Def. Dada una M.T. M que siempre termina su complejidad espacial es una función $f: \mathbb{N} \to \mathbb{N}$ definida por:

$$f(n) = \text{máx} \{ \text{\# celdas de la cinta que la M.T. recorre si se la alimenta con } w \mid w \in \Sigma^*, |w| = n \}.$$

Dada una M.T.N. N que siempre termina su complejidad espacial es una función $f: \mathbb{N} \to \mathbb{N}$ definida por:

$$f(n) = \text{máx} \{ \text{\# celdas de la cinta que recorre } N(w) \text{ en cualquiera de sus ramas} \mid w \in \Sigma^*, |w| = n \}$$

Def. Dada una función $f: \mathbb{N} \to \mathbb{R}^+$ definimos:

$TIME(n^2)$

$NTIME(n^2)$

$$SPACE(f(n)) = \{ A \subseteq \Sigma^* \mid A \text{ se puede decidir con una M.T. usando espacio } O(f(n)) \}$$

$$NSPACE(f(n)) = \{ A \in \Sigma^* \mid A \text{ se puede decidir con una M.T.N. usando espacio } O(f(n)) \}.$$

Recordemos:

$$SAT = \{ \langle \varphi \rangle \mid \varphi \text{ es satisfactible} \}$$

Teorema. SAT se puede decidir en espacio lineal, es decir, $SAT \in SPACE(n)$.

Dem.

Si la fórmula φ tiene ℓ variables $x_1, x_2, \ldots, x_\ell$. $\quad \ell \leq n$

x_1	x_2	x_3	...	$x_{\ell-1}$	x_ℓ
0	0	0	...	0	0
0	0	0	...	0	1
0	0	0	...	1	0
0	0	0	...	1	1
⋮					

$O(\ell) \subseteq O(n)$
ESPACIO

2^ℓ

Esta idea decide SAT (en tiempo exponencial) y espacio lineal.

Recordemos:

$$ALL_{AFN} = \{ \langle N \rangle \mid N \text{ es un autómata finito no determinístico}, \mathcal{L}(N) = \Sigma^* \}.$$

Teorema. $\overline{ALL_{AFN}}$ se puede decidir usando espacio lineal no determinísticamente, es decir, $\overline{ALL_{AFN}} \in NSPACE(n)$.

Dem. Dado un AFN N cuyo cjto. de estados es $\{q_1, q_2, \ldots, q_n\}$, vamos a construir una tabla:

q_1	q_2		...	q_{n-1}	q_n
1	0			0	0
0	1	1	0 ...	0	0
⋮					

Ocupa $O(n)$ espacio.

- No determinísticamente, elegimos un símbolo del alfabeto de entrada y actualizamos la tabla para mantener un registro del conjunto de estados en el que nos podríamos encontrar.

- Repetimos el proceso a lo sumo 2^n veces.

- Si pasamos por algún conjunto de estados que no incluye ningún estado final, quiere decir que existe una $w \in \Sigma^*$ tal que $N(w) = \text{rechaza}$. Por lo tanto, $N \in \overline{ALL_{AFN}}$, y aceptamos.

- Si al cabo de 2^n pasos nunca llegamos a un sitio de estados que no incluya un estado final, quiere decir que No existe una $w \in \Sigma^*$ tal que $N(w) = \text{rechaza}$. Por lo tanto $N \notin \overline{ALL_{AFN}}$, y lo rechazamos.

Obs. $A \in SPACE(f(n)) \Rightarrow A \in NSPACE(f(n))$

$\Leftarrow$?

Teorema. Si $f : \mathbb{N} \to \mathbb{R}^+$ tq. $f(n) \geq n$,
entonces

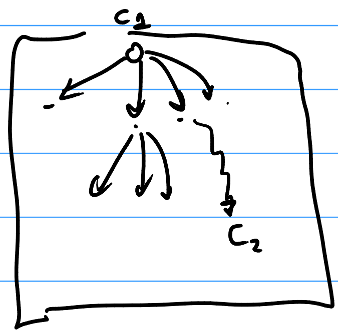
$$NSPACE(f(n)) \subseteq SPACE(f(n)^2).$$

Dem.

