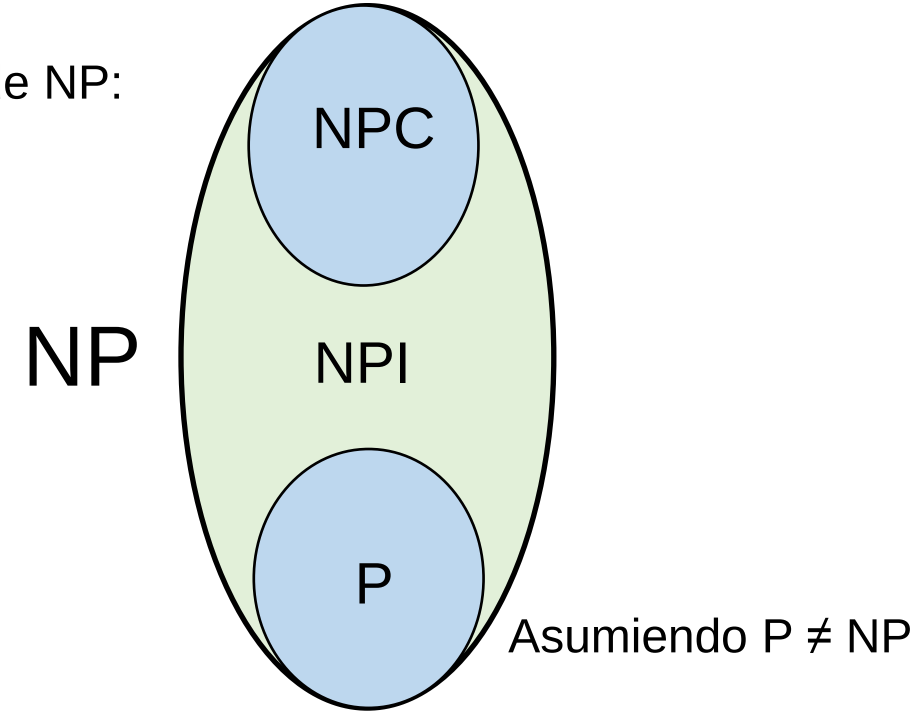
Clase teórica 7

Jerarquía espacio-temporal

La clase NPI

- Una visión aún más detallada de NP:

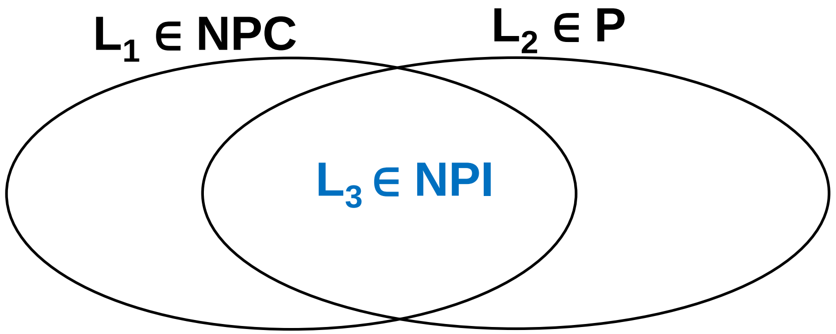


- Asumiendo P ≠ NP, se prueba que además de P y NPC, NP incluye una tercera clase de lenguajes: **NPI.**

- Los lenguajes de NPI **no son ni tan fáciles como los de P ni tan difíciles como los de NPC.**

- La existencia de NPI, asumiendo $P \neq NP$, se basa en el **Teorema de Ladner**, de 1975.

- Idea general: si $\mathbf{L_1 \in NPC}$, entonces existe un lenguaje $\mathbf{L_2 \in P}$, tal que $\mathbf{L_3 = L_1 \cap L_2 \in NP - (P \cup NPC)}$.



