The Power of Self-Reducibility: Selectivity, Information, and Approximation[1]

Lane A. Hemaspaandra
Dept. of Comp. Sci., Univ. of Rochester

February 21, 2019 (last revised March 3, 2019)

In memory of Ker-I Ko (2015–2018), whose indelible contributions to computational complexity included important work on each of this talk's topics: self-reducibility, selectivity, information, and approximation.

[1] This is a set of slides to accompany the book chapter, "The Power of Self-Reducibility: Selectivity, Information, and Approximation," by Lane A. Hemaspaandra, in *Complexity and Approximation*, eds. Ding-Zhu Du and Jie Wang, Springer, in preparation, or to serve as the basis for a stand-alone lecture or two-lecture series. A preliminary version of that chapter appears under the same title as arXiv.org technical report 1902.08299.

What Will The Year Be About?

It is always hard to know what a year will be about. However, as an example, in February 2019, I looked to see what predictions there were for what that year might be about. And I found the following predictions.

Future years might differ somewhat, especially regarding the car and house predictions. But overall, this is probably a pretty typical set of predictions for any year.

What Will The Year Be About?

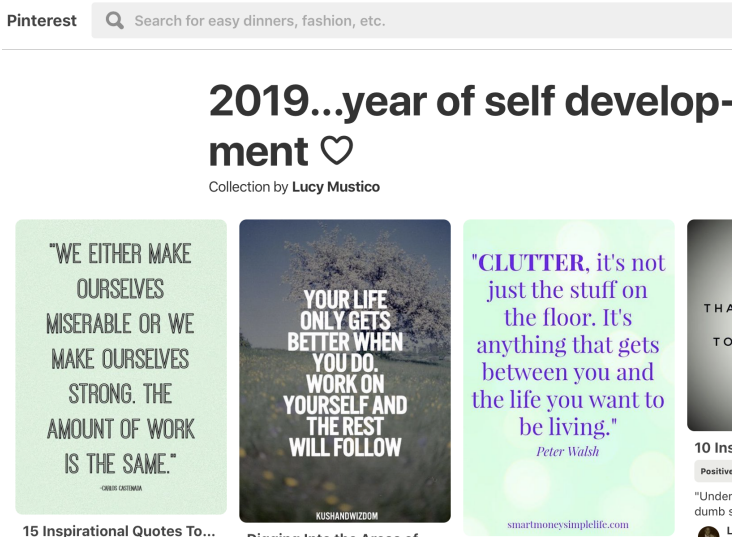
2019: the Year of Self-Improvement

BY THEODORA · PUBLISHED FEBRUARY 6, 2019 · UPDATED FEBRUARY 6, 2019

What Will the Year Be About?

15 Simple Ways to Make 2019 the Year of Self-Care

What Will the Year Be About?



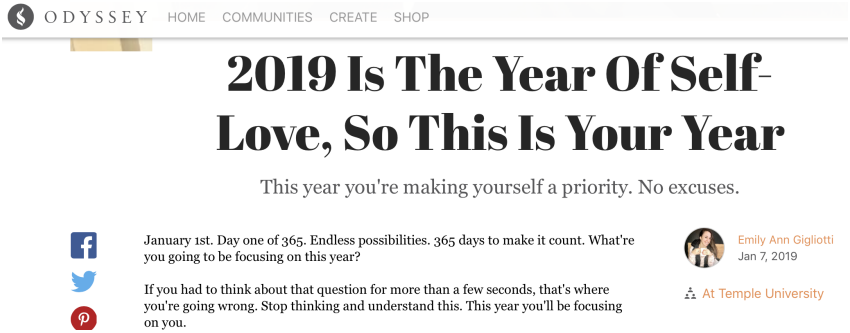
What Will the Year Be About?

JELIFESTYLE Extraordinary Things Happen Here: We Live By Design

2019 - "THE YEAR OF YOUR BEST SELF!"

