Recall From Last Time

Parallel Computation: Many computations at once, we measure *parallel time* and *amount of hardware*.

Models: (time measure, hardware measure)

- Parallel RAM: number of steps, number of processors
- Alternating TM: alternations, $2^{\mathrm{S}pace}$
- Boolean Circuits: depth, size

Uniformity: The n-input circuit in the family must be easily ($F(\mathbf{L})$, $F(\mathrm{FO})$ computable from input 1^n.

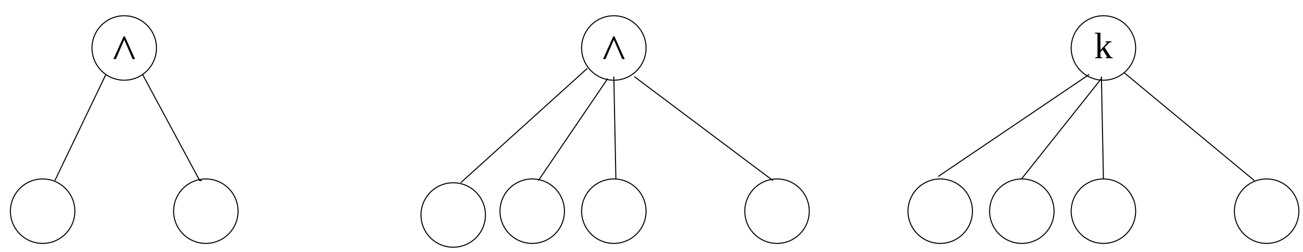
Theorem: **P** is uniform **PSIZE**.

NC Hierarchy: Classes of programs with *fast* ($(\log n)^{O(1)}$ time) parallel algorithms that use *reasonable* ($n^{O(1)}$) hardware.

Definition 23.1 (The NC Hierarchy) Let $t(n)$ be a polynomially bounded function and let $S \subseteq \{0,1\}^\star$ Then S is in the circuit complexity class **NC**$[t(n)]$, **AC**$[t(n)]$, **ThC**$[t(n)]$, respectively iff there exists a uniform family of circuits $C_1, C_2, \ldots$ with the following properties:

1. For all $w \in \{0,1\}^\star, \quad w \in S \quad \Leftrightarrow \quad C_{|w|}(w) = 1$
2. The depth of C_n is $O(t(n))$.
3. $|C_n| \leq n^{O(1)}$
4. The gates of C_n consist of,

NC	AC	ThC
bounded fan-in and, or gates	unbounded fan-in and, or gates	unbounded fan-in threshold gates



For $i = 0, 1, \ldots$,

$$\begin{aligned}
\mathbf{NC}^i &= \mathbf{NC}[(\log n)^i] \\
\mathbf{AC}^i &= \mathbf{AC}[(\log n)^i] \\
\mathbf{ThC}^i &= \mathbf{ThC}[(\log n)^i]
\end{aligned}$$

$$\mathbf{NC} \quad = \quad \bigcup_{i=0}^{\infty} \mathbf{NC}^i \quad = \quad \bigcup_{i=0}^{\infty} \mathbf{AC}^i \quad = \quad \bigcup_{i=0}^{\infty} \mathbf{ThC}^i$$

We will see that the following inclusions hold:

$$\begin{array}{ccccccc}
\mathbf{AC}^0 & \subseteq & \mathbf{ThC}^0 & \subseteq & \mathbf{NC}^1 \subseteq \mathbf{L} \subseteq \mathbf{NL} & \subseteq & \mathbf{AC}^1 \\
\mathbf{AC}^1 & \subseteq & \mathbf{ThC}^1 & \subseteq & \mathbf{NC}^2 & \subseteq & \mathbf{AC}^2 \\
\mathbf{AC}^2 & \subseteq & \mathbf{ThC}^2 & \subseteq & \mathbf{NC}^3 & \subseteq & \mathbf{AC}^3 \\
\vdots & \subseteq & \vdots & \subseteq & \vdots & \subseteq & \vdots \\
\bigcup_{i=1}^{\infty} \mathbf{AC}^i & = & \bigcup_{i=1}^{\infty} \mathbf{ThC}^i & = & \bigcup_{i=1}^{\infty} \mathbf{NC}^i & = & \mathbf{NC}
\end{array}$$

Overall, **NC** consists of those problems that can be solved in *poly-log parallel time* on a parallel computer with *polynomially much hardware*. The question of whether $\mathbf{P} = \mathbf{NC}$ is the *second* most important open question in complexity theory, after the $\mathbf{P} = \mathbf{NP}$ question.