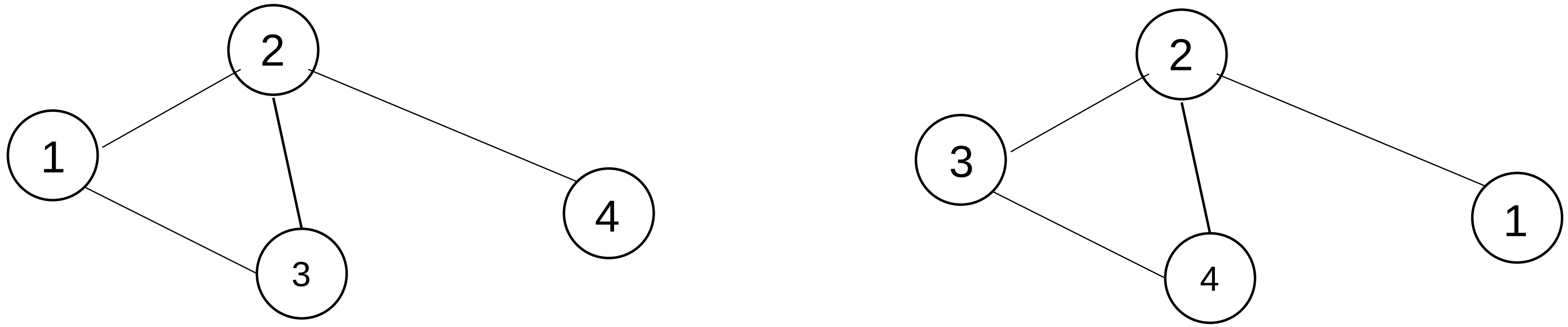
- Por ejemplo: dado SAT, existe un lenguaje L de fórmulas booleanas en P, tal que SAT restringido a las fórmulas de L es un lenguaje de NPI (no es NP-completo ni está en P).

- A diferencia de lo que sucede en NPC, en NPI podría haber **lenguajes dispersos**.

- También podría haber lenguajes L_1 y L_2 **incomparables**, es decir que **no cumplan** $\mathbf{L_1 \leq_P L_2}$ **ni** $\mathbf{L_2 \leq_P L_1}$**.**

El problema de los grafos isomorfos (candidato a estar en NPI)

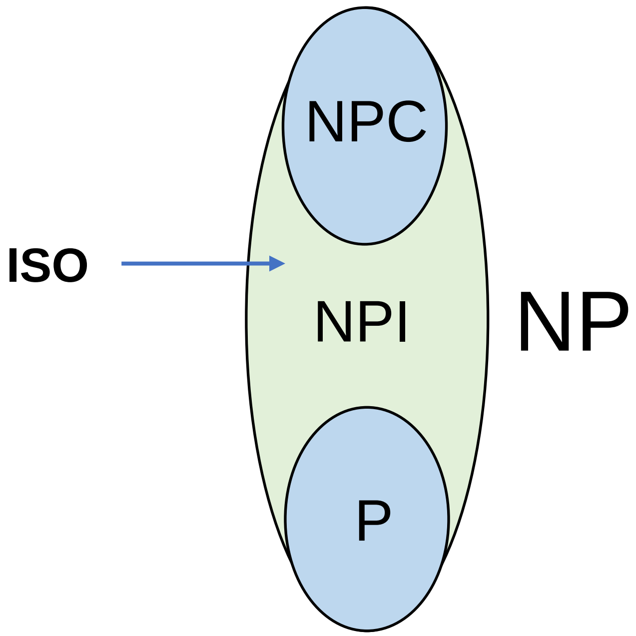
ISO = {(G_1, G_2) | G_1 y G_2 son grafos isomorfos, es decir son idénticos salvo por el nombre de sus arcos}



ISO está en NP
los certificados suscintos son permutaciones de V (ejercicio)

ISO no estaría en P
en el peor caso hay que probar con todas las permutaciones de V (ejercicio)

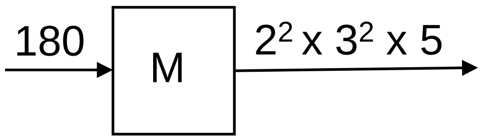
ISO no estaría en NPC
no se ha encontrado un lenguaje NP-completo L tal que L $\leq_P$ ISO



Asumiendo P ≠ NP

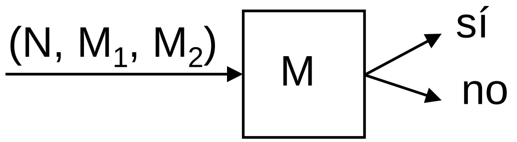
El problema de la factorización (otro candidato a estar en NPI)

Dado N, hay que encontrar todos sus divisores primos. P.ej, **$180 = 2^2 \times 3^2 \times 5$.**



