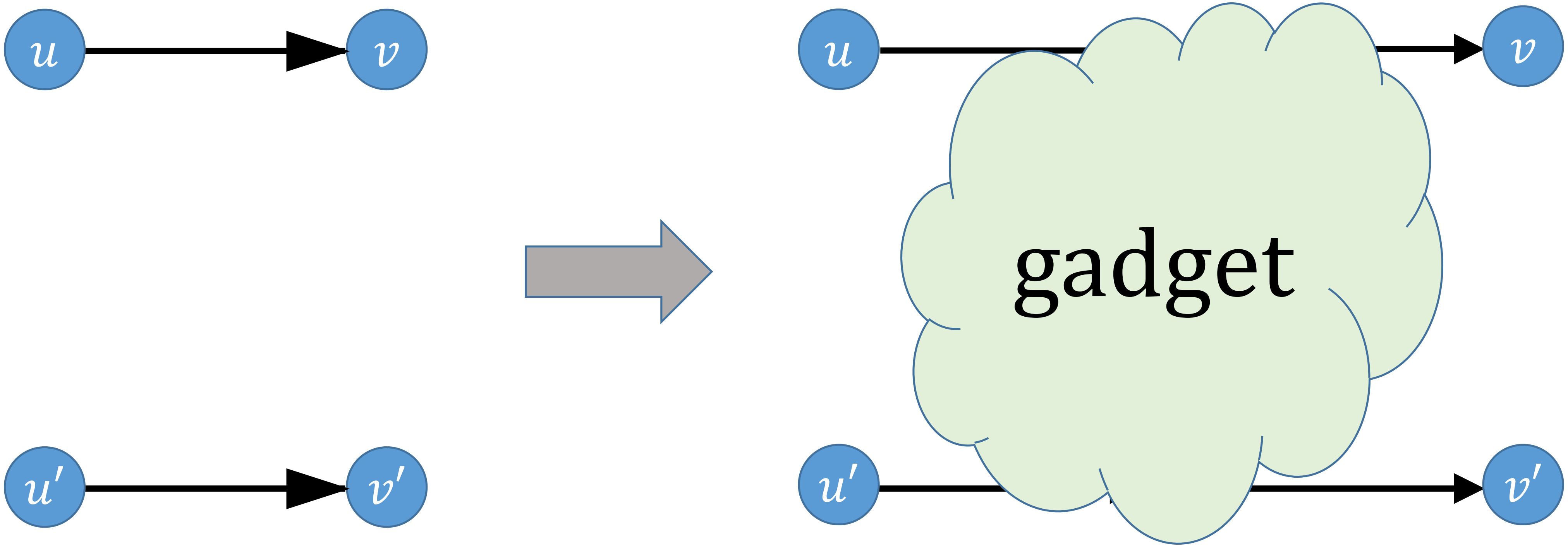
(c) All remaining other "new" cycle covers have total weight **zero**.



Old graph

New graph

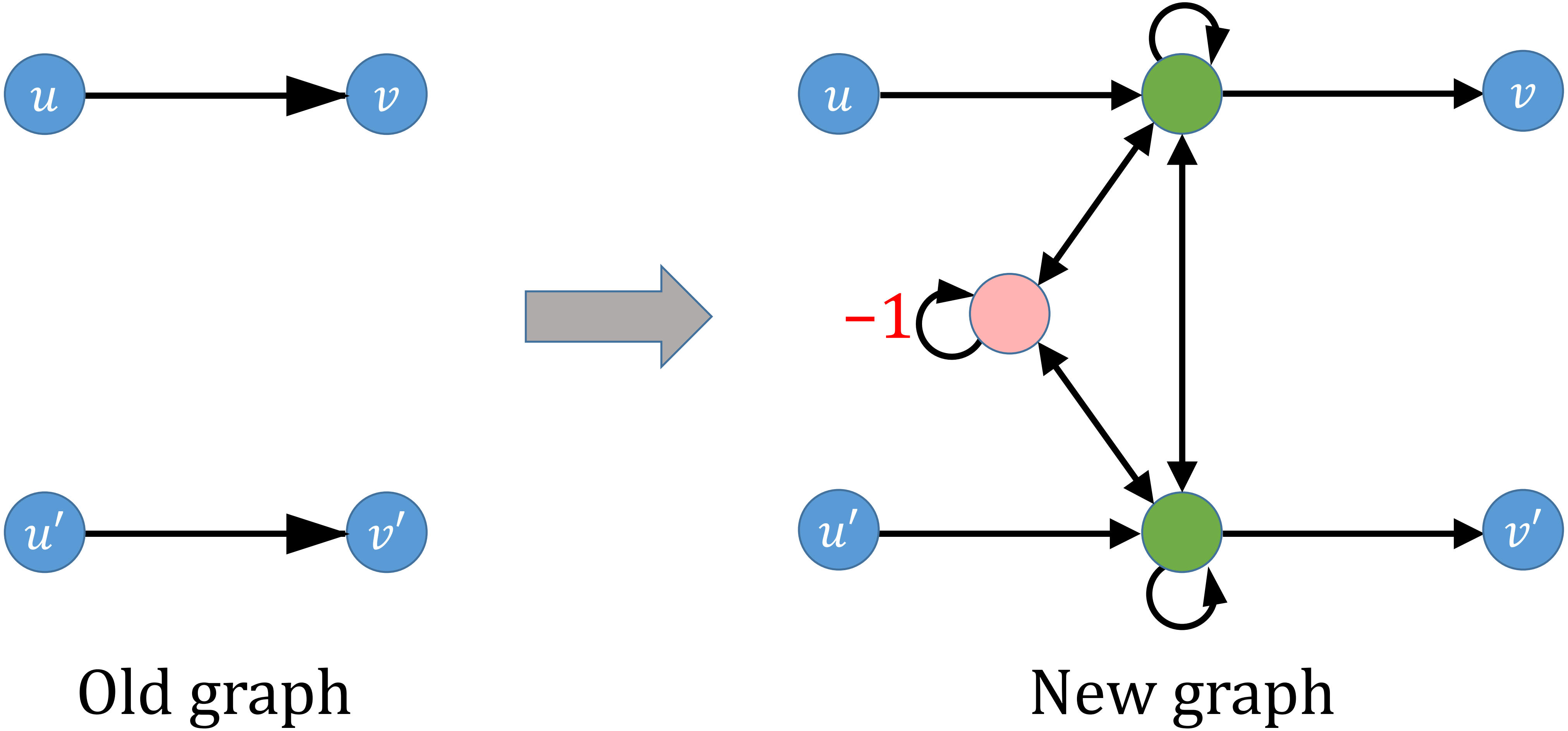
Identification gadget revealed



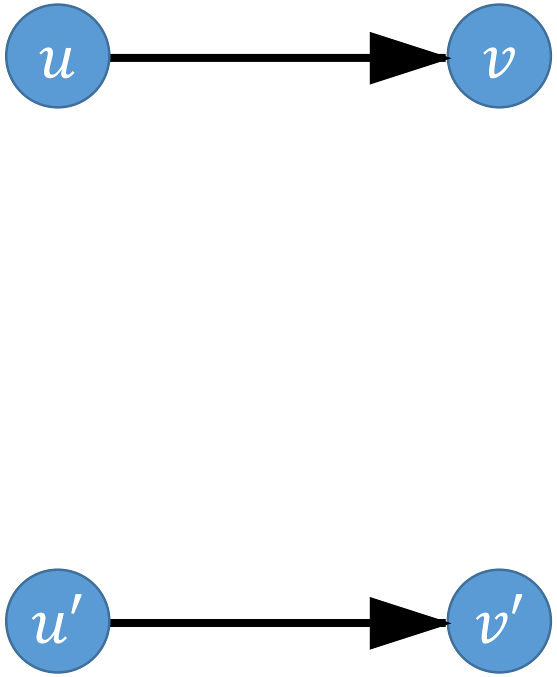
Old graph

New graph

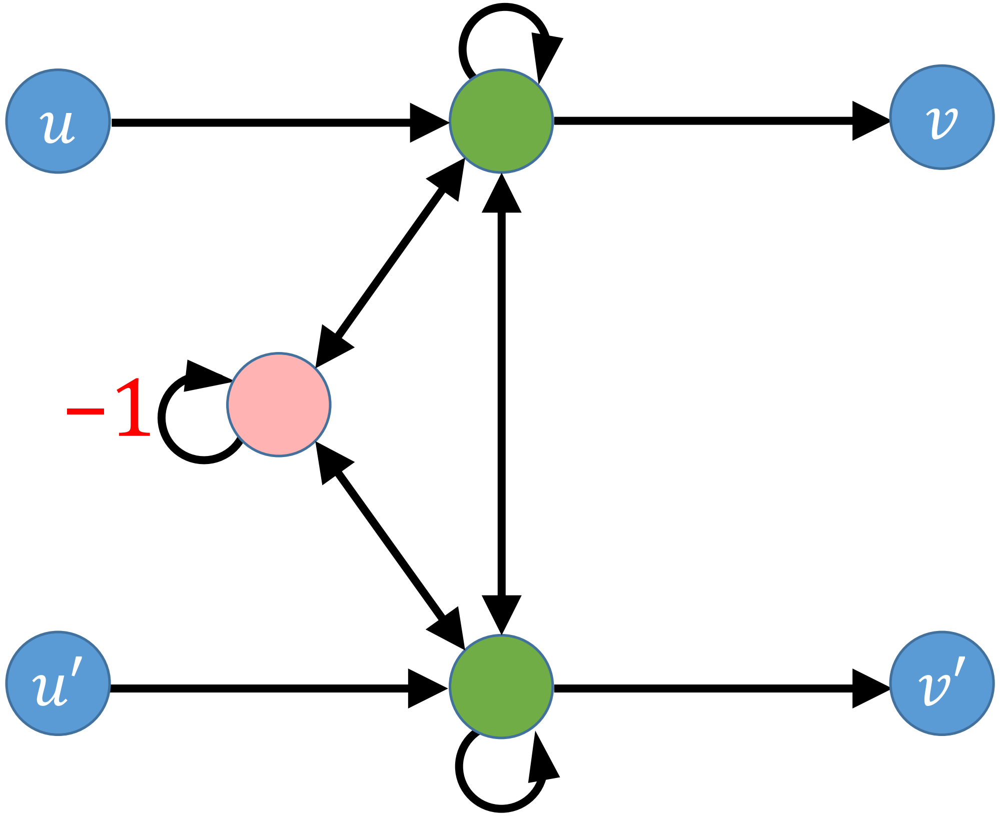
Identification gadget revealed



We must study all possible cycle covers in the new graph.



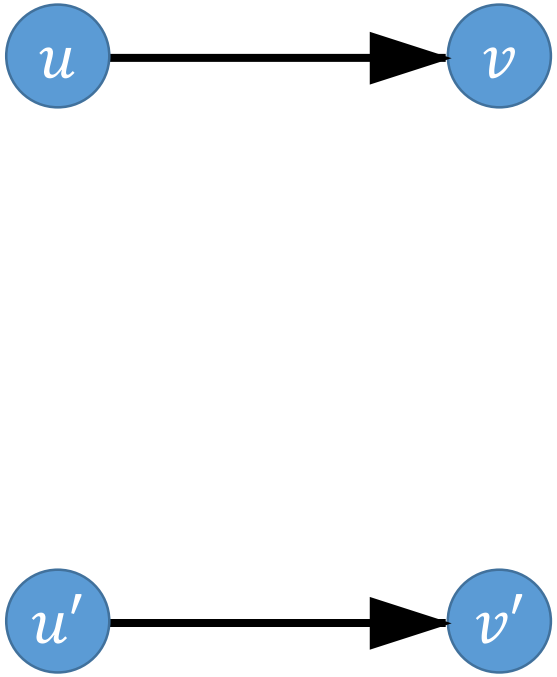
Old graph



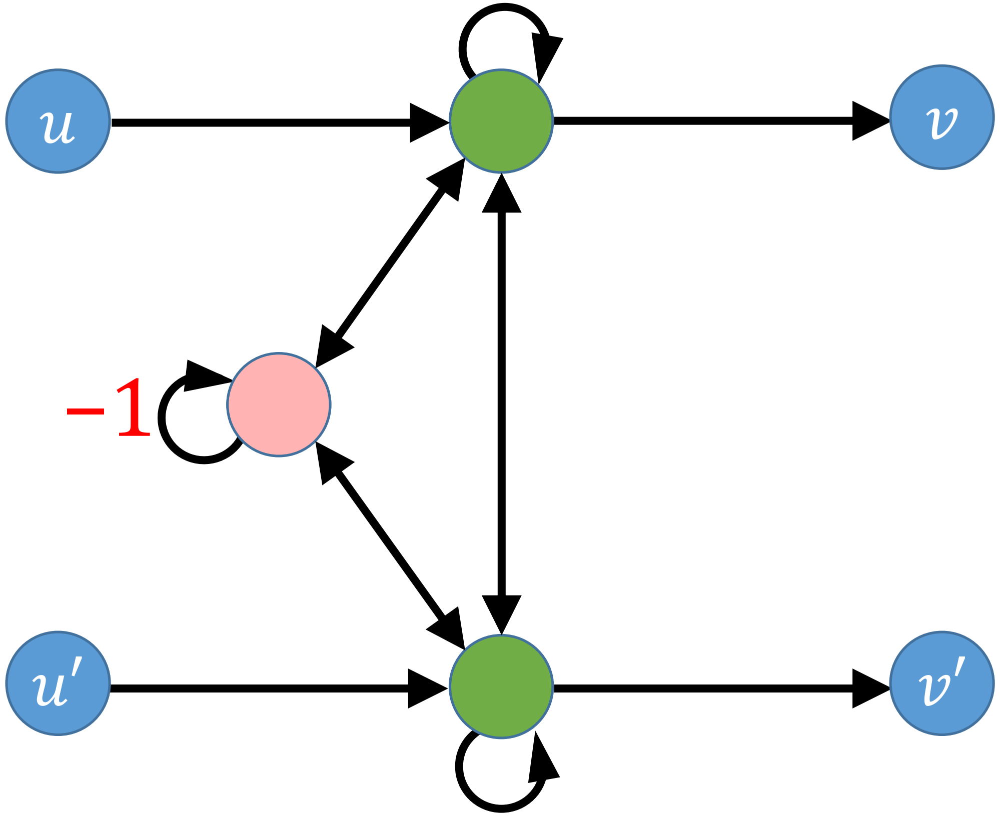
New graph

We must study all possible cycle covers in the new graph.

Divide into cases based on which of the wobbling edges are taken.



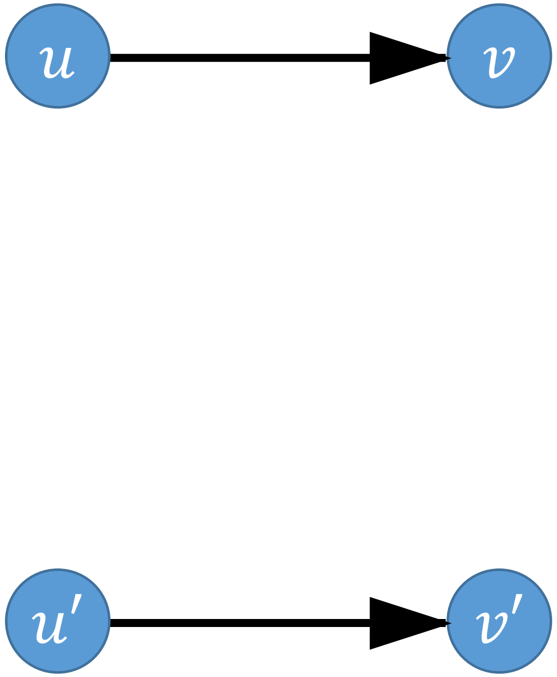
Old graph



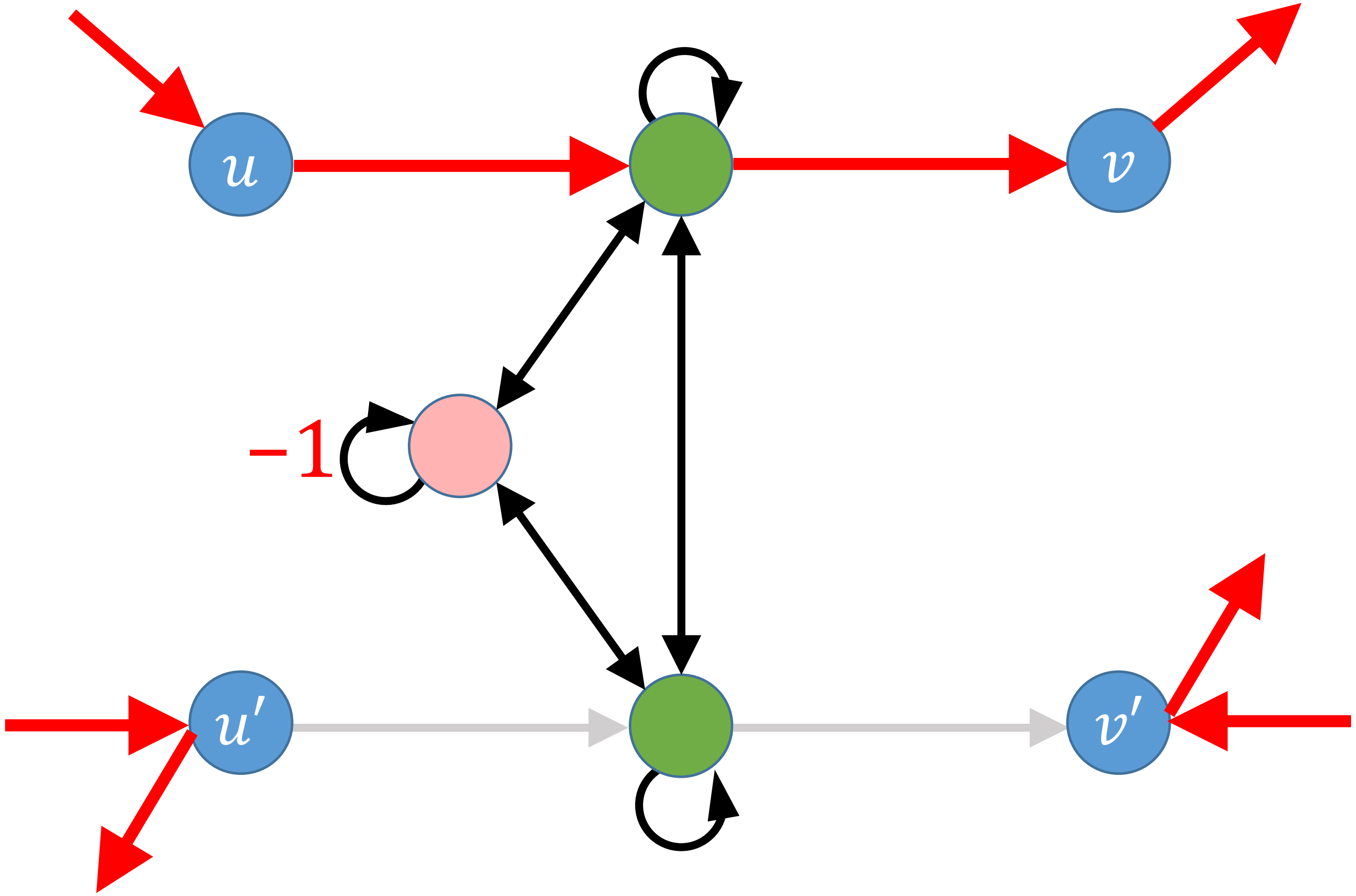
New graph

One case: *Both top edges, neither bottom edge*

What must the rest of the cycle cover in the new graph look like?



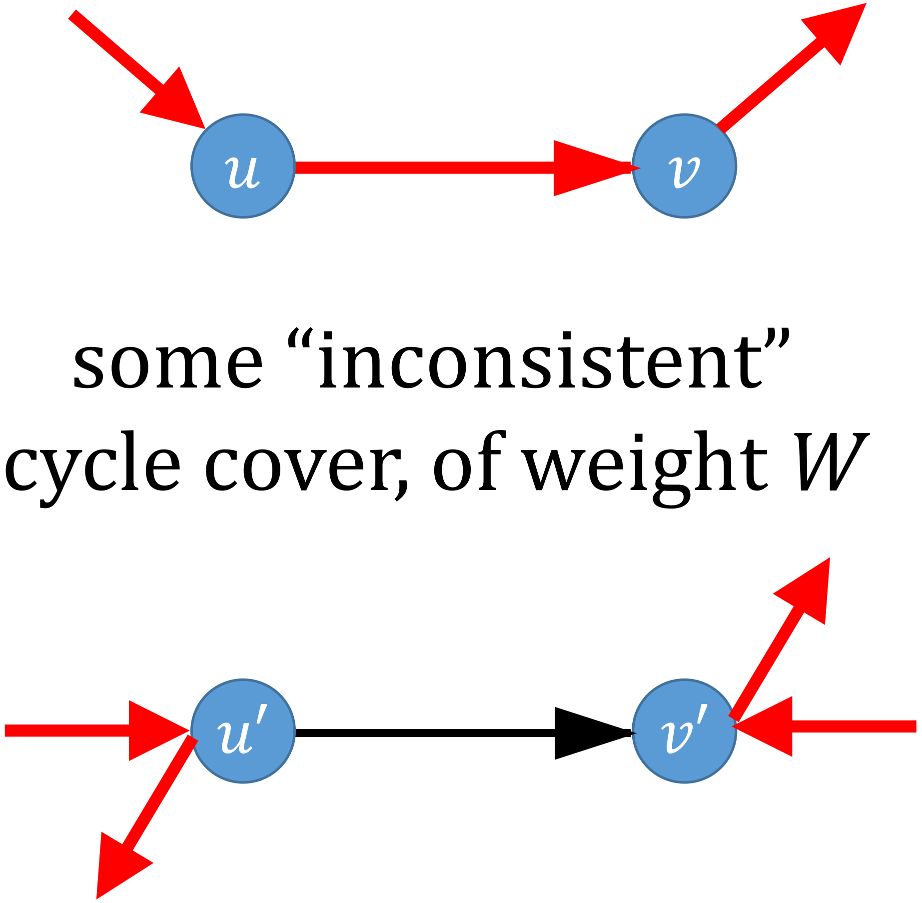
Old graph



New graph

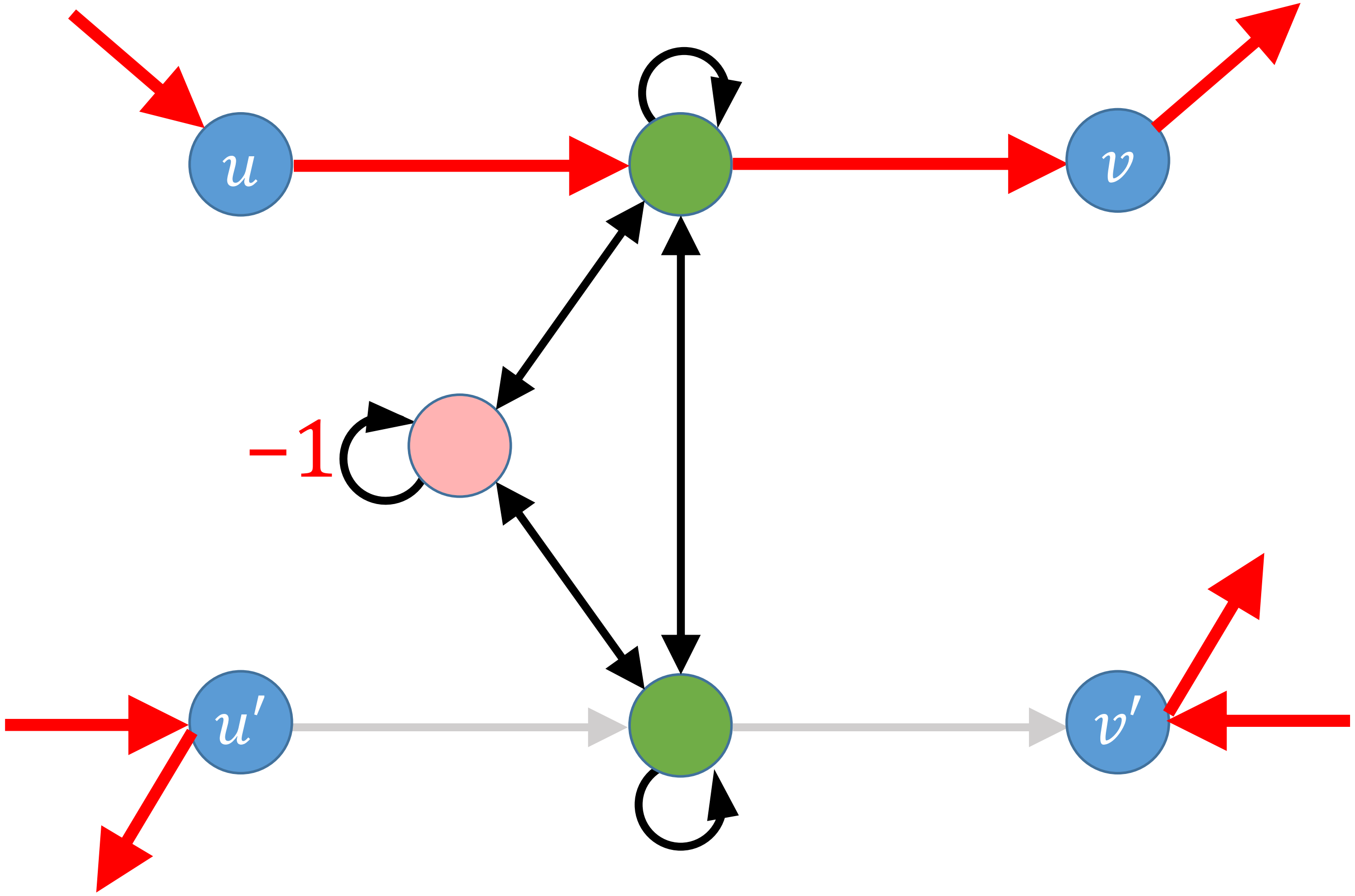
One case: *Both top edges, neither bottom edge*

