Superpowers and relativization

Noah Singer

Computer Science and Mathematics @ Harvard College

October 2020

Outline

Problem 1: Divisibility by 7

- Is 35 divisible by 7?
- Is 1991 divisible by 7?
- Is 16430672648476658943 divisible by 7?

Problem 2: Linear equations over $\{0, 1\}$

- Does the equation $x_0 + x_1 = 1$ have solutions for $x_0, x_1 \in \{0, 1\}$?
- Does the system of equations

$$\begin{aligned} x_9 + x_5 + x_7 + x_{10} + x_9 &= 3 \\ x_8 + x_0 + x_8 + x_9 + x_1 &= 2 \\ x_3 + x_9 + x_1 + x_6 + x_7 &= 4 \\ x_7 + x_2 + x_9 + x_7 + x_6 &= 0 \\ x_8 + x_0 + x_{10} + x_9 + x_6 &= 4 \\ x_9 + x_2 + x_9 + x_{10} + x_7 &= 3 \\ x_6 + x_4 + x_3 + x_5 + x_2 &= 1 \\ x_5 + x_8 + x_5 + x_4 + x_0 &= 1 \end{aligned}$$

have solutions for $x_0, \ldots, x_{10} \in \{0, 1\}$?

P vs. **NP**

Definition (**P**)

P is the class of *efficiently solvable* problems.

Definition (**NP**)

NP is the class of problems *whose solutions can be efficiently verified*.

- $\mathbf{P} \subseteq \mathbf{NP}$
- Think in terms of "powers"
  - **NP**: Power to "guess" potential solutions
- Strict subset? Believed yes! ($$$)
- Theory interest (*complexity*), and many important problems are in **NP**

Examples of problems

Examples of **P** problems:

- Is n divisible by 7?
- Is a graph G connected?
- Is there a path between vertices u and v of graph G of length at most k?
- Is n prime?
- Does a graph G have a cut cutting at most k edges?

Examples of **NP** problems not known to be in **P**:

- Does a system of linear equations have a solution with variables in $\{0, 1\}$?
- Does G contain a path of length at least k?
- Does n have a prime factor less than k?
- Does G have a cut cutting at least k edges?
- *And many more:* Formula/circuit satisfiability, knapsacks, graph coloring, traveling salesperson, etc.

Programs

Definition (Program)

A *program* is a binary string P which encodes a "computational process".

- Given *input* x, a binary string, we can *run* P on input x (write $P(x)$)

Problems

Notation: $\{0,1\}^*$ is the set of finite binary strings, e.g. 0010, 111. *Problems* are maps $\{0,1\}^* \to \{0,1\}$.

Every program* *computes* a corresponding program.

*: That always halts and outputs 0 or 1.

Complexity classes

Classes of problems corresponding to certain "powers":

- **TIME**$(T(n))$: Problems that can be solved in $T(n)$ steps (n is input size)
- **P**: Problems in **TIME**$(p(n))$ for any polynomial p
- **NP**: Can anyone write a definition?

Complexity classes

Classes of problems corresponding to certain "powers":

- **TIME**$(T(n))$: Problems that can be solved in $T(n)$ steps (n is input size)
- **P**: Problems in **TIME**$(p(n))$ for any polynomial p
- **NP**: Can anyone write a definition?

$\mathcal{A} \in$ **NP** means

$$\mathcal{A}(x) = 1 \iff \exists w \text{ s.t. } \mathcal{W}(x, w) = 1$$

for some $\mathcal{W} \in$ **P**.*

NP corresponds to *guessing*. Other "powers": flipping coins, interaction, nonuniformity, quantumness, approximation, promises, space

*: Also need $|w| = \text{poly}(n)$

Time hierarchy theorem

Theorem (Time hierarchy theorem [HS1965])

Suppose $T_1(n) \ll T_2(n)$, *then*

$$\mathbf{TIME}(T_1(n)) \subsetneq \mathbf{TIME}(T_2(n)).$$

E.g., $\mathbf{TIME}(n^2) \subsetneq \mathbf{TIME}(n^4)$.

Oracle programs

Definition

An *oracle* is a “magical box” that can solve a problem $\mathcal{A}$ instantly!

- An $\mathcal{A}$-oracle is another “power”
- What can we do with this power?

Superscript-$\mathcal{A}$ denotes “with an $\mathcal{A}$-oracle”, e.g. $\mathbf{P}^{\mathcal{A}}$, $\mathbf{TIME}^{\mathcal{A}}(n^3)$.