Llevado a un problema de decisión tiene la siguiente forma:

$$\mathbf{FACT = \{(N, M_1, M_2) \mid N \text{ tiene un divisor primo entre } M_1 \text{ y } M_2\}}$$



- Como la sospecha es que no está en P, se lo usa para **encriptar mensajes**:
  Dado un número N muy grande,
  si $N = N_1 \times N_2$, y N_1 y N_2 son números primos de tamaño similar,
  resulta muy difícil obtener N_1 y N_2 conociendo solamente N.

mensaje encriptado (sistema RSA)

1001111010100001110101110001111100101010

se encripta con N — clave pública

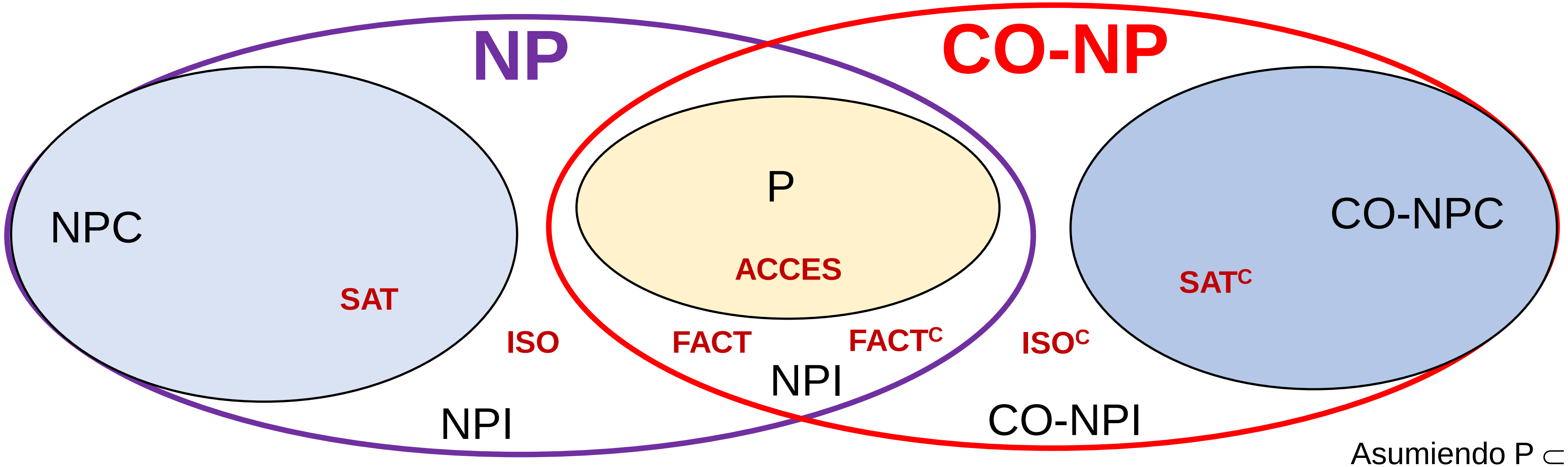
se desencripta con N_1 y N_2 — clave privada

- En base a esto, un esquema de seguridad habitual consiste en **encriptar mensajes con N** (que conoce todo el mundo),
  y **desencriptarlos con N_1 y N_2** (qué sólo conoce el receptor)

- Si se probase que FACT está en P, habría que reemplazar dicho esquema de seguridad.

- En 1994, P. Shor encontró un **algoritmo cuántico** que factoriza los números en tiempo **poly(n).**
  ¿Será que efectivamente lo cuántico acelera exponencialmente los algoritmos clásicos?
  ¿O Será que la factorización está en P?

