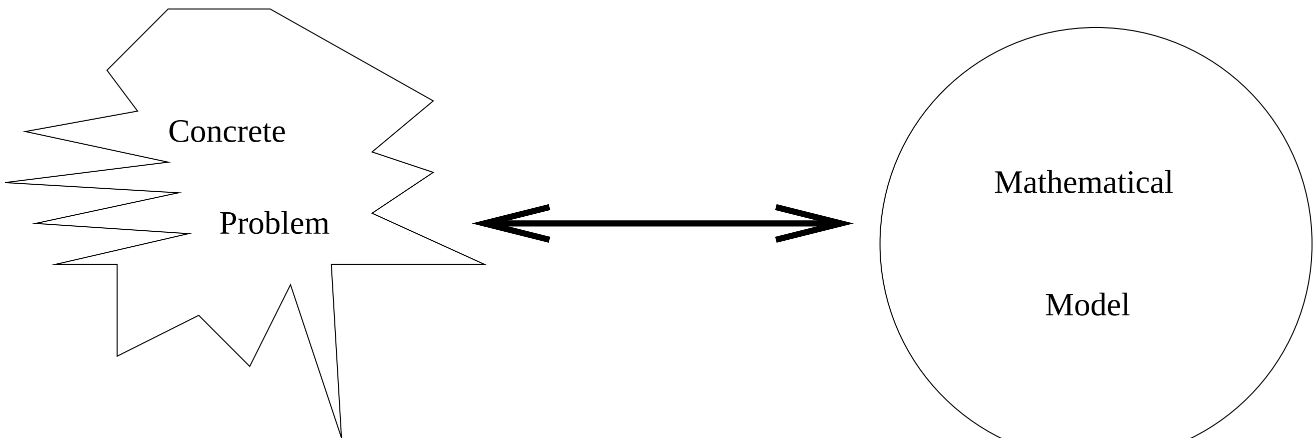
We've studied the main models and concepts of the theory of computation:

- **Computability**: what can be computed in principle
- **Logic**: how can we express our requirements
- **Complexity**: what can be computed in practice



Formal Models of Computation:

- FA $\cong$ Regular Expression
- PDA $\cong$ CFG
- TM $\cong$ Recursive Function $\cong$ Boolean Circuits $\ldots$
- logical formula

Kleene's Theorem: Let $A \subseteq \Sigma^\star$ be any language. Then the following are equivalent:

1. $A = \mathcal{L}(D)$, for some DFA D.
2. $A = \mathcal{L}(N)$, for some NFA N wo ϵ transitions
3. $A = \mathcal{L}(N)$, for some NFA N.
4. $A = \mathcal{L}(e)$, for some regular expression e.

Myhill-Nerode Theorem: The language A is regular iff $\sim_A$ has a finite number of equivalence classes. Furthermore, this number of equivalence classes is equal to the number of states in the minimum-state DFA that accepts A.

Pumping Lemma for Regular Sets: Let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Let $n = |Q|$. Let $w \in \mathcal{L}(D)$ s.t. $|w| \geq n$. Then $\exists x, y, z \in \Sigma^\star$ s.t. the following all hold:

- $xyz = w$
- $|xy| \leq n$
- $|y| > 0$, and
- $(\forall k \geq 0) xy^k z \in \mathcal{L}(D)$

Closure Theorem for Context Free Languages: Let $A, B \subseteq \Sigma^\star$ be context-free languages, let $R \subseteq \Sigma^\star$ be a regular language, and let $h : \Sigma^\star \to \Gamma^\star$ and $g : \Gamma^\star \to \Sigma^\star$ be homomorphisms. Then the following languages are context-free:

1. $A \cup B$
2. AB
3. $A \cap R$
4. $h(A)$
5. $g^{-1}(A)$

CFL Pumping Lemma: Let A be a CFL. Then there is a constant n, depending only on A such that if $z \in A$ and $|z| \geq n$, then there exist strings u, v, w, x, y such that $z = uvwxy$, and,

- $|vx| \geq 1$,
- $|vwx| \leq n$, and
- for all $k \in \mathbf{N}$, $uv^kwx^ky \in A$

A (partial) function is *recursive* iff it is computed by some TM M.

