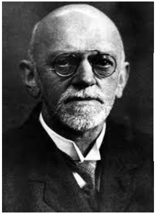
Logic and Complexity

Avijeet Ghosh[1]

[1]Indian Statistical Institute, Kolkata

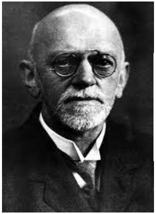
A System of Proofs

- A Formal System of Mathematical Proofs



?

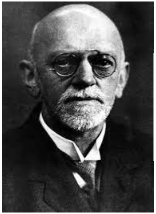
A System of Proofs



?

- A Formal System of Mathematical Proofs
  - Axioms $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$

A System of Proofs

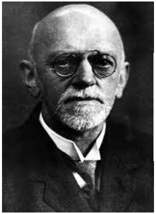


?

- A Formal System of Mathematical Proofs
  - Axioms $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$
  - Truth-preserving rules $\mathcal{I} = \{I_1, \ldots, I_m\}$

$$I_i = \frac{S_1, S_2, \ldots, S_n}{C}$$

A System of Proofs



?

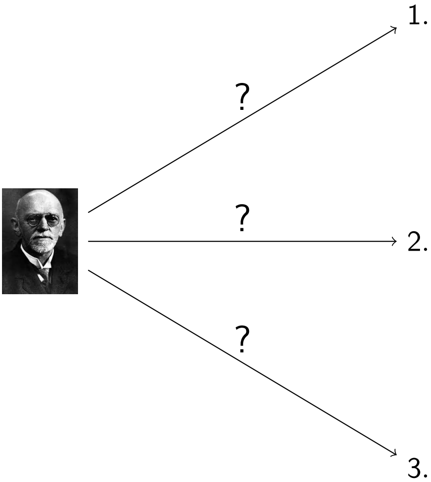
- A Formal System of Mathematical Proofs
  - Axioms $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$
  - Truth-preserving rules $\mathcal{I} = \{I_1, \ldots, I_m\}$

$$I_i = \frac{S_1, S_2, \ldots, S_n}{C}$$

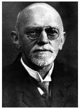
- Objective: Anything true has a proof

$$\begin{tikzcd}
 & & & T & & & \\
 & T_1 \arrow[urr, "I_{k'}"] & & & & T_2 \arrow[ull] & \\
A_i \arrow[ur, "I_{i'}"] & & A_j \arrow[ul] & & A_k \arrow[ur] & & A_\ell \arrow[ul, "I_{j'}"']
\end{tikzcd}$$

Hilbert's Questions



Hilbert's Questions



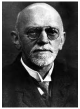
?

?

?

1. Complete
2. Consistent
3. Decidable

Hilbert's Questions



1. Complete X 

   Some Truths cannot be proven

2. Consistent

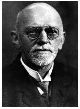
3. Decidable

?

?

?

Hilbert's Questions



?

?

?

1. Complete X

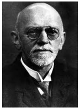
Some Truths cannot be proven

2. Consistent ?

Proving its own consistency beyond system

3. Decidable

Hilbert's Questions



?

?

?

1. Complete X

Some Truths cannot be proven

2. Consistent ?

Proving its own consistency beyond system

3. Decidable 🤔 This talk

Given a statement and axioms, is there a proof?

Computability

- **NEED:** Given S and $\mathcal{A}$, "procedure" to decide whether there is a proof

- **NEED:** Given S and $\mathcal{A}$, "procedure" to decide whether there is a proof
    - For any axiom system $(\mathcal{A}, \mathcal{I})$ and any statement S

$$f(S, \mathcal{A}, \mathcal{I}) = \begin{cases} 1 & \mathcal{A} \vdash_{\mathcal{I}} S \\ 0 & \mathcal{A} \nvdash_{\mathcal{I}} S \end{cases}$$

- **NEED:** Given S and $\mathcal{A}$, "procedure" to decide whether there is a proof
  - For any axiom system $(\mathcal{A}, \mathcal{I})$ and any statement S
    $$f(S, \mathcal{A}, \mathcal{I}) = \begin{cases} 1 & \mathcal{A} \vdash_{\mathcal{I}} S \\ 0 & \mathcal{A} \nvdash_{\mathcal{I}} S \end{cases}$$
  - Is f *computable*?

Computability

- **NEED:** Given S and $\mathcal{A}$, "procedure" to decide whether there is a proof
  - For any axiom system $(\mathcal{A}, \mathcal{I})$ and any statement S
    $$f(S, \mathcal{A}, \mathcal{I}) = \begin{cases} 1 & \mathcal{A} \vdash_{\mathcal{I}} S \\ 0 & \mathcal{A} \nvdash_{\mathcal{I}} S \end{cases}$$
  - Is f *computable*?
- What is *computability*?

- **NEED:** Given S and $\mathcal{A}$, "procedure" to decide whether there is a proof
  - For any axiom system $(\mathcal{A}, \mathcal{I})$ and any statement S
    $$f(S, \mathcal{A}, \mathcal{I}) = \begin{cases} 1 & \mathcal{A} \vdash_{\mathcal{I}} S \\ 0 & \mathcal{A} \nvdash_{\mathcal{I}} S \end{cases}$$
  - Is f *computable*?
- What is *computability*?
- **Example 1**: $\forall x \in \mathbb{N}, f(x) = x^2 + 2x + 1$

Computability

- **NEED:** Given S and $\mathcal{A}$, "procedure" to decide whether there is a proof
  - For any axiom system $(\mathcal{A}, \mathcal{I})$ and any statement S
    $$f(S, \mathcal{A}, \mathcal{I}) = \begin{cases} 1 & \mathcal{A} \vdash_{\mathcal{I}} S \\ 0 & \mathcal{A} \nvdash_{\mathcal{I}} S \end{cases}$$
  - Is f *computable*?
- What is *computability*?
- **Example 1**: $\forall x \in \mathbb{N}, f(x) = x^2 + 2x + 1$
  - $f(13) = ?$