Ultima visión de la jerarquía temporal



Asumiendo P ⊂ NP
Asumiendo P ⊂ NP ∩ CO-NP
Asumiendo NP ≠ CO-NP

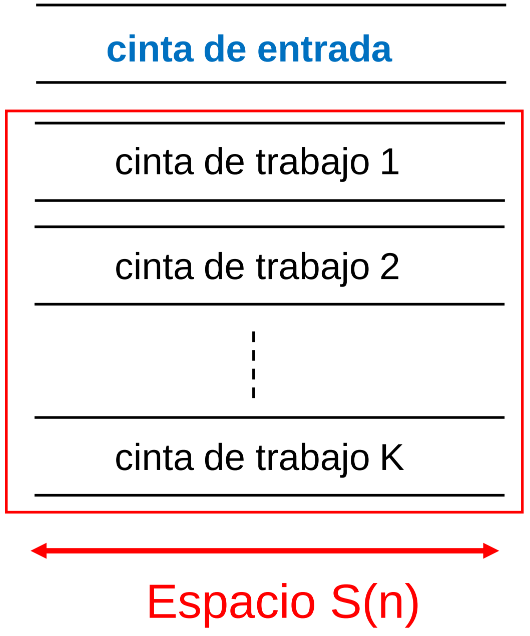
Jerarquía de lenguajes (de menor a mayor dificultad)

1) P
2) (NP ∩ CO-NP) – P
3) NPI - CO-NP
4) NPC
5) CO-NPI – NP
6) CO-NPC

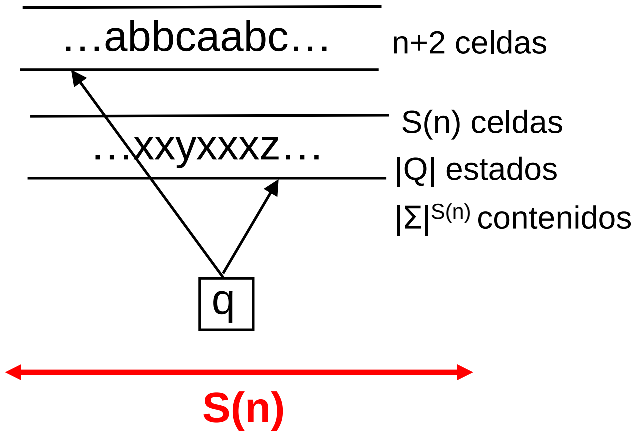
ISO sería más fácil que SAT

FACT sería más fácil que ISO

Introducción a la complejidad espacial



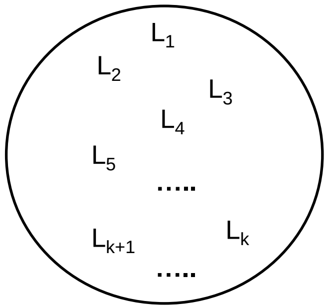
- Se consideran MT tales que la cinta de entrada es de **sólo lectura**. El resto son **cintas de trabajo**.
- Una MT ocupa **espacio S(n)** sii en todas sus cintas de trabajo (no cuenta la cinta de entrada) ocupa **a lo sumo S(n) celdas**, siendo n como siempre el tamaño de las cadenas de entrada.
- La cinta de entrada de sólo lectura permite espacios **menores que O(n).**



- Una MT M que ocupa espacio S(n) **puede no parar**, pero dada M, existe una MT M´ equivalente que ocupa espacio S(n) y **para siempre.**

  P.ej., una MT con 1 cinta de entrada de sólo lectura y 1 cinta de trabajo entra en loop luego de: **$(n+2).S(n).|Q|.|\Sigma|^{S(n)} = O(c^{S(n)})$ pasos.**
  Tener en cuenta entonces que: **espacio S(n) implica tiempo $O(c^{S(n)})$.**

- **Un lenguaje L pertenece a la clase SPACE(S(n))** sii existe una MT que decide L en espacio O(S(n)).



SPACE(S(n))

Los L_i se deciden en espacio O(S(n))

Ejemplo. El lenguaje de los palíndromos o "capicúas".

L = {wcw^R | w tiene cero o más símbolos *a* y b, y w^R es la cadena inversa de w}. P.ej., abbbcbbba está en L.

L se puede decidir en espacio **O(n)** (ejercicio). Pero en realidad alcanza con espacio **O(log$_2$n).** Una MT M que decide L en espacio O(log$_2$n), usando codificación binaria, se comporta de la siguiente manera:

1. Hacer i = 1 en la cinta 1.
2. Hacer j = n en la cinta 2, con n = |w|. Si j es par, rechazar.
3. Copiar el símbolo i de w en la cinta 3.
4. Copiar el símbolo j de w en la cinta 4.
5. Si i = j : si los símbolos son **c,** aceptar, si no, rechazar.
   Si i ≠ j : si los símbolos no son igualmente ***a*** o **b**, rechazar.
6. Hacer i = i + 1 en la cinta 1.
7. Hacer j = j – 1 en la cinta 2.
8. Volver al paso 3.