Let $S \subseteq \{0,1\}^\star$ or $S \subseteq \mathbf{N}$.

S is a *recursive set* iff the function χ_S is a (total) recursive function,

$$\chi_S(x) \;=\; \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{otherwise} \end{cases}$$

S is a *recursively enumerable set* (S is r.e.) iff the function p_S is a (partial) recursive function,

$$p_S(x) \;=\; \begin{cases} 1 & \text{if } x \in S \\ \nearrow & \text{otherwise} \end{cases}$$

Th: **Recursive = r.e. $\cap$ co-r.e.**

Define the *primitive recursive functions* to be the smallest class of functions that

- contains the Initial functions: ζ, σ, and $\pi_i^n, n = 1, 2, \ldots$, $1 \leq i \leq n$, and
- is closed under **Composition**, and
- is closed under **Primitive Recursion**

Define the *Gödel recursive functions* to be the smallest class of functions that

- contains the Initial functions, and
- is closed under **Composition**, and
- is closed under **Primitive Recursion**, and
- is closed under **Unbounded Mimimalization**

Th: [Kleene] COMP(n, x, c, y) is a primitive recursive predicate.

Theorem: A (partial) function is recursive iff it is Gödel recursive.

Cantor's Theorem: $\wp(\mathbf{N})$ is not countable!

Proof: Suppose it were. Let $f : \mathbf{N} \overset{1:1}{\underset{onto}{\rightarrow}} \wp(\mathbf{N})$. Define the diagonal set,

$$D \quad = \quad \{n \mid n \notin f(n)\}$$

Thus $D = f(k)$ for some $k \in \mathbf{N}$.

$$k \in D \quad \Leftrightarrow \quad k \notin f(k) \quad \Leftrightarrow \quad k \notin D$$

$\Rightarrow\Leftarrow$ Therefore, $\wp(\mathbf{N})$ is not countable! ♠

n	0	1	2	3	4	5	6	7	8	$\cdots$	$f(n)$
0	$\boxed{0}$	0	0	0	0	0	0	0	0	$\cdots$	$f(0)$
1	1	$\boxed{1}$	1	1	1	1	1	1	1	$\cdots$	$f(1)$
2	1	0	$\boxed{1}$	0	1	0	1	0	1	$\cdots$	$f(2)$
3	0	1	0	$\boxed{1}$	0	1	0	1	0	$\cdots$	$f(3)$
4	1	0	0	0	$\boxed{0}$	0	0	0	0	$\cdots$	$f(4)$
5	0	1	1	0	1	$\boxed{0}$	0	0	1	$\cdots$	$f(5)$
6	1	0	0	1	0	0	$\boxed{1}$	0	0	$\cdots$	$f(6)$
7	1	1	0	0	0	0	0	$\boxed{0}$	0	$\cdots$	$f(7)$
8	0	1	0	0	0	0	0	0	$\boxed{0}$	$\cdots$	$f(8)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
	1	0	0	0	1	1	0	1	1	$\cdots$	D

$$K \quad = \quad \{n \mid M_n(n) = 1\}$$

Theorem: $\overline{K}$ is not r.e.

Hierarchy Theorems: Let $f(n)$ be a well behaved function, and $\mathcal{C}$ one of DSPACE, NSPACE, DTIME, NTIME.

If $g(n)$ is sufficiently smaller than $f(n)$ then $\mathcal{C}[g(n)]$ is strictly contained in $\mathcal{C}[f(n)]$.

"$g(n)$ sufficiently smaller than $f(n)$" means

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$	$\lim_{n\to\infty} \frac{g(n)\log(g(n))}{f(n)} = 0$
$\mathcal{C} = \mathbf{DSPACE}, \mathbf{NSPACE}, \mathbf{NTIME}$	$\mathcal{C} = \mathbf{DTIME}$

Hence $\mathbf{P} \neq \mathbf{EXPTIME}$, $\mathbf{L} \neq \mathbf{PSPACE}$.