Computability

- **NEED:** Given S and $\mathcal{A}$, "procedure" to decide whether there is a proof
  - For any axiom system $(\mathcal{A}, \mathcal{I})$ and any statement S
    $$f(S, \mathcal{A}, \mathcal{I}) = \begin{cases} 1 & \mathcal{A} \vdash_{\mathcal{I}} S \\ 0 & \mathcal{A} \nvdash_{\mathcal{I}} S \end{cases}$$
  - Is f *computable*?
- What is *computability*?
- **Example 1**: $\forall x \in \mathbb{N}, f(x) = x^2 + 2x + 1$
  - $f(13) = ?$
- **Example 2**: $\forall x \in \mathbb{N}, \forall y \in \mathbb{N} f(x, y) = x^y + xy + 1$

Computability

- **NEED:** Given S and $\mathcal{A}$, "procedure" to decide whether there is a proof
  - For any axiom system $(\mathcal{A}, \mathcal{I})$ and any statement S
    $$f(S, \mathcal{A}, \mathcal{I}) = \begin{cases} 1 & \mathcal{A} \vdash_{\mathcal{I}} S \\ 0 & \mathcal{A} \nvdash_{\mathcal{I}} S \end{cases}$$
  - Is f *computable*?
- What is *computability*?
- **Example 1**: $\forall x \in \mathbb{N}, f(x) = x^2 + 2x + 1$
  - $f(13) = ?$
- **Example 2**: $\forall x \in \mathbb{N}, \forall y \in \mathbb{N} f(x, y) = x^y + xy + 1$
  - $f(13, 12) = ?$

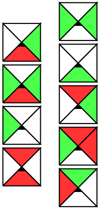
- **NEED:** Given S and $\mathcal{A}$, "procedure" to decide whether there is a proof
  - For any axiom system $(\mathcal{A}, \mathcal{I})$ and any statement S
    $$f(S, \mathcal{A}, \mathcal{I}) = \begin{cases} 1 & \mathcal{A} \vdash_{\mathcal{I}} S \\ 0 & \mathcal{A} \nvdash_{\mathcal{I}} S \end{cases}$$
  - Is f *computable*?
- What is *computability*?
- **Example 1**: $\forall x \in \mathbb{N}, f(x) = x^2 + 2x + 1$
  - $f(13) = ?$
- **Example 2**: $\forall x \in \mathbb{N}, \forall y \in \mathbb{N} f(x, y) = x^y + xy + 1$
  - $f(13, 12) = ?$
  - **HARDER**

Computability

- Proper Tiling:

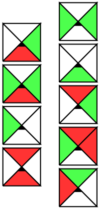
Computability

- Proper Tiling:
  - Given a set of tiles $\mathcal{T}$

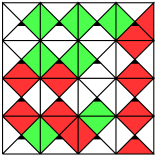


Computability

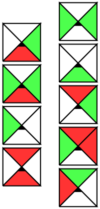
- Proper Tiling:
  - Given a set of tiles $\mathcal{T}$



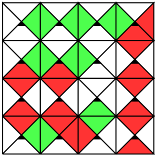
  - A Proper tiling on a finite grid:



- Proper Tiling:
  - Given a set of tiles $\mathcal{T}$



  - A Proper tiling on a finite grid:



- $f(\mathcal{T}) = \begin{cases} 1 & \text{Proper tiling covering infinite grid} \\ 0 & \text{No such proper tiling} \end{cases}$

Model of Computability

- $f(x) = x^2 + 2x + 1.$

Model of Computability

- $f(x) = x^2 + 2x + 1.$
- $f(13) = 13^2 + (2x13) + 1$

Model of Computability

- $f(x) = x^2 + 2x + 1.$
- $f(13) = 13^2 + (2x13) + 1$
- Local rules:

Model of Computability

- $f(x) = x^2 + 2x + 1$.
- $f(13) = 13^2 + (2x13) + 1$
- Local rules:
  - Rule of multiplication

Model of Computability

- $f(x) = x^2 + 2x + 1$.
- $f(13) = 13^2 + (2x13) + 1$
- Local rules:
  - Rule of multiplication
  - Rule of addition

Model of Computability

- $f(x) = x^2 + 2x + 1$.
- $f(13) = 13^2 + (2x13) + 1$
- Local rules:
  - Rule of multiplication
  - Rule of addition
- Local states:

Model of Computability

- $f(x) = x^2 + 2x + 1$.
- $f(13) = 13^2 + (2x13) + 1$
- Local rules:
  - Rule of multiplication
  - Rule of addition
- Local states:
  1. Initial $13^2 + (2x13) + 1$

Model of Computability

- $f(x) = x^2 + 2x + 1$.
- $f(13) = 13^2 + (2x13) + 1$
- Local rules:
  - Rule of multiplication
  - Rule of addition
- Local states:
  1. Initial $13^2 + (2x13) + 1$
  2. All multiplication done $169 + 26 + 1$

Model of Computability

- $f(x) = x^2 + 2x + 1.$
- $f(13) = 13^2 + (2x13) + 1$
- Local rules:
  - Rule of multiplication
  - Rule of addition
- Local states:
  1. Initial $13^2 + (2x13) + 1$
  2. All multiplication done $169 + 26 + 1$
  3. All addition done 196

Model of Computability

- $f(x) = x^2 + 2x + 1.$
- $f(13) = 13^2 + (2x13) + 1$
- Local rules:
  - Rule of multiplication
  - Rule of addition
- Local states:
  1. Initial $13^2 + (2x13) + 1$
  2. All multiplication done $169 + 26 + 1$
  3. All addition done 196
  4. Final value 196

