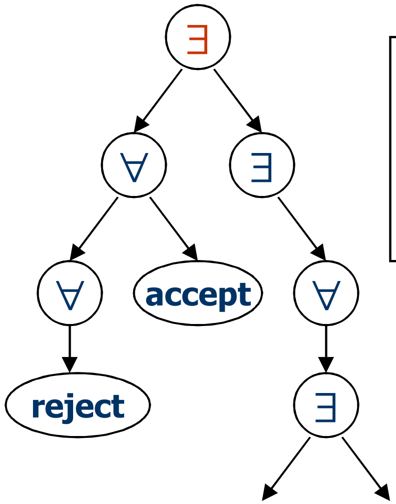
Alternation and Polynomial Time Hierarchy

- **Alternating Turing Machine (ATM)**



An ATM is an NTM with all states, except accept and reject states, divided into universal states and existential states.

  - ∃ node is marked accept iff any of its children is marked accept.
  - ∀ node is marked accept iff all of its children are marked accept.

Definition 2.1. An *alternating Turing machine* (ATM) is a seven-tuple

$$M = (k, Q, \Sigma, \Gamma, \delta, q_0, g),$$

where

k is the number of work tapes,
Q is a finite set of *states*,
Σ is a finite *input alphabet* (¢ $\notin \Sigma$ is an endmarker),
Γ is a finite *work tape alphabet* ($\# \in \Gamma$ is the blank symbol),
$\delta \subseteq (Q \times \Gamma^k \times (\Sigma \cup \{¢\})) \times (Q \times (\Gamma - \{\#\})^k \times \{\text{left, right}\}^{k+1})$ is the *next move relation*,
$q_0 \in Q$ is the *initial state*,
$g : Q \to \{\wedge, \vee, \neg, \textbf{accept}, \textbf{reject}\}$.

If $g(q) = \wedge$ (respectively, $\vee$, $\neg$, **accept**, **reject**), then q is said to be a *universal* (respectively, *existential*, *negating*, *accepting*, *rejecting*) state.

Tautology

- TAUT={<f>|f is a tautology} (in coNP)

  On input <f>:
- Universally select all assignments to the variables of f.
- For a particular assignment evaluate f
- If f evaluates to 1 accept, else reject

MIN-FORMULA={ < ϕ> | ϕ is a minimal Boolean formula, i.e., there is no shorter equivalent}.

(formula is equivalent if they evaluate to same value on assigning the same values to its variables)

MIN-FORMULA$\in$ AP. (not known to be in NP or coNP)

On input ϕ :

1. Universally select all formula f that are shorter than ϕ.
2. Existentially select an assignment to the input variables of ϕ .
3. Evaluate both ϕ and f on this assignment.
4. Accept if the formulas evaluate to different values; else reject.

EXACT-INDSET

- INDSET = {(G, k) : graph G has an independent set of size k} .
- EXACT INDSET = {(G, k) : the largest independent set in G has size exactly k} .

  EXACT INDSET iff *there exists* an independent set of size k in G and *every* other independent set has size at most k.

- It seems that the way to capture such languages is to allow not only an“exists“ quantifier (as in Definition of NP) or only a “for all” quantifier (as Definition of coNP) but a combination of both quantifiers. This

motivates the following definition:

The class $\sum_2^p$ is defined to be the set of all languages L for which there exists a polynomial-time TM M and a polynomial q such that

- x €L , $\exists$ u €{0, 1} $^{q(|x|)}$ $\forall$ v € {0, 1} $^{q(|x|)}$ such that M(x, u, v) = 1 $\forall$ x € {0, 1}*.
- Note that $\sum_2^p$ contains both the classes NP and coNP.

- The language EXACT INDSET above is in since as we noted above, pair (G, k) is in EXACT INDSET iff

$\exists$ S $\forall$ T, set S is an independent set of size k in G and T is not an independent set of size k + 1.