You wouldn't think that *every* problem in **P** can be sped up to polylog time by parallel processing. Some problems appear to be *inherently sequential*. If we prove that a problem is **P**-complete, we know that it is *not* in **NC** unless $\mathbf{P} = \mathbf{NC}$.

Theorem: CVP, MCVP and HORN-SAT are all **P**-complete.

Proposition 23.2 *Every regular language is in* $\mathbf{NC}^1$.

There are several proofs of this: the basic idea is to use divide-and-conquer to determine the behavior of a DFA on a string. This can be rephrased as *composing together* the DFA behaviors on each letter (using a binary tree of composition operators) or as solving a reachability problem on a constant-width graph by Savitch.

By a very similar argument, you can show that every regular language is in $\mathbf{ATIME}(\log n)$. Actually, with a suitable uniformity condition,

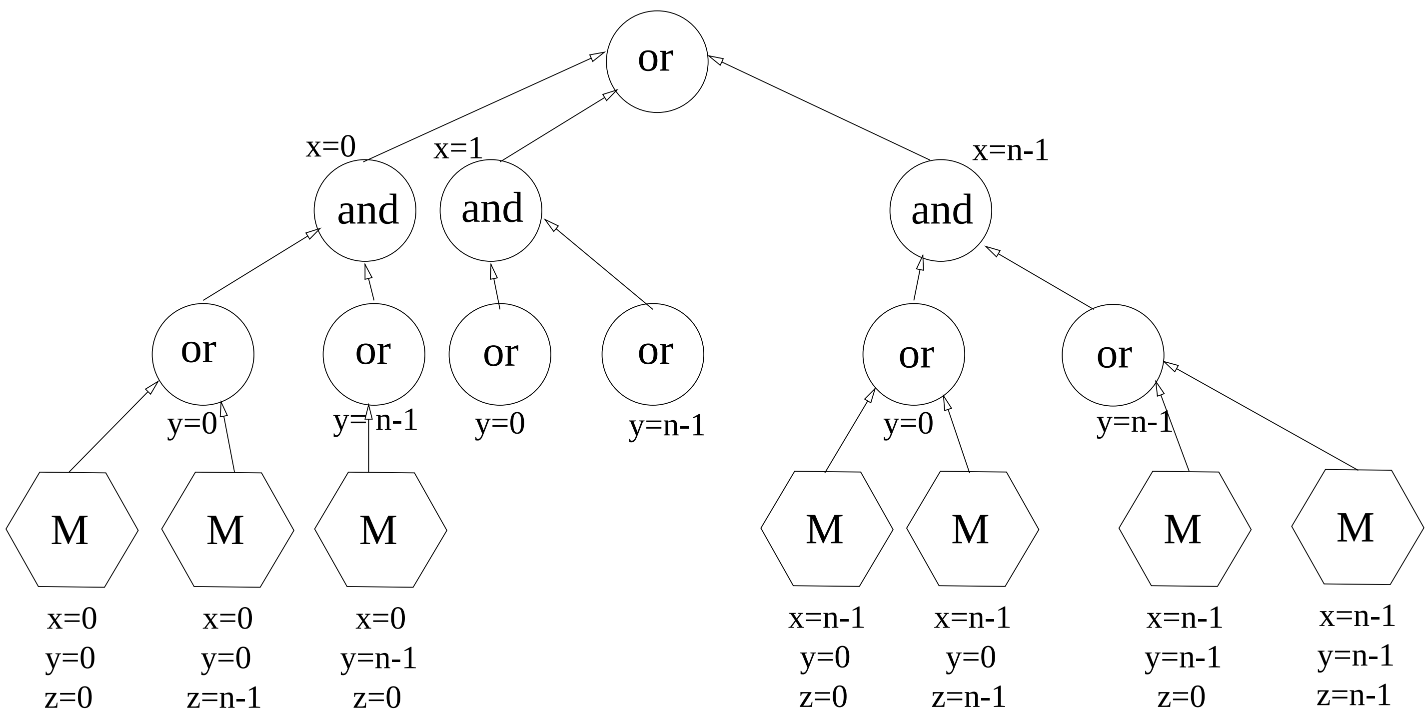
Theorem: (Ruzzo) $\mathbf{NC}^1 = \mathbf{ATIME}(\log n)$.