JELifestyle has declared **2019 as "The Year of Your Best Self!"** It's time for you to **upgrade** in **every area** of your life to becoming your best from the inside out. We are here to help you do the work by equipping you with the information and tools needed to help you transform into your higher self. What are your goals for the new year? Do you have resolutions? At JELifestyle we don't believe in "new years resolutions." In the words of Yoda from the Star Wars Franchise, you either **"Do or Don't Do. There is no 'try.'"** We are going to **do the work** to help you transform your "ordinary" into the "extraordinary" in your **Career, Health, Nutrition, Fitness, Finances, Relationships, and Wellness** which includes self-care, mental, emotional and spiritual well-being —**The 'Whole-y 7' of Wholeness.** We are going to explore the best practices, techniques, strategies and recommendations

What Will the Year Be About?



What Will the Year Be About?

What Will the Year Be About?

Visionary Finance

Call Now: London 0207 100 4

HOME ABOUT US RESIDENTIAL MORTGAGES BUY TO LET FOREIGN INVESTO

Home » News » Is 2019 the year of self-build homes?

IS 2019 THE YEAR OF SELF-BUILD HOMES?

January 31st, 2019

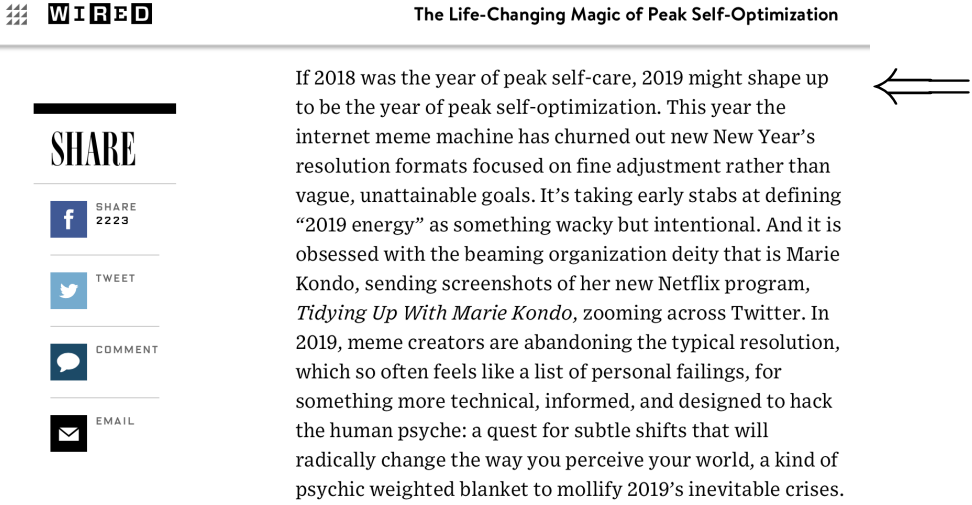
The idea of building your dream home from scratch is an attractive proposition for potential home owners of all ages from all walks of life.

The number of self-build completions have risen every year for the past six years and with new mortgage products coming onto the market that trend is more than likely to continue in 2019.

Why 2019 is the year of data self-sufficiency

Chad Ledford · October 26, 2018 · AddShoppers News, Client Success, Identity Resolution, Behavioral Targeting, Machine Learning + AI

What Will the Year Be About?



WIRED

The Life-Changing Magic of Peak Self-Optimization

SHARE

SHARE 2223

TWEET

COMMENT

EMAIL

If 2018 was the year of peak self-care, 2019 might shape up to be the year of peak self-optimization. This year the internet meme machine has churned out new New Year's resolution formats focused on fine adjustment rather than vague, unattainable goals. It's taking early stabs at defining "2019 energy" as something wacky but intentional. And it is obsessed with the beaming organization deity that is Marie Kondo, sending screenshots of her new Netflix program, *Tidying Up With Marie Kondo*, zooming across Twitter. In 2019, meme creators are abandoning the typical resolution, which so often feels like a list of personal failings, for something more technical, informed, and designed to hack the human psyche: a quest for subtle shifts that will radically change the way you perceive your world, a kind of psychic weighted blanket to mollify 2019's inevitable crises.

What Will the Year Be About?

There seems to some agreement among these varied predictions: "self"!

I can't predict what "self-" theme this year will be the year of for you.

What Will the Year Be About?

There seems to some agreement among these varied predictions: "self"!

I can't predict what "self-" theme this year will be the year of for you.