What must the rest of the cycle cover in the new graph look like?



some “inconsistent” cycle cover, of weight W

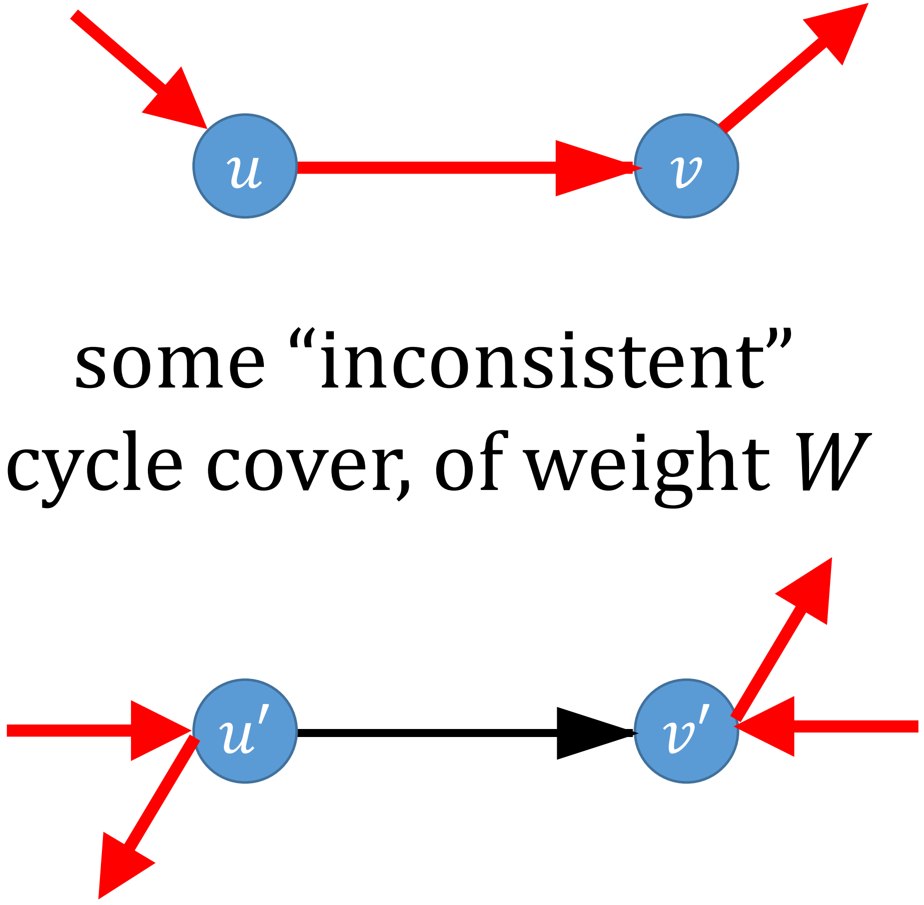
Old graph



New graph

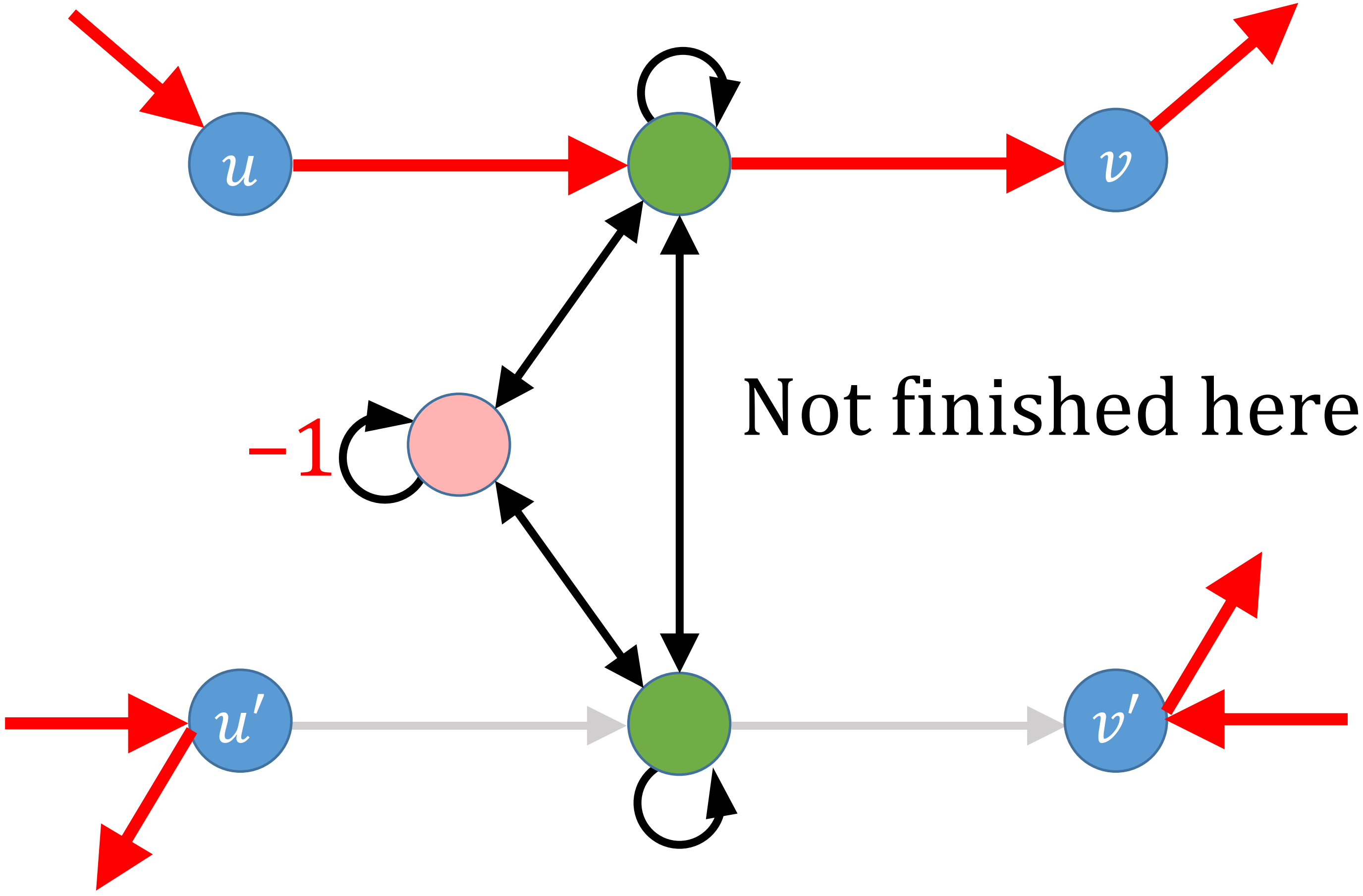
One case: *Both top edges, neither bottom edge*

What must the rest of the cycle cover in the new graph look like?



some “inconsistent” cycle cover, of weight W

Old graph

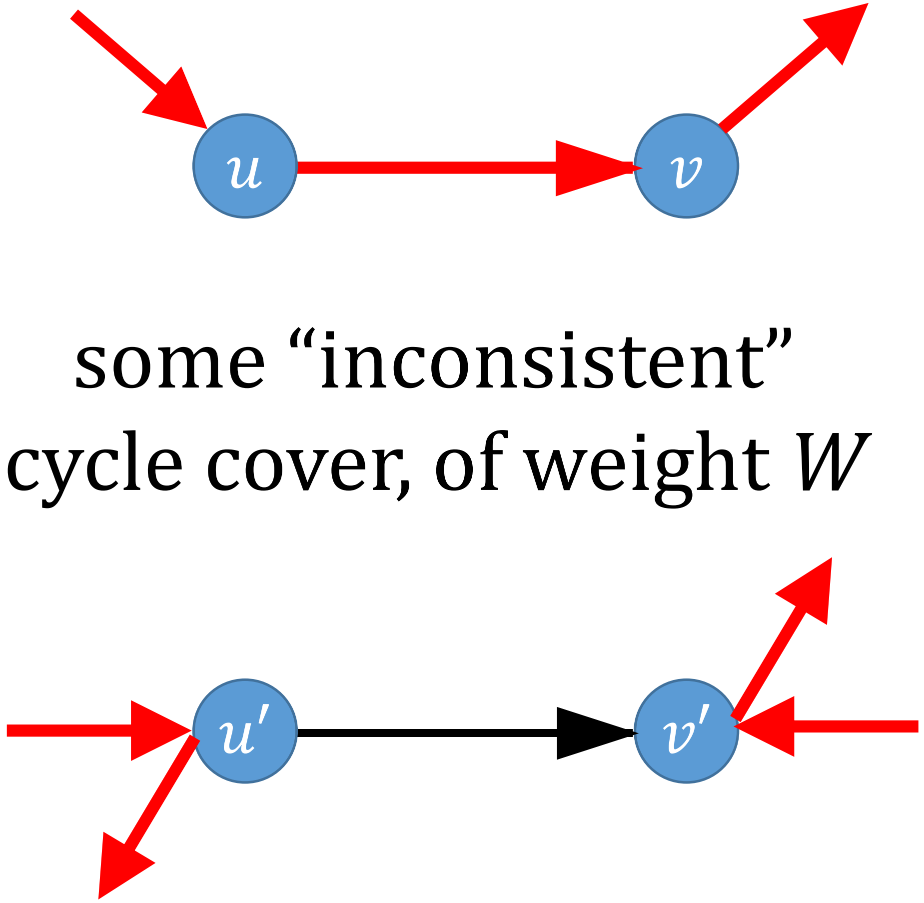


Not finished here!

New graph

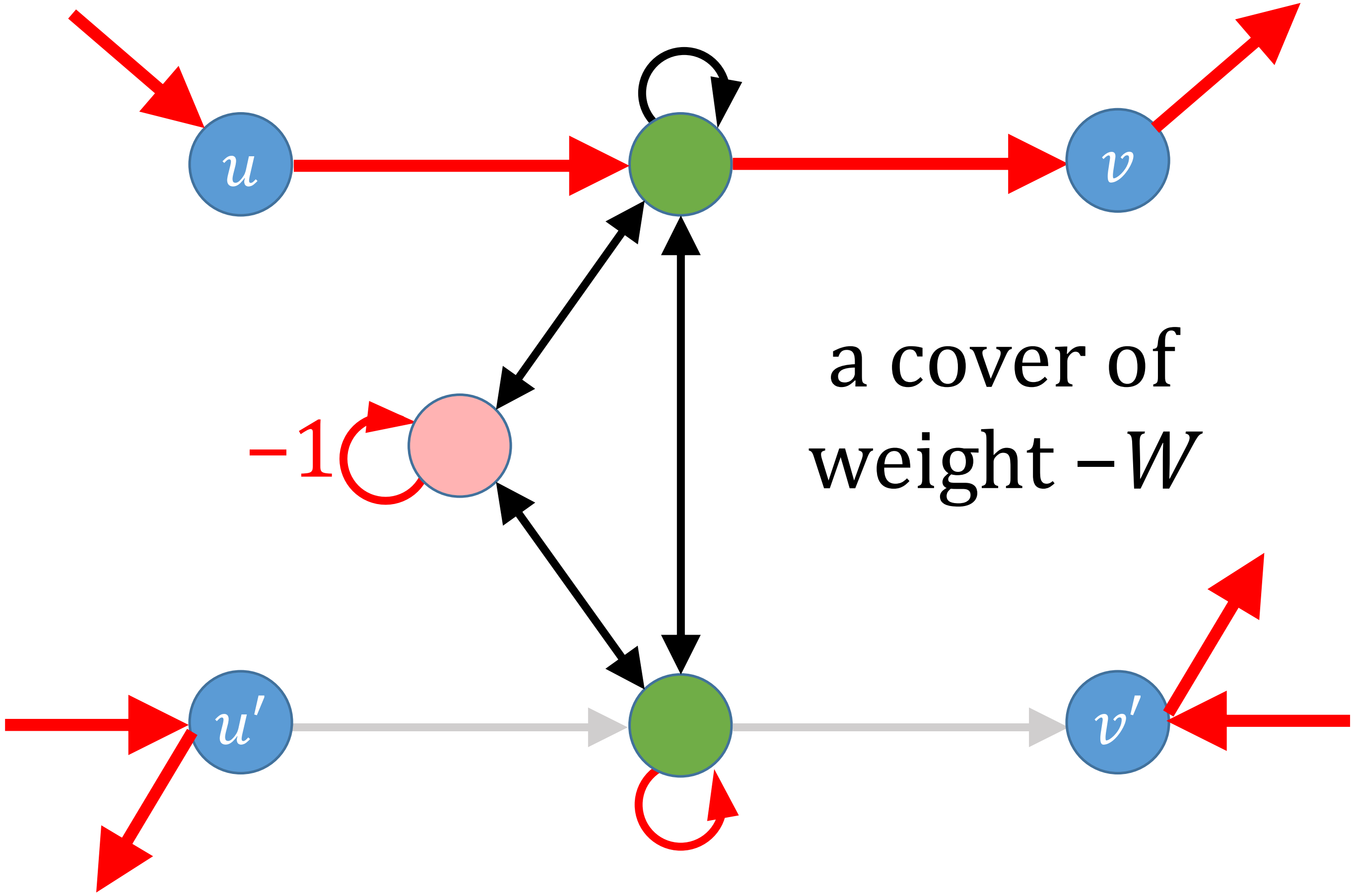
One case: *Both top edges, neither bottom edge*

What must the rest of the cycle cover in the new graph look like?



some “inconsistent” cycle cover, of weight W

Old graph

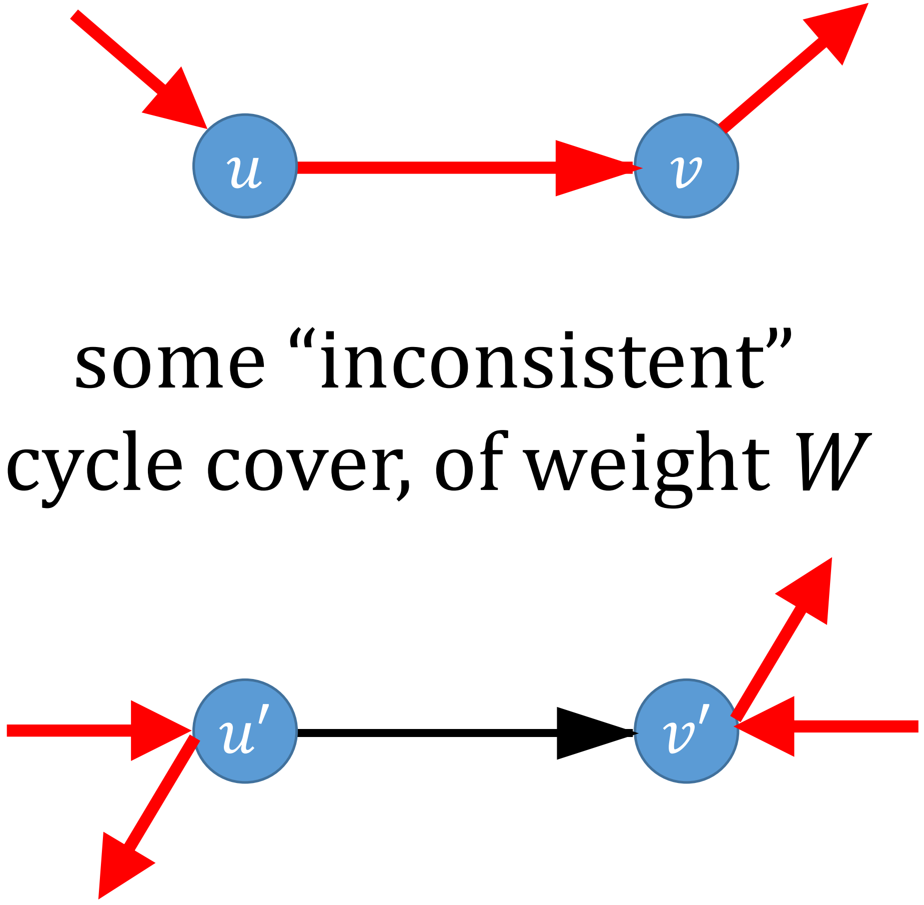


a cover of weight $-W$

New graph

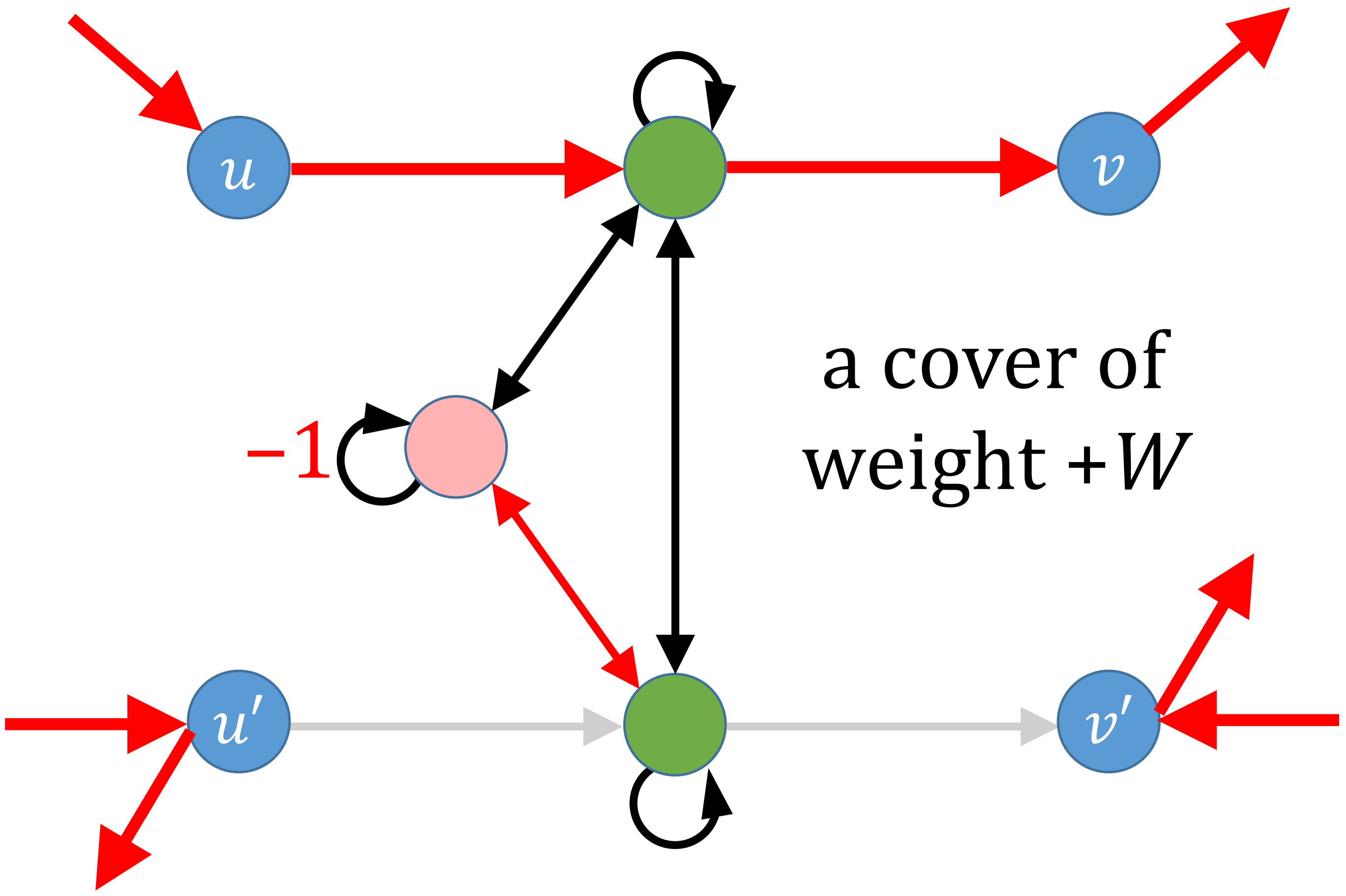
One case: *Both top edges, neither bottom edge*

What must the rest of the cycle cover in the new graph look like?