Model of Computability

- $f(x) = x^2 + 2x + 1$.
- $f(13) = 13^2 + (2x13) + 1$
- Local rules:
  - Rule of multiplication
  - Rule of addition
- Local states:
  1. Initial $13^2 + (2x13) + 1$
  2. All multiplication done $169 + 26 + 1$
  3. All addition done 196
  4. Final value 196
- **NEED: MODEL** Anything that model can compute is computable

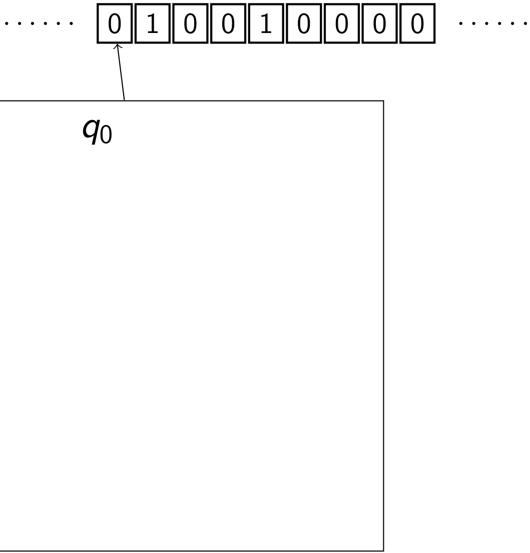
Model of Computability

- $f(x) = x^2 + 2x + 1$.
- $f(13) = 13^2 + (2x13) + 1$
- Local rules:
  - Rule of multiplication
  - Rule of addition
- Local states:
  1. Initial $13^2 + (2x13) + 1$
  2. All multiplication done $169 + 26 + 1$
  3. All addition done 196
  4. Final value 196
- **NEED: MODEL** Anything that model can compute is computable
- Turing Machine, Lambda Calculus

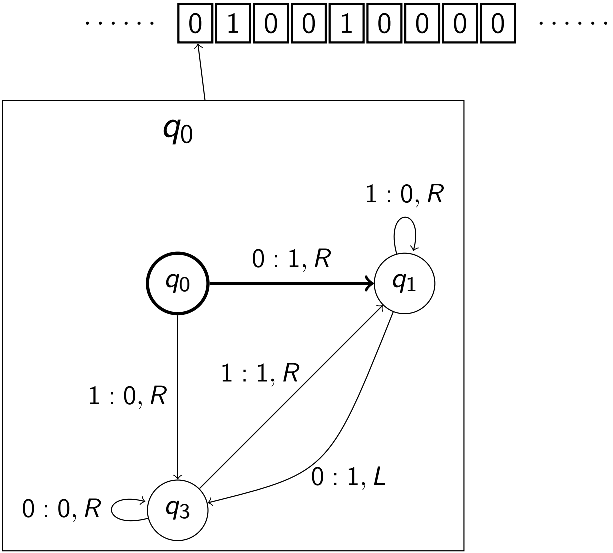
The Turing Machine

$$\cdots\cdots \quad \boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{1}\boxed{0}\boxed{0}\boxed{0}\boxed{0} \quad \cdots\cdots$$