But these are the *only* separations of classes we know! (Except at the p.r. and above level, and for **REG** and **CFL**).

Th: The busy beaver function is eventually larger than any total, recursive function.

Th: Let $S \subseteq \mathbf{N}$. T.F.A.E.

1. S is the domain of a partial, recursive function.
2. $S = \emptyset$ or S is the range of a total, recursive function.
3. S is the range of a partial, recursive function.
4. $S = W_n$, some $n = 0, 1, 2, \ldots$ where

$$W_n \quad = \quad \{m \mid M_n(m) = 1\}$$

Definitions of **Formula, Structure,** and **Truth**

Axioms and Proof Rules

Modus Ponens (M.P.): From φ, $\varphi \to \psi$, conclude ψ.

Proposition: Modus Ponens preserves validity.

Axioms: all generalizations of the following

0	Tautologies on **at most three** boolean variables
1a	$t = t$
1b	$(t_1 = t_1' \wedge \cdots \wedge t_k = t_k') \to f(t_1, \ldots, t_k) = f(t_1', \ldots, t_k')$
1c	$(t_1 = t_1' \wedge \cdots \wedge t_k = t_k') \to R(t_1, \ldots, t_k) \to R(t_1', \ldots, t_k')$
2	$(\forall x)(\varphi) \to \varphi[x \leftarrow t]$
3	$\varphi \to (\forall x)(\varphi)$, x not free in φ
4	$(\forall x)(\varphi \to \psi) \to ((\forall x)(\varphi) \to (\forall x)(\psi))$

Proposition: Every instance of every axiom is valid.

$$\text{FO-THEOREMS} \quad = \quad \{\varphi \mid \vdash \varphi\}$$

Soundness Theorem: If $\vdash \varphi$ then $\models \varphi$.

$$\text{FO-THEOREMS} \subseteq \text{FO-VALID}$$

Completeness Theorem: If $\models \varphi$ then $\vdash \varphi$.

$$\text{FO-THEOREMS} \supseteq \text{FO-VALID}$$

Corollary:

$$\vdash \; = \; \models; \qquad \text{FO-THEOREMS} = \text{FO-VALID}$$

Compactness Th: If every finite subset of Γ has a model, then Γ has a model.

Gödel's Incompleteness Theorem:

Theory($\mathbf{N}$) is not r.e. and thus not axiomatizable.

Th: For $t(n) \geq n, s(n) \geq \log n$,

$$\textbf{DTIME}[t(n)] \subseteq \textbf{NTIME}[t(n)] \subseteq \textbf{DSPACE}[t(n)]$$

$$\textbf{DSPACE}[s(n)] \quad \subseteq \quad \textbf{DTIME}[2^{O(s(n))}]$$

Savitch's Theorem:

For $s(n) \geq \log n$,

$$\textbf{NSPACE}[s(n)] \subseteq \textbf{ATIME}(s(n))^2 \subseteq \textbf{DSPACE}[(s(n))^2]$$

Immerman-Szelepcsényi Theorem:

For $s(n) \geq \log n$,

$$\textbf{NSPACE}[s(n)] \quad = \quad \text{co-}\textbf{NSPACE}[s(n)]$$

Theorem: Let $\mathcal{C}$ be one of the following complexity classes: L, NL, P, NP, co-NP, PSPACE, EXPTIME, Primitive-Recursive, RECURSIVE, r.e., co-r.e.

$$\text{Suppose } A \leq B.$$
$$\text{If } B \in \mathcal{C} \quad \text{Then} \quad A \in \mathcal{C}$$

All these complexity classes are **closed under reductions.**

Lower Bounds: If A is hard then B is hard.

Upper Bounds: If B is easy then A is easy.