some “inconsistent” cycle cover, of weight W

Old graph

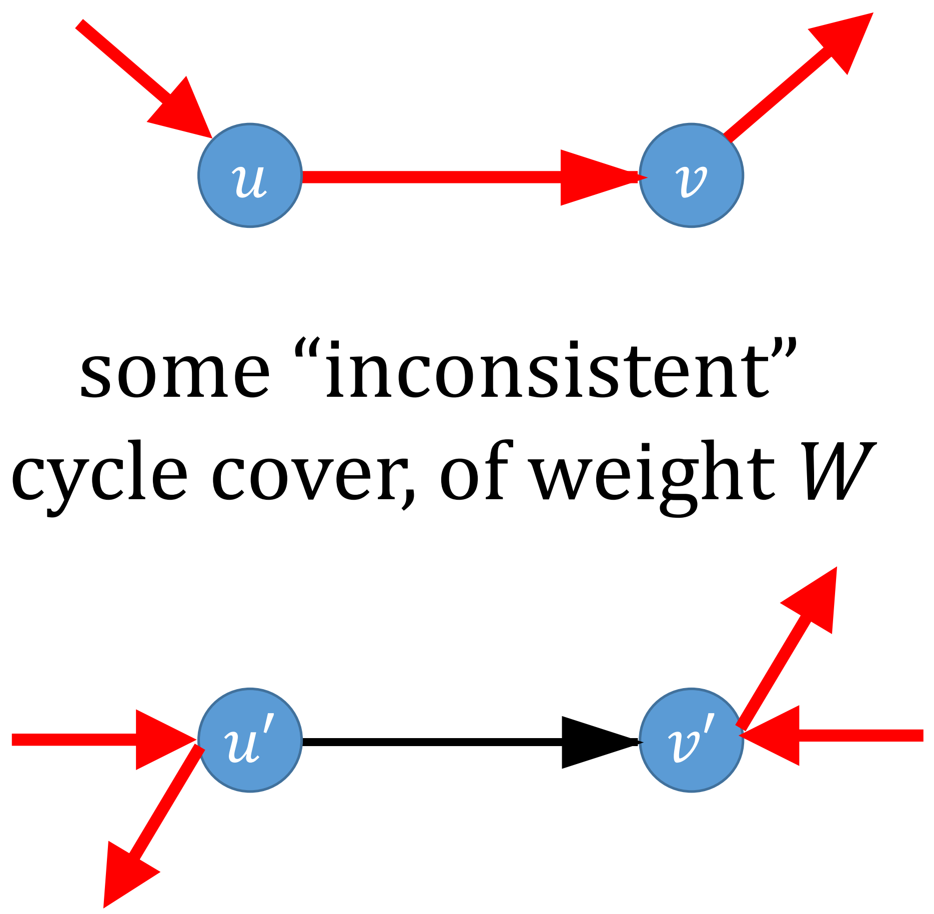


a cover of weight $+W$

New graph

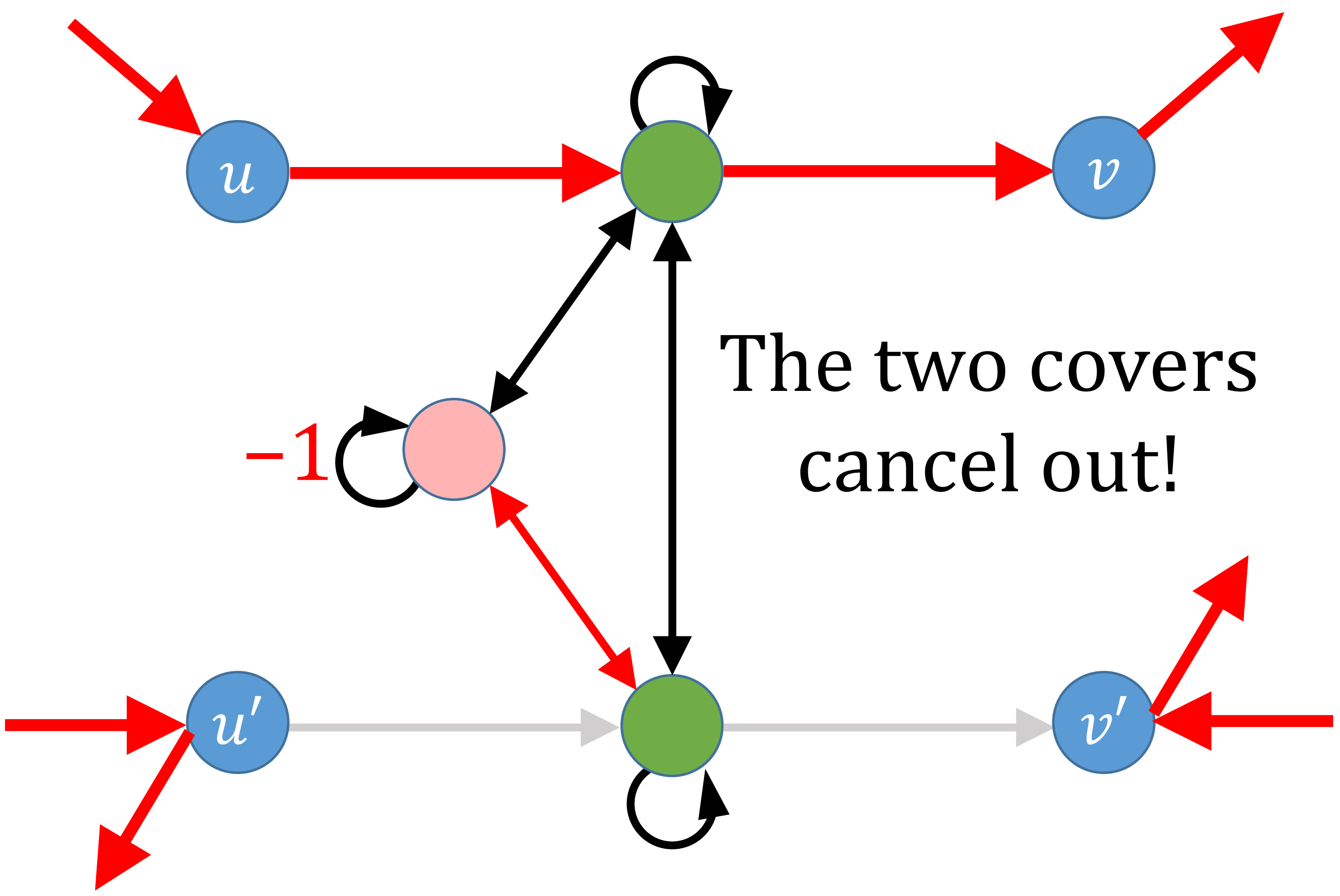
One case: *Both top edges, neither bottom edge*

What must the rest of the cycle cover in the new graph look like?



some “inconsistent” cycle cover, of weight W

Old graph

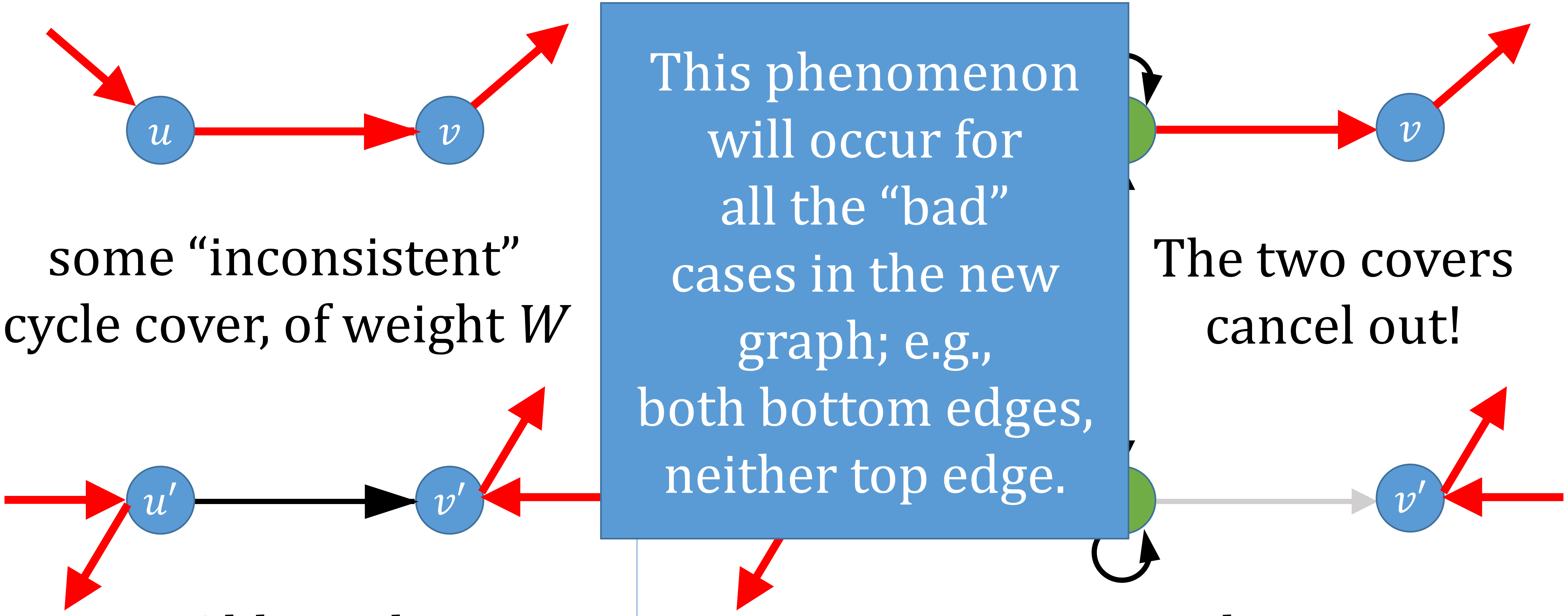


The two covers cancel out!

New graph

One case: *Both top edges, neither bottom edge*

What must the rest of the cycle cover in the new graph look like?



some “inconsistent” cycle cover, of weight W

Old graph

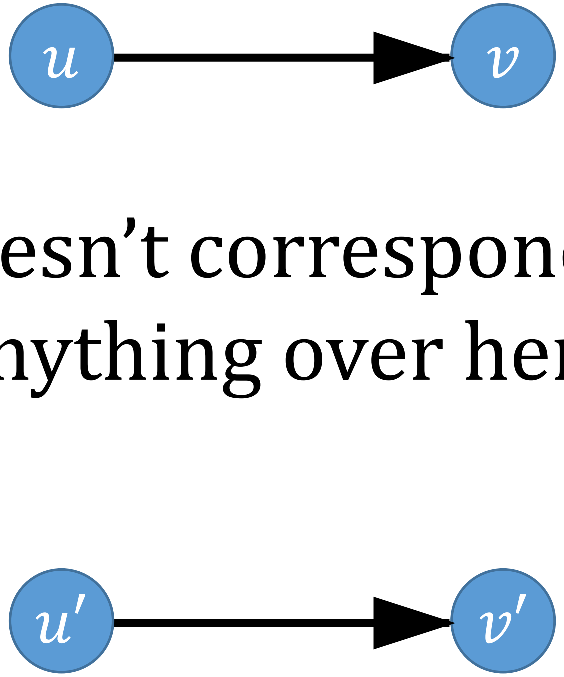
This phenomenon will occur for all the “bad” cases in the new graph; e.g., both bottom edges, neither top edge.

The two covers cancel out!

New graph

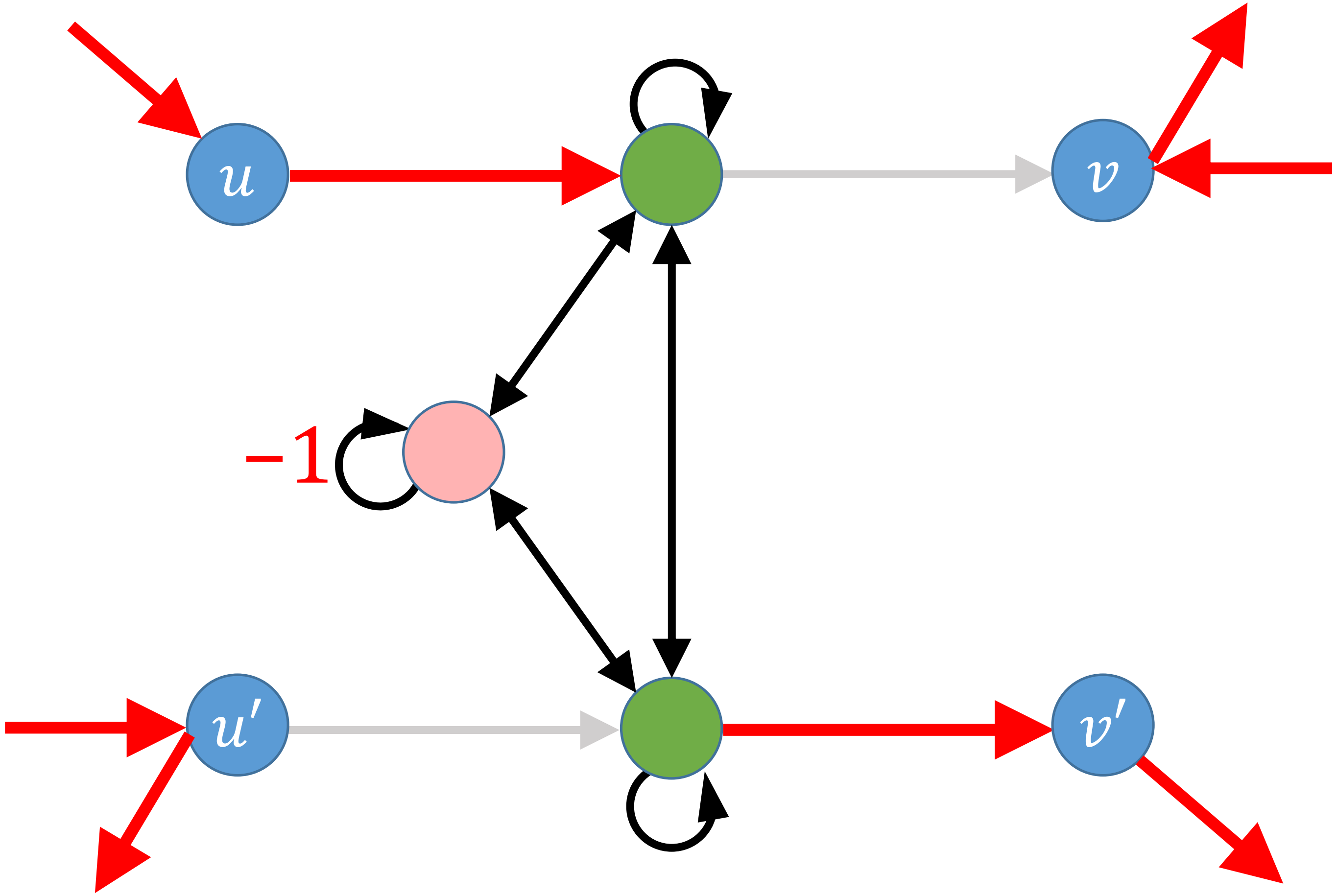
Another bad case: *Top left edge, bottom right edge*

What must the rest of the cycle cover in the new graph look like?



Doesn't correspond to anything over here!

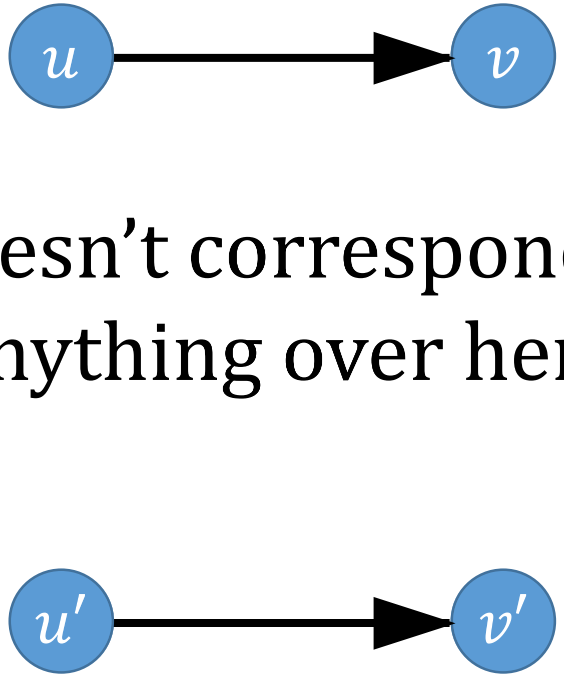
Old graph



New graph

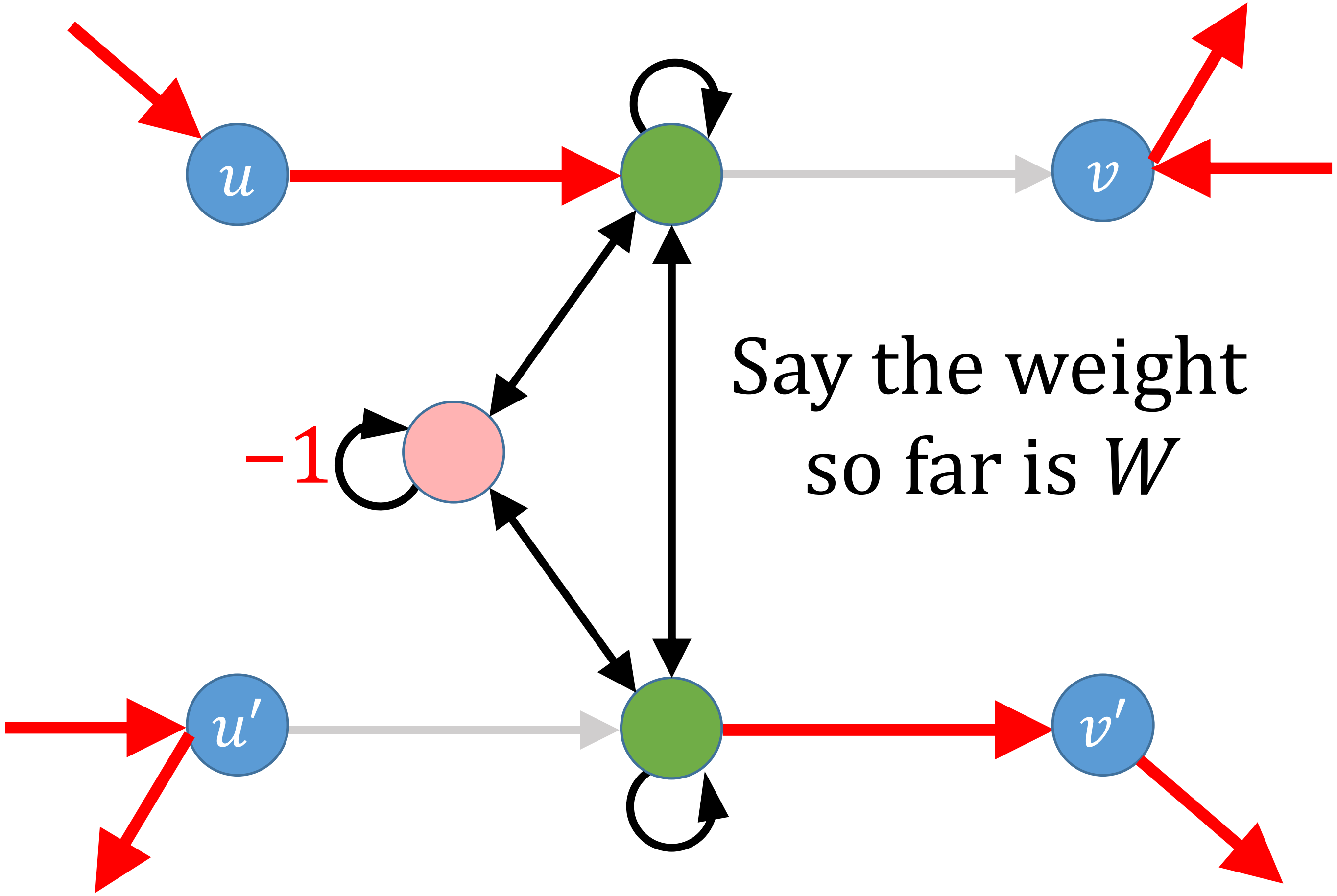
Another bad case: *Top left edge, bottom right edge*

What must the rest of the cycle cover in the new graph look like?



Doesn't correspond to anything over here!

Old graph

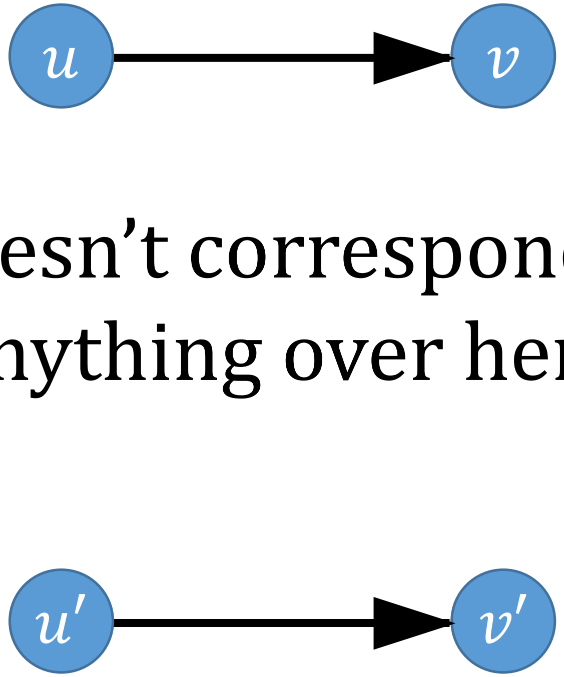


Say the weight so far is W

New graph

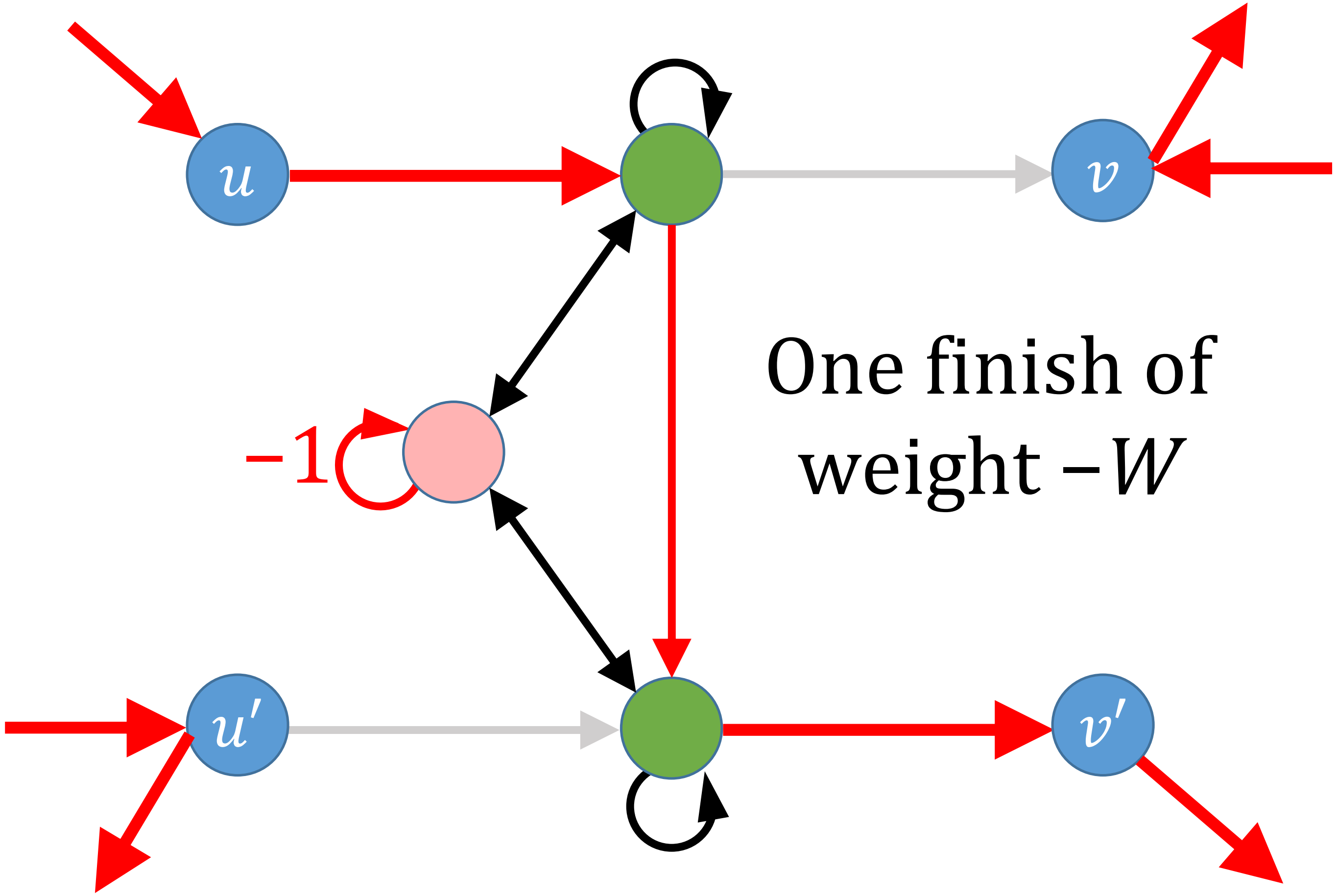
Another bad case: *Top left edge, bottom right edge*

What must the rest of the cycle cover in the new graph look like?