Ejemplo (1ra iteración)

	a b b b c b b b a
1	i = 1
2	j = 9
3	a
4	a

Ejemplo (2da iteración)

	a b b b c b b b a
1	i = 2
2	j = 8
3	b
4	b

Ejemplo (última iteración)

	a b b b c b b b a
1	i = 5
2	j = 5
3	c
4	c

La MT M ocupa el espacio de los contadores i y j, que en binario miden **O(log$_2$n)**, **más 2 celdas** para alojar a los símbolos que se van comparando en cada iteración.

Así, **L ∈ SPACE(log$_2$n).** A esta clase también se la llama **LOGSPACE.**

Jerarquía espacial

- **LOGSPACE** es la clase de los lenguajes aceptados en espacio **$O(\log_2 n)$**
- **PSPACE** es la clase de los lenguajes aceptados en espacio **poly(n)**
- **EXPSPACE** es la clase de los lenguajes aceptados en espacio **exp(n)**

- Por lo dicho antes: **espacio S(n) implica tiempo $O(c^{S(n)})$, con c constante**

  Así, si en particular una MT M ocupa espacio **$\log_2 n$**, entonces M tarda tiempo **$O(c^{\log_2 n})$**

  Y como $c^{\log_2 n} = n^{\log_2 c}$ = poly(n), queda: **LOGSPACE $\subseteq$ P**

Los problemas tratables en espacio son los que pertenecen a la clase LOGSPACE

Existe una clase NLOGSPACE (homóloga a la clase NP), también incluida en P.

Tiempo T(n) implica espacio T(n). ¿Por quié?
Espacio S(n) implica tiempo $O(c^{S(n)})$, para alguna constante c.

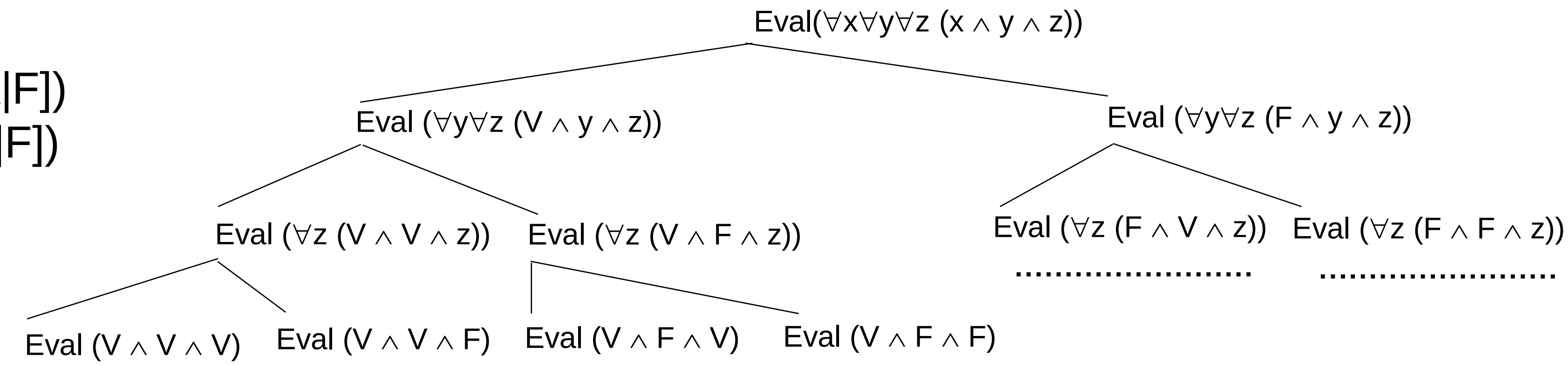
Ejemplo. El lenguaje QSAT.

QSAT = {φ | φ es una fórmula booleana con cuantificadores, no tiene variables libres, y es verdadera}.