Complete for NL: REACH, EMPTY-DFA, EMPTY-NFA, 2-SAT

Complete for P: CVP, MCVP, EMPTY-CFL, Horn-SAT, REACH_a

Complete for NP: TSP, SAT, 3-SAT, 3-COLOR, CLIQUE, Subset Sum, Knapsack

Complete for PSPACE: QSAT, GEOGRAPHY, SUCCINT-REACH, REG-EXP-$\Sigma^\star$

Complete for r.e.: K, HALT, $A_{0,17}$, FO-VALID

Complete for co-r.e.: $\overline{K}$, $\Sigma^\star$CFL, EMPTY, FO-SAT

Theorem:

$$\begin{aligned}
\textbf{r.e.} &= \mathrm{FO}\exists(\mathbf{N}) \\
\text{co-}\textbf{r.e.} &= \mathrm{FO}\forall(\mathbf{N}) \\
\\
\mathrm{PH} &= \mathrm{SO} \\
\mathbf{NP} &= \mathrm{SO}\exists \\
\mathbf{P} &= \mathrm{SO}\exists\text{-Horn} \\
\mathbf{AC}^0 &= \mathbf{CRAM}[1] \;=\; \mathrm{LH} \;=\; \mathrm{FO}
\end{aligned}$$

One can understand the complexity of a problem as the richness of a logical language that is needed to describe the problem.

Theorem: For $s(n) \geq \log n$, and for $t(n) \geq n$,

$$\bigcup_{k=1}^{\infty} \textbf{ATIME}[(t(n))^k] \;=\; \bigcup_{k=1}^{\infty} \textbf{DSPACE}[(t(n))^k]$$

$$\textbf{ASPACE}[s(n)] \;=\; \bigcup_{k=1}^{\infty} \textbf{DTIME}[k^{s(n)}]$$

Corollary: In particular,

$$\textbf{ASPACE}[\log n] \;=\; \textbf{P}$$

$$\textbf{ATIME}[n^{O(1)}] \;=\; \textbf{PSPACE}$$

$$\textbf{ASPACE}[n^{O(1)}] \;=\; \textbf{EXPTIME}$$

$$\text{depth} \quad = \quad \text{parallel time}$$

$$\text{width} \;=\; \text{hardware}$$

$$\text{number of gates} \;=\; \text{computational work} \;=\; \text{sequential time}$$

Theorem: For all i, $\mathbf{CRAM}[(\log n)^i] \quad = \quad \mathbf{AC}^i$

$$\begin{array}{cccccccccccc}
\mathbf{AC}^0 & \subseteq & \mathbf{ThC}^0 & \subseteq & \mathbf{NC}^1 & \subseteq & \mathbf{L} & \subseteq & \mathbf{NL} & \subseteq & \mathbf{sAC}^1 & \subseteq \\
\mathbf{AC}^1 & \subseteq & \mathbf{ThC}^1 & \subseteq & \mathbf{NC}^2 & & & \subseteq & & & \mathbf{sAC}^2 & \subseteq \\
\vdots & \subseteq & \vdots & \subseteq & \vdots & & & \subseteq & & & \vdots & \subseteq \\
\mathbf{AC}^i & \subseteq & \mathbf{ThC}^i & \subseteq & \mathbf{NC}^{i+1} & & & \subseteq & & & \mathbf{sAC}^{i+1} & \subseteq \\
\vdots & \subseteq & \vdots & \subseteq & \vdots & & & \subseteq & & & \vdots & \subseteq \\
\mathbf{NC} & = & \mathbf{NC} & = & \mathbf{NC} & & & = & & & \mathbf{NC} & =
\end{array}$$

$$\mathbf{NC} \quad \subseteq \quad \mathbf{P} \quad \subseteq \quad \mathbf{NP}$$

Alternation/Circuit Theorem:

Log-space ATM's with:

- $O(\log^i n)$ time give $\mathbf{NC}^i$ ($i \geq 1$)
- $O(\log^i n)$ alternations give $\mathbf{AC}^i$ ($i \geq 1$)

Alternating TM's are one good way to design uniform families of circuits. We used this method to prove $\mathbf{CFL} \subseteq \mathbf{sAC}^1$.