Doesn't correspond to anything over here!

Old graph

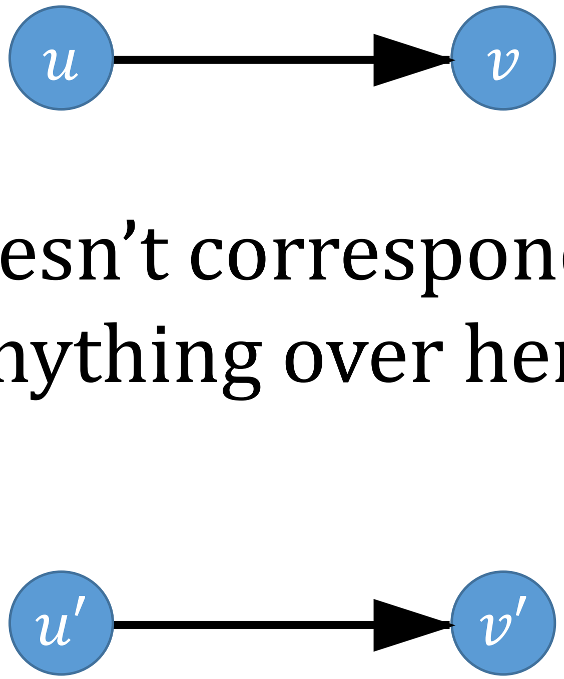


One finish of weight $-W$

New graph

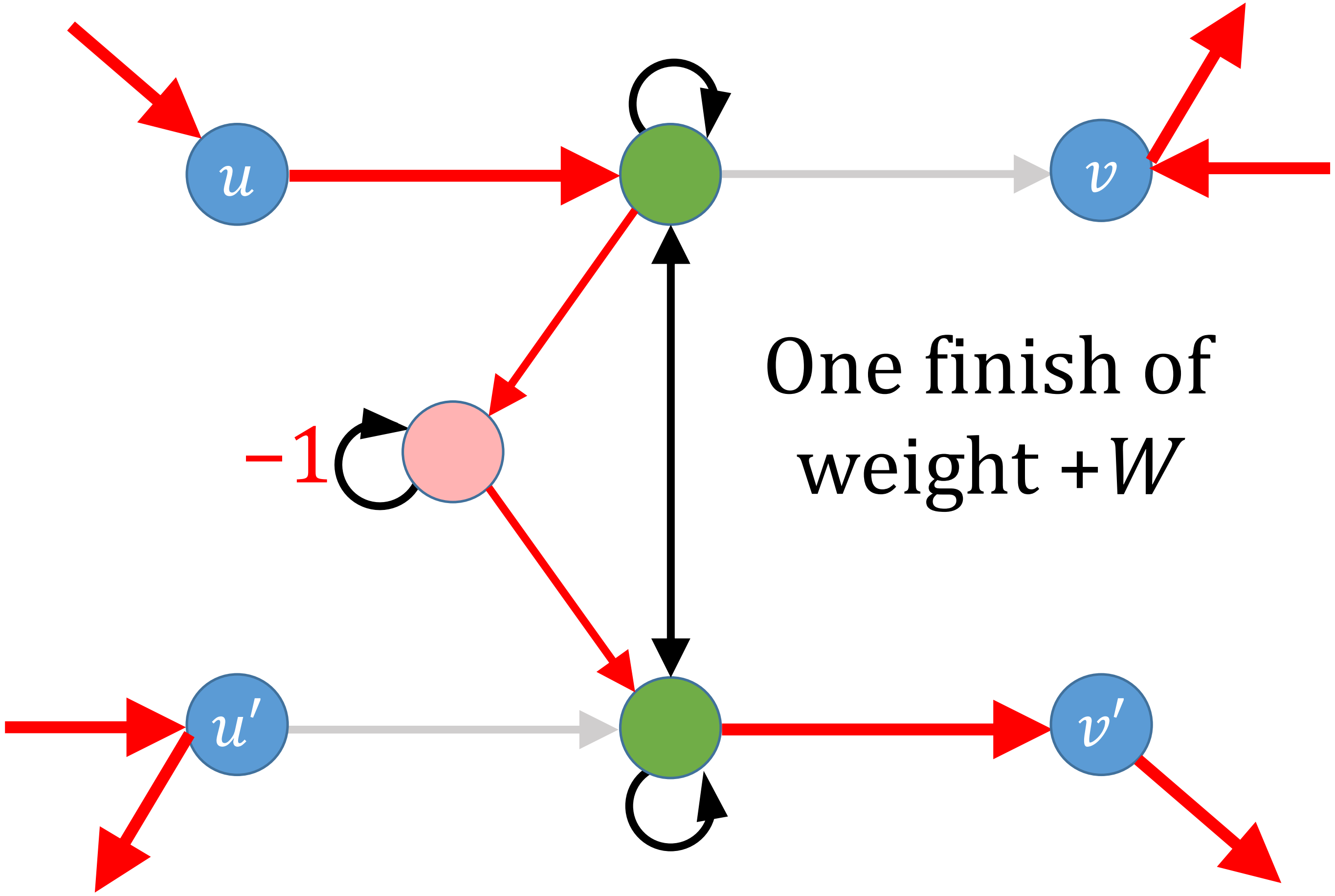
Another bad case: *Top left edge, bottom right edge*

What must the rest of the cycle cover in the new graph look like?



Doesn't correspond to anything over here!

Old graph

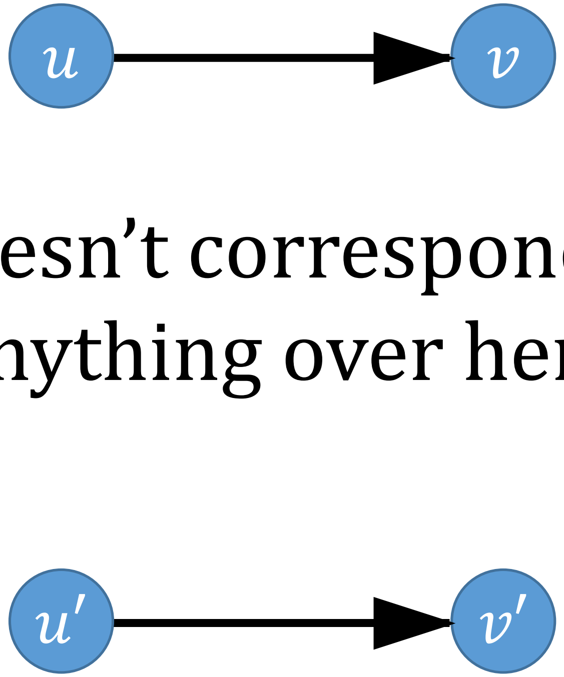


One finish of weight $+W$

New graph

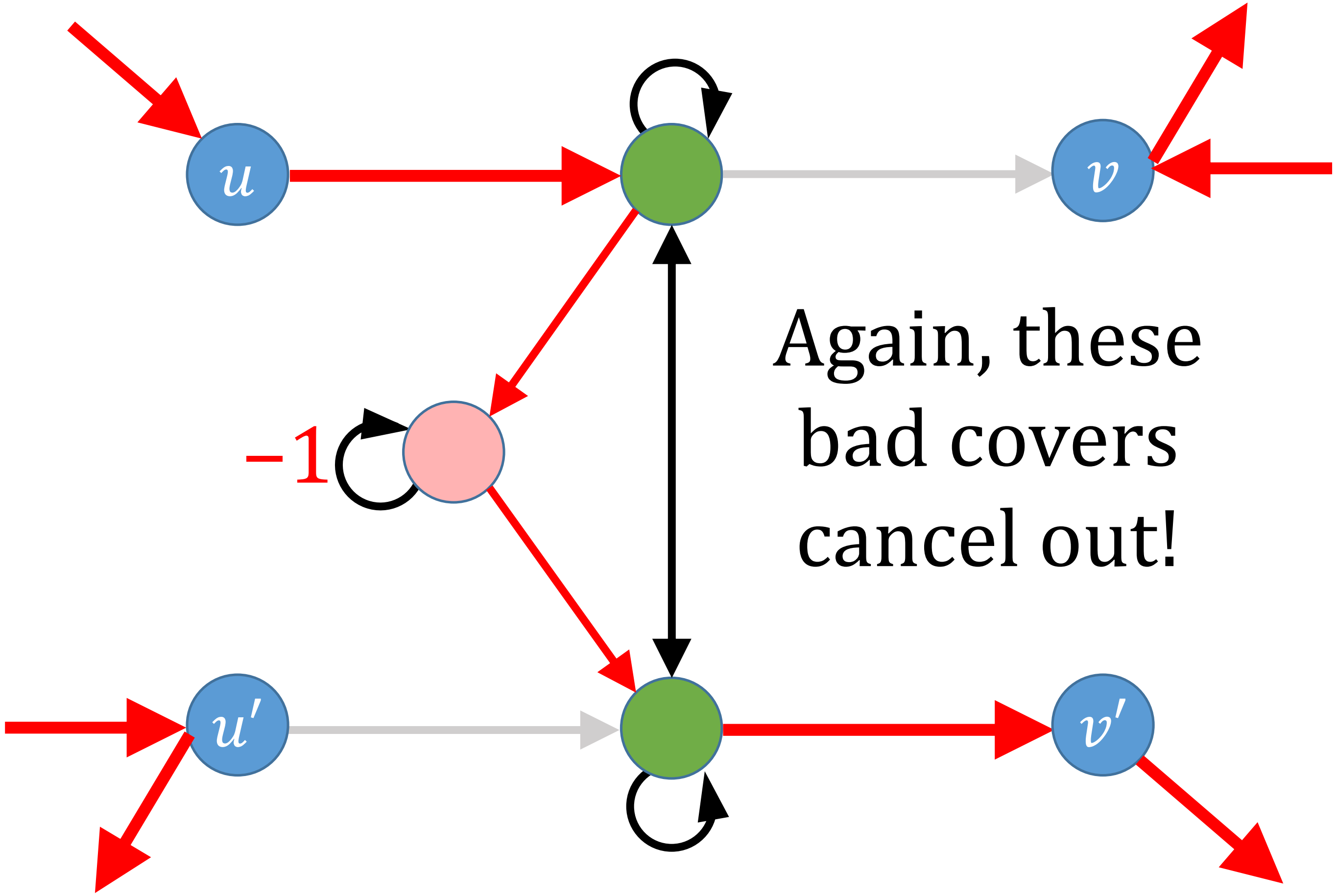
Another bad case: *Top left edge, bottom right edge*

What must the rest of the cycle cover in the new graph look like?



Doesn't correspond to anything over here!

Old graph



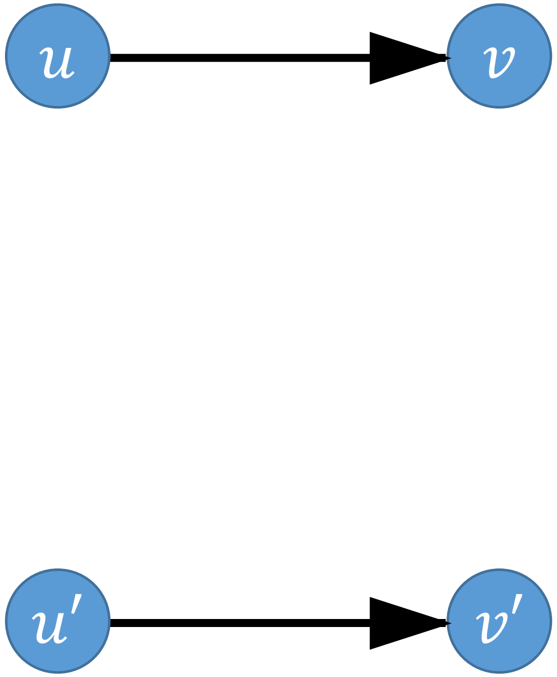
Again, these bad covers cancel out!

New graph

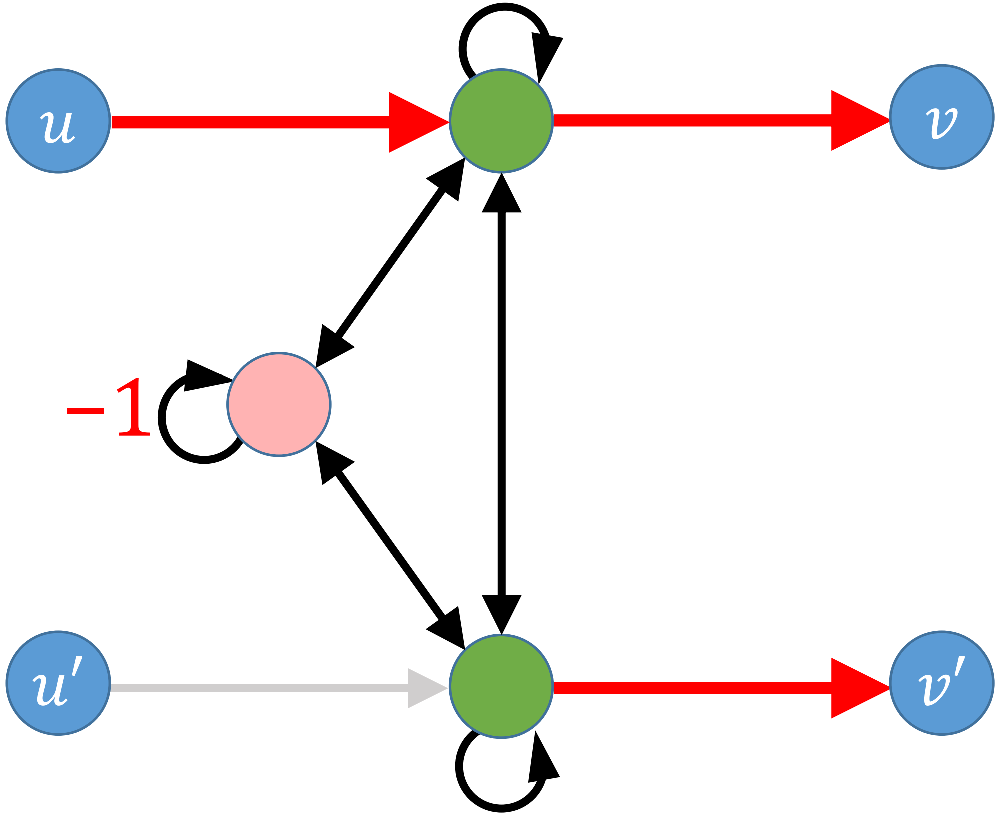
Yet another bad case: *Both top edges, bottom right edge*

What must the rest of the cycle cover in the new graph look like?

Actually, it's impossible!



Old graph

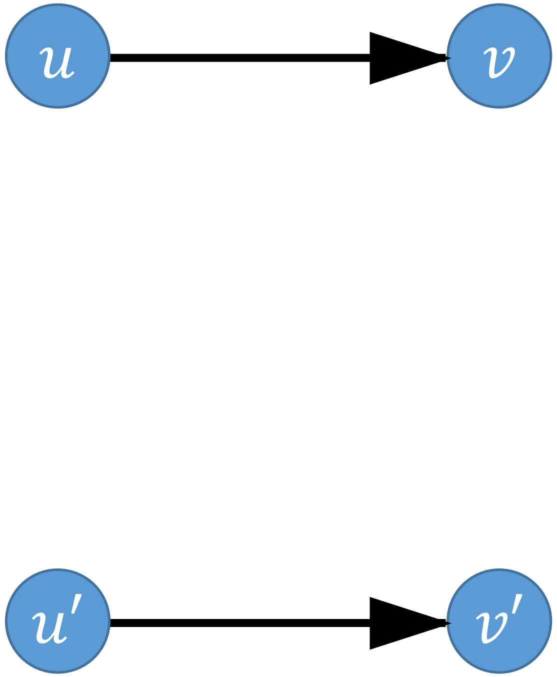


New graph

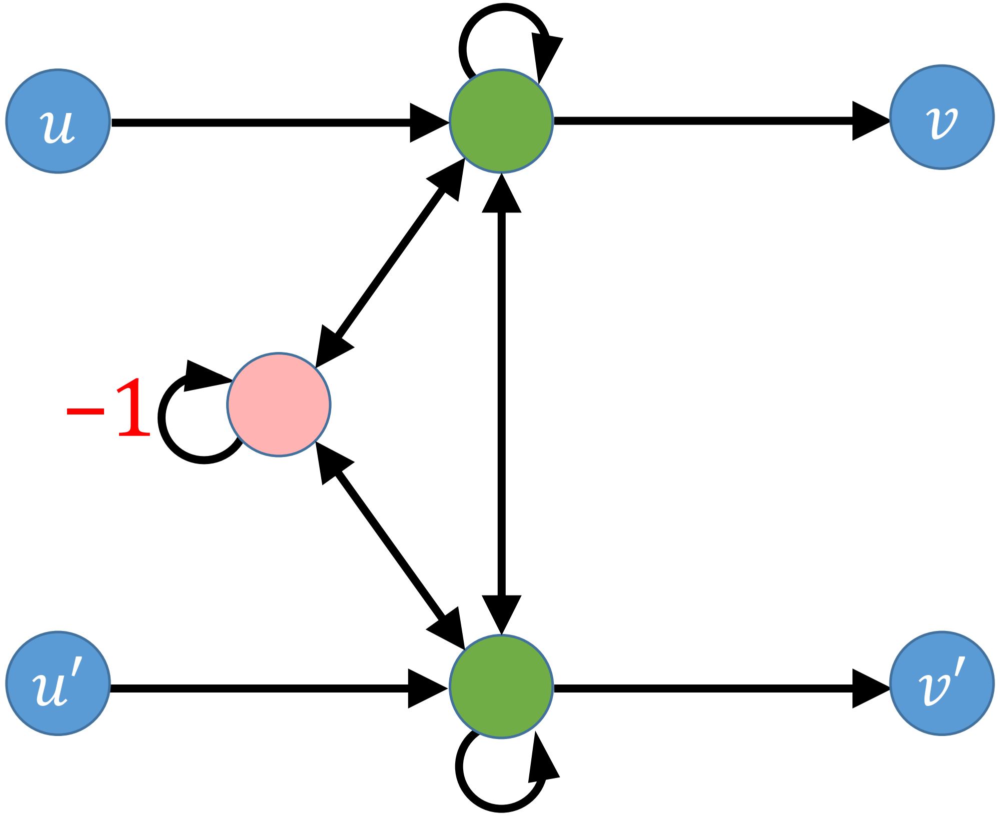
All “bad” cases complete; they contribute **zero** total weight.

This was part **(c)** of the Identification gadget properties.

What remains: two “good” cases.



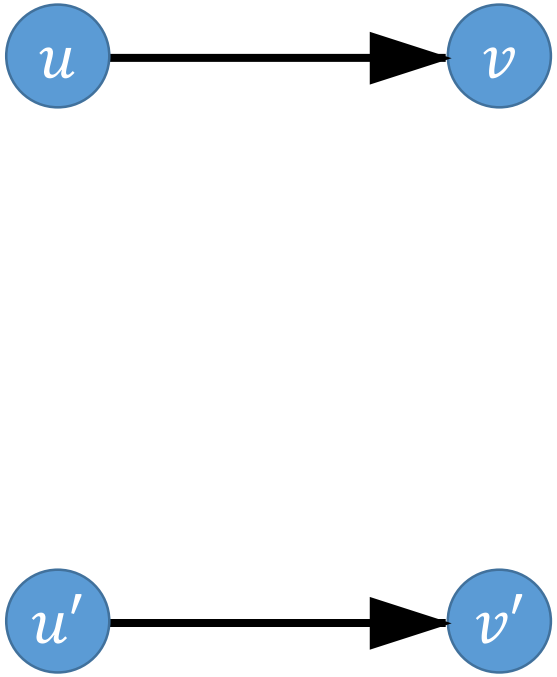
Old graph



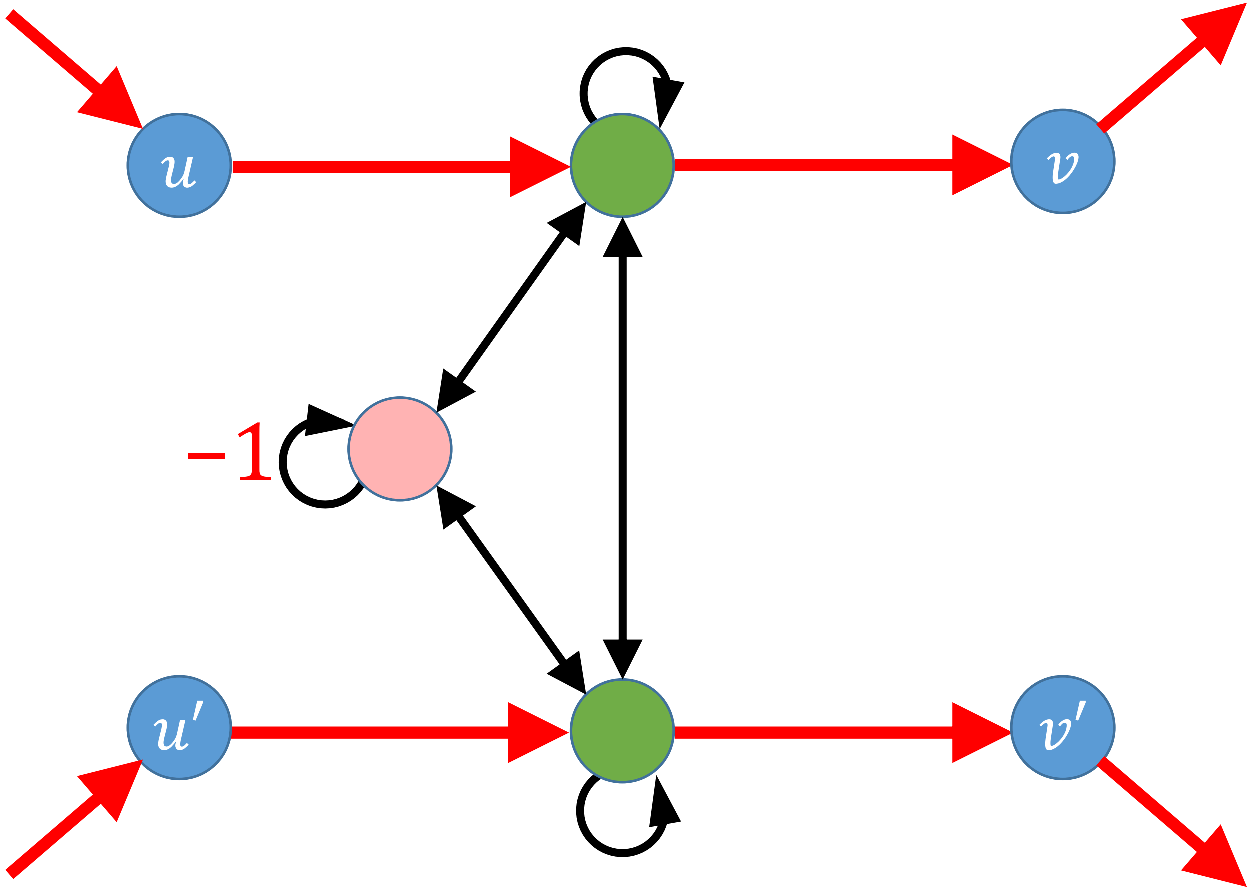
New graph

One good case: *All four edges taken.*

What must the rest of the cycle cover in the new graph look like?



