First-Order Logic: A vocabulary to talk about a particular type of domain, formulas with atomic predicates, boolean operators, and quantifiers.

Fitch: A proof system including the rules we had for boolean operators plus new rules for identity and quantifiers.

Main Theorems:

- **Soundness:** (Lecture 14) If $P \vdash Q$ in Fitch, and $\mathcal{M} \models P$, then $\mathcal{M} \models Q$.
- **Completeness:** (Lecture 15) If $P \models Q$, meaning that Q is true in any model where P is true, then $P \vdash Q$ in Fitch.
- **Compactness:** (Lecture 16) If every finite subset of Γ has a model, then Γ has a model.
- **Incompleteness:** (Lecture 16) Any r.e. set of sentences Γ in the language of number theory either contains some statement that is false in **N** or cannot prove some statement that is true in **N**.

Relating Logic to Computability:

We observed that it is easy, certainly primitive recursive, to test whether an alleged proof is valid according to the rules of Fitch.

Corollary to Completeness:

$$\Gamma \models \varphi \qquad \Leftrightarrow \qquad \Gamma \vdash \varphi$$

$$\models \varphi \qquad \Leftrightarrow \qquad \vdash \varphi$$

$$\textbf{FO-VALID} \qquad = \qquad \textbf{FO-THEOREMS}$$

Note that **FO-VALID** and **FO-THEOREMS** are statements that are true in any model of the given vocabulary. While we *constructed* a special model where *every* statement was either provably true or provably false, this is not true in general. An **FO-VALID** statement about graphs would be true for all graphs, but most interesting statements are true for some graphs and not for others.

Theorem 17.1 FO-THEOREMS *is* **r.e.** *complete.*

Proof: We have already seen that FO-THEOREMS is **r.e.**, because we can semi-decide whether a formula S is a theorem by searching all strings in parallel to see whether any is a proof of S.

Recall that the language K is represented by a bounded formula φ_K.

$$n \in K \quad \Leftrightarrow \quad \mathbf{N} \models \varphi_K(n) \quad \Leftrightarrow \quad \mathrm{NT} \vdash \varphi_K(n)$$

$$n \in K \qquad \Leftrightarrow \qquad \text{“}\mathrm{NT} \rightarrow \varphi_K(n)\text{”} \in \text{FO-THEOREMS}$$

We have thus shown that $K \leq$ FO-THEOREMS, by defining f so that:

$$f(n) = \text{“}\mathrm{NT} \rightarrow \varphi_K(n)\text{”}$$

♠

Note that this function is only well-defined because NT is a *finite* set of formulas.

Definition A set $A \subseteq \Sigma^\star$ is in **DTIME**$[t(n)]$ iff there exists a deterministic, multi-tape TM, M, and a constant c, such that,

1. $A = \mathcal{L}(M) \equiv \{w \in \Sigma^\star \mid M(w) = 1\}$, and
2. $\forall w \in \Sigma^\star$, $M(w)$ halts within $c(1 + t(|w|))$ steps.

Definition A set $A \subseteq \Sigma^\star$ is in **DSPACE**$[s(n)]$ iff there exists a deterministic, multi-tape TM, M, and a constant c, such that,

1. $A = \mathcal{L}(M)$, and
2. $\forall w \in \Sigma^\star$, $M(w)$ uses at most $c(1 + s(|w|))$ work-tape cells.

(The input tape is considered "read-only" and not counted as space used.)

$$
\begin{array}{lclcl}
\mathbf{L} & \equiv & \mathbf{DSPACE}[\log n] & & \\
\mathbf{P} & \equiv & \mathbf{DTIME}[n^{O(1)}] & \equiv & \bigcup_{i=1}^{\infty} \mathbf{DTIME}[n^i] \\
\mathbf{PSPACE} & \equiv & \mathbf{DSPACE}[n^{O(1)}] & \equiv & \bigcup_{i=1}^{\infty} \mathbf{DSPACE}[n^i]
\end{array}
$$

Theorem For any functions $t(n) \geq n, s(n) \geq \log n$, we have

$$
\begin{array}{rcl}
\mathbf{DTIME}[t(n)] & \subseteq & \mathbf{DSPACE}[t(n)] \\
\mathbf{DSPACE}[s(n)] & \subseteq & \mathbf{DTIME}[2^{O(s(n))}]
\end{array}
$$

Proof: Let M be a $\mathbf{DSPACE}[s(n)]$ TM, let $w \in \Sigma^\star$, let $n = |w|$

$M(w)$ has at most,

$$
|Q| \cdot (n + cs(n) + 2)^k \cdot |\Sigma|^{cs(n)} \quad < \quad 2^{c's(n)}
$$

possible configurations.