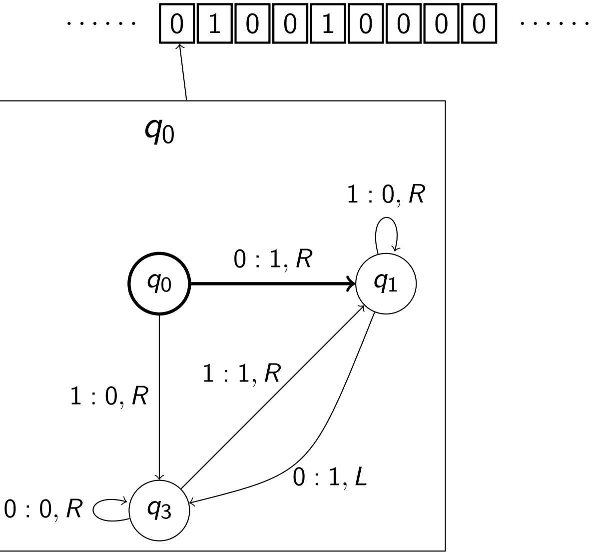
The Turing Machine



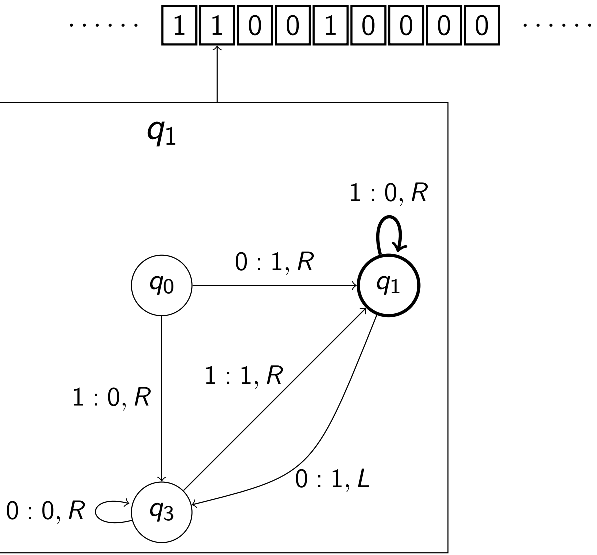
The Turing Machine



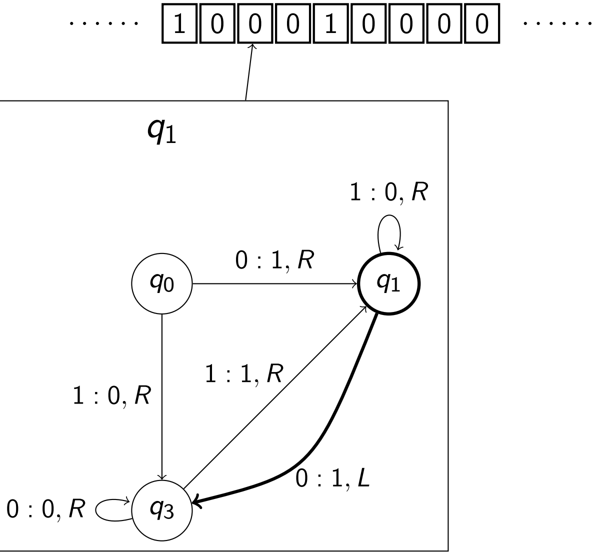
The Turing Machine



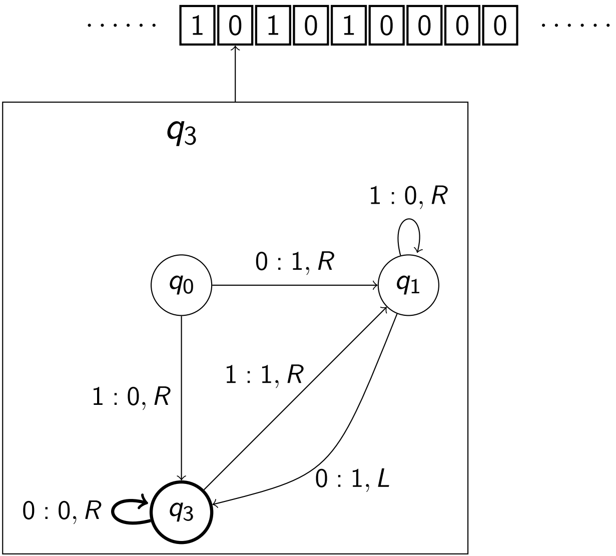
The Turing Machine



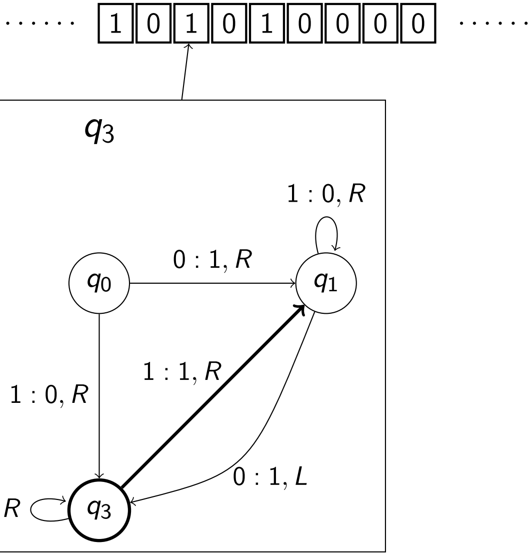
The Turing Machine



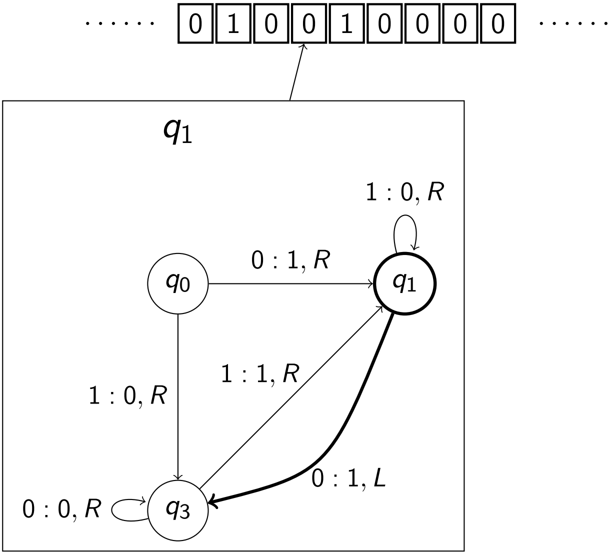
The Turing Machine



The Turing Machine



The Turing Machine



The Turing Machine

- $\mathcal{M} = \langle Q, q_0 \in Q, \Sigma, \delta, F \subseteq Q \rangle$
  - Q: finite set of states
  - Σ: alphabet
  - $\delta : Q \times \Sigma \to Q \times \Sigma \cup \{R, L\}$ transition rule
- Decision functions: $f(x \in \{0,1\}^\star) \in \{0,1\}$
- **Other MODELS**: Church's Lambda Calculus

Church-Turing Thesis

Any computable function can be computed by a Turing Machine/λ-Calculus.

- **Model:** Turing Machine (TM)

Complexity Theory

- **Model:** Turing Machine (TM)
- Bounding resource usage of computation wrt input size n

Complexity Theory

- **Model:** Turing Machine (TM)
- Bounding resource usage of computation wrt input size n
  - **Time**: Steps a TM takes

Complexity Theory

- **Model:** Turing Machine (TM)
- Bounding resource usage of computation wrt input size n
  - **Time**: Steps a TM takes
  - **Space**: Tape space used

Complexity Theory

- **Model:** Turing Machine (TM)
- Bounding resource usage of computation wrt input size n
  - **Time**: Steps a TM takes
  - **Space**: Tape space used
  - **Asymptotical**: $2\mathcal{C}(n) + 3 \equiv 55\mathcal{C}(n) + 7$
    $2^{\mathcal{C}(n)} \not\equiv 3^{\mathcal{C}(n)}$

Complexity Theory

- **Model:** Turing Machine (TM)
- Bounding resource usage of computation wrt input size n
  - **Time**: Steps a TM takes
  - **Space**: Tape space used
  - **Asymptotical**: $2\mathcal{C}(n) + 3 \equiv 55\mathcal{C}(n) + 7$
    $2^{\mathcal{C}(n)} \not\equiv 3^{\mathcal{C}(n)}$
- Difficulty between classes of problems (Decision functions)