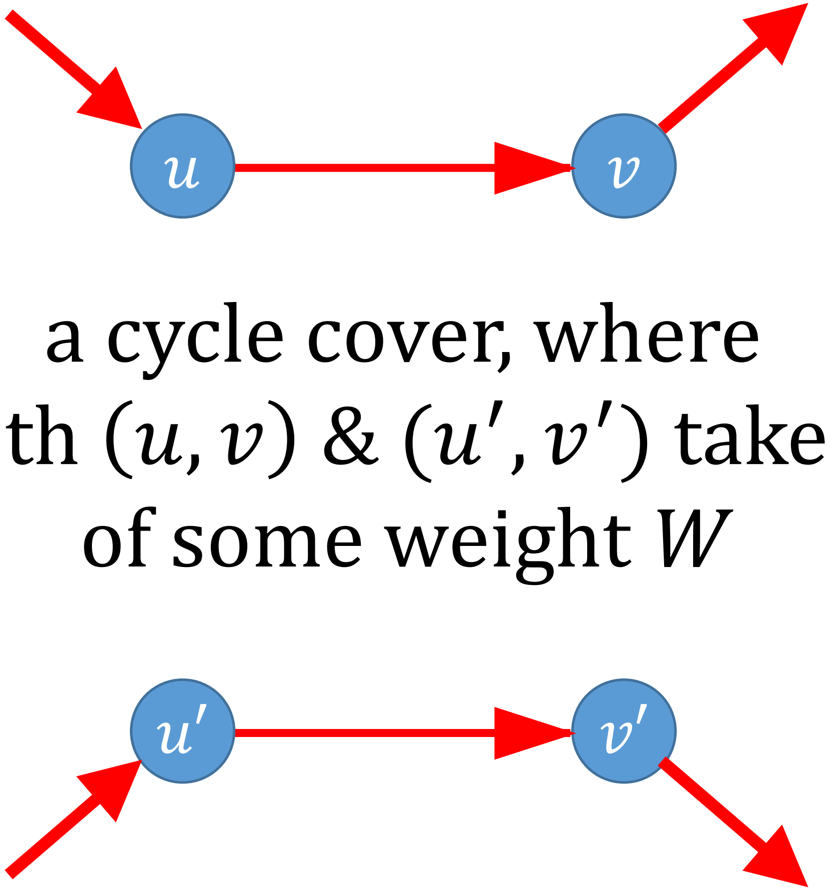
Old graph



New graph

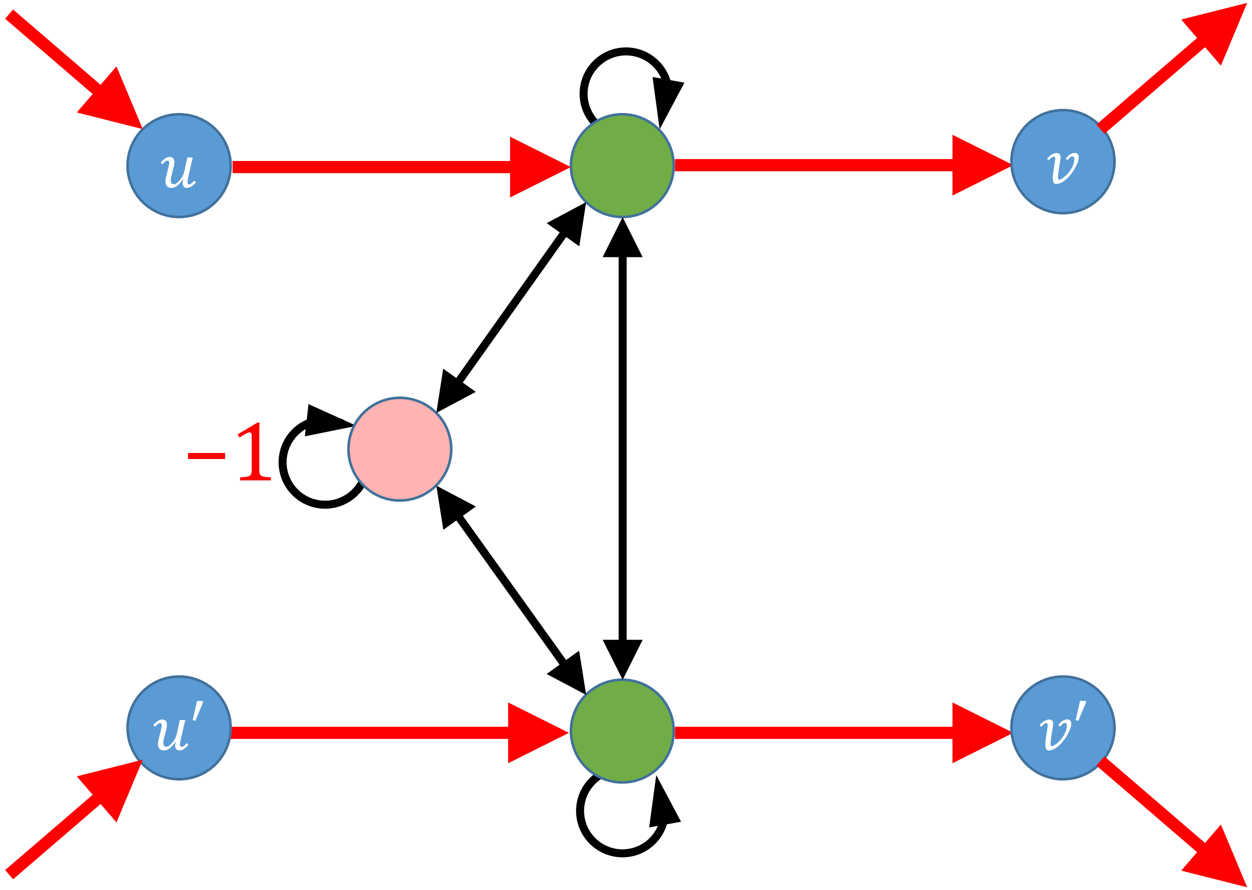
One good case: *All four edges taken.*

What must the rest of the cycle cover in the new graph look like?



a cycle cover, where
both (u, v) & (u', v') taken,
of some weight W

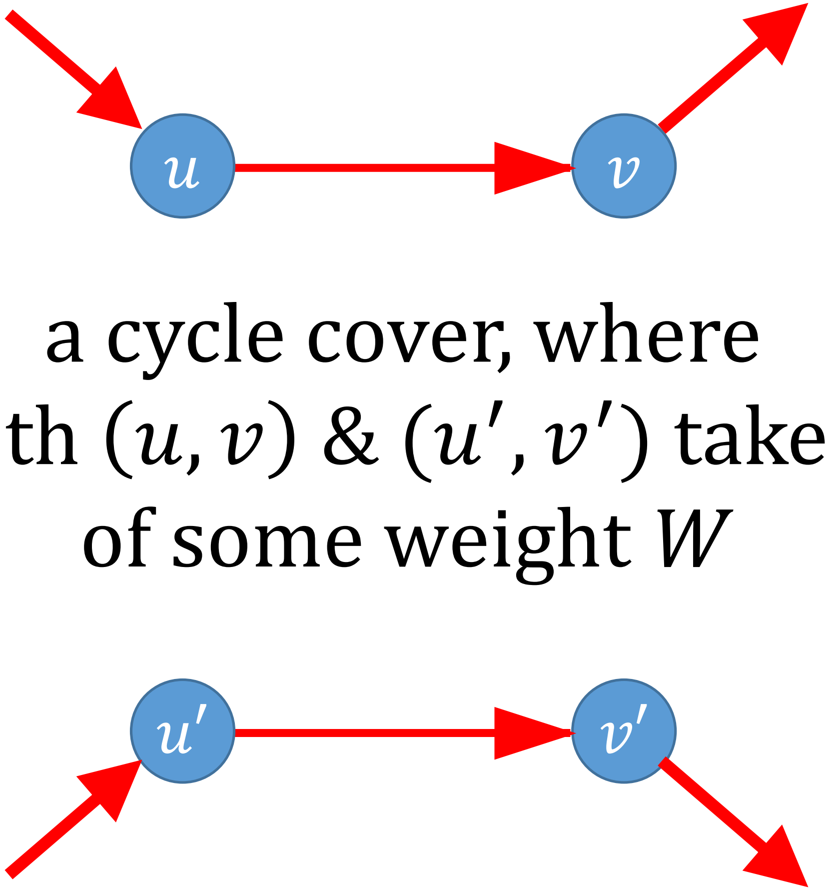
Old graph



New graph

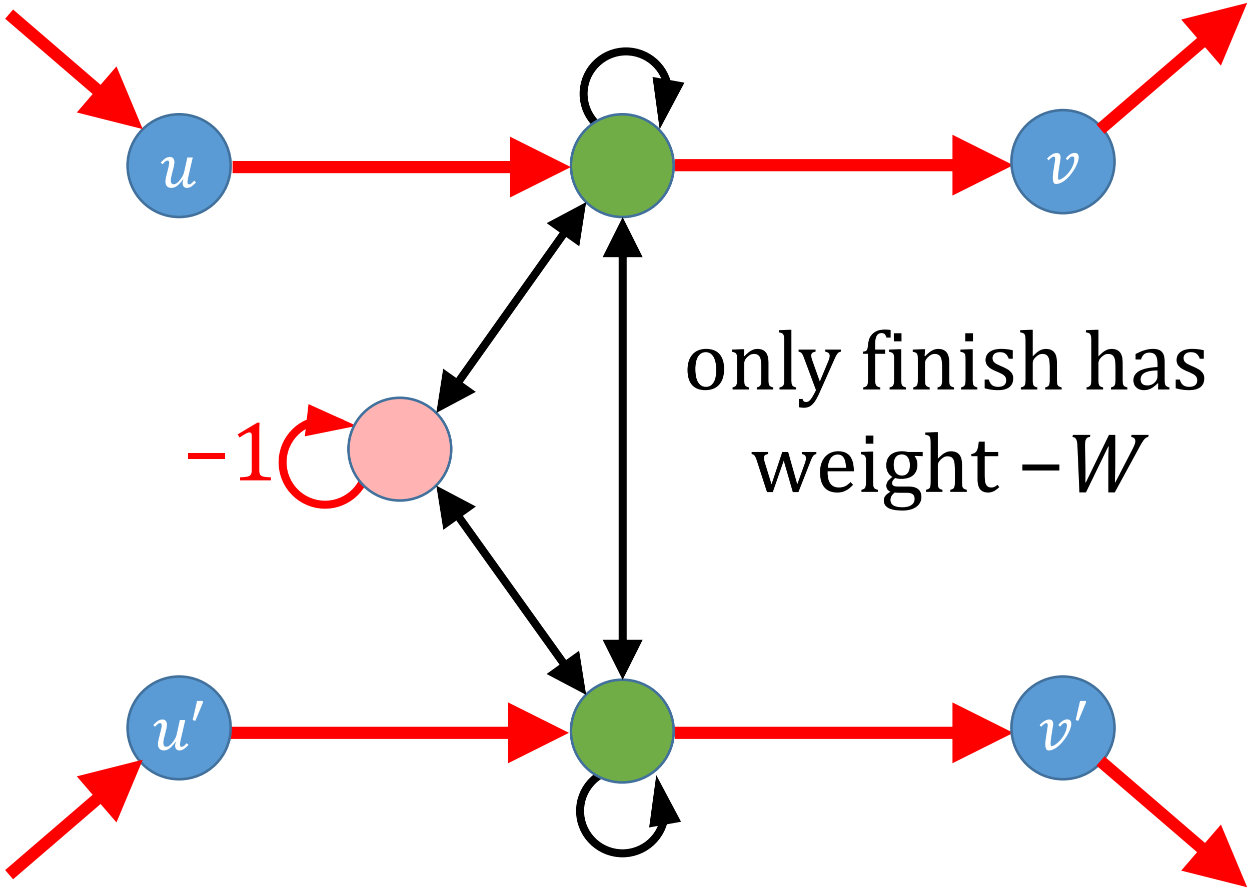
One good case: *All four edges taken.*

What must the rest of the cycle cover in the new graph look like?



a cycle cover, where
both (u, v) & (u', v') taken,
of some weight W

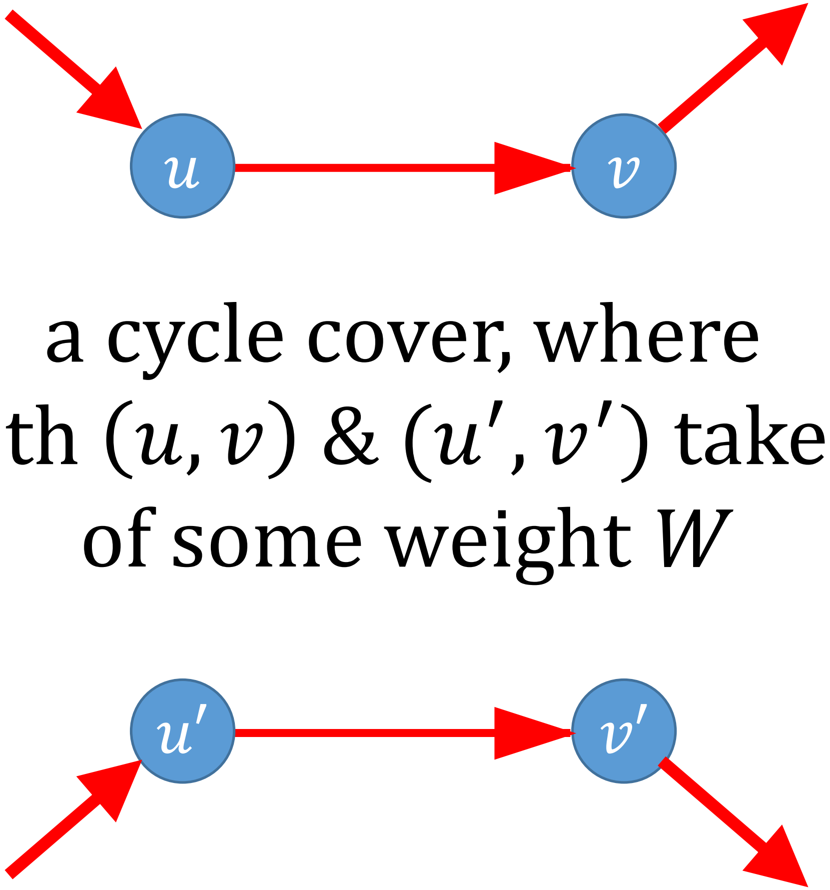
Old graph



only finish has
weight $-W$

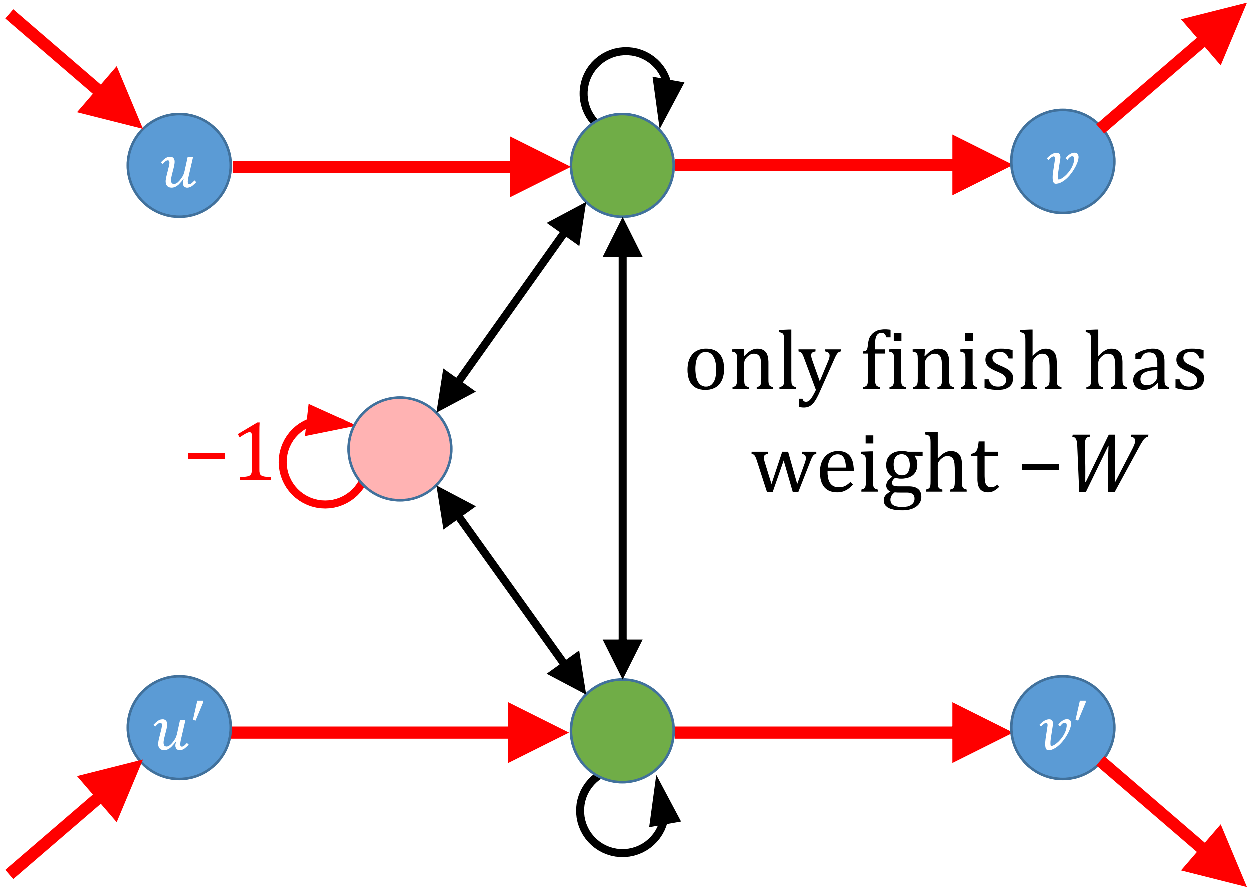
New graph

Gadget property (a): Each old cycle cover with weight W, where $(u, v), (u', v')$ *both taken*, yields a new cycle cover with weight $-W$.



a cycle cover, where both (u, v) & (u', v') taken, of some weight W

Old graph

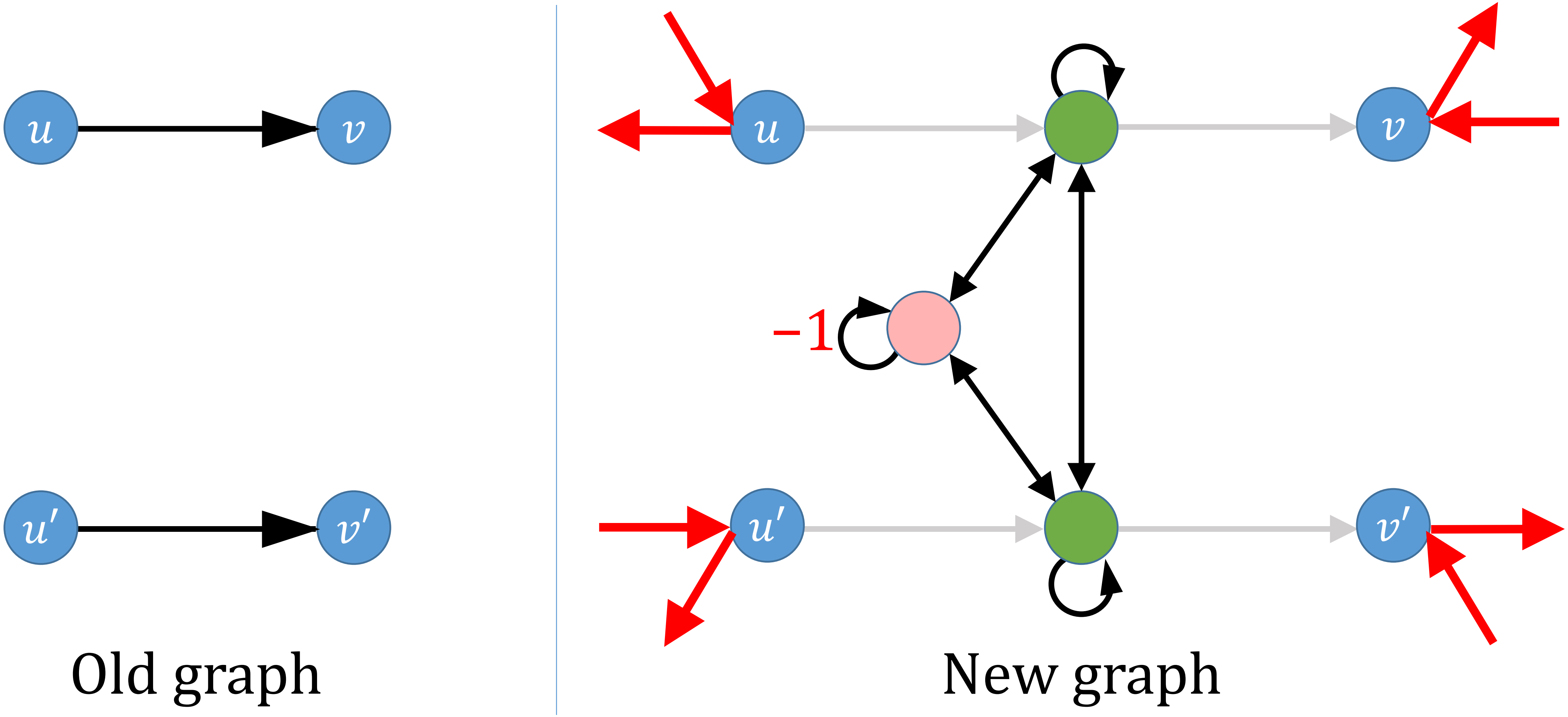


only finish has weight $-W$

New graph

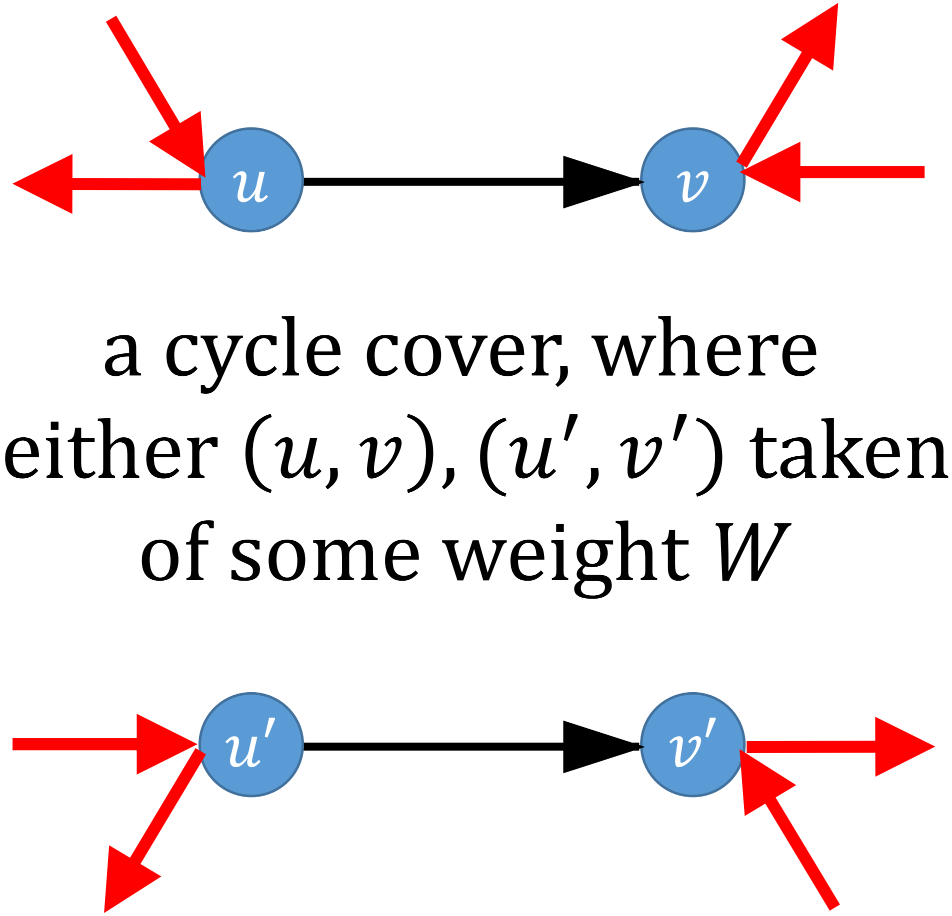
Final (good) case: *None of the four edges is taken.*

What must the rest of the cycle cover in the new graph look like?