- Sea N una M.T.N. que usa espacio $f(n)$, y veamos que podemos construir una M.T.D. M que usa espacio $O(f(n)^2)$ y que decide el mismo lenguaje que N.

- Vamos a definir un procedimiento auxiliar
  - $\text{ALCANZA}(\underbrace{c_1, c_2}_{\text{Configuraciones de la máquina } N}, \underbrace{n}_{\text{número},\ n \in \mathbb{N}})$

- Este procedimiento determina si desde la configuración c_1 se puede llegar hasta la configuración c_2 en la máquina N, en a lo sumo n pasos.



- Por simplicidad, vamos a suponer que n es una potencia de 2:

$\underline{\text{ALCANZA}(c_1, c_2, t):}$

```
if t = 1 {
    si c1 = c2, ó de c1 se alcanza c2 en un
    paso, devolver True; si no, devolver False.
} else {
    foreach c' configuración de N
              tq. |c'| ≤ f(n) {
        if ALCANZA(c1, c', t/2) && ALCANZA(c', c2, t/2) {
            return True
        }
    }
```

$c_1 \xrightarrow{n/2} c' \xrightarrow{n/2} c_2$

} return False
}

- ALCANZA es una función recursiva.
- Cada vez que se anida un llamado recursivo dentro de otro, se usa $O(f(n))$ memoria.

- Para que la máquina M simule la máquina N, lo primero que vamos a hacer es modificar la máquina N para que cuando N llegue a una configuración de aceptación:
  - borre todo el contenido de la cinta
  - pase a un estado q_{accept}.

- Además, consideramos un número d tal que N tenga a lo sumo $2^{d \cdot f(n)}$ configuraciones de tamaño $f(n)$.

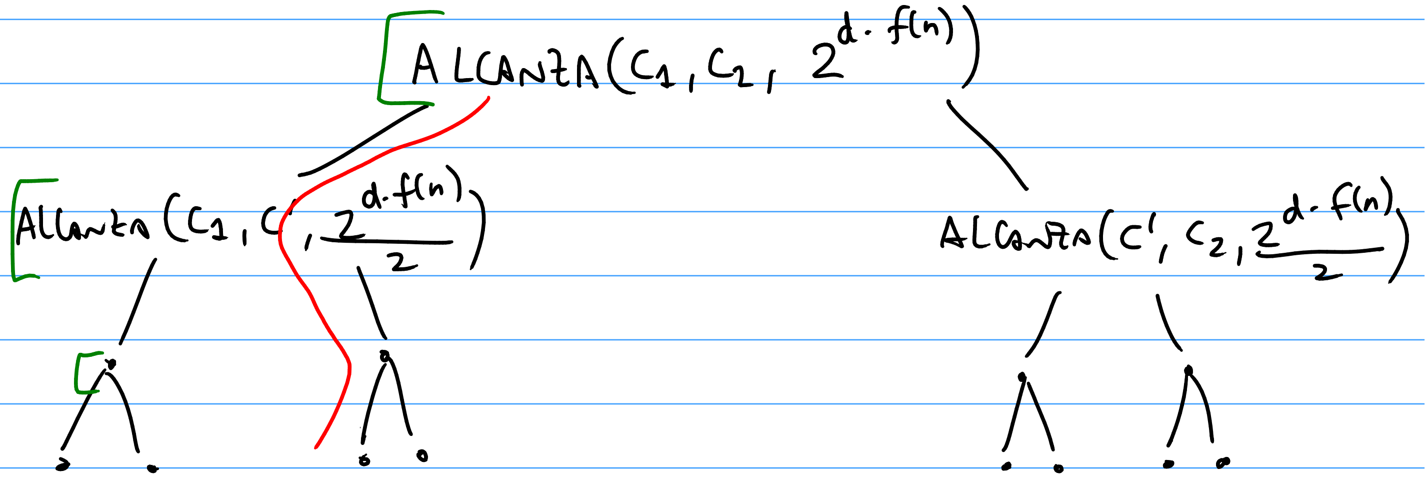
- Observemos que el tiempo de ejecución de N es a lo sumo $2^{d \cdot f(n)}$.