But I hope to make today be your Day of Self-Reducibility!

What Will the Year Be About?

There seems to some agreement among these varied predictions: "self"!

I can't predict what "self-" theme this year will be the year of for you.

But I hope to make today be your Day of Self-Reducibility!

And, beyond that, I hope you'll keep the tool/technique of self-reducibility in mind for the rest of your year, decade, and lifetime—and on each new challenge will spend at least a few moments asking, "Can self-reducibility play a helpful role in my study of this problem?" And with luck, sooner or later, the answer may be, "Yes! Wow... what a surprise!"

What Will the Year Be About?

There seems to some agreement among these varied predictions: "self"!

I can't predict what "self-" theme this year will be the year of for you.

But I hope to make today be your Day of Self-Reducibility!

And, beyond that, I hope you'll keep the tool/technique of self-reducibility in mind for the rest of your year, decade, and lifetime—and on each new challenge will spend at least a few moments asking, "Can self-reducibility play a helpful role in my study of this problem?" And with luck, sooner or later, the answer may be, "Yes! Wow... what a surprise!"

So... let us define self-reducibility, and then set you to work, in teams, on using it to solve some famous, important problems (whose solutions via self-reducibility indeed are already known... but this will be a workshop-like "talk," with the goal of each of you becoming hands-on familiar with using self-reducibility in proofs).

Overview

Introduction: Sit Back and Relax

Game Plan

For each of a few challenge problems (theorems), I'll give you definitions and perhaps some other background, and then the state challenge problem (theorem), and then you in groups will spend the (verbally) mentioned amount of time trying to prove the challenge problem. And then we will go over an answer from one of the groups that solved the problem (or if none did, we'll together to get to an answer).

Note

You don't have to take notes on the slides, since during each challenge problem, I'll leave up a slide that summarizes the relevant definitions/notions that have been presented up to that point in the talk, and the challenge question.

Introduction: SAT and Self-Reducibility

SAT is the set of all satisfiable (propositional) Boolean formulas. For example, $x \wedge \overline{x} \notin \mathrm{SAT}$ but $(x_1 \wedge x_2 \wedge \overline{x_3}) \vee (x_4 \wedge \overline{x_4}) \in \mathrm{SAT}$.

Introduction: SAT and Self-Reducibility

SAT is the set of all satisfiable (propositional) Boolean formulas. For example, $x \wedge \overline{x} \notin \mathrm{SAT}$ but $(x_1 \wedge x_2 \wedge \overline{x_3}) \vee (x_4 \wedge \overline{x_4}) \in \mathrm{SAT}$. SAT has the following "divide and conquer" property.

Fact (2-disjunctive length-decreasing self-reducibility)

Let $k \geq 1$. Let $F(x_1, x_2, \ldots, x_k)$ be a Boolean formula (wlog assume that each of the variables actually occurs in the formula). Then

$$F(x_1, x_2, \ldots, x_k) \in \mathrm{SAT} \iff \big(F(\mathrm{True}, x_2, \ldots, x_k) \in \mathrm{SAT} \vee F(\mathrm{False}, x_2, \ldots, x_k) \in \mathrm{SAT}\big).$$

The above says that SAT is self-reducible (in particular, in the lingo, it says that SAT is 2-disjunctive length-decreasing self-reducible).

Introduction: SAT and Self-Reducibility

SAT is the set of all satisfiable (propositional) Boolean formulas. For example, $x \wedge \overline{x} \notin \mathrm{SAT}$ but $(x_1 \wedge x_2 \wedge \overline{x_3}) \vee (x_4 \wedge \overline{x_4}) \in \mathrm{SAT}$. SAT has the following "divide and conquer" property.

Fact (2-disjunctive length-decreasing self-reducibility)

Let $k \geq 1$. Let $F(x_1, x_2, \ldots, x_k)$ be a Boolean formula (wlog assume that each of the variables actually occurs in the formula). Then

$$F(x_1, x_2, \ldots, x_k) \in \mathrm{SAT} \iff \big(F(\mathrm{True}, x_2, \ldots, x_k) \in \mathrm{SAT} \vee F(\mathrm{False}, x_2, \ldots, x_k) \in \mathrm{SAT}\big).$$

The above says that SAT is self-reducible (in particular, in the lingo, it says that SAT is 2-disjunctive length-decreasing self-reducible).