Complexity Theory

- **Model:** Turing Machine (TM)
- Bounding resource usage of computation wrt input size n
  - **Time**: Steps a TM takes
  - **Space**: Tape space used
  - **Asymptotical**: $2\mathcal{C}(n) + 3 \equiv 55\mathcal{C}(n) + 7$
    $2^{\mathcal{C}(n)} \not\equiv 3^{\mathcal{C}(n)}$
- Difficulty between classes of problems (Decision functions)
- Decision functions gives Language

Complexity Theory

- **Model:** Turing Machine (TM)
- Bounding resource usage of computation wrt input size n
  - **Time**: Steps a TM takes
  - **Space**: Tape space used
  - **Asymptotical**: $2\mathcal{C}(n) + 3 \equiv 55\mathcal{C}(n) + 7$
    $2^{\mathcal{C}(n)} \not\equiv 3^{\mathcal{C}(n)}$
- Difficulty between classes of problems (Decision functions)
- Decision functions gives Language
  - Language of f $\mathcal{L}_f \subseteq \Sigma^\star$

Complexity Theory

- **Model:** Turing Machine (TM)
- Bounding resource usage of computation wrt input size n
  - **Time**: Steps a TM takes
  - **Space**: Tape space used
  - **Asymptotical**: $2\mathcal{C}(n) + 3 \equiv 55\mathcal{C}(n) + 7$
    $2^{\mathcal{C}(n)} \not\equiv 3^{\mathcal{C}(n)}$
- Difficulty between classes of problems (Decision functions)
- Decision functions gives Language
  - Language of f $\mathcal{L}_f \subseteq \Sigma^\star$
  - $f(x) = 1 \equiv \mathcal{M}_f(x)$ accepts $\equiv x \in \mathcal{L}_f$

Polynomial Complexity

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?

Polynomial Complexity

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?
  - $\langle 23, 45, 79, 127 \rangle$

Polynomial Complexity

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?
  - $\langle 23, 45, 79, 127 \rangle$
  - Easily solvable

Polynomial Complexity

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?
  - $\langle 23, 45, 79, 127 \rangle$
  - Easily solvable
  - Compare adjascent pair (constant steps c)

Polynomial Complexity

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?
  - $\langle 23, 45, 79, 127 \rangle$
  - Easily solvable
  - Compare adjacent pair (constant steps c)
  - Check for all adjacent pairs ($\leq n$)

Polynomial Complexity

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?
  - $\langle 23, 45, 79, 127 \rangle$
  - Easily solvable
  - Compare adjacent pair (constant steps c)
  - Check for all adjacent pairs ($\leq n$)
  - Total steps $\sim cn$

Polynomial Complexity

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?
  - $\langle 23, 45, 79, 127 \rangle$
  - Easily solvable
  - Compare adjacent pair (constant steps c)
  - Check for all adjacent pairs ($\leq n$)
  - Total steps $\sim cn$
- Given a propositional formula φ, **is it SATISFIABLE**?

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?
  - $\langle 23, 45, 79, 127 \rangle$
  - Easily solvable
  - Compare adjacent pair (constant steps c)
  - Check for all adjacent pairs ($\leq n$)
  - Total steps $\sim cn$
- Given a propositional formula φ, **is it SATISFIABLE**?
  - $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_1) \wedge (x_3 \vee x_1 \vee x_2)$

Polynomial Complexity

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?
  - $\langle 23, 45, 79, 127 \rangle$
  - Easily solvable
  - Compare adjacent pair (constant steps c)
  - Check for all adjacent pairs ($\leq n$)
  - Total steps $\sim cn$
- Given a propositional formula φ, **is it SATISFIABLE**?
  - $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_1) \wedge (x_3 \vee x_1 \vee x_2)$
  - Harder to solve

Polynomial Complexity

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?
  - $\langle 23, 45, 79, 127 \rangle$
  - Easily solvable
  - Compare adjacent pair (constant steps c)
  - Check for all adjacent pairs ($\leq n$)
  - Total steps $\sim cn$
- Given a propositional formula φ, **is it SATISFIABLE**?
  - $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_1) \wedge (x_3 \vee x_1 \vee x_2)$
  - Harder to solve
  - If **SATISFIABLE**, what is the certificate?

Polynomial Complexity

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?
  - $\langle 23, 45, 79, 127 \rangle$
  - Easily solvable
  - Compare adjacent pair (constant steps c)
  - Check for all adjacent pairs ($\leq n$)
  - Total steps $\sim cn$
- Given a propositional formula φ, **is it SATISFIABLE**?
  - $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_1) \wedge (x_3 \vee x_1 \vee x_2)$
  - Harder to solve
  - If **SATISFIABLE**, what is the certificate?
  - How big?

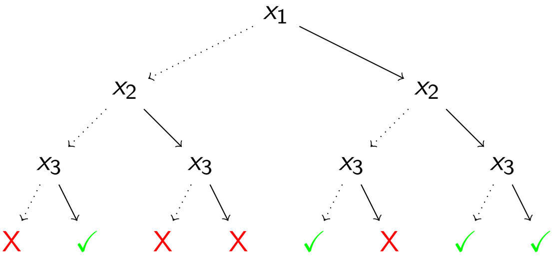
Polynomial Complexity

Consider the following problems:

- Given a tuple of n integers, **are they SORTED**?
  - $\langle 23, 45, 79, 127 \rangle$
  - Easily solvable
  - Compare adjacent pair (constant steps c)
  - Check for all adjacent pairs ($\leq n$)
  - Total steps $\sim cn$
- Given a propositional formula φ, **is it SATISFIABLE**?
  - $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_1) \wedge (x_3 \vee x_1 \vee x_2)$
  - Harder to solve
  - If **SATISFIABLE**, what is the certificate?
  - How big?
  - If certificate given, how much time to verify?