We'll prove this later in this lecture, though without the uniformity details.

Theorem 23.3 *(Barrington-Immerman-Straubing)* FO $=$ **AC**^{0}

Proof: (Sketch of one direction, with some uniformity details skipped)

$$\varphi \quad = \quad (\exists x)(\forall y)(\exists z)M(x,y,z)$$



♠

Proposition 23.4 *For* $i = 0, 1, \ldots,$

$$\mathbf{NC}^i \quad \subseteq \quad \mathbf{AC}^i \quad \subseteq \quad \mathbf{ThC}^i \quad \subseteq \quad \mathbf{NC}^{i+1}$$

Proof:

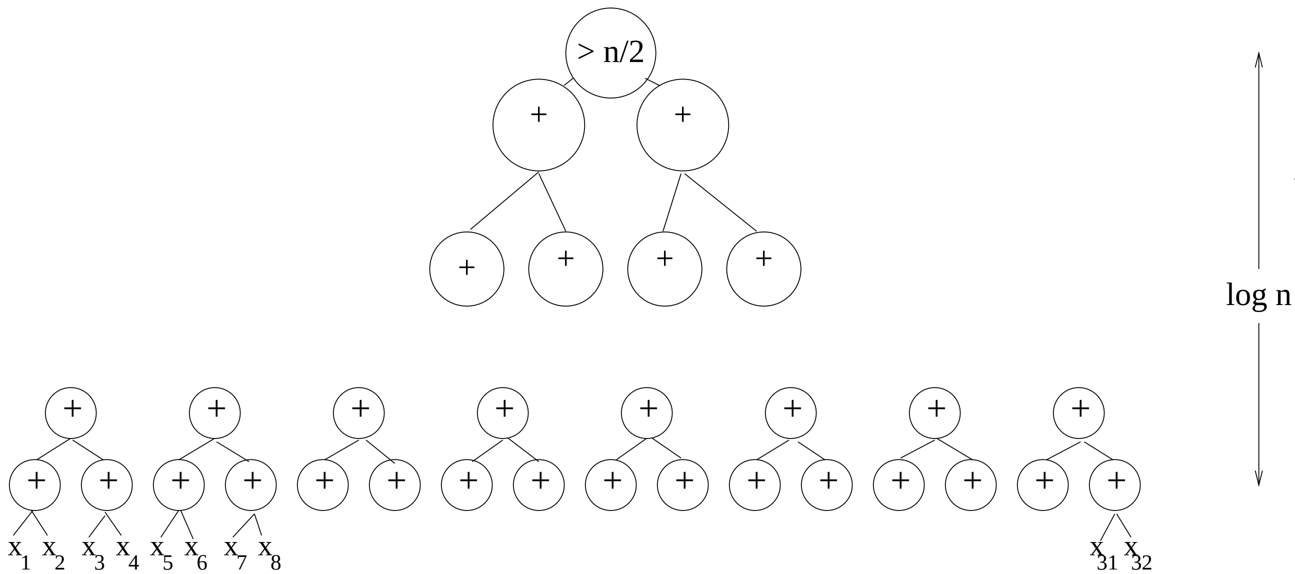
All inclusions but $\mathbf{ThC}^i \subseteq \mathbf{NC}^{i+1}$ are clear.

$\mathrm{MAJ} \ = \ \{w \in \{0,1\}^\star \mid w$ has more than $|w|/2$ "1"s$\} \ \in \mathbf{ThC}^0$

Lemma 23.5 $\quad \mathrm{MAJ} \in \mathbf{NC}^1$

(and the same for any other threshold gate).

The obvious way to try to build an $\mathbf{NC}^1$ circuit for majority is to add the n input bits via a full binary tree of height $\log n$. The problem with this, is that while the sums being added have more and more bits, we must still add them in constant depth.