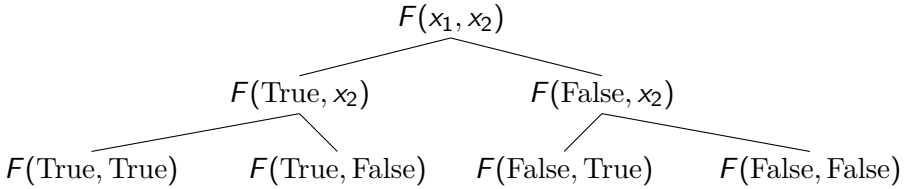
Note

We typically won't focus on references in this talk. But just to be explicit: none of the notions/theorems in this talk, other than in Challenge 4, are due to me. FYI, self-reducibility dates back to, from the 1970s, Schnorr (ICALP) and Meyer & Paterson (an MIT TR).

Note

We won't at all focus here on details of the encoding of formulas and other objects.

Introduction: SAT and Self-Reducibility



The above is what is called a self-reducibility tree. We know for each nonleaf node that it is satisfiable iff at least one of its children is satisfiable. (Inductively, the root is iff some leaf is. And of course that is clear—the leaves are enumerating all possible assignments!) But wait... the tree can be exponentially large in the number of variables, and so we can't hope to build fast algorithms to brute-force explore it.

But rather magically—and this is central to all the challenge problems—one can often find ways to solve problems via exploring just a very small portion of this tree. Tree-pruning will be the order of the day during this talk! So please do keep this tree, and the need to prune it, closely in mind!

Challenge 1: Is SAT P-Selective (i.e., Is SAT Semi-Feasible)?

Challenge 1: Definitions

A set is said to be feasible (in the sense of belonging to P) if there is a poly-time algorithm that decides membership.
A set is said to be semi-feasible (aka P-selective) if there is a poly-time algorithm that semi-decides membership, i.e., that given any two strings, outputs one that is "more likely" (to be formally cleaner, since the probabilities are all 0 and 1 and can tie, what is really meant is "no less likely") to be in the set.

Definition (Selman)

A set L is P-selective if there exists a poly-time function, $f : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that,

$$(\forall a, b \in \Sigma^*)[f(a, b) \in \{a, b\} \wedge \big(\{a, b\} \cap L \neq \emptyset \implies f(a, b) \in L\big)].$$

Note: P-selective sets can be hard! There exist undecidable sets that are P-selective (e.g., the set of left cuts of the real number implicit in the characteristic function of the halting problem).

Challenge 1: Can SAT Be P-Selective?

Challenge Problem

(Prove that) if SAT is P-selective, then $\mathrm{SAT} \in \mathrm{P}$.

Challenge 1: Can SAT Be P-Selective?

Challenge Problem

(Prove that) if SAT is P-selective, then $\mathrm{SAT} \in \mathrm{P}$.

So that you have them easily at hand while working on this, here are some of the definitions and tools from previous slides:

SAT SAT is the set of all satisfiable (propositional) Boolean formulas.

Self-reducibility Let $k \geq 1$. Let $F(x_1, x_2, \ldots, x_k)$ be a Boolean formula (wlog assume that each of the variables occurs in the formula). Then $F(x_1, x_2, \ldots, x_k) \in \mathrm{SAT} \iff \big(F(\mathrm{True}, x_2, \ldots, x_k) \in \mathrm{SAT} \vee F(\mathrm{False}, x_2, \ldots, x_k) \in \mathrm{SAT}\big)$.

P-selectivity A set L is P-selective if there exists a poly-time function, $f : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that, $(\forall a, b \in \Sigma^*)[f(a, b) \in \{a, b\} \wedge \big(\{a, b\} \cap L \neq \emptyset \implies f(a, b) \in L\big)]$.

Challenge 2: Can SAT Reduce to a Tally Set?

Challenge 2: Can SAT Reduce to a Tally Set?