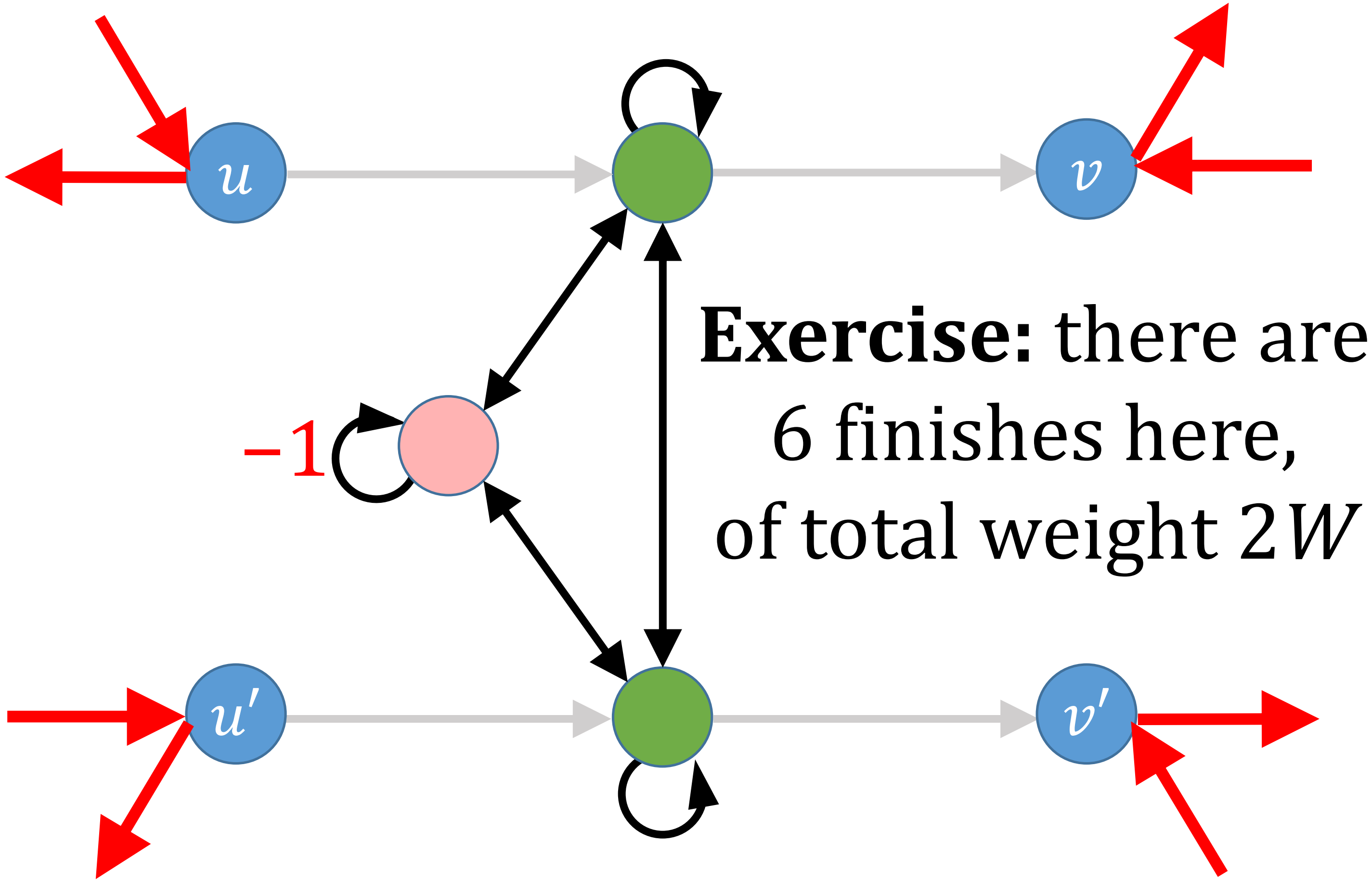
Final (good) case: *None of the four edges is taken.*

What must the rest of the cycle cover in the new graph look like?



a cycle cover, where neither $(u, v), (u', v')$ taken, of some weight W

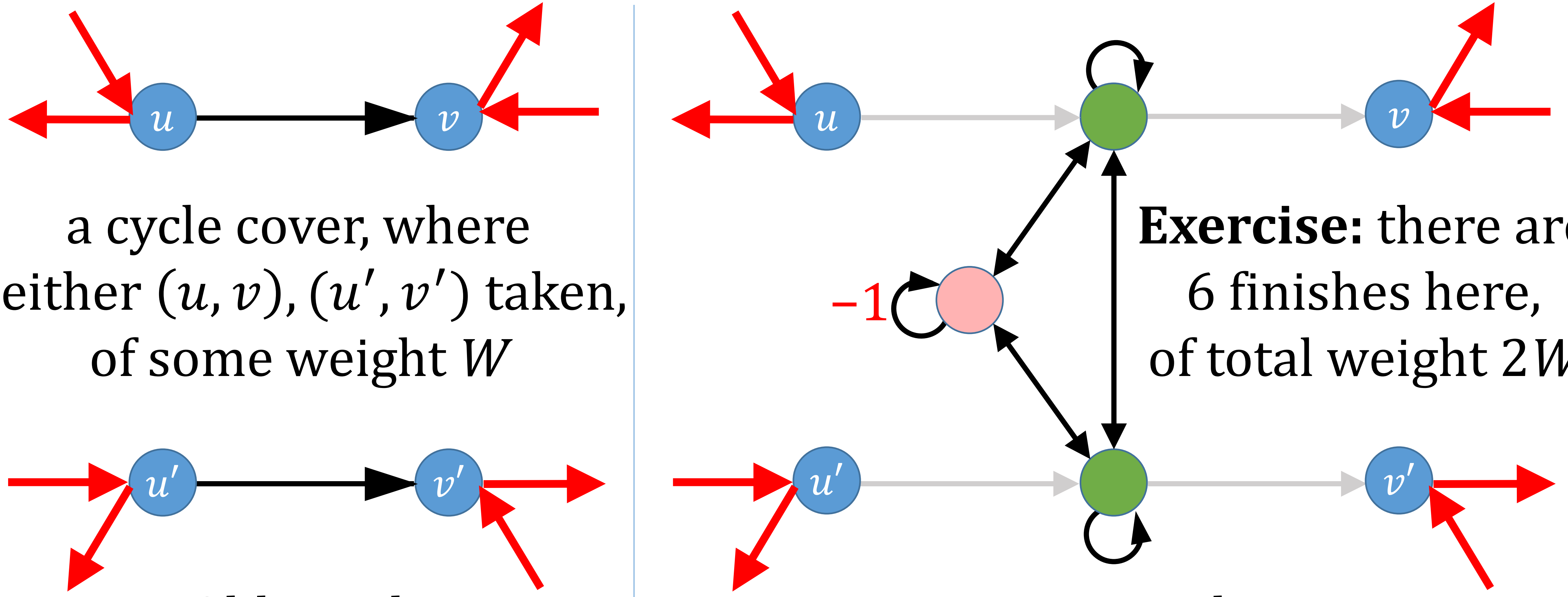
Old graph



Exercise: there are 6 finishes here, of total weight $2W$

New graph

Gadget property (b): Each old cycle cover with weight W, where $(u, v), (u', v')$ *both not taken* yields new cycle covers with weight $2W$.



a cycle cover, where neither $(u, v), (u', v')$ taken, of some weight W

Exercise: there are 6 finishes here, of total weight $2W$

Old graph

New graph

Identification gadget properties:

(a) To every "old" cycle cover of weight W using **both** edges, there corresponds a "new" cycle cover of weight $-W$.

(b) To every "old" cycle cover of weight W using **neither** edge, there correspond several "new" cycle covers of weight $2W$.

(c) All remaining other "new" cycle covers have total weight **zero**.

All gadget properties verified!

The reduction is complete.

∎

Graduate Computational Complexity Theory **CMU 15-855*, Fall 2017**

Homework 1

Due: 10:00am, Tuesday September 12

---

1. **(Valiant's Depth-Reduction Lemma.)** This problem is concerned with (simple) directed acyclic graphs. The "depth" of such a graph $G = (V, E)$ is defined to be the length of the longest path in the graph. A "labeling" of G is a mapping $\ell : V \to \mathbb{N}$. A labeling ℓ is "legal" if $\ell(u) < \ell(v)$ for all directed edges $(u, v) \in E$.

   (a) Show that if G has a legal labeling using at most d distinct labels, then its depth is less than d. Conversely, show that if G has depth less than d, then it has a legal labeling using at most d distinct labels. (Hint for the latter: consider the "canonical labeling", in which $\ell(v)$ equals the length of the longest path ending at v.)

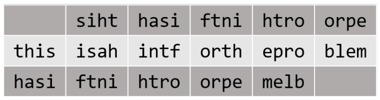
   (b) Suppose we take the canonical labeling of a graph G and consider the labels to be written in binary. For $j = 0, 1, 2, \ldots$, let E_j be all edges (u, v) such that the most significant bit where $\ell(u)$ and $\ell(v)$ differ is the jth. Show that if edges E_j are deleted from G, we can get a legal labeling of the new graph by deleting the jth bit from all labels.

   (c) Deduce the following "depth reduction lemma": Let G be a directed acyclic graph with m edges and depth less than d, where $d = 2^k$. Then for any $1 \leq r \leq k$, one can reduce the depth to less than $d/2^r$ by the deletion of at most $(r/k)m$ edges.

2. **(Block-respecting TMs.)** Given a "block-size" function $B : \mathbb{N} \to \mathbb{N}^+$, we say a multitape TM is "$B(n)$-block-respecting" if, on length-n inputs, all its tapes are divided into contiguous blocks of $B(n)$ cells, and the tape heads only cross block boundaries at times that are integer multiples of $B(n)$. (In other words, in each segment of $B(n)$ time steps, tape heads always stay within a single block of cells.)

   Let M be a k-tape Turing Machine with running time $T(n)$. Let $1 \leq B(n) \leq T(n)/2$ be a block-size function. Show there is another Turing Machine M' with $O(k)$ tapes[1] that is $B(n)$-block-respecting, decides the same language as M, and has running time $O(T(n))$.[2]

   The TAs will pay extra attention to the *quality of your exposition* in this problem.



	siht	hasi	ftni	htro	orpe
this	isah	intf	orth	epro	blem
hasi	ftni	htro	orpe	melb	

3. **(Improving the Time Hierarchy Theorem via padding.)**

   (a) Let $t_1 : \mathbb{N} \to \mathbb{N}$ be a nondecreasing time-constructible function with $t_1(n) \geq n$, and let t_2 and f be two more such functions.[3] Show that $\mathsf{TIME}(t_1(n)) = \mathsf{TIME}(t_2(n))$ implies $\mathsf{TIME}(t_1(f(n))) = \mathsf{TIME}(t_2(f(n)))$. (Hint: padding.)

   (b) Show that $\mathsf{TIME}(n^3 \log^{3/4} n) \neq \mathsf{TIME}(n^3)$. You may use the Time Hierarchy Theorem (Theorem 3.1 in Arora–Barak), and you may take it for granted that any normal-looking functions are time-constructible. (Hint: you may need to use part (a) *several* times.)

   It is a fact (you don't need to prove it) that a suitable elaboration of this problem shows $\mathsf{TIME}(n^a \log^{\varepsilon} n) \neq \mathsf{TIME}(n^a)$ for all $a \geq 1$ and $\varepsilon > 0$.

Footnotes

[1] $3k + 1$, or even $k + 1$, should be possible.

[2] Technicalities: First, you may assume $B(n)$ has the following "time- and space-constructibility" properties: There is a 2-tape TM that, on inputs of length n, uses $O(T(n))$ time and exactly $B(n)$ space (on its second tape, only reading the input tape), and writes $B(n)$ in unary on the second tape. Further, your M' may use this routine at the beginning, and only *then* become $B(n)$-block-respecting.

[3] We would like to also talk about functions like $\sqrt{t_1}$ or $t_2 \log t_2$ without worrying about the fact that these could be real-valued. Assume that real values arising in such expressions are always rounded up; or, just choose not to worry about it.

Graduate Computational Complexity Theory **CMU 15-855*, Fall 2017**

HOMEWORK 2
Due: 10:00am, Tuesday September 19

---

1. **(Almost-Everywhere Time Hierarchy Theorems.)**

   (a) The standard (Deterministic) Time Hierarchy Theorem we considered in class shows that if $T(n)$ is time-constructible and $t(n)\log t(n) = o(T(n))$ then there is a language $L \in \mathsf{TIME}(T(n))$ such that $L \notin \mathsf{TIME}(t(n))$. If we unpack the definition of $L \notin \mathsf{TIME}(t(n))$, it means this:

   $$\text{for any TM } M \text{ with running time } O(t(n)), \quad \exists x \; M(x) \neq L(x), \tag{1}$$

   Here we're abusing notation a little by writing $L(x)$ for the answer to the question $x \overset{?}{\in} L$. Actually, if you inspect the proof of the theorem, it showed something stronger:

   $$\text{for any TM } M \text{ with running time } O(t(n)), \quad \exists^\infty x \; M(x) \neq L(x), \tag{2}$$

   where the symbol $\exists^\infty$ means "there exists infinitely many" (or synonymously, "infinitely often").[1] Show that even if you didn't remember the proof of the THT, you could deduce (2) in a purely "black-box" fashion from (1). (You may assume that $t(n) \geq n$.)

   (b) Similarly show that you can deduce the following in a purely "black-box" fashion:

   $$\text{for any } M \text{ deciding } L, \text{ and any } C, \quad \exists^\infty x \; M(x) \text{ takes } > Ct(|x|) \text{ time steps.} \tag{3}$$

   (c) Arguably even (2) is pretty weak. Here is an upgraded statement that one might desire:

   $$\text{for any TM } M \text{ with running time } O(t(n)), \quad \forall^\infty x \; M(x) \neq L(x), \tag{4}$$

   where the symbol "$\forall^\infty$ means "for all but finitely many x" (or synonymously, "almost everywhere"). Show that (4) is provably too much to hope for.

   (d) Here is an upgrade of (3):

   $$\text{for any } M \text{ deciding } L, \text{ and any } C, \quad \forall^\infty x \; M(x) \text{ takes } > Ct(|x|) \text{ time steps.} \tag{5}$$

   This *can* be achieved, but the proof is much harder (it took 13 years after the original THT). Short of that, you are asked to prove a weaker statement in this problem.

   Say that a language A is in the class i.o.-P if there is a polynomial-time Turing Machine M that computes A correctly for infinitely many input lengths (i.e., $A \cap \{0,1\}^n = L(M) \cap \{0,1\}^n$ for infinitely many n). Prove that there is a language $L \in \mathsf{EXP}$ that is not in i.o.-P.

2. **(Superiority.)** Do Exercise 3.4 in Arora–Barak.[2]

---

[1] In fact, the proof kind of needed to show this, to take care of the fact that you need to diagonalize against all $O(t(n))$ running times.

[2] Of course, you may assume $n^{1.1}$ is time-constructible.

3. **(Awesome circuit lower bounds from depth-3 circuit lower bounds.)** Suppose $f : \{0,1\}^n \to \{0,1\}$ can be computed by a circuit of logarithmic depth $c_1 \log n$ and linear size $c_2 n$. The goal of this problem is to show that f can also be computed by a depth-3 circuit of subexponential size, namely $2^{O(n/\log\log n)}$.[3] In fact, you should be able to make the depth-3 circuit an OR of CNFs, where each CNF has at most $2^{O(n^{.01})}$ clauses, and where the circuit has the additional property that on all inputs, at most one of the CNFs outputs True.

   By the way, this result shows that to get a superlinear circuit lower bound for log-depth circuits (which would be awesome), "all" you have to do is get an essentially-fully-exponential circuit lower bound for depth-3 circuits. Later in the class we will show that depth-3 circuits require size $2^{\Omega(\sqrt{n})}$ to compute the Parity function $f(x) = \sum_i x_i \bmod 2$. Close, but no cigar.

   (a) In the log-depth, linear-size circuit for f, show that it is possible to "cut" $O(n/\log\log n)$ wires, leaving a collection of subcircuits each of which depends on at most $O(n^{.01})$ inputs. (Hint: an earlier homework problem.)

   (b) Complete the proof — i.e., the construction of the depth-3 circuit for f. (Hint: consider "enumerating" all possible values for the cut wires.)

---

[3] As per usual conventions, in the log-depth linear-size circuit, we assume the allowed gates are NOT and fan-in-2 AND/OR, whereas in the depth-3 circuit we assume the allowed gates are NOT and unbounded-fan-in AND/OR. Also, NOT gates are not counted toward depth in constant-depth circuits.

Graduate Computational Complexity Theory **CMU 15-855*, Fall 2017**

Homework 3

Due: 10:00am, Tuesday September 26

---

1. Define $\mathsf{P}/\log$ (polynomial time with logarithmic advice) to be $\bigcup_{b,c} \mathsf{TIME}(n^b)/c \log n$. Show that $\mathsf{NP} \subseteq \mathsf{P}/\log \implies \mathsf{NP} = \mathsf{P}$. (Hint: use the fact that SAT is NP-complete.)

2. Exercise 6.17 in Arora–Barak asks you to prove their Theorem 6.32. We'll get to that result later in the class, but for now, prove the following variant:

   $L \in \mathsf{NP}$ iff L can be computed by a "DC-uniform"[1] circuit family (C_n) with the same four conditions as in Theorem 6.32, except condition #3 is changed to "semi-unbounded fan-in", which means the OR gates can have arbitrary (exponential) fan-in, but the AND gates can only have poly(n) fan-in.

3. A *restarting* probabilistic Turing Machine is a standard probabilistic Turing Machine (Definition 7.1 in Arora–Barak) with the following additional feature: It has a special state called Restart, and if ever the TM transitions into that state, the entire computation is completely restarted (tapes/heads/state all reset to their initial values, nothing remembered from the prior computation). Each computation path from the initial state to Accept/Reject/Restart is called a *run*, and depending on the machine's "coin flips", it may well have multiple runs (restarts) before it finally accepts/rejects. The running time $t(n)$ on inputs of length n is defined to be the maximum possible number of steps in a single run (over all inputs and coin flips). Note that we do *not* sum the times over all runs; we only "pay for" the longest run. (This makes restarting TMs a rather non-realistic model.) Finally, if a restarting TM has the property that it restarts with probability 1 on at least one input, we deem it to be an *invalid* machine.[2]

   Let $\mathcal{C}$ be a probabilistic complexity class such as RP, BPP, PP (or even NP) that can be defined as follows, for some constants $0 \le s \le c < 1$: "$L \in \mathcal{C}$ if there exists a polynomial-time probabilistic TM M such that $x \in L \implies \mathbf{Pr}[M(x) \text{ accepts}] > c$ and $x \notin L \implies \mathbf{Pr}[M(x) \text{ accepts}] \le s$". We then define $\mathsf{Restarting}\mathcal{C}$ to be the same class, except that M is allowed to be a (valid) restarting probabilistic TM.

   (a) Prove that $\mathsf{RestartingPP} = \mathsf{PP}$.

   (b) Prove that $\mathsf{RestartingNP} = \mathsf{NP}$.

   (c) Prove that $\mathsf{RestartingRP} = \mathsf{NP}$.

   (d) $\mathsf{RestartingBPP}$ is a funny class. Prove that $\mathsf{NP}, \mathsf{coNP} \subseteq \mathsf{RestartingBPP}$.

   (e) Prove that $\mathsf{RestartingZPP} = \mathsf{NP} \cap \mathsf{coNP}$, where $\mathsf{RestartingZPP}$ means the class of languages L for which there is a polynomial-time restarting TM with the following properties: Besides "Accept" and "Reject", it has a third halting (output) state called "?". And on input x, the machine (eventually, after possible restarts) outputs "?" with probability at most 1/2, and otherwise outputs the correct answer to $x \overset{?}{\in} L$.

---

[1] This is basically the same as "DLOGTIME-uniform" as discussed in class, except DC-uniformity allows you polylog time, as opposed to logarithmic time. Go with "DC-uniformity" here (as in Arora–Barak's Definition 6.31) for simplicity. However, please augment their definition as follows: not only should the "TYPE" function give a gate's type, it should also give the *number* of incoming wires (i.e., the fan-in amount).

[2] This is nothing unusual; it's exactly like how deterministic TMs may fail to halt on some inputs, in which case we deem them "non-deciders" and say they don't compute any language.