Thus, after $2^{c's(n)}$ steps, $M(w)$ must be in an infinite loop. ♠

Corollary $\mathbf{L} \subseteq \mathbf{P} \subseteq \mathbf{PSPACE}$

$$\textbf{NTIME}[t(n)] \equiv \text{problems accepted by NTMs in time } t(n)$$

$$\textbf{NP} \equiv \textbf{NTIME}[n^{O(1)}] \equiv \bigcup_{i=1}^{\infty} \textbf{NTIME}[n^i]$$

Theorem For any function $t(n)$,

$$\begin{aligned} \textbf{DTIME}[t(n)] &\subseteq \textbf{NTIME}[t(n)] \\ &\subseteq \textbf{DSPACE}[t(n)] \subseteq \textbf{DTIME}[2^{O(t(n))}] \end{aligned}$$

Corollary

$$\textbf{L} \subseteq \textbf{P} \subseteq \textbf{NP} \subseteq \textbf{PSPACE}$$

For any complexity class $\mathcal{C}$, define $F(\mathcal{C})$, the total, polynomially-bounded functions computable in $\mathcal{C}$ as follows:

$$F(\mathcal{C}) \;=\; \left\{ h : \Sigma^\star \to \Sigma^\star \;\middle|\; \begin{array}{c} (\exists k)(\forall x)(|h(x)| \le k|x|^k \\ \text{and} \quad \text{bit-graph}(h) \in \mathcal{C}) \end{array} \right\}$$

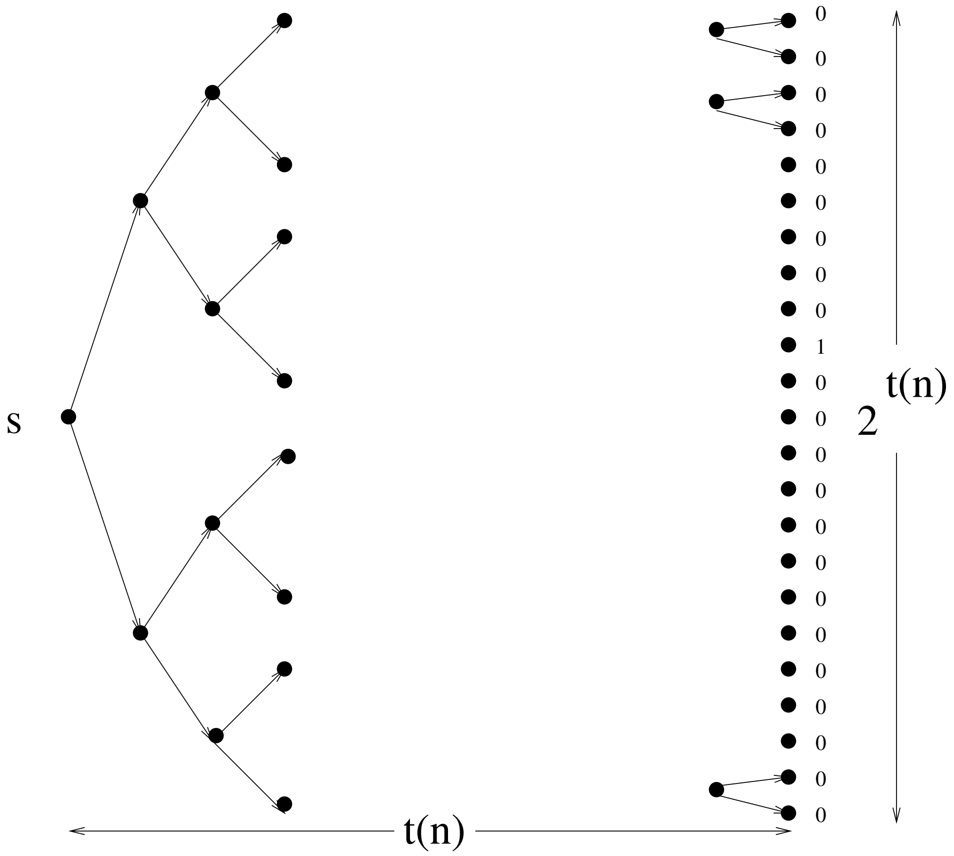
$$\text{bit-graph}(h) \;=\; \{\langle x, i, b\rangle \;\mid\; \text{bit } i \text{ of } h(x) \text{ is } b\}$$

Idea: $f \in F(\mathcal{C})$ iff

1. f is polynomially bounded, and,
2. bit i of $f(w)$ is uniformly computable in $\mathcal{C}$ and co-$\mathcal{C}$.