A solution to this problem is via ambiguous arithmetic notation. Consider a representation of natural numbers in binary, except that digits $0, 1, 2, 3$ may be used. For example 3213 and 3221 are different representations of the decimal number 37 in this ambiguous notation,

$$3213 \;=\; 3 \cdot 2^3 + 2 \cdot 2^2 + 1 \cdot 2^1 + 3 \cdot 2^0 \;=\; 37$$

$$3221 \;=\; 3 \cdot 2^3 + 2 \cdot 2^2 + 2 \cdot 2^1 + 1 \cdot 2^0 \;=\; 37$$

Lemma 23.6 *Adding two n bit numbers in ambiguous notation can be done via an* $\mathbf{NC}^0$ *circuit, i.e., with bounded depth and bounded fan-in.*

Example:

$$
\begin{array}{rrrrrr}
\text{carries:} & 3 & 2 & 2 & 3 & \\
\\
 & & 3 & 2 & 1 & 3 \\
+ & & 3 & 2 & 1 & 3 \\
\hline
 & 3 & 2 & 2 & 1 & 0
\end{array}
$$

This is doable in $\mathbf{NC}^0$ because the carry from column i can be computed by looking only at columns i and $i+1$.

Translating from ambiguous notation back to binary, which must be done only once at the end, is just an addition problem. This is first-order, and thus $\mathbf{AC}^0$, and thus $\mathbf{NC}^1$.

♠

Theorem 23.7 *(Borodin)* $\mathbf{NC}^1 \subseteq \mathbf{L}$

Proof: Use a recursive evaluation of the circuit:

```
boolean eval()
{// a method in the Gate class
   if (type == OR)
      return eval(left) || eval(right);
   if (type == AND)
      return eval(left) && eval(right);
   if (type == INPUT)
      return inputValue;}
```

We must save $O(1)$ bits each time we recurse, and our recursion depth is the depth of the circuit, $O(\log n)$. Thus we use $O(\log n)$ space. ♠

Theorem 23.8 *(Savitch)* **NL** $\subseteq$ **AC**^{1}

Proof: Express the Savitch middle-first search algorithm for REACH as a circuit. For every two nodes u and v and every number d up to $\log n$, have a gate $G(u, v, d)$ that will evaluate to true iff there is a path from u to v of length at most 2^d.

Then $G(u, v, d+1)$ is the OR, over all nodes w, of $G(u, w, d)$ AND $G(w, v, d)$. $G(u, v, 0)$ is the input bit $E(u, v)$ OR'ed with "$u = v$". There are only polynomially many gates and our depth (using unbounded fan-in) is clearly $O(\log n)$. ♠

Note that this circuit uses unbounded fan-in only for OR gates, so it is in a subclass of **AC**^{1} called **sAC**^{1}. By a proof similar to Immerman-Szelepcsenyi, it can be shown that **sAC**^{1} is closed under complement – it can be defined with big-ORs-only or big-ANDs-only.

We know that **ASPACE**$(\log n) =$ **P**, and we have defined subclasses of **P** in terms of circuits with limited depth. It turns out that these *same* subclasses can also be defined in terms of alternating Turing machines. We define subclasses of **ASPACE**$(\log n)$ in terms of limited *time* and limited *number of alternations*.

Theorem 23.9 *(Ruzzo) For all* $i \geq 1$,

- **NC**^{i} *equals the languages of ATM's with space* $O(\log n)$ *and time* $O(\log^i n)$.
- **AC**^{i} *equals the languages of ATM's with space* $O(\log n)$ *and* $O(\log^i n)$ *alternations.*

Proof: (This is a sketch, omitting many uniformity details. For example, the exact uniformity definition used for **NC**^{1} is a messy one designed specifically to make **NC**^{1} equal **ATIME**$(\log n)$.)

First, consider simulating ATM's by circuits. If we make a gate for each of the $n^{O(1)}$ configurations, we can connect these gates into a circuit in an obvious way. The type of the gate for configuration c is AND if c is a Black-move (universal) configuration and OR if it is White-move (existential). A terminal configuration becomes an constant gate. The children of c are the two configurations that the ATM can move to from c. The output gate is the start configuration.

But we have a problem in that the children of a gate depend on the input, and we can't let the *structure* of the circuit depend on the input (only on its size). However, following Problem 5 on HW#7, we can assume that the ATM is a *one-look* machine. (To be complete we would need to prove a lemma that we can enforce the one-look restriction preserving time or preserving alternations.)

What is the depth of the resulting circuit? It is equal to the running time of the ATM, assuming we make the circuit have fan-in two. This shows the $\mathbf{NC}^i$ part of this half of the theorem.