First-order logic gives us another way to design uniform families of circuits. We've used this to construct $\mathbf{AC}^0$ circuits by showing a problem to be in FO.

We need uniformity definitions on our circuit classes to relate them to ordinary classes. For example, poly-size circuit families compute languages in **P** only if they are at least **P**-uniform.

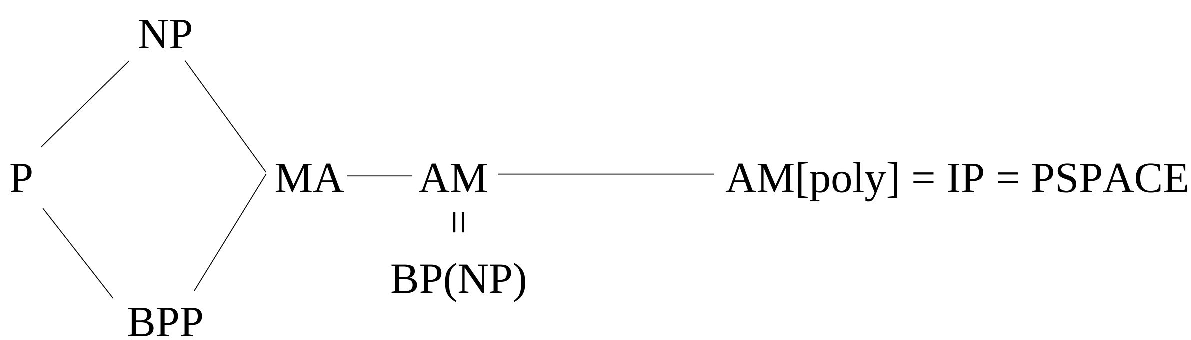
Theorem: PRIME and Factoring are in **NP** $\cap$ co-**NP**. (PRIME is now in **P** as well.)

Theorem: [Solovay-Strassen, Miller]