Can SAT (or more generally, can any NP-complete set) have low information content? To answer that, one needs to formalize what notion of low information content one wishes to study. One such notion is whether a given set can many-one polynomial-time reduce to a tally set (a set over a 1-letter alphabet).

Challenge 2: Definitions

- ϵ will denote the empty string. A set T is a tally set if $T \subseteq \{\epsilon, 0, 00, 000, \dots\}$.
- We say that $A \leq_m^p B$ (A many-one polynomial-time reduces to B) if there is a polynomial-time function g such that,

$$(\forall x \in \Sigma^*)[x \in A \iff g(x) \in B].$$

  (Informally, this says that B is so powerful that each membership query to A can be efficiently transformed into a membership query to B.)
- The complexity class NP is $\{L \mid L \leq_m^p \mathrm{SAT}\}$. (Note: NP more commonly is defined as the class of sets accepted by nondeterministic, polynomial-time Turing machines. But that in fact yields the same class of sets as the alternate definition just given.)
- A set H is said to be hard for a class $\mathcal{C}$ if for each set $L \in \mathcal{C}$ it holds that $L \leq_m^p H$. (If in addition $H \in \mathcal{C}$, then we say that H is $\mathcal{C}$-complete.)

Challenge 2: Can SAT Reduce to a Tally Set?

Challenge Problem (Berman)

(Prove that) if there exists a tally set T such that $\mathrm{SAT} \leq_m^p T$, then $\mathrm{SAT} \in \mathrm{P}$. (Equivalently, if there exists an NP-hard tally set, then $\mathrm{P} = \mathrm{NP}$).

Challenge 2: Can SAT Reduce to a Tally Set?

Challenge Problem (Berman)

(Prove that) if there exists a tally set T such that $\mathrm{SAT} \leq_m^p T$, then $\mathrm{SAT} \in \mathrm{P}$. (Equivalently, if there exists an NP-hard tally set, then $\mathrm{P} = \mathrm{NP}$).

So that you have them easily at hand while working on this, here are some of the definitions and tools from previous slides:

SAT SAT is the set of all satisfiable (propositional) Boolean formulas.

Self-reducibility Let $k \geq 1$. Let $F(x_1, x_2, \ldots, x_k)$ be a Boolean formula (wlog assume that each of the x_i actually occurs in the formula). Then $F(x_1, x_2, \ldots, x_k) \in \mathrm{SAT} \iff \big(F(\mathrm{True}, x_2, \ldots, x_k) \in \mathrm{SAT} \vee F(\mathrm{False}, x_2, \ldots, x_k) \in \mathrm{SAT}\big)$.

Tally sets A set T is a tally set if $T \subseteq \{\epsilon, 0, 00, 000, \ldots\}$.

Many-one reductions We say that $A \leq_m^p B$ if there is a polynomial-time function g such that, $(\forall x \in \Sigma^*)[x \in A \iff g(x) \in B]$.

Challenge 3: Can $\overline{\mathrm{SAT}}$ Reduce to a Sparse Set?

Challenge 3: Definitions

- Let $\|S\|$ denote the cardinality of set S, e.g., $\|\{\epsilon, 0, 0, 0, 00\}\| = 3$.
- Let $|x|$ denote the length string x, e.g., $|\text{moon}| = 4$. (Used in Challenge 4: $|F|$ will denote the length of (the encoding of) formula F.)
- A set S is sparse if there exists a polynomial q such that, for each natural number $n \in N$, it holds that $\|\{x \mid x \in S \wedge |x| \leq n\}\| \leq q(n)$. (Informally put, the sparse sets are the sets whose number of strings up to a given length is at most polynomial.)

Example (Example)

$\{0,1\}^*$ is, for example, not a sparse set; but all tally sets are sparse, indeed all via the polynomial $q(n) = n + 1$.

- For any set L, let $\overline{L}$ denote the complement of L.
- $\text{coNP} = \{L \mid \overline{L} \in \text{NP}\}$.

Challenge 3: Can $\overline{\mathrm{SAT}}$ Reduce to a Sparse Set?

Challenge Problem (Fortune)

(Prove that) if there exists a sparse set S such that $\overline{\mathrm{SAT}} \leq_m^p S$, then $\mathrm{SAT} \in \mathrm{P}$.
(Equivalently, if there exists a coNP-hard sparse set, then $\mathrm{P} = \mathrm{NP}$).