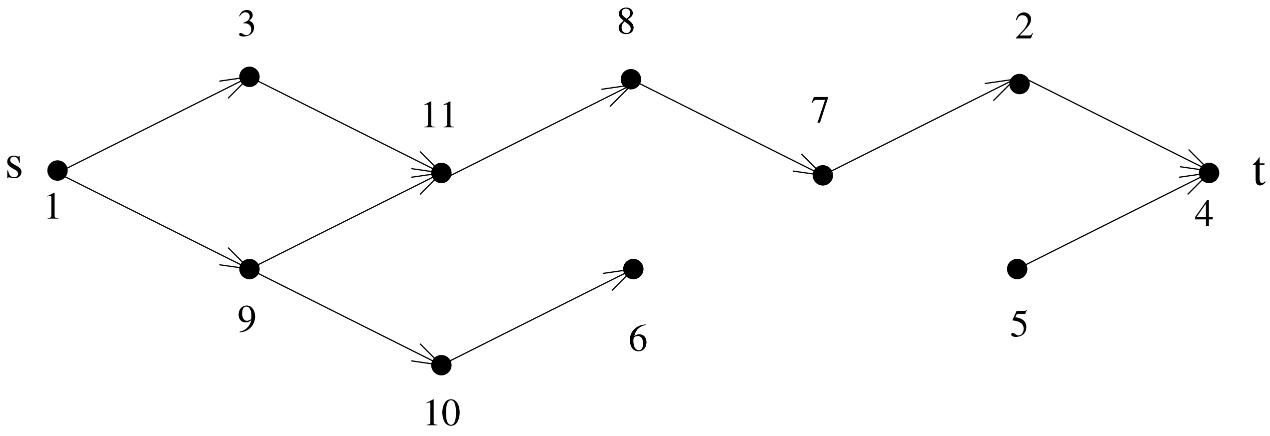
We rule out functions that are not polynomially bounded because we want to be able to define the complexity class in terms of either the input length or the output length.

NSPACE$[s(n)]$ is the set of problems accepted by NTMs using at most $O(s(n))$ space on each branch.



$s(n)$ space on each branch

Definition REACH is the set of directed graphs G such that there is a path in G from s to t. (Remember that in $\mathcal{L}_G$, graphs always have constants for the vertices s and t.)



We observed earlier that REACH is in the class **P**, because we can depth-first search G starting from s, in time $O(e) = O(n^2)$, and see whether we ever encounter t. This algorithm is not space-efficient, however, as we need $O(n)$ space to keep track of whether each vertex has been visited.

Proposition REACH $\in$ **NL** $=$ **NSPACE**$[\log n]$

Proof: Let $n = |V|$, and let a and b be variables ranging over vertices.

```
boolean reach() {
    b = s;
    for (int c=0; c < n; c++) {
        if (b==t) return true;
        a = b;
        b = nondeterministicChoice();
        if (not edge(a,b)) return false;}
    return false;}
```

This algorithm *might* return true if the input graph is in REACH, and *cannot* return true if it is not. It uses three variables of $O(\log n)$ bits each, and thus puts the language REACH in the class **NSPACE**$[\log n]$. ♠

Definition 17.2 A problem T is *complete* for a complexity class $\mathcal{C}$ iff

1. $T \in \mathcal{C}$, and
2. $(\forall A \in \mathcal{C})(A \leq T)$

♠

We have to redefine the notion of reduction used in defining the $\leq$ symbol. Total recursive reductions would make the concept of **NP**-completeness unreasonable.

NP-completeness is usually defined in terms of reductions in the class $F(\mathbf{P})$, functions computable in polynomial time. But we're going to want to talk about languages complete for **P** and **NL**, so we redefine our reductions to be in $F(\mathbf{L})$.

We're most interested in **natural** complete problems for these classes, meaning problems that someone might have posed other than as examples of complete problems. It turns out that the natural complete problems we will see remain complete under any **reasonable** notion of reduction.

Theorem 17.3 REACH *is complete for* **NL**.