$$\text{PRIME} \in \textbf{BPP}$$

Fact: $\text{REACH}_u \in \text{BPL}$

Interactive Proofs



Fact: PCP[$\log n, 1$] = NP

A is an *optimization problem* iff

For each instance x, $F(x)$ is the set of *feasible solutions*

Each $s \in F(x)$ has a cost $c(s) \in \mathbf{Z}^+$

For minimization problems,

$$\mathrm{OPT}(x) \quad = \quad \min_{s \in F(x)} c(s)$$

For maximization problems,

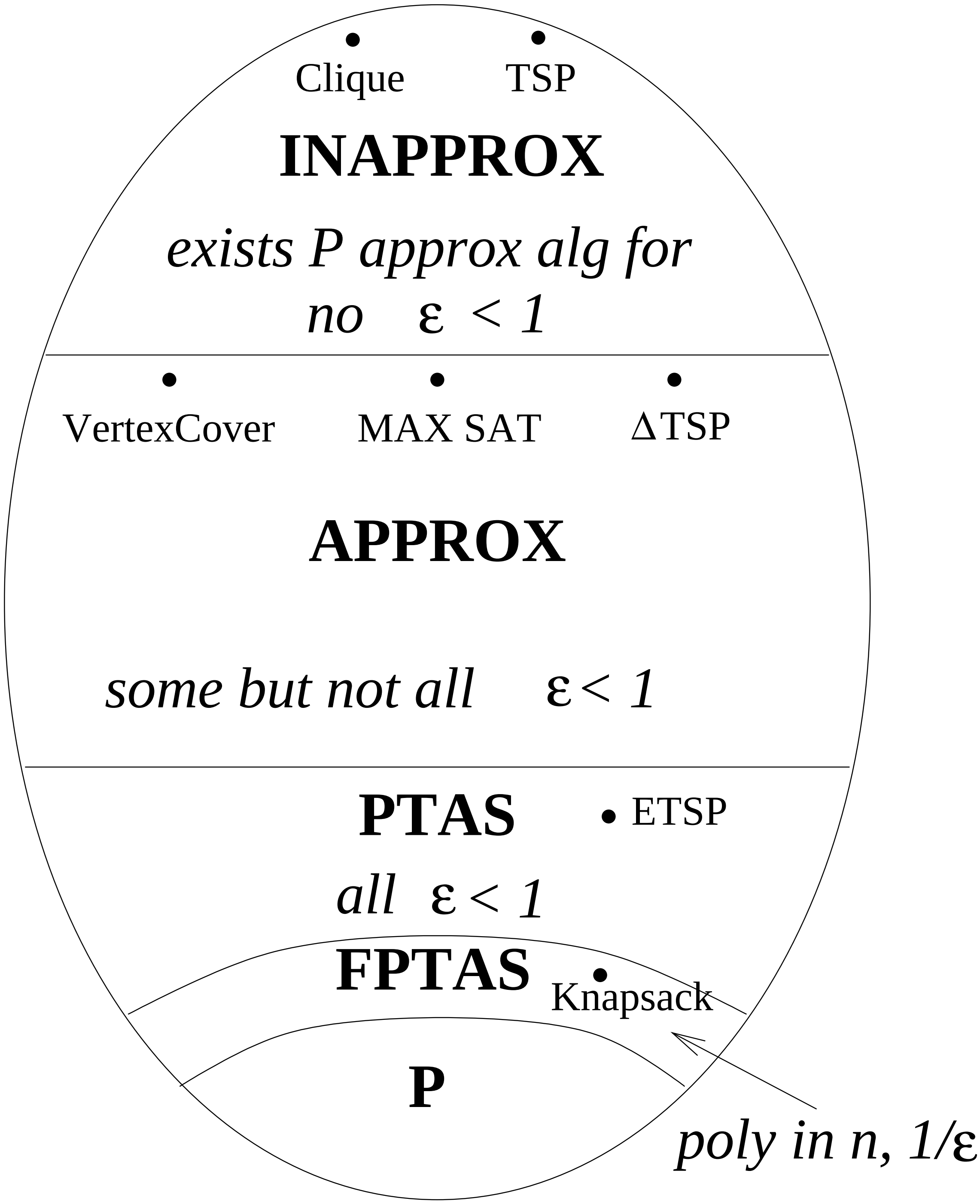
$$\mathrm{OPT}(x) \quad = \quad \max_{s \in F(x)} c(s)$$