Graduate Computational Complexity Theory **CMU 15-855*, Fall 2017**

Homework 4

Due: 10:00am, Tuesday October 3

---

1. **(Could EXP have poly-size circuits?)** We know that $\mathsf{EXP} \not\subseteq \mathsf{P}$ by the Time Hierarchy Theorem. And we sometimes think of P as being roughly comparable to P/poly. Now the latter contains undecidable languages, so they're definitely not the same. Still, it doesn't seem so likely that getting to use a different poly-time algorithm for each input length would be especially helpful for solving an EXP-complete language (like SUCCINCT-CIRCUIT-EVAL, or GENERALIZED-CHESS). But can we get more convincing evidence that $\mathsf{EXP} \subseteq \mathsf{P}/\text{poly}$ is indeed unlikely?

   (a) Let M be a deterministic 1-tape Turing Machine running in time at most 2^{cn^c}. Appropriately formalize — and then also prove — a statement encapsulating the idea that the entries in $M(x)$'s "computation tableau" are computable in EXP. (You might want to look at Sipser's Theorem 7.37 (Cook–Levin) for some terminology and inspiration.)

   (b) Prove that $\mathsf{EXP} \subseteq \mathsf{P}/\text{poly} \implies \mathsf{EXP} = \mathsf{PSPACE}$. (Hence $\mathsf{EXP} \subseteq \mathsf{P}/\text{poly}$ is probably not true, because we consider $\mathsf{EXP} = \mathsf{PSPACE}$ unlikely.)

2. **(Circuit characterization of PH.)** Do Exercise 6.17 in Arora–Barak (called Exercise 6.13 in the "draft version").

3. **(HPV in an alternate universe.)** In this problem you are asked to show the following: There exists a language A such that for all (time- and space-)constructible[1] $f(n)$,

$$\mathsf{TIME}^A(f(n)) = \mathsf{SPACE}^A(f(n)).$$

   Here $\mathsf{TIME}^A(f(n))$ means all languages decidable by a multitape, time-$O(f(n))$ oracle Turing Machine with oracle access to A, and $\mathsf{SPACE}^A(f(n))$ is similarly defined. It is very easy to see that the theorem $\mathsf{TIME}(f(n)) \subseteq \mathsf{SPACE}(f(n))$ "relativizes" (namely, $\mathsf{TIME}^B(f(n)) \subseteq \mathsf{SPACE}^B(f(n))$ for every language B and bound $f(n)$), so the goal is to show that there exists A with $\mathsf{SPACE}^A(f(n)) \subseteq \mathsf{TIME}^A(f(n))$.

   We now describe — roughly — the A you will want to use, although we leave it to you to define A completely precisely. Basically, A should be the language of all tuples $(M, D, x, 1^s)$ such that M is a multitape oracle TM, D is a multitape TM running in space 2^n (how can you ensure this?), x is a string, and (this being the main point) $M^{L(D)}(x)$ accepts while using at most s tape cells. After formalizing A, the first step will be to show that A can be computed in space 2^n. (You might want to add some "padding" into the definition of A to help you show a bound of literally 2^n, not $O(2^n)$.) Then complete the proof, using the fact that there is some D^* deciding A...

---

[1] In fact, the result is known to hold with no constructibility assumptions at all. For your proof I doubt you'll need space-constructibility. However time-constructibility definitely simplifies the proof. And honestly, I don't even know why I'm typing this footnote, because no one cares about non-constructible functions.

HOMEWORK 5

Due: 10:00am, Tuesday October 10

---

1. **(Courtroom complexity.)** In this problem we study a slightly peculiar complexity class that we'll call $\mathsf{S_2P}$. Informally, we say $L \in \mathsf{S_2P}$ whenever the following circumstances hold. There are two lawyers, Yolanda and Zeyuan, whose job is to argue in front of judge Victor about whether or not $x \in L$. Whenever $x \in L$, there is something Yolanda can say that will convince judge Victor that indeed $x \in L$, no matter what Zeyuan says. Conversely, whenever $x \notin L$, there is something Zeyuan can say that will convince judge Victor that $x \notin L$, no matter what Yolanda says.

   More precisely, we say that $L \in \mathsf{S_2P}$ if there is a polynomial $p(n)$ and a polynomial-time algorithm V such that

   $$x \in L \implies \exists^p y\ \forall^p z\ V(x,y,z) = 1,$$
   $$x \notin L \implies \exists^p z\ \forall^p y\ V(x,y,z) = 0.$$

   (Recall "$\exists^p y$" means "$\exists y$ with $|y| \leq p(|x|)$", etc.)

   (a) Show that $\mathsf{S_2P}$ is closed under complement: $\mathsf{coS_2P} = \mathsf{S_2P}$.

   (b) Show that $\mathsf{S_2P} \subseteq \Sigma_2\mathsf{P} \cap \Pi_2\mathsf{P}$.

   (c) Show $\mathsf{NP} \subseteq \mathsf{P/poly} \implies \mathsf{PH} = \mathsf{S_2P}$. (This is an improvement on the Karp–Lipton Theorem, by part (b)... but in fact, you can solve this problem by almost literally repeating the proof of Karp–Lipton.)

   (d) Show that $\mathsf{P^{NP}} \subseteq \mathsf{S_2P}$.

2. **(A route to $\mathsf{P} \neq \mathsf{NP}$?)** Let c_n denote the maximum number of gates needed by a Boolean circuit to compute any function $f : \{0,1\}^n \to \{0,1\}$. Shannon and Lupanov showed that $c_n \approx 2^n/n$, but we will be interested in the literal exact value of c_n. Let us say that a language L has *maximal circuit complexity* if $L \cap \{0,1\}^n$ requires circuits of size c_n for every n.

   Show that if every language in E has *non*-maximal circuit complexity (i.e., just *one* gate can be saved somewhere in the circuit family) then $\mathsf{P} \neq \mathsf{NP}$. (Recall that $\mathsf{E} = \bigcup_c \mathsf{TIME}(2^{cn})$.)

3. **(Limited SAT queries.)** When C is a complexity class, the notation $\mathsf{C}^{A[k]}$ means the same class where *at most* k oracle queries to the language A are allowed. As usual, $\mathsf{C}^{\mathsf{NP}[k]}$ denotes the union of $\mathsf{C}^{A[k]}$ over all $A \in \mathsf{NP}$; equivalently, it's $\mathsf{C}^{\mathrm{SAT}[k]}$. In studying the Polynomial Time Hierarchy, we observed that when $\mathsf{C} = \mathsf{NP}$, we could massively reduce the number of queries used: $\mathsf{NP^{NP}} = \mathsf{NP}^{\mathsf{NP}[\mathrm{poly}(n)]} = \mathsf{NP}^{\mathsf{NP}[1]}$. The same is (seemingly) not true when $\mathsf{C} = \mathsf{P}$; it is believed that $\mathsf{P}^{\mathsf{NP}[1]} \subsetneq \mathsf{P}^{\mathsf{NP}[2]} \subsetneq \mathsf{P}^{\mathsf{NP}[3]} \subsetneq \cdots$

   In this problem, we will look at an interesting class: $\mathsf{P}^{\mathsf{NP}[\log]}$, which is short for $\mathsf{P}^{\mathsf{NP}[O(\log n)]}$, the class of languages decidable in polynomial time by a SAT-oracle Turing Machine that makes at most $O(\log n)$ oracle queries on inputs of length n.

   (a) Show that the following two problems are in $\mathsf{P}^{\mathsf{NP}[\log]}$: UNIQUE-MAX-CLIQUE, the language of all graphs whose largest clique is unique; ODD-MAX-CNF-SAT, the language of all CNF formulas for which the maximum number of clauses that can be satisfied by any truth assignment is odd.

(b) Define $\mathsf{P}_{\|}^{\mathsf{NP}[r]}$ to be the class of all languages decidable in polynomial time by a SAT-oracle Turing Machine that makes at most r *nonadaptive* oracle queries. This means that the machine can only interact with "the oracle" one time, in the following way: it can submit r separate oracle queries, and get back the r answers. Show that $\mathsf{P}^{\mathsf{NP}[k]} \subseteq \mathsf{P}_{\|}^{\mathsf{NP}[2^k-1]}$, even for $k = O(\log n)$, and hence $\mathsf{P}^{\mathsf{NP}[\log]} \subseteq \mathsf{P}_{\|}^{\mathsf{NP}}$.

(c) Building on work of Gilbert, Michael Fischer showed the following result: For every n, there is an n-input, n-output Boolean circuit, consisting of poly(n) AND gates, poly(n) OR gates, and $\lceil \log_2(n+1) \rceil$ NOT gates, such that on input $(x_1, x_2, \ldots, x_n)$, the output is $(\neg x_1, \neg x_2 \ldots, \neg x_n)$.[1] If you have never seen this before, I very strongly urge you to try to prove this result in the case $n = 3$; it's a great puzzle! But anyway, you can assume Fischer's result.

Show an almost-opposite containment to part (b): $\mathsf{P}_{\|}^{\mathsf{NP}[2^k-1]} \subseteq \mathsf{P}^{\mathsf{NP}[k+1]}$, even for $k = O(\log n)$, and hence $\mathsf{P}^{\mathsf{NP}[\log]} = \mathsf{P}_{\|}^{\mathsf{NP}}$.

(0-point bonus problem: Can you get the exact-opposite containment, $\mathsf{P}_{\|}^{\mathsf{NP}[2^k-1]} \subseteq \mathsf{P}^{\mathsf{NP}[k]}$ in case $k = 2$? Can you get it in general?)

---

[1] Also, the construction is P-uniform.

Graduate Computational Complexity Theory **CMU 15-855*, Fall 2017**

TEST #1

Due: 10:00am, Tuesday October 17

No collaboration or Internet-usage allowed! You may consult the textbook.
You may cite past homework problems and results from lecture.

Solve four problems total, namely #1, #2, and two out of three from $\{\#3, \#4, \#5\}$.

---

1. **(MA in PP, 10 points.)** Show that $\mathsf{MA} \subseteq \mathsf{PP}$. (You are not required to do it this way, but one way you *can* show this is to show $\mathsf{MA} \subseteq \mathsf{BPP}_{\text{path}} \subseteq \mathsf{PP}$, where $\mathsf{BPP}_{\text{path}}$ is the "real" name of the class called "RestartingBPP" on Homework 3.)

2. **(Collapses, 10 points total.)**

   (a) **(4 points.)** Show that $\mathsf{PSPACE} \subseteq \mathsf{P}/\text{poly}$ implies that $\mathsf{PSPACE} = \Sigma_2 = \Pi_2$.

   (b) **(2 points.)** Show that $\mathsf{EXP} \subseteq \mathsf{P}/\text{poly}$ implies that $\mathsf{EXP} = \Sigma_2 = \Pi_2$.

   (c) **(4 points.)** Show that $\mathsf{NP} \subseteq \mathsf{P}/\text{poly}$ implies $\mathsf{AM} = \mathsf{MA}$.

3. **(Superiority II, 10 points.)** Recall that for Homework #2, Problem 2, you did Exercise 3.4 in Arora–Barak. Now, show that $\mathsf{NTIME}(n^{1.1})$ *is* in fact "superior" to $\mathsf{NTIME}(n)$. You will need a new proof of the Nondeterministic Time Hierarchy Theorem. What follows are some hints for it; besides completing the proof, please note that even the hints need several details to be filled in.

   Consider a nondeterministic TM D which parses its input into the form 1^i01^j0y, where $i, j \in \mathbb{N}$ and $y \in \{0,1\}^*$. This is henceforth written as (i, j, y). The number i encodes an NTM M, the number j is for "junk" (to ensure arbitrarily long strings), and y is interpreted as a sequence of nondeterministic "guess bits". If $|y| < \lceil (i+j+2)^{1.05} \rceil$ then D accepts iff M accepts both $(i, j, y0)$ and $(i, j, y1)$ in $(i+j+2+|y|)^{1.05}$ steps. If $|y| = \lceil (i+j+2)^{1.05} \rceil$ then D accepts iff M rejects (i, j, ε) when using y as its nondeterministic guesses.

   In solving this problem, please have a clear section where you give what you feel is/are the "main idea(s)" in the proof. Of course, you must fill in all the details in the other sections.

4. **(The Exponential Time Hierarchy collapses, 10 points total.)** Throughout this problem, let $N^?$ denote some nondeterministic oracle-TM running in time kn^k, let A denote some language in NEXP, and let U denote any fixed NEXP-complete language (under poly-time mapping reductions, as usual).

   (a) **(3 points.)** Show there is a poly-time deterministic oracle-TM $C^?$ with the following property: $C^U(x)$ computes the exact number of strings in A of length at most $k|x|^k$.

   (b) **(3 points.)** Say that a nondeterministic TM K "computes the answer to $y \in A$" if, on input y, it runs nondeterministically, accepts on at least one computation branch, and on *all* branches where it accepts, its working tape contains nothing but the correct answer (0/1) to the question of whether $y \in A$.

   Show that you can extend your machine $C^?$ from part (a) so that after $C^U(x)$ is finished, it can deterministically construct a nondeterministic TM K which, on input y of length at most $k|x|^k$, runs in time at most $\exp(k'|x|^{k'})$ (for some constant k') and "computes the answer to $y \in A$".

(c) **(3 points.)** Show that $C^?$ can be *further* extended so that $C^U(x)$ constructs the description of a nondeterministic machine N^* (running in some $\exp(k'|x|^{k'})$ time) which, on input x, has the same overall answer as $N^A(x)$.

(d) **(1 point.)** Show that $\mathsf{NP}^{\mathsf{NEXP}} = \mathsf{P}^{\mathsf{NEXP}}$.

5. **(Trading error for advice.)** Show that $\mathsf{RSPACE}(n) \subseteq \mathsf{ZPSPACE}(n)/(n+1)$.

(Now to explain carefully the meaning of these complexity classes. First of all, roughly speaking, $\mathsf{RSPACE}(n)$ is to linear space as RP is to polynomial time; similarly $\mathsf{ZPSPACE}(n)$ and ZPP. However, it turns out there is a major subtlety in defining randomized space classes. The issue is whether you require the randomized machines to *always halt* or just to *halt with probability* 1. These are actually not the same; a randomized machine that flips coins until it gets a 0 and then halts has the property that it "halts with probability 1", but it doesn't "always halt". It turns out that the distinction is unimportant for time-bounded classes, but quite important for space-bounded classes. To make a long story short, the "better" definition turns out to be the one where you require machines to *always halt*, meaning that for every input and every possible sequence of random bits they might flip, they halt in finitely many steps. In fact, once you decide on this definition, it is not too hard to show that for space-$s(n)$ machines you can assume that the machine always halts in at most $2^{O(s(n))}$ steps. (This is pleasant, because it's something we rely on in the deterministic case, too.) Therefore, finally: We say $L \in \mathsf{RSPACE}(s(n))$ if there is a randomized Turing Machine M that, on every input x of length n and every possible sequence of random bits, uses at most $O(s(n))$ space and at most $2^{O(s(n))}$ time, and has the following properties:

$$x \in L \implies \mathbf{Pr}[M(x) \text{ acc.}] \geq 2/3, \quad x \notin L \implies \mathbf{Pr}[M(x) \text{ acc.}] = 0.$$

By the way, the most famous example of this kind of class is RL, randomized log-space. With our definition, this class includes the demand that the algorithm runs in polynomial time. If this were eliminated, and the machine were only required to halt with probability 1, then the resulting class would actually equal NL! This is not too hard to prove, and is entertaining to think about.

Next, as for ZPSPACE(n), please use the following definition: a ZPSPACE(n) machine is a randomized $O(n)$-space, $2^{O(n)}$-time machine that halts on every input and every sequence of random bits, and has three kinds of final states: "accept", "reject", and "?". We say that $L \in$ ZPSPACE(n) if such a machine has the following property: on every input x, the machine *never* outputs a "wrong" answer (i.e., accepts when $x \notin L$, or rejects when $x \in L$); and, on every input x, the probability the machine outputs "?" is at most 1/3.

Finally, ZPSPACE(n)/($n+1$) is the same class, but where the machine takes $n+1$ bits of advice on inputs of length n. That is, $L \in$ ZPSPACE(n)/($n+1$) if there exists a ZPSPACE(n) machine M and sequence of advice strings (a_n) with $|a_n| = n+1$ such that, when provided with $a_{|x|}$ on input x, the machine M has the aforementioned accept/reject/? properties.)

Graduate Computational Complexity Theory **CMU 15-855*, Fall 2017**

HOMEWORK 6

Due: 10:00am, Tuesday October 24

Notation: in these problems, C always denotes a Boolean circuit, and $\#C$ denotes the number of input strings that cause C to output 1. Also, if M is a nondeterministic Turing Machine and x is an input, then $\#M(x)$ denotes the number of accepting nondeterministic computation paths of M on x.

1. **(Derandomized restarting via approximate counting.)** Show that $\mathsf{BPP}_{\text{path}}$ (the class called "$\mathsf{RestartingBPP}$" in Homework #3, Problem 3b) is a subset of the class $\mathsf{P}_{\|}^{\text{ApproxCount}}$. By the latter, we mean the class of all decision problems solvable in polynomial time given the ability to nonadaptively query an oracle for approximate counting (meaning that, when circuit C is submitted to the oracle, it returns a number α such that $\#C \leq \alpha < 2 \cdot \#C$).

   (Remark: In fact, you might like to try to show that $\mathsf{P}_{\|}^{\text{ApproxCount}} \subseteq \mathsf{BPP}_{\text{path}}$, and thus the two classes are equal.)

2. **(An odd problem.)** Recall the complexity class $\oplus\mathsf{P}$: a language L is in the class if and only if there is a polynomial-time nondeterministic Turing Machine M such that $x \in L$ if $\#M(x)$ is odd. As we discussed in class, the Cook–Levin Theorem is "parsimonious", and therefore the language ODD-CIRCUIT-SAT $= \{C : \#C \text{ is odd}\}$ is $\oplus\mathsf{P}$-complete.

   (a) Show that $\oplus\mathsf{P}$ is closed under complement and under intersection.

   (b) Show that $\oplus\mathsf{P}^{\oplus\mathsf{P}} = \oplus\mathsf{P}$. Here, as usual, $\oplus\mathsf{P}^{\oplus\mathsf{P}}$ denotes $\bigcup_{A \in \oplus\mathsf{P}} \oplus\mathsf{P}^{A}$, and $\oplus\mathsf{P}^{A}$ has the same definition as $\oplus\mathsf{P}$ given at the beginning of the problem, except that the machine M has oracle access to language A.

3. **(Mind the gap.)** Recall that $f : \{0,1\}^* \to \mathbb{N}$ is in the class $\#\mathsf{P}$ if there is a polynomial-time nondeterministic Turing Machine M such that $f(x) = \#M(x)$ for all x. We introduce a new function class called GapP, defined to be all $f : \{0,1\}^* \to \mathbb{Z}$ such that there is a polynomial-time nondeterministic Turing Machine M with $f(x) = \Delta M(x)$ for all x, where $\Delta M(x)$ is defined to be the number of accepting paths minus the number of rejecting paths of M on x.

   (a) Show that GapP is the closure of $\#\mathsf{P}$ under subtraction. More precisely, show that $\#\mathsf{P} \subseteq \mathsf{GapP}$, that every $f \in \mathsf{GapP}$ is the difference of two $\#\mathsf{P}$ functions, and that the difference of two GapP functions is in GapP.

   (b) Show that every $f \in \mathsf{GapP}$ can in fact be written as $g - h$, where $g \in \#\mathsf{P}$ and $h \in \mathsf{FP}$. (Here FP is the class of integer-valued functions computable in polynomial time; i.e., those h for which there is a deterministic Turing Machine that on input x, prints out $h(x)$ in binary.) Conclude that $\mathsf{GapP} \subseteq \mathsf{FP}^{\#\mathsf{P}[1]}$.

Graduate Computational Complexity Theory **CMU 15-855*, Fall 2017**

Homework 7

Due: 10:00am, Tuesday October 31

---

1. **(Subset sum.)** Consider the following task: The input is a function $f : 2^{[n]} \to \mathbb{N}$, given explicitly as a table of length $N = 2^n$. (Here $2^{[n]}$ denotes the set of all subsets of $[n] = \{1, 2, \dots, n\}$.) You may assume that each integer $f(S)$ is expressible with $O(n)$ bits. The goal is to output (also in table format) the function $g : 2^{[n]} \to \mathbb{N}$ defined by

$$g(T) = \sum_{S \subseteq T} f(S).$$

Give an algorithm for solving this problem in $N \cdot \mathrm{polylog}(N)$ time (i.e., in $2^n \cdot \mathrm{poly}(n)$ time). You may work in the random-access Turing Machine model (which means you can basically give a "normal" algorithmic description without really worrying about how the data is laid out on TM tapes). Hint: induction/recursion on n.

2. **(Computing a univariate polynomial.)** Let f be any univariate polynomial in X of degree n with complex coefficients. Show that f can be computed by an algebraic circuit that uses at most $2\sqrt{n}$ multiplications, no divisions, and with additions and multiplications by complex scalars being free of charge.

(Remarks: It's possible to improve this to $\sqrt{2n} + \log_2 n + O(1)$; the proof is tricky, but elementary. On the other side, it is known that "almost all" degree-n polynomials need at least $\sqrt{n} - 1$ multiplications to compute, and Strassen showed that the following specific polynomial requires at least $(1 - o(1))\sqrt{n}$ multiplications: $f(X) = 2^{2^n} X + 2^{2^{2n}} X^2 + 2^{2^{3n}} X^3 + \cdots + 2^{2^{n^2}} X^n$.)

3. **(Why determinants are everywhere.)** In this problem you may take for granted the following properties of the determinant: multiplicativity ($\det(AB) = \det(A)\det(B)$); if A' is formed from A by multiplying some row by scalar c, then $\det(A') = c\det(A)$; if A' is formed from A by swapping two rows, then $\det(A') = -\det(A)$; and, cofactor expansion.

For this problem, an algebraic formula F over indeterminates $X_1, \dots, X_n$ and coefficient field K means an algebraic circuit which is a binary tree, with the internal nodes being labeled $\times$ or $+$, and the leaves labeled either with an indeterminate or a scalar from K. The *size* of F is the number of leaves. The goal of this problem is to show the following:

Claim: Any F of size L is expressible by the determinant of a $(3L-1) \times (3L-1)$ matrix A whose entries are either scalars or scalar-times-indeterminates.[1] Furthermore, the matrix A has the following special form:

$$A = \begin{bmatrix} * & * & * \cdots & * & * \\ 1 & * & * \cdots & * & * \\ 0 & 1 & * \cdots & * & * \\ 0 & 0 & 1 \ \ddots & * & * \\ \vdots & \vdots & \ddots \ \ddots & * & * \\ 0 & 0 & 0 \cdots & 1 & * \end{bmatrix}.$$

---

[1] Remark: it is known that this can be improved to $(L+3) \times (L+3)$, one can have just scalars *or* indeterminates, and that one can also replace "determinant" by "permanent".

(That is: arbitrary on and above the main diagonal; all 1's on the diagonal below the main one; and, all 0's below that.)

(a) Prove that for block matrices

$$Z = \left[\begin{array}{c|c} P & 0 \\ \hline Q & R \end{array}\right]$$

it holds that $\det(Z) = \det(P)\det(R)$. Hint: factorize Z using the matrices

$$\left[\begin{array}{c|c} P & 0 \\ \hline Q & I \end{array}\right], \quad \left[\begin{array}{c|c} I & 0 \\ \hline 0 & R \end{array}\right],$$

where I is the identity matrix.