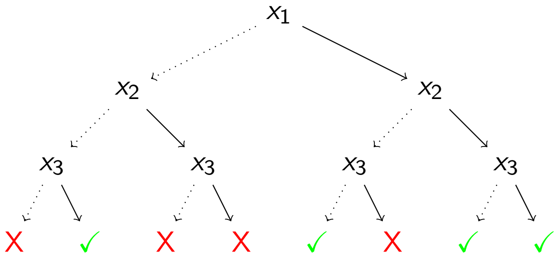
Non-determinism

- $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_1) \wedge (x_3 \vee x_1 \vee x_2)$
- TM steps:



Non-determinism

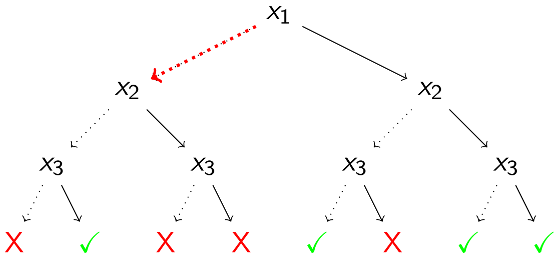
- $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_1) \wedge (x_3 \vee x_1 \vee x_2)$
- Lucky TM steps:



- This Lucky TM is *quick* iff small certificate, quick verify
- This Lucky TM is called **Non-deterministic**

Non-determinism

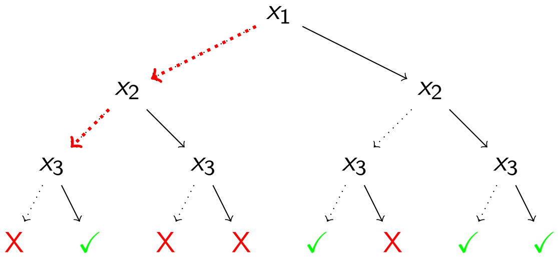
- $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_1) \wedge (x_3 \vee x_1 \vee x_2)$
- Lucky TM steps:



- This Lucky TM is *quick* iff small certificate, quick verify
- This Lucky TM is called **Non-deterministic**

Non-determinism

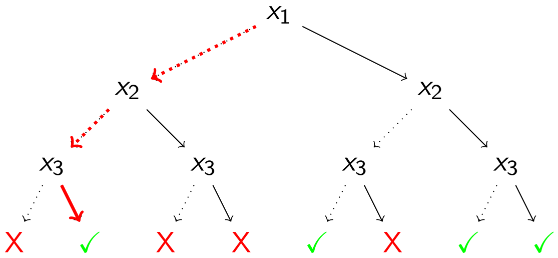
- $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_1) \wedge (x_3 \vee x_1 \vee x_2)$
- Lucky TM steps:



- This Lucky TM is *quick* iff small certificate, quick verify
- This Lucky TM is called **Non-deterministic**

Non-determinism

- $\varphi = (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_1) \wedge (x_3 \vee x_1 \vee x_2)$
- Lucky TM steps:



- This Lucky TM is *quick* iff small certificate, quick verify
- This Lucky TM is called **Non-deterministic**

Complexity Classes

- $\mathcal{L} \in$ P:= Steps$(\mathcal{M}_{\mathcal{L}}(x)) \leq O(|x|^c)$
- $\mathcal{L} \in$ NP:= Steps$(\mathcal{M}_{\mathcal{L}}(x)) \leq O(|x|^c)$, $\mathcal{M}_{\mathcal{L}}$ is Non-deterministic
- P $\subset_?$ NP
- $\mathcal{L} \in$ EXPTIME:= Steps$(\mathcal{M}_{\mathcal{L}}(x)) \leq O(2^{|x|^c})$
- $\mathcal{L} \in$ NEXPTIME:= Steps$(\mathcal{M}_{\mathcal{L}}(x)) \leq O(2^{|x|^c})$, $\mathcal{M}_{\mathcal{L}}$ is ND
- P $\subseteq_?$ NP $\subseteq_?$ EXPTIME $\subseteq_?$ NEXPTIME
- P $\subset$ EXPTIME and NP $\subset$ NEXPTIME

Hardness and Completeness

- $\mathcal{L}_1 \geq_{hard} \mathcal{L}_2$

Hardness and Completeness

- $\mathcal{L}_1 \geq_{hard} \mathcal{L}_2$

  $\mathcal{L}_2$ can be computed using $\mathcal{M}_{\mathcal{L}_1}$

Hardness and Completeness

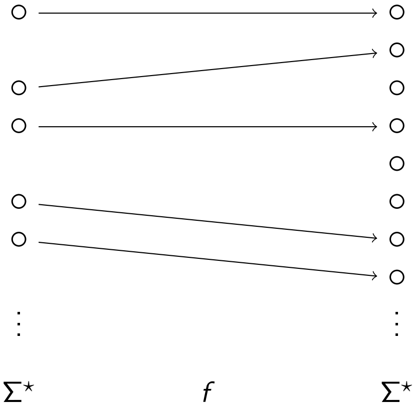
- $\mathcal{L}_1 \geq_{hard} \mathcal{L}_2$

  $\mathcal{L}_2$ can be computed using $\mathcal{M}_{\mathcal{L}_1}$
- *Reducing* $\mathcal{L}_2$ to $\mathcal{L}_1$.

Hardness and Completeness

- $\mathcal{L}_1 \geq_{hard} \mathcal{L}_2$

  $\mathcal{L}_2$ can be computed using $\mathcal{M}_{\mathcal{L}_1}$
- *Reducing* $\mathcal{L}_2$ to $\mathcal{L}_1$.