Oracles

Figure 1: *Lycurgus Consulting the Pythia* (Eugéne Delacroix) (public domain, from Google Art Project via Wikimedia Commons)

Oracle classes: warmup

- Let $\mathcal{A} \in \textbf{TIME}(n^4)$ but $\mathcal{A} \notin \textbf{TIME}(n^2)$. Compare $\textbf{TIME}(n^2)$ and $\textbf{TIME}^{\mathcal{A}}(n^2)$.
- Let $\mathcal{A} \in \textbf{P}$. Compare $\textbf{P}^{\mathcal{A}}$ and $\textbf{NP}^{\mathcal{A}}$. (What about $\textbf{EXP}^{\textbf{EXP}}$?)

Relativization

Theorem (Relative time hierarchy theorem)

If $T_1(n) \ll T_2(n)$, and $\mathcal{A}$ is any problem, then $\mathbf{TIME}^{\mathcal{A}}(T_1(n)) \subsetneq \mathbf{TIME}^{\mathcal{A}}(T_2(n))$.

Many other results "relativize" (find and replace); here are some compiled from Fortnow, Moshkovitz, Arora-Barak, Aaronson, ...:

- $\mathbf{EXPSPACE}, \mathbf{NEXP}^{\mathbf{NP}} \not\subset \mathbf{P}/\mathbf{poly}$
- $\overline{\textsc{st-Con}} \in \mathbf{NL}$
- *Savitch's theorem*: $\mathbf{NSPACE}(T(n)) \subseteq \mathbf{SPACE}(T(n)^2)$
- *Ladner's theorem*: There are **NP**-intermediate problems
- *Toda's theorem*: $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$

Victory!

Figure 2: Washington Capitals Stanley Cup Victory Parade, 2018 (Michael Saffle) (public domain via Wikimedia Commons)

BUT...

Baker-Gill-Solovay result

Theorem ([BGS1975])

There exists problems $\mathcal{A}, \mathcal{B}$ such that:

- $\mathbf{P}^{\mathcal{A}} = \mathbf{NP}^{\mathcal{A}}$
- $\mathbf{P}^{\mathcal{B}} \subsetneq \mathbf{NP}^{\mathcal{B}}$

Relativization barrier

- "All the theorems we know how to prove relativize!" $\implies$ "We don't know how to prove anything interesting"
- Some more questions that don't relativize (from [AIV1992] and [Aar2017]):
  - Does $\mathbf{P} = \mathbf{PSPACE}$?
  - Does $\mathbf{NP} = \mathbf{EXP}$?
  - Does $\mathbf{BPP} = \mathbf{NEXP}$?
  - Does $\mathbf{IP} = \mathbf{PSPACE}$? *
  - Is $\mathbf{NEXP} \subseteq \mathbf{P}/\mathbf{poly}$?

*: Resolved using nonrelativizing techniques.

Constructing $\mathcal{A}$

- Pick "really hard" problem
- Claim 1: If $\mathcal{A} \in \mathbf{EXP}$, then $\mathbf{NP}^{\mathcal{A}} \subseteq \mathbf{EXP}$
  - Not too tricky given $\mathbf{NP} \subseteq \mathbf{EXP}$
- Claim 2: Can pick $\mathcal{A} \in \mathbf{EXP}$ such that $\mathbf{P}^{\mathcal{A}} = \mathbf{EXP}$ (**EXP**-*hard problem*)
- Together, have

$$\mathbf{EXP} = \mathbf{P}^{\mathcal{A}} \subseteq \mathbf{NP}^{\mathcal{A}} \subseteq \mathbf{EXP} \implies \mathbf{P}^{\mathcal{A}} = \mathbf{NP}^{\mathcal{A}}$$

Following description of Arora-Barak [AB2009].

Constructing $\mathcal{B}$

- Pick "really bizarre" problem
- E.g. pick uniformly random* $\mathcal{B}$ and consider problem $\mathcal{C}$ related to $\mathcal{B}$
- Show
$$\Pr_{\mathcal{B}}[\mathcal{C} \in \mathbf{NP}^{\mathcal{B}} \text{ but } \mathcal{C} \notin \mathbf{P}^{\mathcal{B}}] = 1$$
- "**NP** gives you extra power to learn something about $\mathcal{B}$"

Following description of Aaronson [Aar2017].

Other barriers

- *Natural proofs* — combinatorial approaches seeking to bound special classes of programs
- *Algebrization* [AW08] — algebraic extension of relativization handling e.g. **IP** = **PSPACE**

Time hierarchy theorem — very rough sketch I

Theorem

$\mathbf{TIME}(n^2) \subsetneq \mathbf{TIME}(n^4)$.

Define the problem*

$$\textsc{Halt}_{n^3}(P, x) := \begin{cases} 1 & P \text{ runs for fewer than } n^3 \text{ steps on input } x \\ 0 & \text{otherwise.} \end{cases}$$

Firstly, $\textsc{Halt}_{n^3} \in \mathbf{TIME}(n^4)$.