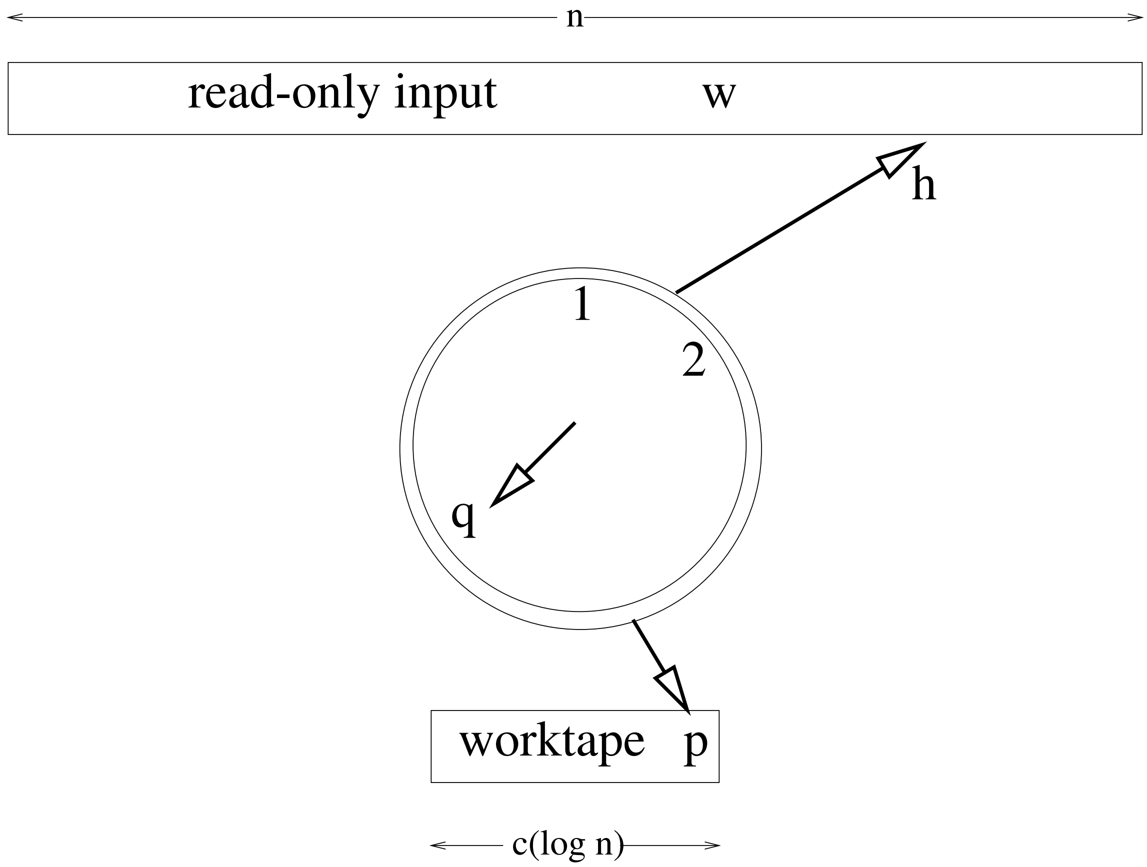
Proof: Let $A \in$ **NL**, $A = \mathcal{L}(N)$, uses $c \log n$ bits of worktape.

Input w, $\quad n = |w|$

$$w \;\mapsto\; \text{CompGraph}(N, w) \quad = \quad (V, E, s, t)$$

$$\begin{aligned}
V &= \{\text{ID} = \langle q, h, p \rangle \mid q \in \text{States}(N), h \leq n, |p| \leq c\lceil \log n \rceil\} \\
E &= \{(\text{ID}_1, \text{ID}_2) \mid \text{ID}_1(w) \xrightarrow[N]{} \text{ID}_2(w)\} \\
s &= \text{initial ID} \\
t &= \text{accepting ID}
\end{aligned}$$

An accepting computation of N corresponds exactly to a path through the configuration graph, from the start configuration to the accepting configuration, following an edge for each computation step.



$$\text{CompGraph}(N, w) \quad = \quad (V, E, s, t)$$

$$V \;=\; \{\text{ID} = \langle q, h, p\rangle \;\mid\; q \in \text{States}(N), h \le n, |p| \le c\lceil \log n\rceil\}$$