$x \in \mathcal{L}_2$ iff $f(x) \in \mathcal{L}_1$

$\mathcal{L}_2$ is reduced to $\mathcal{L}_1$

Completeness

- $\mathcal{L}_1 \geq_{\mathrm{P}} \mathcal{L}_2$: Reduction f is poly computable
- $\mathcal{L} \in \mathcal{C} - \textit{complete}$
  - $\mathcal{L} \in \mathcal{C}$
  - $\mathcal{L} \geq_{\mathrm{P}} \mathcal{L}'$ for any $\mathcal{L}' \in \mathcal{C}$

Cook-Levin

Propositional SAT is $\mathrm{NP} - \textit{complete}$

- A $\mathcal{C} - \textit{complete}$ problem is one of the hardest in $\mathcal{C}$

Definability of Classes

- $\mathcal{L} = \{x \in \Sigma^\star \mid x \vDash \varphi\}$
  - φ: Property or **Query**
  - x can represent any finite structure
  - **Example**: x: Graph
    φ: "Is there a triangle?"
  - **Example**: x: Graph
    φ: "Is it 3-colorable?"
- A $\mathcal{L}$ is *definable* by a query φ

Definability of Classes

- Triangle in a graph
  - **INPUT:** $G = \langle V, E \subseteq V \times V \rangle$
  - The property/query

$$\varphi = \exists x \exists y \exists z (\neg(x = y) \land \neg(y = z) \land \neg(x = z) \\ \land E(x, y) \land E(y, z) \land E(x, z))$$

Definability of Classes

- Triangle in a graph
  - **INPUT:** $G = \langle V, E \subseteq V \times V \rangle$
  - The property/query

$$\varphi = \exists x \exists y \exists z (\neg(x = y) \wedge \neg(y = z) \wedge \neg(x = z) \\ \wedge E(x, y) \wedge E(y, z) \wedge E(x, z))$$

- 3-Colorable graph
  - **INPUT:** $G = \langle V, E \subseteq V \times V \rangle$
  - The property/query

$$\varphi = \exists R \subseteq V \exists G \subseteq V \exists B \subseteq V (\forall x (R(x) \vee B(x) \vee G(s)) \wedge \\ \forall x (\neg(R(x) \wedge B(x)) \wedge \neg(B(x) \wedge G(x)) \wedge \neg(R(x) \wedge G(x))) \wedge \\ \forall x \forall y (E(x, y) \rightarrow (\neg(R(x) \wedge R(y) \wedge \neg(G(x) \wedge G(y)) \wedge \\ \neg(B(x) \wedge B(y)))))$$

Definability of Classes

- Triangle in a graph P
  - **INPUT:** $G = \langle V, E \subseteq V \times V \rangle$
  - The property/query

$$\varphi = \exists x \exists y \exists z (\neg(x = y) \land \neg(y = z) \land \neg(x = z) \\ \land E(x, y) \land E(y, z) \land E(x, z))$$

- 3-Colorable graph NP-complete
  - **INPUT:** $G = \langle V, E \subseteq V \times V \rangle$
  - The property/query

$$\varphi = \exists R \subseteq V \exists G \subseteq V \exists B \subseteq V (\forall x (R(x) \lor B(x) \lor G(s)) \land \\ \forall x (\neg(R(x) \land B(x)) \land \neg(B(x) \land G(x)) \land \neg(R(x) \land G(x))) \land \\ \forall x \forall y (E(x, y) \to (\neg(R(x) \land R(y) \land \neg(G(x) \land G(y)) \land \\ \neg(B(x) \land B(y))))))$$

Second Order Logic

- Extends FOL formulas with quantification over relations

Second Order Logic

- Extends FOL formulas with quantification over relations

$$\exists X \varphi$$

Second Order Logic

- Extends FOL formulas with quantification over relations

$$\exists X \varphi$$

- X can be any relation of any arity over the domain

Second Order Logic

- Extends FOL formulas with quantification over relations

$$\exists X \varphi$$

- X can be any relation of any arity over the domain
- Given a finite structure $\mathcal{U}$:

Second Order Logic

- Extends FOL formulas with quantification over relations

$$\exists X \varphi$$

- X can be any relation of any arity over the domain
- Given a finite structure $\mathcal{U}$:
  - $\mathcal{U} \models \exists X \varphi$ iff $\mathcal{U} \models \varphi[X \backslash R]$ for some n-ary R.

Second Order Logic

- Extends FOL formulas with quantification over relations

$$\exists X \varphi$$

- X can be any relation of any arity over the domain
- Given a finite structure $\mathcal{U}$:
  - $\mathcal{U} \models \exists X \varphi$ iff $\mathcal{U} \models \varphi[X \backslash R]$ for some n-ary R.

$$\begin{aligned}\varphi = \exists R \subseteq V \exists G \subseteq V \exists B \subseteq V (&\forall x (R(x) \vee B(x) \vee G(s)) \wedge \\ &\forall x (\neg(R(x) \wedge B(x)) \wedge \neg(B(x) \wedge G(x)) \wedge \neg(R(x) \wedge G(x))) \wedge \\ &\forall x \forall y (E(x,y) \to (\neg(R(x) \wedge R(y) \wedge \neg(G(x) \wedge G(y)) \wedge \\ &\neg(B(x) \wedge B(y))))))\end{aligned}$$

Logical Characterising Complexity Classes

- Existential SOL ($\exists SOL$)
  - $\exists X_1 \exists X_2 \ldots \exists X_n \varphi$, $\varphi \in FOL$

Fagin's Theorem

NP $\equiv \exists SOL$

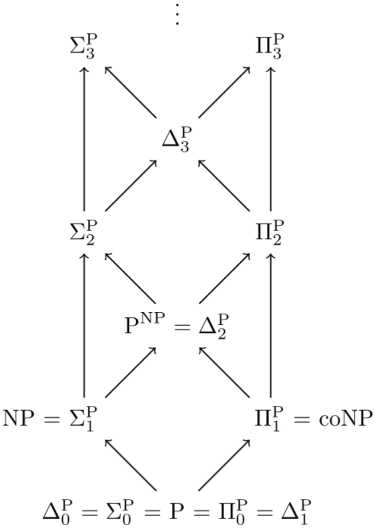
- Universal SOL ($\forall SOL$)
  - $\forall X_1 \forall X_2 \ldots \forall X_n \varphi$, $\varphi \in FOL$

Fagin's Theorem

$co -$ NP $\equiv \forall SOL$

- $UNSAT \in co -$ NP

- $\Sigma_1^{\mathsf{P}} = \exists SO$
- $\Pi_1^{\mathsf{P}} = \forall SO$
- $\Sigma_n^{\mathsf{P}} = \exists X_1 \dots \exists X_n \Pi_{n-1}^{\mathsf{P}}$
- $\Pi_n^{\mathsf{P}} = \exists X_1 \dots \exists X_n \Sigma_{n-1}^{\mathsf{P}}$
- $\Sigma_n^{\mathsf{P}} \cup \Pi_n^{\mathsf{P}} \subseteq \Sigma_{n+1}^{\mathsf{P}} \cap \Pi_{n+1}^{\mathsf{P}}$
- What tops it?