If we take the circuit we have constructed and collapse it to an unbounded fan-in circuit by merging ANDs with ANDs or ORs with ORs on consecutive levels, then our depth is reduced from the running time to the number of alternations. Each phase of all-AND or all-OR gates becomes a single level of the new circuit. There are some details to check to make sure that this construction is sufficiently uniform. But we only care in the case of $\mathbf{AC}^1$ and above, and in $\mathbf{AC}^1$ we can test REACH and thus decide whether one gate can be reached from another by a path of all AND or all OR gates.

This (with some details missing) concludes the simulation of ATM's by circuits.

We now need to show the other half of the theorem, that we can simulate a circuit with an ATM. But we've really already done this, in defining the Circuit Game to solve MCVP in **ASPACE**$(\log n)$.

If the input circuit is fan-in two and depth $O(\log^i n)$ (an **NC**^{i} circuit), the exact same Circuit Game will be completed in $O(\log^i n)$ moves of the game. But can we implement a move of the game in $O(1)$ time on the machine? We can have the players make their choices by writing down a bit for each move. But how do we know whose move it is, and where we are in the circuit? If we wrote down the new gate number each time, we would necessarily take $O(\log^{i+1} n)$ time in all as each gate number has $O(\log n)$ bits.

The trick is to *amortize* the cost of writing down the gate number by doing it only every $\log n$ moves. In between, the players operate by looking at the last gate number recorded and the sequence of moves since then. We allow *challenges* to any claims about whose move it is, or what the new gate number should be, so we need the circuit to be uniform enough that we can decide these challenges.

If the input circuit has unbounded fan-in, the player in the Circuit Game picking a child must write down its entire gate number, and then claim (subject to challenge) that this gate is really a child. Now the moves of the game take time $O(\log n)$ each, but each move is only a single alternation so the number of alternations is bounded by the depth of the circuit.

This completes the proof of the theorem. ♠

We'll conclude the discussion of parallel complexity by showing where another one of our existing classes, the context-free languages, fit into the **NC** hierarchy.

Theorem 23.10 *(Ruzzo) If G is any context-free grammar, $\mathcal{L}(G) \in$* **sAC**^{1}.

Proof: Using the Alternation/Circuit theorem, we'll prove this by designing an ATM game for $\mathcal{L}(G)$ that has the following properties:

- White wins the game on input w iff $w \in \mathcal{L}(G)$,
- the game uses $O(\log n)$ space,
- the number of alternations is $O(\log n)$, and
- all Black's alternation phases consist of a single bit move.

When we covert this game to a circuit, the last clause ensures that all the AND gates have fan-in two, so we are in **sAC**^{1}. (Though our best upper bound for REACH is also **sAC**^{1}, it is believed that REACH is not complete for **sAC**^{1} while there are CFL's that are complete for it.)

Let's assume G is in Chomsky normal form. We have an input string w, and White claims there is a way to derive $S \to w$ using the rules of G. Black, as usual, disputes this.

White advances her claim by naming a node in the middle of the parse tree and saying what it does. Specifically, for some i, j, and A she says $S \to w_1 \dots w_i A w_{j+1} \dots w_n$ and $A \to w_{i+1} \dots w_j$. Black picks one of these two claims to challenge.

If White is telling the truth about the orginal claim, she can get two true claims by telling the truth. But if she is lying, one of her two subsidiary claims must be a lie. We continue the process until we have a claim about a single input letter, such as $A \to w_i$, which can be verified by looking up the input letter and checking the rules of G.

This is a valid ATM game that decides whether $w \in \mathcal{L}(G)$, but it does not yet meet our specification. There are two problems:

- The game could last as long as $n-1$ moves, rather than the $O(\log n)$ we need, and
- The subclaim under dispute might not be specifiable in space $O(\log n)$, as it has the form

  $$A \rightarrow w_{i_1} \ldots w_{i_2} B w_{i_3} \ldots w_{i_4} C w_{i_5} \ldots w_{i_k}.$$

  We need $O(\log n)$ bits to record each "scar" in the string.

We solve the first problem by setting a fair time limit on White. If she has not reduced the claim to one letter in $O(\log n)$ moves, she loses. But why is this fair? On her move, she is dividing the *parse tree* of w into two pieces by cutting an edge.

Lemma: (Lipton-Tarjan) Any binary tree can be cut on some edge into two pieces, each at most 2/3 the original size. (Proof on HW#8.)