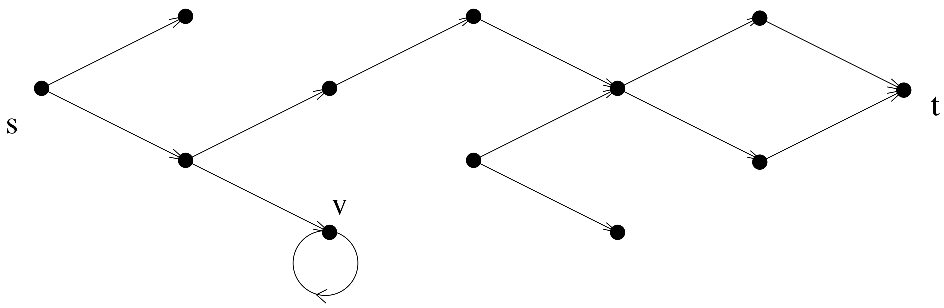
$$E \;=\; \{(\text{ID}_1, \text{ID}_2) \;\mid\; \text{ID}_1(w) \underset{N}{\longrightarrow} \text{ID}_2(w)\}$$

$$s \;=\; \text{initial ID}$$

$$t \;=\; \text{accepting ID}$$

Claim:

$$w \in A \;\Leftrightarrow\; w \in \mathcal{L}(N) \;\Leftrightarrow\; \text{CompGraph}(N, w) \in \text{REACH}$$



♠

Corollary 17.4

$$\mathbf{NL} \quad \subseteq \quad \mathbf{P}$$

Proof: We say that REACH $\in$ **P**

P is closed under (logspace) reductions.
That is

$$(B \in \mathbf{P} \quad \wedge \quad A \leq B) \quad \Rightarrow \quad A \in \mathbf{P}$$ ♠

Definition 17.5 Function $f : \mathbf{N} \to \mathbf{N}$ is $\mathcal{C}$**-constructible** if the map

$$1^n \;\mapsto\; f(n)$$

is computable in the complexity class $\mathcal{C}[f(n)]$. For example a function $f(n)$ is DSPACE-constructible if the function $f(n)$ can be deterministically computed from the input 1^n, using space at most $O[f(n)]$. ♠

Fact 17.6 *All reasonable functions greater than or equal to* $\log n$ *are DSPACE-constructible, and all reasonable functions greater than or equal to* n *are DTIME-constructible.*

The Four Hierarchy Theorems:

Theorem 17.7 *If $f(n)$ is a $\mathcal{C}$-constructible function; $\mathcal{C}$ is DSPACE, NSPACE, DTIME, or NTIME; and, 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:

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$$

$$\textit{for } \mathcal{C} = \mathbf{DSPACE}, \mathbf{NSPACE}, \mathbf{NTIME}, \textit{and}$$

$$\lim_{n\to\infty} \frac{g(n)\log(g(n))}{f(n)} = 0$$

$$\textit{for } \mathcal{C} = \mathbf{DTIME}$$

We'll only prove one of these four in lecture:

Theorem 17.8 (Space Hierarchy Theorem)
Let $f \geq \log n$ *be a space constructible function. If*

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$$

Then, $\quad \mathbf{DSPACE}[g(n)] \quad \subsetneq \quad \mathbf{DSPACE}[f(n)].$

Proof: Construct following $\mathbf{DSPACE}[f(n)]$ machine, D:

Input: $w, \quad n = |w|$

1. Mark off $6f(n)$ tape cells, (f space constructible)
2. Simulate $M_w(w)$ using space $3f(n)$, time $\leq 2^{3f(n)}$
3. **if** ($M_w(w)$ needs more space or time) **then accept**
4. **else if** ($M_w(w)$ = **accept**) **then reject**
5. **else accept** // ($M_w(w)$ = **reject**)

$$\underbrace{\boxed{\text{space to simulate } M_w(w)}}_{3f(n)}\underbrace{\boxed{\qquad\text{counter}\qquad}}_{3f(n)}$$

Claim: $\mathcal{L}(D) \in \mathbf{DSPACE}[f(n)] - \mathbf{DSPACE}[g(n)]$

Clearly $\mathcal{L}(D) \in \mathbf{DSPACE}[f(n)]$ by the construction.

Suppose that $\mathcal{L}(D) \in \mathbf{DSPACE}[g(n)]$.

Let $\mathcal{L}(M_w) = \mathcal{L}(D)$, where M_w uses $cg(n)$ space.

Choose a number N such that $(\forall n > N)(cg(n) < f(n))$.