- Para que M simule $N(w)$:
  - Consideramos la configuración $c_1 := q_0 w$. (estado inicial de N)
  - Consideremos la configuración $c_2 := q_{accept}$.
  - M verifica si $ALCANZA(c_1, c_2, 2^{d \cdot f(n)})$. En tal caso, acepta. Si no, rechaza.

- Es fácil ver que $\begin{cases} M(w) = \text{acepta} \iff N(w) = \text{acepta} \\ M(w) = \text{rechaza} \iff N(w) = \text{rechaza} \end{cases}$

  Con lo cual $\mathcal{L}(M) = \mathcal{L}(N)$.

- Lo que resta ver es que M usa espacio $O(f(n)^2)$.

- 

- Cada nodo del árbol requiere $O(f(n))$ memoria.
- La altura del árbol es $O\left(\log_2\left(2^{d\cdot f(n)}\right)\right)$

$$= O(d\cdot f(n)) = O(f(n)).$$

- El uso total de espacio en una rama es $O(f(n)^2)$
- Por lo tanto, $M \in SPACE(f(n)^2)$.

Def. $\mathrm{PSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{SPACE}(n^k)$

Def. $\mathrm{NPSPACE} = \bigcup_{k \in \mathbb{N}} \mathrm{NSPACE}(n^k)$

Teorema. $\mathrm{PSPACE} = \mathrm{NPSPACE}$.

Dem. $(\subseteq)$ Si $A \in \mathrm{PSPACE}$, también $A \in \mathrm{NPSPACE}$.

$(\supseteq)$ Sea $A \in \mathrm{NPSPACE}$.
Por lo tanto, $A \in \mathrm{NSPACE}(n^k)$ para algún $k \in \mathbb{N}$.
Por el teorema de Savitch

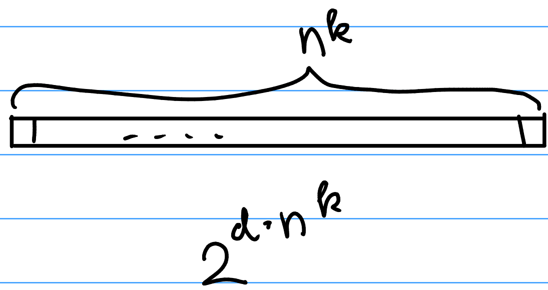
$$A \in \mathrm{SPACE}((n^k)^2) = \mathrm{SPACE}(n^{2k}).$$

Por lo tanto

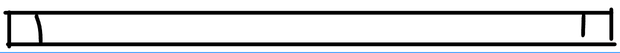
$$A \in \mathrm{PSPACE}.$$

Def. $\mathrm{EXPTIME} = \bigcup_{k \in \mathbb{N}} \mathrm{TIME}(2^{n^k})$

Teorema. $\mathrm{PSPACE} \subseteq \mathrm{EXPTIME}$.



Obs. $P \subseteq \mathrm{PSPACE}$



En n^k pasos, una M.T. puede usar a lo sumo n^k celdas de la cinta.

Obs. $NP \subseteq \mathrm{NPSPACE} = \mathrm{PSPACE}$

$$P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$$

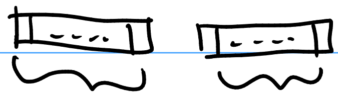
<u>Def</u>.

$$L = SPACE(\log n)$$

$$NL = NSPACE(\log n).$$

· <u>Ej</u>. El lenguaje $A = \{ a^n b^n \mid n \in \mathbb{N} \}$ está en la clase L.

$a \dots a\, b \dots b$



$0 \dots n \qquad 0 \dots n$

$\log_2(n)$

· <u>Ej</u>. El lenguaje $PATH = \{ \langle G, v, w \rangle \mid G$ es un grafo dirigido tq. existe un camino de v a $w \}$

está en la clase NL.

Si el grafo tiene n vértices,
puedo guardar un puntero a un vértice,
el puntero ocupa $O(\log_2 n)$ bits.