- The class $\prod_2^p$ is defined to be the set of all languages L for which there exists a polynomial-time TM M and a polynomial q such that

x €L , $\forall$ u €{0, 1} $^{q(|x|)}$ $\exists$ v € {0, 1} $^{q(|x|)}$ such that M(x, u, v) = 1 $\forall$ x € {0, 1}*.

- **Polynomial Hierarchy:**

  Let $\Sigma_k T(n)$ be a class of language L accepted by an Alternating Turing Machine that begins in an existential state, alternates between $\exists$ and $\forall$ states $\leq$ k-1 times and halts within O(T(n)) .

  – Define $\Sigma_k P = \sum_{i=1}^{\infty} \Sigma_k n^i$

  The class of complement of languages within $\Sigma_k P$ is called $\Pi_k P = \text{co-}\Sigma_k P$.

  $\Pi_k P$: where ATM begins in a $\forall$ state.

  Note that $\Sigma_k P \subseteq \Pi_{k+1} P$ and $\Pi_k P \subseteq \Sigma_{k+1} P$,

  $\Sigma_1 P = NP$ and $\Pi_1 P = \text{co-}NP$.

  MIN-CIRCUIT is in $\Pi_2 P$.

– Def: The polynomial hierarchy

$$\mathrm{PH} = {}_{k} \sum\nolimits_{k} \mathrm{P} = {}_{k} \Pi_{k} \mathrm{P} \; .$$

- **Alternating Time and Space :**

  ATIME(f(n))={ L : ∃ an ATM that decides L in O(f(n)) time}

  ASPACE(f(n)) is defined similarly.

  AL= ASPACE(log n), AP = $\bigcup_k$ ATIME(n^k) .

- **Thm:** ATIME(f(n)) ⊆ SPACE(f(n)) ⊆ ATIME(f^2(n)) .

  Thus, AP = PSPACE .

- **Thm:** ASPACE(f(n)) = TIME($2^{O(f(n))}$) .

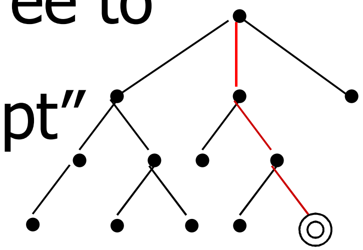
  Thus, ASPACE = EXP , AL=P

Lemma: For f(n) >= n we have ATIME(f(n)) $\subseteq$ SPACE(f(n)).

Proof:

Convert an O(f(n))-time ATM M to a O(f(n))-space DTM S.

S makes a depth-first search of M's computation tree to determine whether the start configuration is "accept" or not. It is done by recursion and the depth is O(f(n)), since the computation tree has depth O(f(n)).



For each level of recursion, the stack store the non-det choice that M made to reach that configuration from its parent and this uses only constant space. S can recover the configuration by "replaying" the computation from the start and following the recorded "signposts." $\square$

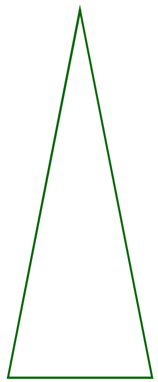
Lemma: Let $f(n) >= n$ we have $SPACE(f(n)) \subseteq ATIME(f^2(n))$.

Proof:

- Let M be an $O(f(n))$-space DTM. We want to construct an $O(f^2(n))$-time ATM S to simulate M.
- It is very similar to the proof of Savitch's Theorem.

  $\phi_{c1,c2,t} = \exists m\, [\phi_{c1,m,\,t/2} \wedge \phi_{m,c2,t/2}]$: indicate if C2 is reachable from C1 in t steps for M.

- S uses the above recursive alternating procedure to test whether the start configuration can reach an accepting conf. within $2^{df(n)}$ steps. The recursive depth is $O(f(n))$.



- For each level it takes $O(f(n))$ time to write a conf. Thus the algorithm runs in $O(f^2(n))$ alternating time.

Lemma: $f(n) >= \log n$ we have ASPACE$(f(n)) \subseteq$ TIME $(2^{O(f(n))})$.

Proof:

•Construct a $2^{O(f(n))}$ -time DTM S to simulate an $O(f(n))$-space ATM M. On input w, S constructs the following graphs of the computation M on w.