Choose a string w' such that $M_{w'}$ and M_w compute the same function, and that $|w'| > N$. (Add useless states to M_w, for example.)

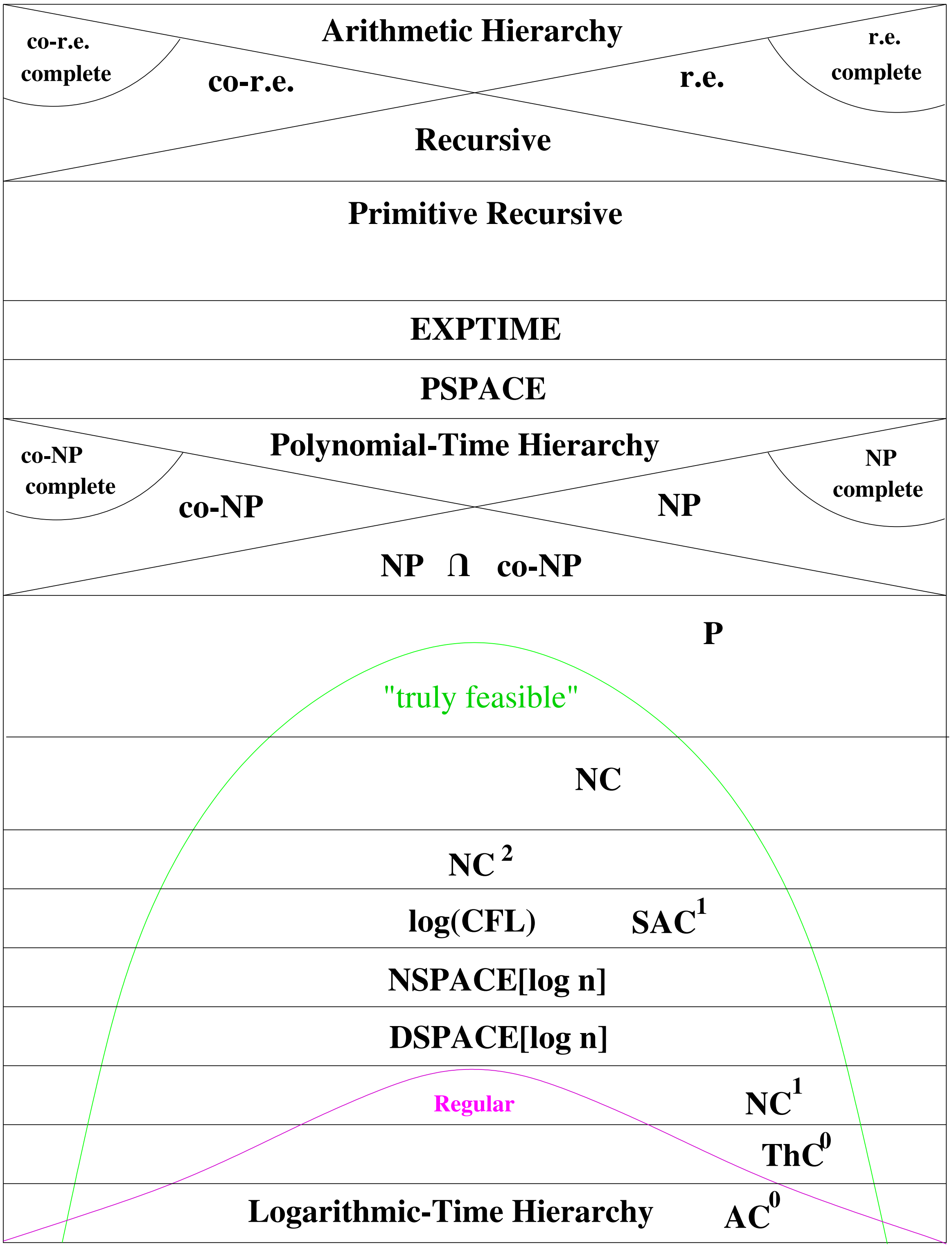
On input w', D successfully simulates $M_{w'}(w')$ in $3f(n)$ space and $2^{3f(n)}$ time.

But now we have a contradiction:

$$w' \in \mathcal{L}(D) \;\Leftrightarrow\; w' \notin \mathcal{L}(M_{w'}) \;\Leftrightarrow$$

$$w' \notin \mathcal{L}(M_w) \;\Leftrightarrow\; w' \notin \mathcal{L}(D)$$

♠


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