Challenge 3: Can $\overline{\text{SAT}}$ Reduce to a Sparse Set?

Challenge Problem (Fortune)

(Prove that) if there exists a sparse set S such that $\overline{\text{SAT}} \leq_m^p S$, then $\text{SAT} \in \text{P}$.
(Equivalently, if there exists a coNP-hard sparse set, then $\text{P} = \text{NP}$).

So that you have them easily at hand, here are some of the defs/tools from previous slides:

SAT and Complements SAT is the set of all satisfiable (propositional) Boolean formulas. $\overline{\text{SAT}}$ denotes the complement of SAT. $\text{coNP} = \{L \mid \overline{L} \in \text{NP}\}$.

Self-reducibility Let $k \geq 1$. Let $F(x_1, x_2, \ldots, x_k)$ be a Boolean formula (wlog assume that each of the x_i actually occurs in the formula). Then $F(x_1, x_2, \ldots, x_k) \in \text{SAT} \iff \big(F(\text{True}, x_2, \ldots, x_k) \in \text{SAT} \vee F(\text{False}, x_2, \ldots, x_k) \in \text{SAT}\big)$.

Sparse sets A set S is sparse if there exists a polynomial q such that, for each natural number $n \in N$, it holds that $\|\{x \mid x \in S \wedge |x| \leq n\}\| \leq q(n)$.

Many-one reductions We say that $A \leq_m^p B$ if there is a polynomial-time function g such that, $(\forall x \in \Sigma^*)[x \in A \iff g(x) \in B]$.

Challenge 4: Is #SAT as Hard to (Enumeratively) Approximate as It Is to Solve Exactly?

Challenge 4: Definitions

This final challenge is harder than the other ones. You'll in solving it have to have multiple insights—as to what approach to use, what building blocks to use, and how to use them.

- $\#\mathrm{SAT}$ is the function that given as input a Boolean formula $F(x_1, x_2, \ldots, x_k)$—wlog assume each of the variables appears in F—outputs the number of satisfying assignments the formula has (i.e., of the 2^k possible assignments of the variables to True/False, the number of those under which F evaluates to True; so the output will be a natural number in the interval $[0, 2^k]$). For example, $\#\mathrm{SAT}(x_1 \vee x_2) = 3$ and $\#\mathrm{SAT}(x_1 \wedge \overline{x_1}) = 0$.
- We say that $\#\mathrm{SAT}$ has a polynomial-time 2-enumerator (aka, is p-time 2-enumerably approximable) if there is a polynomial-time function h such that on each input x, (a) $h(x)$ outputs a list of two (perhaps identical) natural numbers, and (b) $\#\mathrm{SAT}(x)$ appears in the list output by $h(x)$.
  (Informally put, such a function h outputs a list of (at most) two candidate values for the value of $\#\mathrm{SAT}$ on the given input, and the actual output is always in that list. This notion generalizes in the natural way to other list cardinalities, e.g., $\max(1, \sqrt{|F|})$-enumerators and poly-enumerators.)

Challenge 4: Food for Thought

- You'll certainly want to use some analogue of the key self-reducibility observation, except now respun by you to be about the number of solutions of a formula and how it relates or is determined by the number of solutions of its two "children" formulas.
- But doing that is just the first step of your quest. So... please play around together with your groupmates with ideas and approaches. Don't be afraid to be bold and ambitious. For example, your group might say "Hmmmm, if we could do/build XYZ (where perhaps XYZ might be some particular insight about combining formulas), that would be a powerful tool in solving this, and I suspect we can do/build XYZ." And then your group might want to have half its members work on building XYZ, while the other half worked on showing in detail how, if you did have tool XYZ in hand, you can use it to show the theorem.

Challenge 4: Is #SAT as Hard to (Enumeratively) Approximate as It Is to Solve Exactly?

Challenge Problem (Cai & Hemachandra)

(Prove that) If #SAT *is has a polynomial-time* 2*-enumerator, then there is a polynomial-time algorithm for* #SAT*.*

Challenge 4: Is #SAT as Hard to (Enumeratively) Approximate as It Is to Solve Exactly?