*: **Not accounting for input lengths at all** — this is just to give an idea of the contradiction.

Time hierarchy theorem — very rough sketch II

Suppose $\textsc{Halt}_{n^3}$ is decided by program $P_{FASTHALT}$ running in time n^2.

Define the program

```
def P_EVIL(x):
    h = P_FASTHALT(x,x)
    if h = 0:
        loop_forever()
    else:
        output(0)
```

On input x, P_{EVIL} either runs in n^2 steps or never ends. What does P_{EVIL} do on input P_{EVIL}?

Time hierarchy theorem — very rough sketch III


```
def P_EVIL(x):
    h = P_FASTHALT(x,x)
    if h == 1:
        loop_forever()
    else:
        output(0)
```

If ...

1. ... $P_{EVIL}(P_{EVIL})$ runs in n^2 steps, then

Time hierarchy theorem — very rough sketch III

```
def P_EVIL(x):
    h = P_FASTHALT(x,x)
    if h == 1:
        loop_forever()
    else:
        output(0)
```

If …

1. … $P_{EVIL}(P_{EVIL})$ runs in n^2 steps, then $P_{FASTHALT}(P_{EVIL}, P_{EVIL}) = 1$, so

Time hierarchy theorem — very rough sketch III

```
def P_EVIL(x):
    h = P_FASTHALT(x,x)
    if h == 1:
        loop_forever()
    else:
        output(0)
```

If …

1. … $P_{EVIL}(P_{EVIL})$ runs in n^2 steps, then $P_{FASTHALT}(P_{EVIL}, P_{EVIL}) = 1$, so $P_{EVIL}(P_{EVIL})$ never ends
2. … $P_{EVIL}(P_{EVIL})$ never ends, then

Time hierarchy theorem — very rough sketch III


```
def P_EVIL(x):
    h = P_FASTHALT(x,x)
    if h == 1:
        loop_forever()
    else:
        output(0)
```

If ...

1. ... $P_{EVIL}(P_{EVIL})$ runs in n^2 steps, then $P_{FASTHALT}(P_{EVIL}, P_{EVIL}) = 1$, so $P_{EVIL}(P_{EVIL})$ never ends
2. ... $P_{EVIL}(P_{EVIL})$ never ends, then $P_{FASTHALT}(P_{EVIL}, P_{EVIL}) = 0$, so

Time hierarchy theorem — very rough sketch III

```
def P_EVIL(x):
    h = P_FASTHALT(x,x)
    if h == 1:
        loop_forever()
    else:
        output(0)
```

If ...

1. ... $P_{EVIL}(P_{EVIL})$ runs in n^2 steps, then $P_{FASTHALT}(P_{EVIL}, P_{EVIL}) = 1$, so $P_{EVIL}(P_{EVIL})$ never ends
2. ... $P_{EVIL}(P_{EVIL})$ never ends, then $P_{FASTHALT}(P_{EVIL}, P_{EVIL}) = 0$, so $P_{EVIL}(P_{EVIL})$ runs in n^2 steps

Contradiction! $\square$

Further reading

- [AB2009, §3.5] for more technical details on time hierarchy and relativization
- [Aar2017, §6] for discussion of other barriers, including *algebrization*, and other relativization barriers, such as **NEXP** vs. **P**/**poly**
- [AIV1992] for interpretations of relativization in terms of axiomatic systems about complexity classes
- Take CS121! Thanks to Boaz Barak for teaching this and for the notation

Conclusion

"The magic of diagonalization, self-reference, and counting arguments is how abstract and general they are: they never require us to 'get our hands dirty' by understanding the inner workings of algorithms or circuits. But as was recognized early in the history of complexity theory, the price of generality is that the logical techniques are extremely limited in scope." — Scott Aaronson [Aar2017]

Thanks

Thanks for listening!

Thanks to Voula Collins, Iian Smythe, Elliot Kienzle, and Jyotsna Rao for their feedback while I prepared this presentation.

References I

Scott Aaronson. $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$. 2017. URL: https://www.scottaaronson.com/papers/pnp.pdf.

Sanjeev Arora and Boaz Barak. *Computational Complexity*. Cambridge, UK: Cambridge University Press, Apr. 2009, p. 594.

Sanjeev Arora, Russell Impagliazzo, and Umesh Vazirani. "Relativizing versus Nonrelativizing Techniques: The Role of Local Checkability". 1992.

Theodore Baker, John Gill, and Robert Solovay. "SIAM Journal on Computing". In: 4.4 (1975), pp. 431–442.

References II

Juris Hartmanis and Richard E. Stearns. "On the computational complexity of algorithms". In: *Transactions of the American Mathematical Society* 117 (1965), pp. 285–306.