Fixed Points

- $f : \mathcal{P}(\mathcal{D}) \to \mathcal{P}(\mathcal{D})$
- **Monotone:** $X \subseteq Y$ implies $f(X) \subseteq f(Y)$
- **Inflationary:** $\forall X \in \mathcal{P}(\mathcal{D}) : X \subseteq f(X) \subseteq f(f(X)) \subseteq \ldots$
- **Fixed point:** $X \in \mathcal{P}(D)$ is an FP of f, $f(X) = X$

Fixed Points

- $X_0 = \emptyset$, $X_i = f^i(X_0)$
- If f is monotone, $lfp(f) = \bigcup_{i \geq 0} X_i$
- What if not monotone?
  - **Inflationary Fixed Point (IFP):** $ifp(f) = \bigcup_{i=0}^{n} (X_i \cup f(X_i))$
  - **Partial Fixed Point (PFP):** $pfp(f) = \begin{cases} X_n & X_n = X_{n+1} \\ \emptyset & \forall n \leq 2^{|D|} : X_n \neq X_{n+1} \end{cases}$
- Extending logics using FP operators for higher definability
- **NOTE:** For MONOTONE functions: $lfp = ifp = pfp$

Below PH:

Immerman and Vardi

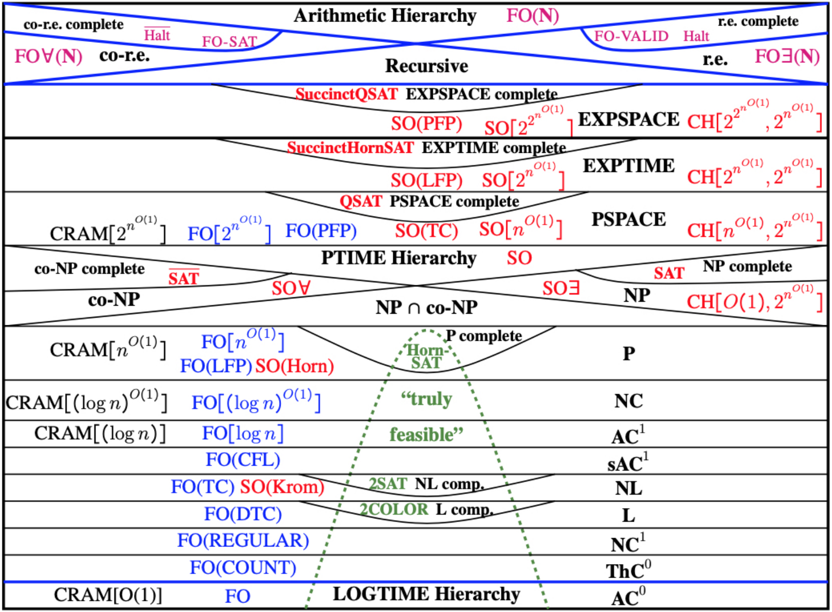
If query being done over *ordered finite* structures then

$$LFP \equiv \mathrm{P}$$

Above PH:

PSPACE Characterisation

$$SO[TC] \equiv \mathrm{PSPACE}$$


Arithmetic Hierarchy FO(N)
co-r.e. complete
Halt
FO-SAT
FO-VALID
Halt
r.e. complete
FO∀(N)
co-r.e.
Recursive
r.e.
FO∃(N)
SuccinctQSAT EXPSPACE complete
SO(PFP)
$SO[2^{2^{n^{O(1)}}}]$
EXPSPACE
$CH[2^{2^{n^{O(1)}}}, 2^{n^{O(1)}}]$
SuccinctHornSAT EXPTIME complete
SO(LFP)
$SO[2^{n^{O(1)}}]$
EXPTIME
$CH[2^{n^{O(1)}}, 2^{n^{O(1)}}]$
QSAT PSPACE complete
$CRAM[2^{n^{O(1)}}]$
$FO[2^{n^{O(1)}}]$
FO(PFP)
SO(TC)
$SO[n^{O(1)}]$
PSPACE
$CH[n^{O(1)}, 2^{n^{O(1)}}]$
PTIME Hierarchy SO
co-NP complete
$\overline{SAT}$
SO∀
SAT
NP complete
co-NP
SO∃
NP
$CH[O(1), 2^{n^{O(1)}}]$
NP ∩ co-NP
$CRAM[n^{O(1)}]$
$FO[n^{O(1)}]$
P complete
Horn-SAT
FO(LFP) SO(Horn)
P
$CRAM[(\log n)^{O(1)}]$
$FO[(\log n)^{O(1)}]$
"truly
NC
$CRAM[(\log n)]$
$FO[\log n]$
feasible"
AC^1
FO(CFL)
sAC^1
FO(TC) SO(Krom)
2SAT NL comp.
NL
FO(DTC)
2COLOR L comp.
L
FO(REGULAR)
NC^1
FO(COUNT)
ThC^0
CRAM[O(1)]
FO
LOGTIME Hierarchy
AC^0

Conclusion

- **Other Aspects**: Proof Complexity
- **Open**: Exact Characterisations for P