•Nodes are conf of M on w and edges go from a conf to those configurations that it can yield in a step.

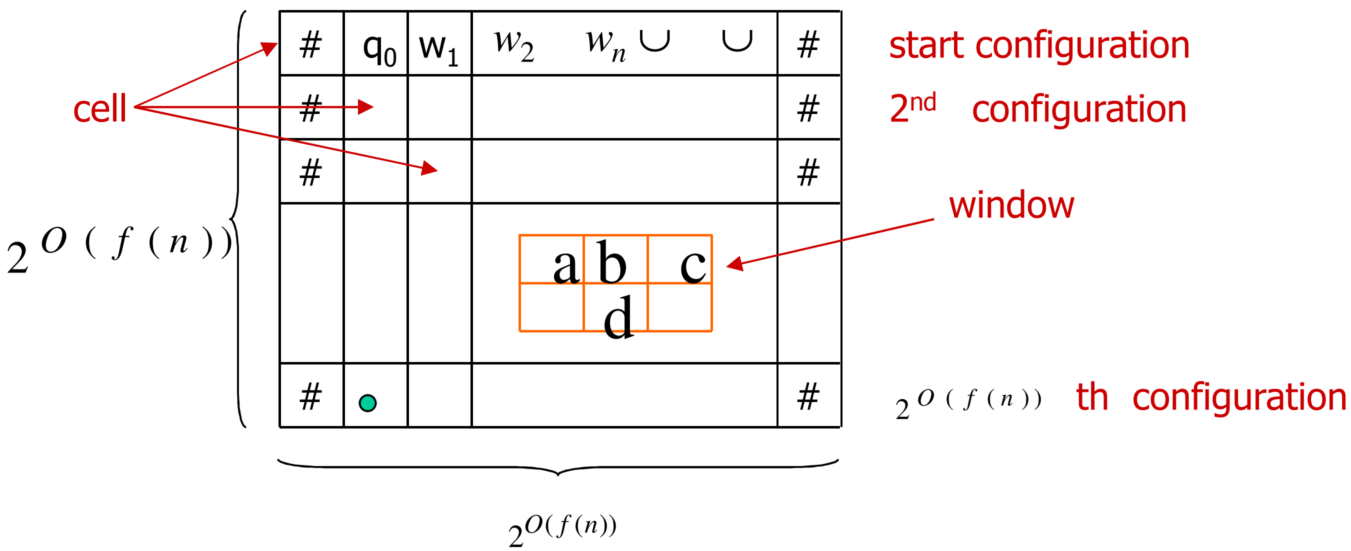
•After the graph is constructed, S repeatedly scans it and marks certain conf as accepting. Initially, only actual accepting conf are marked "accepting". A conf that performs universal branching is marked "accepting" if all its children are marked and an existential conf is marked if any of its children are marked.

•S continues until no additional nodes are marked in a scan. Finally, S checks if the start conf is marked. There are $2^{O(f(n))}$ conf of M on w, which is also the size of the graph. Hence the total time used is $2^{O(f(n))}$.

Lemma: f(n) >= log n we have TIME ($2^{O(f(n))}$) $\subseteq$ ASPACE(f(n))

Proof:

Construct an O(f(n))-space ATM S to simulate a $2^{O(f(n))}$ -time DTM M. On input w, S has only enough space to store pointers into a tableau of the computation M on w as depicted in the following.



The content of d is determined by the contents of its parents a, b, and c.

S operates recursively to guess and then verify the contents of the individual cells of the tableau. It is easy to verify the cells of the first row, since it knows the start conf of M on w.

For other cell d, S existentially guesses the contents of the parents, checks whether their contents would yield d's contents according to M's transition, and then universally branches to verify these guesses recursively.

Assume M moves its head to the leftmost cell after entering accepting state. Thus S can determine whether M accepts w by checking the contents of the lower leftmost cell of the tableau. Hence S never needs to store more than a pointer to a cell in the tableau. So it uses at most O(f(n)) space.