(b) Show that the Claim is true for formulas of size 1 (i.e., single-leaf formulas).

(c) Show that if $F = \det(A)$ and $G = \det(B)$ where A, B are $m \times m$ and $n \times n$ matrices of the special form (respectively), then $F \times G$ is expressible as $\det(C)$ for an $(m+n) \times (m+n)$ matrix of the special form.

(d) Show that if $F = \det(A)$ and $G = \det(B)$ where A, B are $m \times m$ and $n \times n$ matrices of the special form (respectively), then $F + G$ is expressible as $\det(C)$ for an $(m+n+1) \times (m+n+1)$ matrix of the special form. Hint: consider the block matrix

$$C = \left[\begin{array}{c|c|c} 0 & 0\ \ 0\ \ \cdots\ \ 0\ \ 1 & 0\ \ 0\ \ \cdots\ \ 0\ \ 1 \\ \hline 1 & & \\ 0 & & \\ \vdots & A & 0 \\ 0 & & \\ 0 & & \\ \hline 1 & & \\ 0 & & \\ \vdots & 0 & B \\ 0 & & \\ 0 & & \end{array}\right].$$

Show that it works, and that it can be fixed up to the special form with a "swap" or two. . .

(e) Complete the proof of the Claim.

Homework 8

Due: 10:00am, Tuesday November 7

---

1. **(Interactive proofs vs. instance checkers.)** Suppose languages L and $\overline{L}$ have polynomial-round interactive proofs in which Merlin's strategy is implementable in P^L. Show that L has an instance checker. You may use a slightly weaker definition of "instance checker" wherein, if the provided oracle C actually computes L exactly, the checker only has to output the correct answer about $x \overset{?}{\in} L$ with high probability (rather than with probability 1).

2. **(Derandomization implies circuit lower bounds.)** Suppose you wanted to prove $\mathsf{BPP} = \mathsf{P}$. Well, you'd better be able to at least prove $\mathsf{coRP} = \mathsf{P}$. And hence you'd better be able to at least prove that the PIT problem (Polynomial Identity Testing, which we know is in coRP) is in P. And hence you'd better be able to at least prove that it's in NP. And hence you'd better be able to at least prove that it's in $\mathsf{NSUBEXP} \coloneqq \bigcap_{\varepsilon>0} \mathsf{NTIME}(2^{n^\varepsilon})$. In this problem, you'll show this implies that you'd better be able to prove superpolynomial circuit lower bounds.

   In this problem, let $\mathsf{AlgP}^0/\text{poly}$ denote the class of all polynomial-degree families computable by polynomial-size algebraic circuits using $+$, $-$, $\times$ over $\mathbb{Z}$, where the only constants allowed are 0 and 1 (equivalently, where the constants must be of poly(n) bit-length).

   (a) Show that if PERMANENT $\in \mathsf{AlgP}^0/\text{poly}$ and PIT $\in \mathsf{NSUBEXP}$, then $\Sigma_2\mathsf{P} \subseteq \mathsf{NSUBEXP}$. (You can definitely use Valiant's Theorem on $\#\mathsf{P}$-completeness of $\text{PERMANENT}_{0,1}$. You can also use Toda's 1st and 2nd Theorems if you like, though you don't need them.)

   (b) Show that if, furthermore, $\mathsf{NEXP} \subseteq \mathsf{P}/\text{poly}$, then $\Sigma_2\mathsf{P} \subseteq \mathsf{NE} \subseteq \mathsf{SIZE}(n^c)$ for some constant c. (Here $\mathsf{NE} = \mathsf{NTIME}(2^{O(n)})$.)

   (c) Deduce that

   $$\text{PIT} \in \mathsf{NSUBEXP} \quad \Longrightarrow \quad \Big(\text{PERMANENT} \notin \mathsf{AlgP}^0/\text{poly} \quad \vee \quad \mathsf{NEXP} \not\subseteq \mathsf{P}/\text{poly}\Big).$$

3. **(Worst-case hardness to slight hardness-on-average for EXP.)** Suppose that $L \in \mathsf{EXP}$ but L requires superpolynomial-size circuits; more precisely, for all c and all sufficiently large n it holds that there is no Boolean circuit of size n^c computing $L_n : \{0,1\}^n \to \{0,1\}$, the indicator function for presence in $L \cap \{0,1\}^n$.

   (a) Show that there is a language $L' \in \mathsf{E} \coloneqq \mathsf{TIME}(2^{O(n)})$ with the same property.

   (b) Let p stand for the first prime larger than $n+1$ (this can certainly be deterministically computed in poly(n) time, as we'll have $p < 2n$) and write $\mathbb{Z}_p$ for the field of integers modulo p. Show that there is a multilinear polynomial $f_n : \mathbb{Z}_p^n \to \mathbb{Z}_p$, agreeing with L'_n on all inputs in $\{0,1\}^n$, such that the family of functions (f_n) can be computed in $2^{O(n)}$ time.

   (c) Show that for every polynomial-size circuit family (C_n) (where C_n has $n(\log n+1)$ inputs and $\log n + 1$ outputs[1])

   $$\Pr_{\boldsymbol{x} \sim \mathbb{Z}_p^n}[C_n(\boldsymbol{x}) = f_n(\boldsymbol{x})] < 1 - \frac{1}{3n}.$$

---

[1] Here $\log n + 1$ is enough to encode an element of $\mathbb{Z}_p$; I'm too lazy to put ceilings/floors in the right spots here, and you may be equally lazy about this point.

(Hint: recall where this $1 - \frac{1}{3n}$ came up elsewhere in class; also recall $\mathsf{BPP} \in \mathsf{P/poly}$.)

(d) Define a decision problem (language) H as follows: on input $x \in \mathbb{Z}_p^n$ and integer $0 \leq j \leq \log n$, output the jth bit of $f_n(x)$. Show that $H \in \mathsf{E}$, and that for every polynomial-size circuit family (D_n) it holds that

$$\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{x} \sim \mathbb{Z}_p^n \\ \boldsymbol{j} \sim \{0,\dots,\log n\}}} [D_{n'}(\boldsymbol{x}, \boldsymbol{j}) = H(\boldsymbol{x}, \boldsymbol{j})] < 1 - \frac{1}{O(n \log n)}$$

(where $n' = n(\log n + 1) + \log\log n$).

Remark: Thus from a language in EXP that is hard for polynomial-size circuits in the worst case, we may construct a language in E that is slightly hard-on-average for polynomial-size circuits, where "slightly" involves error at least $\frac{1}{O(n')}$ on inputs of length n'.

(Incredibly minor notes: Strictly speaking, we have not quite shown hardness-on-average with respect to the purely uniform distribution on inputs, because of the issue of how exactly to encode the pair $\langle x, j \rangle$ by a single string. Also, strictly speaking, H might be trivial for some input lengths (those not of the appropriate form $n(\log n + 1) + \log\log n$), and we'd rather have it hard for circuits at almost all input lengths. Both issues are easy and boring to fix.)

Homework 9
Due: 10:00am, Tuesday November 14

Recall that a Boolean formula F is a binary tree, where the internal nodes are labeled with $\vee$ or $\wedge$, and the leaves are labeled by either literals x_i or $\overline{x}_i$, or by constants 0 or 1. It computes a Boolean function f in the natural way. The *size* of a Boolean formula is the number of literal-leaves. (Constant-leaves are "free".) The least possible size of a formula computing f is denoted $L(f)$.

You may take it for granted that there is a "simplification" operation on formulas that: (i) preserves the function being computed; (ii) gets rid of all constant-leaves (except when the function is itself a constant function, in which case the formula becomes a single constant-leaf); (iii) does not increase the size of the formula. This simplification operation just does the obvious thing: if a 1 enters into an $\vee$ gate, the gate is replaced by 1; if a 1 enters into an $\wedge$ gate, the gate is replaced by its other child; similarly for 0's.

1. **(Random restrictions shrink formulas.)**
   (a) Argue that if F is a Boolean formula, there is an equivalent Boolean formula F' of no larger size with the following property: in F', whenever some internal node has one child a literal $x_i/\overline{x}_i$ and the other child a subformula G, the literals $x_i/\overline{x}_i$ do not appear in G.
   (b) Suppose f is an n-variable Boolean function with $L(f) > 1$. Let ρ be a random restriction formed by fixing exactly one (randomly chosen) variable (to a uniformly random 0/1 value). Show that $\mathbf{E}[L(f|_\rho)] \leq (1 - \frac{1.5}{n}) \cdot L(f)$. (Hint: getting $(1 - \frac{1}{n}) \cdot L(f)$ should be easy. If a variable gets fixed, think about what might happen to its sibling-subformulas. Technically, you will need part (a) here.)
   (c) Taking for granted that $(1 - \frac{1.5}{n}) \leq (1 - \frac{1}{n})^{1.5}$, show the following: If f is an n-variable Boolean function, and ρ is a random restriction formed by fixing exactly $n-k$ (randomly chosen) variables, then

$$\mathbf{E}[L(f|_\rho)] \leq \max\left\{\left(\frac{k}{n}\right)^{1.5} \cdot L(f), 1\right\}.$$

   (d) Use this (i.e., no fair citing problem 3(c)), with $k = 2$, to prove that $L(\text{Parity}_n) \geq n^{1.5}$.

2. **(Alice and Bob and Parity I.)** A *rectangle* is a set $A \times B$, where $A, B \subseteq \{0,1\}^n$ are disjoint. For $i \in [n]$, we call it *i-colorable* if $x_i \neq y_i$ for all pairs of strings $x \in A$, $y \in B$; we call it *colorable* if it is i-colorable for some i. If a rectangle R can *partitioned* into s (sub)rectangles, each of which is colorable, we say that R is s-tileable. We write $\chi(R)$ for the least s such that R is s-tileable. If f is a Boolean function, we write $\chi(f)$ for $\chi(f^{-1}(0) \times f^{-1}(1))$.
   (a) Prove that $\chi(\text{Parity}_2) = 4$ and $\chi(\text{And}_3) = 3$; draw figures to illustrate the upper bounds.
   (b) Prove that $\chi(f) \leq L(f)$. (Hint: induction.)

3. **(Alice and Bob and Parity II.)** Continuing the previous problem...
   (a) Let $R = f^{-1}(0) \times f^{-1}(1)$. Say we "mark" each entry $(x, y) \in R$ where the Hamming distance between x and y is 1. For a subrectangle $A \times B$ of R, write $M(A \times B)$ for the number of marked entries in it. Show that if $A \times B$ is colorable, then $M(A \times B) \leq \min\{|A|, |B|\} \leq \sqrt{|A| \cdot |B|}$.

(b) Show that $M(R) \leq \sqrt{\chi(f)}\sqrt{|f^{-1}(0)| \cdot |f^{-1}(1)|}$.

(c) Show that $L(\text{Parity}_n) \geq n^2$.

(d) Show that $L(\text{Parity}_n) = n^2$ when n is a power of 2.

Homework 10

Due: 10:00am, Tuesday November 21

In this homework, you may wish to consult Lecture 4. Also, you may take for granted the following result, which you basically proved in Homework 9.1:

Theorem. *Let $f : \{0,1\}^n \to \{0,1\}$, let $\varepsilon \in [\frac{1}{n}, 1]$, and let ρ be an ε-random restriction. (Recall this means each coordinate is independently set to '$\star$' (unfixed) with probability ε, and is otherwise set to 0 or 1 with probability $\frac{1-\varepsilon}{2}$ each.) Then*

$$\mathbf{E}[L(f|_\rho)] \le 2\varepsilon^{1.5} L(f) + 1,$$

where, recall, $L(g)$ is the minimum size of a Boolean formula computing g.

(In fact, Johan Håstad and Avishay Tal have shown that $\mathbf{E}[L(f|_\rho)] \le O(\varepsilon^2)L(f) + O(1)$.)

1. **(More on shrinking formulas.)** Let $b > 1$, $m = 2^b$, $n = bm$. Given some Boolean function $\psi : \{0,1\}^b \to \{0,1\}$, define the function $f_\psi : \{0,1\}^n \to \{0,1\}$ as follows: Think of $x \in \{0,1\}^n$ as being divided into b "blocks" of m bits each. Then $f_\psi(x) = \psi(z_1, \dots, z_b)$, where z_i is the parity (XOR) of the ith block of bits in x.
   (a) Let $\varepsilon = \frac{b\ln(3b)}{n}$ and let ρ be an ε-random restriction on $n = bm$ variables. Show that with probability at least 2/3, the restriction ρ gives at least one $\star$ to each of the b blocks.
   (b) Show that there exists a restriction σ of the n coordinates such that both of the following hold: (i) $L(f_\psi|_\sigma) \le 6(\frac{b\ln(3b)}{n})^{1.5}L(f_\psi) + 3$; (ii) σ gives at least one $\star$ to each of the b blocks.
   (c) Show that $L(f_\psi) \ge \widetilde{\Omega}(n^{1.5})(L(\psi) - O(1))$. Deduce that there exists ψ such that $L(f_\psi) \ge \widetilde{\Omega}(n^{2.5})$.

2. **(Andreev's function.)**
   (a) Does the function L_ψ produced in the previous problem count as "explicit"?[1] Anyway, let us define an explicit function $\alpha : \{0,1\}^{n+m} \to \{0,1\}$, as follows: $\alpha(x,y) = f_y(x)$, where $x \in \{0,1\}^n$, $y \in \{0,1\}^m$ is interpreted as the truth-table of a function $\{0,1\}^b \to \{0,1\}$, and f_y refers to the "f_ψ" notation from the previous question. Show that $L(\alpha) \ge \widetilde{\Omega}(n^{2.5})$.

      Remark. *Using the Håstad–Tal result, one can deduce that in fact $L(\alpha) \ge n^3/\widetilde{O}(\log^3(n))$.*

   (b) Show that $L(\alpha) \le O(n^3/\log^2 n)$. (Bonus: show that $L(\alpha) \le O(n^3/\log^3 n)$.)

3. **(Detecting triangles.)** Prove that any monotone circuit that detects whether a v-vertex graph (given by its $v \times v$ adjacency matrix) contains a triangle must have size at least $v^3/\text{polylog}(v)$. (Hint: complete bipartite graphs contain no triangles.)

---

[1] This is a rhetorical question; you are not required to provide an answer.

Homework 11

Due: 10:00am, Thursday November 30

---

1. **(Just mod 6 things.)**
   (a) Let $f : \{0,1\}^n \to \{0,1\}$ and let p be a prime. As you showed in HW8.3(b), there is a multilinear polynomial $F(x_1, \dots, x_n)$ over $\mathbb{F}_p$ such that $F(x) = f(x)$ for all $x \in \{0,1\}^n$. Show that such a multilinear representation is unique. (Hint: if $F_1(x) = F_2(x)$, key in on the least-degree nonzero monomial in $F_1(x) - F_2(x)$.) Deduce that any multilinear polynomial over $\mathbb{F}_p$ computing the AND function must have degree n.
   (b) Show that AND functions cannot be computed by constant-depth circuits (of *arbitrary* size) consisting only of input gates, the constant 1 gate, and mod_p gates, where p is a fixed prime. Recall that a mod_m gate outputs 0 or 1 depending on whether the number of input 1's is zero or nonzero modulo m. (Hint: show that such a circuit computes a polynomial of constant degree.)
   (c) Show that AND functions *can* be computed by depth-2 circuits (albeit of exponential size) consisting only of input gates, the constant 1 gate, and mod_6 gates. (Hint: first show how to get mod_3 and mod_2 gates; then show that if you take the mod_2 of *every* subset of the inputs, then mod_3-together the 2^n results, you basically get the OR function.)

   Remark: It is open to show that AND is not computable by depth-3, poly-size circuits consisting only of mod_6 gates. It is also open to show this about SAT.

2. **(Circuit lower bounds for Permanent.)** Prove that the Permanent function (of integer matrices) is not computable by (uniform) ACC circuits, even with $2^{n^{o(1)}}$ size. You may take for granted the following facts: (i) the Time Hierarchy Theorem holds relative to any oracle; (ii) many reductions in classic complexity theorems (e.g., the Cook–Levin Theorem, Valiant's $\#\mathsf{P}$-completeness of Permanent for integer matrices, ...) can be carried out in (uniform) AC^0.

   Remark: In fact, it has been shown that Permanent is not even in the larger circuit class of (uniform) TC^0: namely, $O(1)$-depth poly-size circuits of Majority gates.

3. **(Fighting perebor for ACC-SAT.)** In this problem, your algorithms may be in the random-access Turing Machine model.
   (a) Show that there is a $2^m \cdot \text{poly}(m)$ time algorithm for deciding whether a given m-input, "size-$2^{\sqrt{m}}$ SYM+ circuit" is satisfiable. Recall that such a circuit is of the form $h(p(x_1, \dots, x_m))$, where p is a multilinear polynomial given by the sum of at most $2^{\sqrt{m}}$ monomials (each of degree at most $\sqrt{m}$) and h is an explicitly given function $\{0, 1, 2, \dots, 2^{\sqrt{m}}\} \to \{0,1\}$. (Hint: you may appeal to a problem from Homework 7.)
   (b) Fix a depth $d \in \mathbb{N}^+$ and a modulus r. Show that for a sufficiently small constant $\delta > 0$, there is a $2^m \cdot \text{poly}(m)$ time algorithm for deciding whether a given m-input, depth-$(d+1)$, size-$2^{O(m^\delta)}$ $\mathsf{AC}^0[r]$ circuit is satisfiable. (Hint: you may appeal to theorems from class.)
   (c) Show that there is a $2^{n-\Omega(n^\delta)}$ time algorithm for deciding whether a given n-input, depth-d, size-2^{n^δ} $\mathsf{AC}^0[r]$ circuit is satisfiable. (Hint: given C, consider C' which is an OR over all possible settings to the first n^δ variables of C.)

TEST #2

Due: 10:00am, Thursday December 7

No collaboration or Internet-usage allowed!
You may cite results from lecture, past homework problems, and the textbook.

Solve four problems total, namely #1, #2, and two out of three from $\{\#3, \#4, \#5\}$.

---

1. **(PH collapsing when efficient means expected polynomial time.)**

   Show $\mathsf{NP} \subseteq \mathsf{ZPP} \implies \mathsf{PH} = \mathsf{ZPP}$.

   (Hint: don't be surprised if your proof fits on one line.)

2. **(Optimal Karp–Lipton for NEXP.)**

   (a) (This part is worth 1 point, as the proof is about one sentence long.) Read the proof of the IKW Theorem, Lemma 20.20 in the textbook, which uses the "easy witness method" to show that $\mathsf{NEXP} \subseteq \mathsf{P}/\text{poly} \implies \mathsf{NEXP} = \mathsf{EXP}$. Now show that in fact $\mathsf{NEXP} \subseteq \mathsf{P}/\text{poly} \implies \mathsf{NEXP} = \mathsf{MA}$.

   (b) In lecture we focused on showing that strong hardness assumptions imply deterministic poly-time algorithms for BPP; but, we mentioned that if one works the parameters, one gets that weak hardness assumptions imply deterministic subexponential-time algorithms for BPP. Specifically, one can show that if there is a language $L \in \mathsf{EXP}$ that requires superpolynomial circuit size for almost all input lengths n, then for all $\varepsilon > 0$ there is a pseudorandom generator G with seed length $\ell(n) \leq n^{\varepsilon}$. Under this assumption, conclude that $\mathsf{MA} \subseteq \mathsf{NTIME}(2^n)$.

   (c) The above says that if EXP requires superpolynomial circuit size for almost all input lengths, then MA is nondeterministically simulable in $O(2^n)$ time for almost all n. You may now take it for granted that the "infinitely often" version is also true (the proof is essentially the same); namely, that $\mathsf{EXP} \not\subseteq \mathsf{P}/\text{poly} \implies \mathsf{MA} \subseteq \text{i.o.-}\mathsf{NTIME}(2^n)$. Here i.o.-$\mathcal{C}$ denotes the class of all languages A such that there exists $B \in \mathcal{C}$ with $A \cap \{0,1\}^n = B \cap \{0,1\}^n$ for infinitely many n.

   You may also take for granted (cf. Homework 2, #1(d)) the following Time Hierarchy Theorem result: for all $c \in \mathbb{N}$ it holds that $\mathsf{EXP} \not\subseteq \text{i.o.-}\mathsf{TIME}(2^{n^c})$. (Remark: we do not know the nondeterministic version of this result.)

   Now prove the following: $\mathsf{NEXP} = \mathsf{MA} \implies \mathsf{NEXP} \subseteq \mathsf{P}/\text{poly}$.

3. **(One-way functions and complexity classes.)** A "worst-case one-way function" is a function $f : \{0,1\}^* \to \{0,1\}^*$ with the following properties: (i) f is one-to-one (injective); (ii) f does not stretch or shrink by more than a polynomial amount, i.e., there exists $k > 0$ such that $|x|^{1/k} \leq |f(x)| \leq |x|^k$ for all x; (iii) f is computable in polynomial time; (iv) the inverse function $f^{-1} : \{0,1\}^* \to (\{0,1\}^* \cup \{\perp\})$ is *not* computable in polynomial time, where $f^{-1}(y)$ is defined to be x if $f(x) = y$, or else $\perp$ if $y \notin \text{range}(f)$.

The complexity class UP (not its real name) is defined to be the set of all languages L for which there exists a polynomial-time nondeterministic Turing Machine M with the following properties: (i) if $x \in L$ then $M(x)$ accepts on exactly one "nondeterministic branch"; (ii) if $x \notin L$ then $M(x)$ accepts on exactly zero "nondeterministic branches". As a remark, it is immediate that $\mathsf{UP} \subseteq \mathsf{NP}$, and it's also easy to see that $\mathsf{P} \subseteq \mathsf{UP}$.

(a) Prove that if $\mathsf{UP} \neq \mathsf{P}$ then there is a worst-case one-way function.

(b) Conversely, prove that if $\mathsf{UP} = \mathsf{P}$ then worst-case one-way functions do not exist.

4. **(O1.)** Remember that complexity class "$\mathsf{S_2P}$" from Homework 5, Problem 1? Here we describe a variant of it called "$\mathsf{O_2P}$". The class $\mathsf{O_2P}$ is just like $\mathsf{S_2P}$ except Yolanda and Zeyuan are too lazy to even look at the input x; they only look at its length, n. More precisely, we say that $L \in \mathsf{O_2P}$ if there is a polynomial $p(n)$ and a polynomial-time algorithm V such that for all n, there exist strings $y^*, z^* \in \{0,1\}^{p(n)}$ such that for all $x \in \{0,1\}^n$,

$$\begin{aligned} x \in L &\implies \forall z \in \{0,1\}^{p(n)}\ V(x, y^*, z) = 1, \\ x \notin L &\implies \forall y \in \{0,1\}^{p(n)}\ V(x, y, z^*) = 0. \end{aligned}$$

Prove that $\mathsf{BPP} \subseteq \mathsf{O_2P}$.

5. **(O2: Revenge of Karp–Lipton.)**

(a) Show that $\mathsf{NP} \subseteq \mathsf{P}/\text{poly} \implies \mathsf{PH} = \mathsf{O_2P}$.

(b) Show that $\mathsf{PH} = \mathsf{O_2P} \implies \mathsf{NP} \subseteq \mathsf{P}/\text{poly}$.