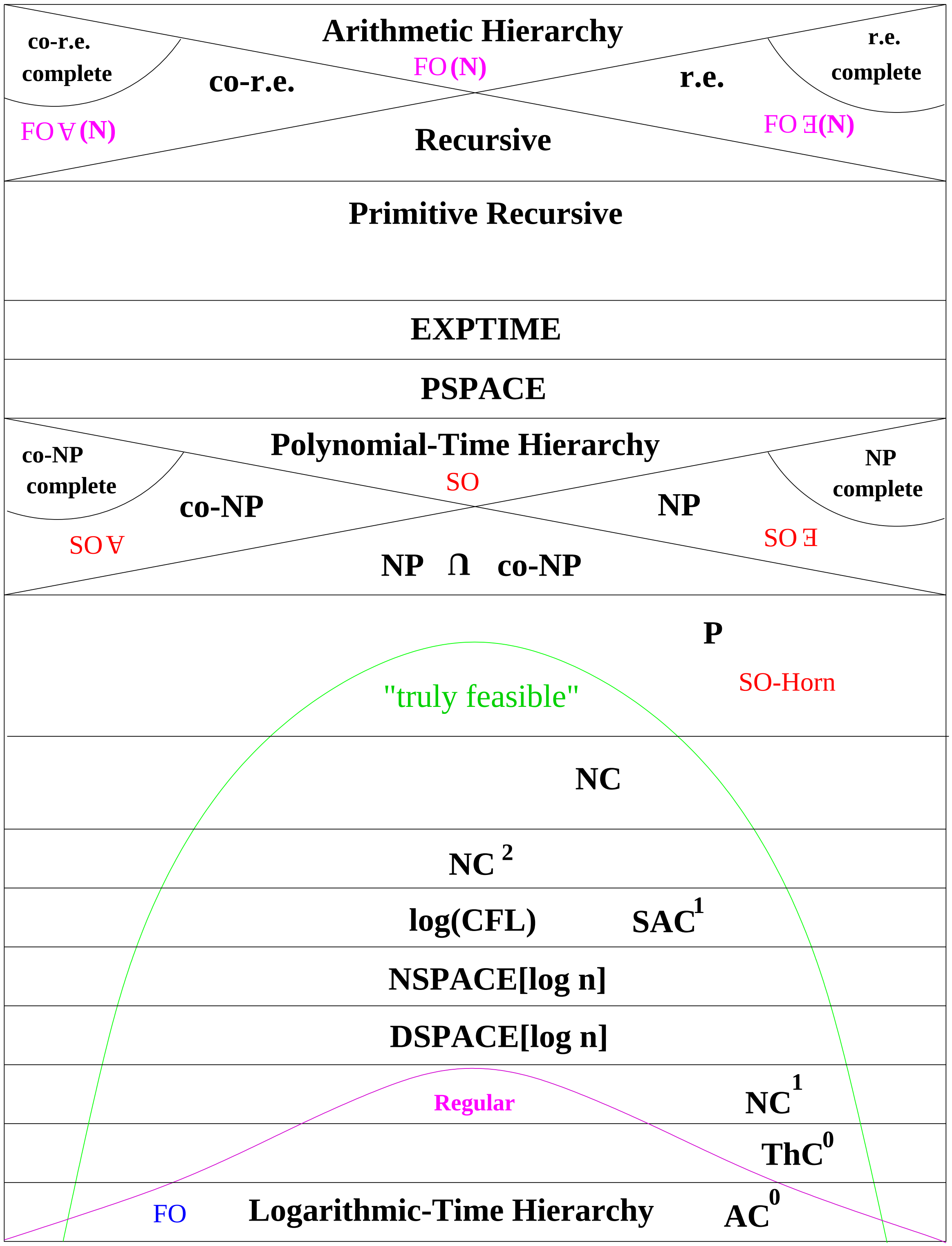
Let M be an algorithm s.t. on any instance x,

$$M(x) \in F(x)$$

M is an ϵ-*approximation algorithm* iff for all x,

$$\frac{|c(M(x)) - \mathrm{OPT}(x)|}{\max(\mathrm{OPT}(x), c(M(x)))} \;\leq\; \epsilon \qquad \spadesuit$$


Clique
TSP
INAPPROX
exists P approx alg for
no ε < 1
VertexCover
MAX SAT
ΔTSP
APPROX
some but not all ε < 1
PTAS
ETSP
all ε < 1
FPTAS
Knapsack
P
poly in n, 1/ε


Arithmetic Hierarchy
co-r.e. complete
co-r.e.
FO(N)
r.e.
r.e. complete
FO∀(N)
Recursive
FO∃(N)
Primitive Recursive
EXPTIME
PSPACE
Polynomial-Time Hierarchy
co-NP complete
co-NP
SO
NP
NP complete
SO∀
NP ∩ co-NP
SO∃
P
SO-Horn
"truly feasible"
NC
NC 2
log(CFL)
SAC1
NSPACE[log n]
DSPACE[log n]
Regular
NC1
ThC0
FO
Logarithmic-Time Hierarchy
AC0

Why are the following so hard to prove?

- **P** $\neq$ **NP**
- **P** $\neq$ **PSPACE**
- **ThC**^{0} $\neq$ **NP**
- **BPP** $=$ **P**

We do know a lot about computation. Reductions and complete problems are a key tool. So is the equivalence of apparently different models and methods. Yet much remains unknown.