Challenge Problem (Cai & Hemachandra)

(Prove that) If #SAT is has a polynomial-time 2-enumerator, then there is a polynomial-time algorithm for #SAT.

So that you have them easily at hand, here are some of the defs/tools from previous slides:

#SAT #SAT is the function that given as input a Boolean formula $F(x_1, x_2, \ldots, x_k)$—wlog assume each of the variables appears in F—outputs the number of satisfying assignments the formula has (i.e., of the 2^k possible assignments of the variables to True/False, the number of those under which F evaluates to True; so the output will be a natural number in the interval $[0, 2^k]$). For example, $\#\mathrm{SAT}(x_1 \vee x_2) = 3$ and $\#\mathrm{SAT}(x_1 \wedge \overline{x_1}) = 0$.

Enumerative approximation We say that #SAT has a polynomial-time 2-enumerator (aka, is p-time 2-enumerably approximable) if there is a polynomial-time function h such that on each input x, (a) $h(x)$ outputs a list of two (perhaps identical) natural numbers, and (b) $\#\mathrm{SAT}(x)$ appears in the list output by $h(x)$.

Challenge 4: Why One Natural Approach Is Hopeless

- One natural approach would be to run the hypothetical 2-enumerator h on the input formula F and both of its x_1-assigned subformulas, and to argue that *purely based on the 2 options that h gives for each of those three (i.e., viewing the formulas for a moment as black boxes)* (side comment: wlog, we may assume that each of the 3 has two distinct outputs; the other cases are even easier), we can either output $\|F\|$ or can identify at least one of the subformulas such that we can show a particular 1-to-1 linkage between which of the two predicted numbers of solutions it has and which of the two predicted numbers of solutions F has. And then we would iteratively walk down the tree, doing that.
- But the following example, based on one suggested by Gerhard Woeginger, shows that that is impossible. Suppose h predicts outputs $\{0, 1\}$ for F, $\{0, 1\}$ for the left subformula, and $\{0, 1\}$ for the right subformula. The values of the root can't be based purely on the numbers being linked 1-to-1 to those of the left subformula, since 0 for the left subformula can be linked to either root value, 0 ($0 + 0 = 0$) or 1 ($0 + 1 = 1$). The same holds for the right subformula. The three separate number-pairs just don't have enough information to make the desired link! But don't despair: we can make h help us far more!

Challenge 4: XYZ Idea/Statement

In particular, we can trick the enumerator into giving us *linked/coordinated* guesses! Let us explore that! What I was thinking of, when I mentioned XYZ in the food-for-thought hint, is the fact that we can efficiently combine two Boolean formulas into a new one such that from the number of satisfying assignments of the new formula we can easily "read off" the number of satisfying assignments of both of the original formulas. In particular, we can just do this in such a way that if we concatenate the (appropriately padded as needed) bitstrings capturing the numbers of solutions of the two formulas, we get the (appropriately padded as needed) bitstring capturing the number of solutions of the new "combined" formula. (Notation: $\|F\|$ is the number of satisfying assignments of F.) The following lemma of Cai and Hemachandra captures this.

Lemma

There are polynomial-time functions combiner and decoder such that for any Boolean formulas F and G, $combiner(F, G)$ is a Boolean formula and $decoder(F, G, \|combiner(F, G)\|)$ prints $\|F\|, \|G\|$.

Challenge 4: XYZ Proof

Lemma

There are polynomial-time functions combiner and decoder such that for any Boolean formulas F and G, $combiner(F, G)$ is a Boolean formula and $decoder(F, G, \|combiner(F, G)\|)$ prints $\|F\|, \|G\|$.

Proof Let $F = F(x_1, \ldots, x_n)$ and $G = G(y_1, \ldots, y_m)$, where $x_1, \ldots, x_n$, $y_1, \ldots, y_m$ are distinct. Let z and z' be two new Boolean variables. Then

$$H = (F \wedge z) \vee (\bar{z} \wedge x_1 \wedge \cdots \wedge x_n \wedge G \wedge z')$$

is the desired combination, since $\|h\| = \|f\|2^{m+1} + \|g\|$ and $\|g\| \leq 2^m$. □

We can easily extend this technique to combine three, four, or even polynomially many formulas.