- Por ejemplo, $\varphi_1 = \exists x \exists y \exists z \colon x \wedge y \wedge z$, es verdadera

  $\varphi_2 = \forall x \forall y \forall z \colon x \wedge y \wedge z$, es falsa

- **QSAT pertenece a PSPACE.** La prueba se basa en la construcción de una función recursiva **Eval**. La idea de la recursión es la siguiente (usamos como ejemplo la fórmula φ_2):

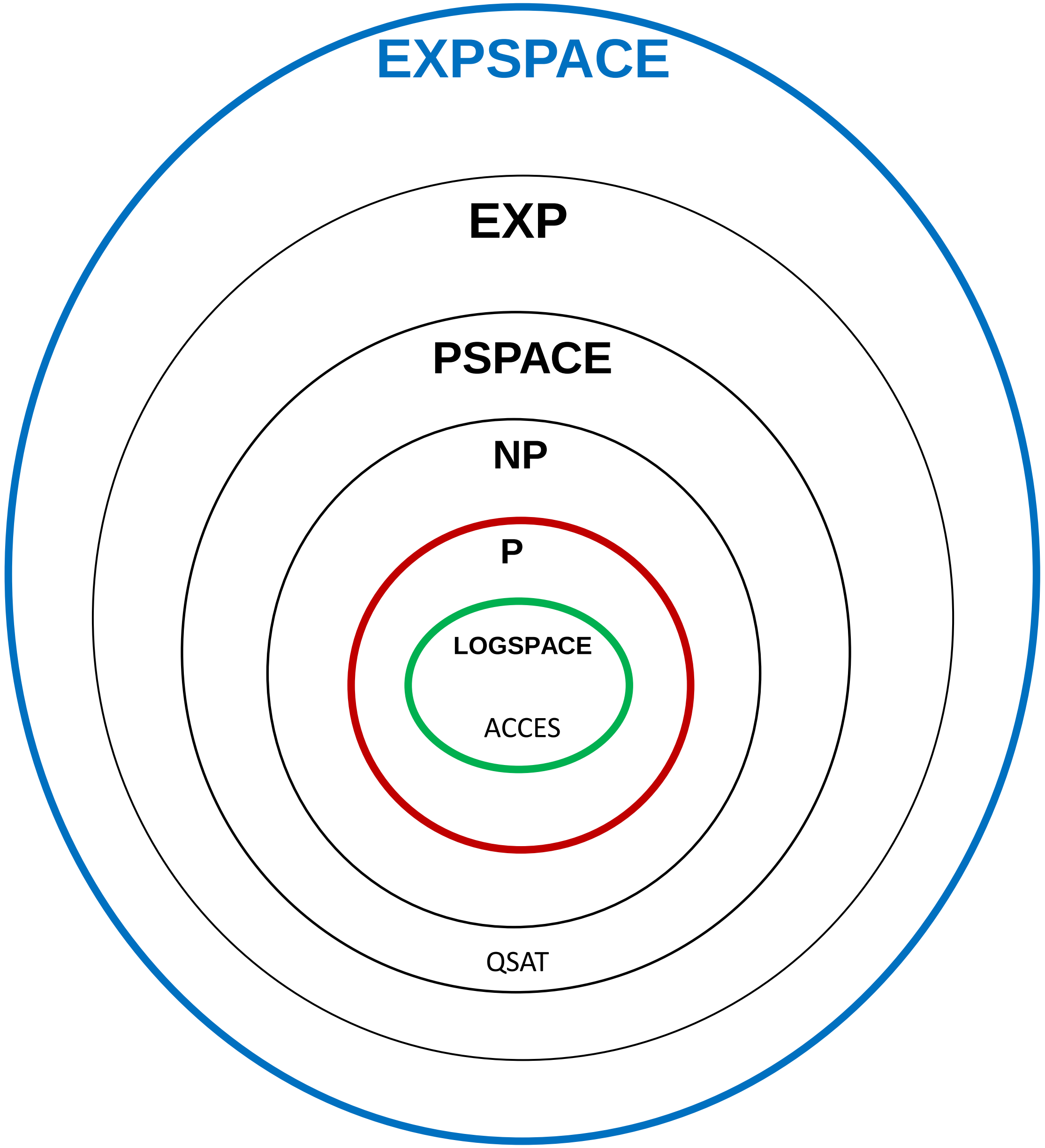


$\text{Eval}(\varphi,\forall x) = \text{Eval}(\varphi[x|V]) \wedge \text{Eval}(\varphi[x|F])$
$\text{Eval}(\varphi,\exists x) = \text{Eval}(\varphi[x|V]) \vee \text{Eval}(\varphi[x|F])$
$\text{Eval}(\varphi_1,\varphi_2,\wedge) = \text{Eval}(\varphi_1) \wedge \text{Eval}(\varphi_2)$
$\text{Eval}(\varphi_1,\varphi_2,\vee) = \text{Eval}(\varphi_1) \vee \text{Eval}(\varphi_2)$
$\text{Eval}(\varphi,\neg) = \neg\text{Eval}(\varphi)$
$\text{Eval}(V) = V$
$\text{Eval}(F) = F$