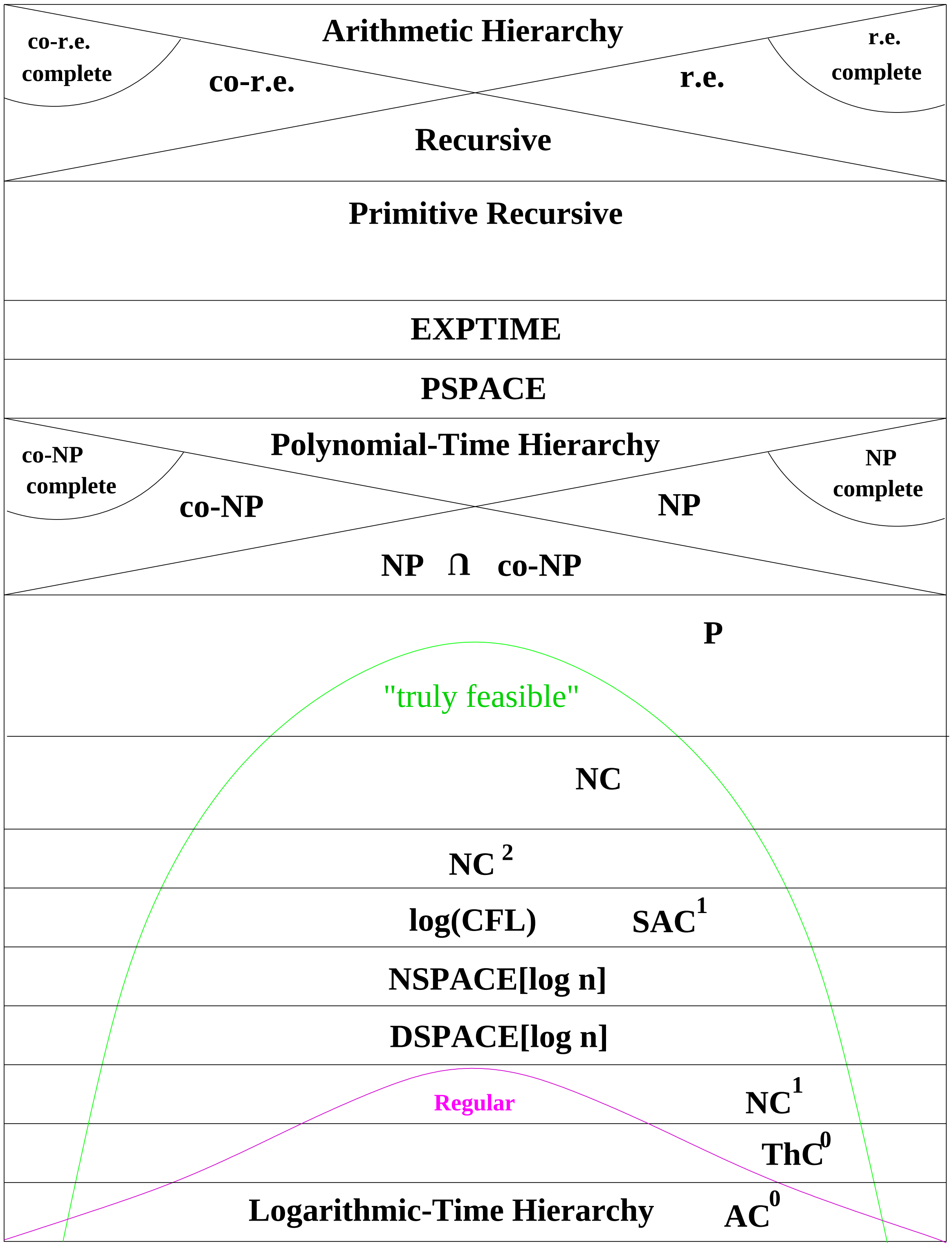
So since White is so smart, she can choose her division to leave smaller subtrees, and after $O(\log n)$ moves she can reduce the subtree to one node.

To solve the second problem, we force White to make sure that the current claim is about a tree with at most three scars, giving her $O(\log n)$ more moves to spend on this goal.

Lemma: Let T be any rooted binary tree and let a, b, and c be any three nodes none of which is an ancestor of another. Then there exists a node d that is an ancestor of exactly two of a, b, and c. (Proof on HW#8.)

Now if White is faced with a tree with scars at a, b, and c, we force her to find some d and divide the tree there. This may not shrink the tree under dispute very much, but it makes sure that on the *next* move, the two subclaims have only two scars each.

White still wins the revised game iff she should, and the revised game now fits all the specifications. ♠


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