Challenge 4: Proof Sketch of the Theorem

Challenge Problem (Cai & Hemachandra)

(Prove that) If #SAT *is has a polynomial-time* 2*-enumerator, then there is a polynomial-time algorithm for* #SAT*.*

And, in case time ran out, here is the quick proof sketch. Start with our input formula, F, whose number of solutions we wish to compute in polynomial time.
Self-reduce the formula on its first variable. Using the XYZ trick (twice), combine the original formula and the two subformulas into a single formula, H, whose number of solutions gives the number of solutions of all three. Run the 2-enumerator on H. If either of its output's two decoded guesses are inconsistent ($a \neq b + c$) ignore that line and the other one is the truth. If both are consistent and agree on $\|F\|$, then we're also done. Otherwise, the lines must differ in their claims about at least one of the two subformulas. Thus if we know the number of solutions of that one, shorter formula, we know the number of solutions of $\|F\|$.
Repeat the above on that formula, and so on, and then at the end ripple all the way back up. The entire process is a polynomial number of polynomial-cost steps. □

Challenge 4: Proof Sketch of the Theorem

Challenge Problem (Cai & Hemachandra)

(Prove that) If $\#\mathrm{SAT}$ *is has a polynomial-time* 2*-enumerator, then there is a polynomial-time algorithm for* $\#\mathrm{SAT}$*.*

Example of the key step from the proof sketch:

Which of the Guesses	$\|F(x_1, x_2, x_3, \dots)\|$	$\|F(\mathrm{True}, x_2, x_3, \dots)\|$	$\|F(\mathrm{False}, x_2, x_3, \dots)\|$
First	100	83	17
Second	101	85	16

In this example, note that we can conclude that $\|F(x_1, x_2, x_3, \dots)\| = 100$ if $\|F(\mathrm{False}, x_2, x_3, \dots)\| = 17$, and $\|F(x_1, x_2, x_3, \dots)\| = 101$ if $\|F(\mathrm{False}, x_2, x_3, \dots)\| = 16$; and we know that $\|F(\mathrm{False}, x_2, x_3, \dots)\| \in \{16, 17\}$.
So we have in polynomial time completely linked $\|F(x_1, x_2, x_3, \dots)\|$ to the issue of the number of satisfying assignments of the (after simplifying) shorter formula $F(\mathrm{False}, x_2, x_3, \dots)$. This completes our example of the key linking step.

Challenge 4: Extensions

Challenge Problem (Cai & Hemachandra)

(Prove that) If $\#\mathrm{SAT}$ *is has a polynomial-time* 2*-enumerator, then there is a polynomial-time algorithm for* $\#\mathrm{SAT}$*.*

Extensions (Cai & Hemachandra)

The above theorem holds, by the same flavor of approach (but doing more aggressively broad groupings), even if one looks at k-enumerators... or even $\sqrt{|F|}$-enumerators... or even, for any $\epsilon > 0$, to $\mathcal{O}(|F|^{1-\epsilon})$-enumerators. Indeed, one can even show that the result holds for polynomial-time enumerators that have no limit on their number of allowed elements in the list.

Conclusions

- Self-reducibility is a powerful tool across a broad range of settings.
- Myself, I have found it to be useful many times. (For example: Search versus Decision for Election Manipulation Problems; The Opacity of Backbones; Existence versus Exploitation: The Opacity of Backbones and Backdoors Under a Weak Assumption Easily Checked Generalized Self-Reducibility; Space-Efficient Recognition of Sparse Self-Reducible Languages; Strong Self-Reducibility Precludes Strong Immunity; P-Immune Sets with Holes Lack Self-Reducibility Properties.)
- My guess/hope is that perhaps you may too! That is, please, if it is not already there, consider adding this tool to *your* personal research toolkit: When you face a problem, think (if only for a moment) whether the problem happens to be one where the concept of self-reducibility will help you gain insight. Who knows?—One of these years, you might be happily surprised in finding that your answer to such a question is "Yes"!

Thank you for your time!