La MT correspondiente <u>reutiliza</u> espacio, característica esencial en la complejidad espacial

La cantidad de cuantificadores más la cantidad de conectivos de φ es a lo sumo |φ| = n.
Así, tanto la profundidad de la pila como el espacio ocupado en cada instancia miden O(n).
Por lo tanto, **Eval consume espacio $O(n^2)$.** ¿QSAT pertenece a P? ¿Pertenece a NP? ¿Pertenece a CO-NP?

Jerarquía espacio-temporal



- QSAT está entre los lenguajes más difíciles de PSPACE.

- En efecto, **QSAT es PSPACE-completo**, todos los lenguajes de PSPACE se reducen polinomialmente a QSAT.

- QSAT es una instancia de un problema más general: **la búsqueda de una estrategia ganadora en una competencia entre dos jugadores J_1 y J_2 (ajedrez, damas, go, hexágono, geografía, etc.):**

¿Existe una jugada 1 de J_1 tal que para toda jugada 1 de J_2 existe una jugada 2 de J_1 tal que para toda jugada 2 de J_2 existe una jugada 3 de J_1 tal que para toda jugada 3 de J_2 … el jugador J_1 gana?

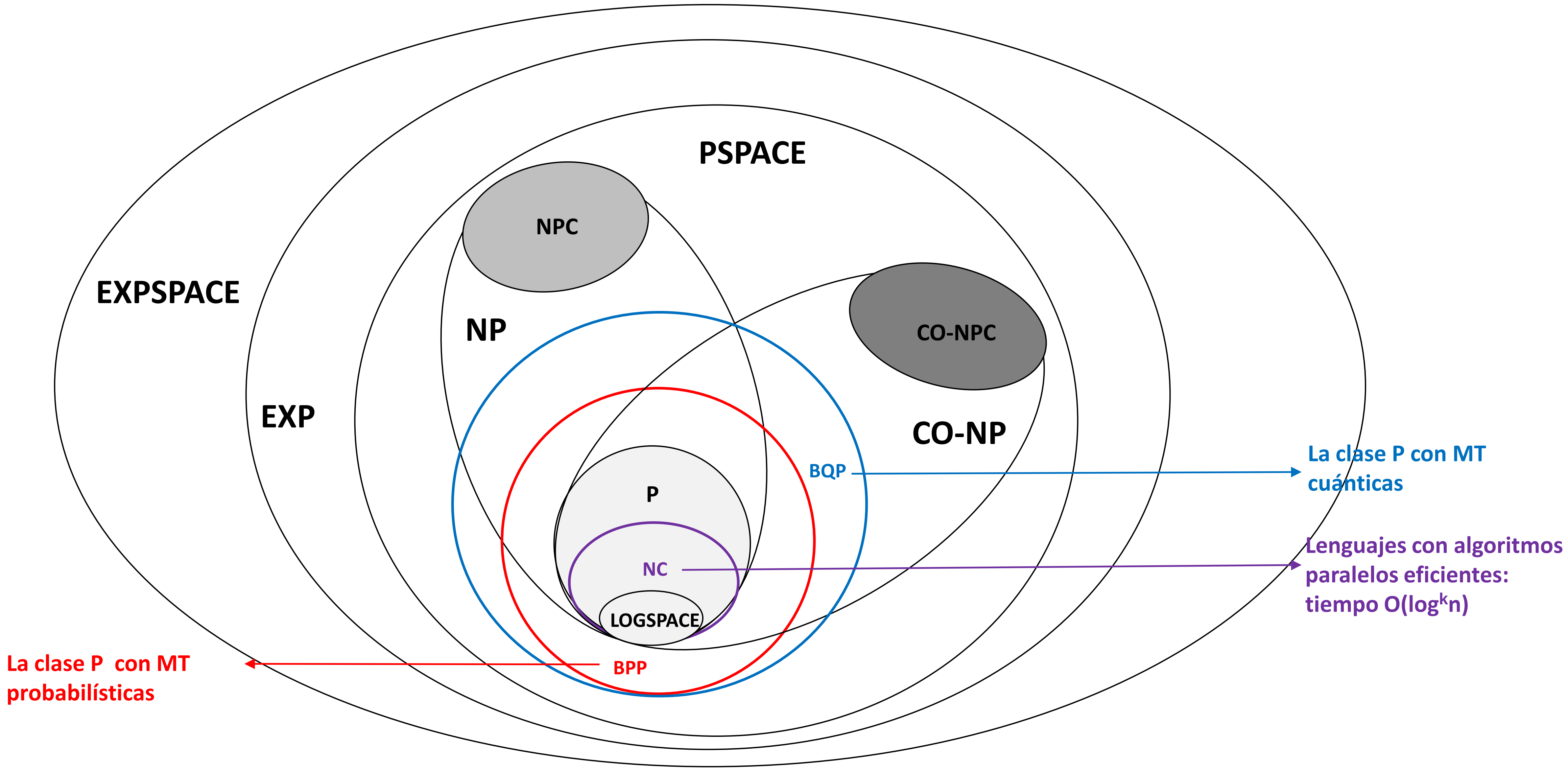
$$\varphi = [\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 \forall x_6 \ldots : \Psi(x_1, x_2, x_3, x_4, x_5, x_6, \ldots)]$$

En el caso particular de QSAT se puede plantear que $\exists$ busca que φ sea verdadera y $\forall$ que φ sea falsa.

Anexo de la clase teórica 7

Jerarquía espacio-temporal

Jerarquías temporales con varios modelos de ejecución



Clase práctica 7

Jerarquía espacio-temporal

Una forma alternativa para probar pertenencia a la clase NP

- Una manera alternativa de probar que un lenguaje L está en NP es utilizando **MTN (MT no determinísticas).**

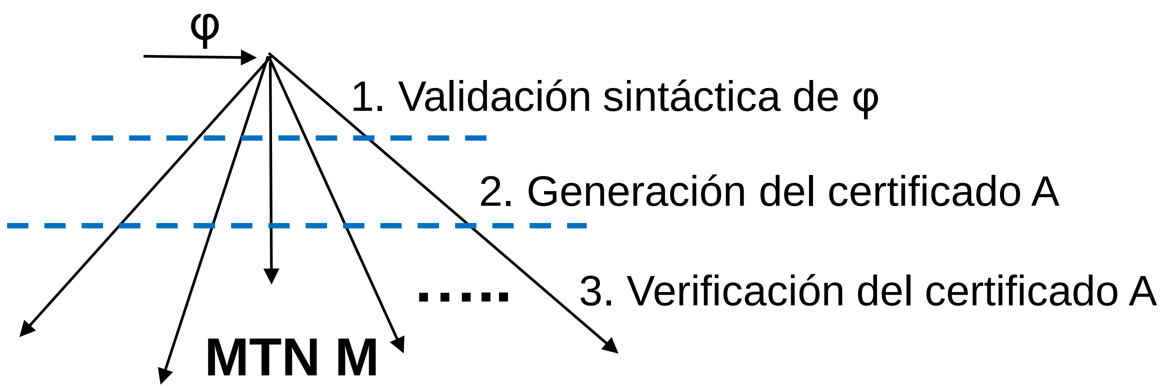
- Se define alternativamente que **L ∈ NP sii existe una MTN que acepta L en tiempo poly(n).**

- **Una MTN se ejecuta en tiempo poly(n) sii todas sus computaciones ejecutan poly(n) pasos.**

- Por ejemplo, la siguiente MTN M
  decide SAT en tiempo poly(n).
  Dada una entrada φ, M hace:

1. Si φ no es una fórmula correcta, rechaza: **O(n)**
2. Genera no determinísticamente una asignación A: **O(n)**
3. Acepta sii A satisface φ: **$O(n^2)$**

Por lo tanto, en total M hace **$O(n^2)$ pasos**



Todas las computaciones de M tardan tiempo poly(n)

- Es decir, en lugar de construir una MTD (MT determinística) que recibe un certificado x,
  se construye una MTN (MT no determinística) que directamente genera el certificado.