Teoría de la Complejidad Computacional

Tema 6: Clases de complejidad computacional en espacio

David Orellana Martín

Grupo de Investigación en Computación Natural
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

dorellana@us.es

Máster Universitario en Lógica, Computación e Inteligencia Artificial

Curso 2023-2024

Índice

* **Complejidad en espacio**.
* **Espacio versus tiempo**.
* Las clases **PSPACE** y **NPSPACE**.
* **PSPACE completitud**.
* **Estrategias ganadoras en juegos**.

Complejidad en espacio

Complejidad computacional en términos de cantidad de espacio/memoria.

- ⋆ Complejidad en **espacio** vs complejidad en **tiempo**.

Definición: *Sea M una MT determinista. Sea C una computación finita de M.*

- ⋆ *El espacio usado por la computación C, $s(C)$, es el número de casillas diferentes consultadas.*

Nota: A veces se considera $s(C) =$ número de casillas diferentes utilizadas.

Definición: *Sea M una MT determinista. La complejidad en espacio de M es la función parcial, $e_M : \mathbf{N} - \to \mathbf{N}$ definida así:*

$$e_M(n) = máx\,\{s(C) :\ \exists x \in \Sigma^n (C\ comput.\ finita\ de\ M\ sobre\ x)\}$$

si para cada $x \in \Sigma^n$ se tiene que $M \downarrow x$ (no definida en otro caso).

Definición: *Sea M una MT no determinista. La complejidad en espacio de M es la función total, $e_M : \mathbf{N} \to \mathbf{N}$, definida así: $e_M(n) = máx\,\{s_M(x) : x \in \Sigma^n\}$, en donde $s_M(x)$ es*

$$\begin{cases} mín\{s(C) : \ C \text{ computación finita de aceptación de } M \text{ sobre } x\} & si\ x \in L(M) \\ 1 & si\ x \notin L(M) \end{cases}$$

En lo que sigue, sea f una función computable 1-aria.

Definición:

- ⋆ **SPACE**$(f) = \{L \subseteq \Sigma^* : \exists M\ (M \text{ máquina Turing determ. } \wedge\ L = L(M) \wedge e_M \in O(f))\}$.
- ⋆ **NSPACE**$(f) = \{L \subseteq \Sigma^* : \exists M\ (M \text{ máq. Turing no determ. } \wedge\ L = L(M) \wedge e_M \in O(f))\}$.

Primeras consideraciones:

- ⋆ **SPACE**$(f) \subseteq$ **NSPACE**(f).
- ⋆ El problema **SAT** es resoluble por una MTD usando una cantidad de **espacio polinomial**.
- ⋆ El espacio **parece** una medida de complejidad **más potente** que el tiempo (**reutilización**).

Teorema: *Toda MTND puede ser* simulada *por una MTD con tres cintas.*

Idea de la demostración:

Sea M una MTND (p: número máximo de elecciones no deterministas en M).

Para cada entrada u de M se considera el árbol de computación T_u asociado:

- ⋆ La raíz es la configuración inicial asociada a u.
- ⋆ Los nodos son las distintas configuraciones.
- ⋆ Existe un arco de C_1 a C_2 sii existe una transición en M de C_1 a C_2.

Se simulará el árbol de computación T_u mediante un *recorrido en anchura*.

- ⋆ En primer lugar, todas las configuraciones obtenidas al ejecutar **un** paso.
- ⋆ En primer lugar, todas las configuraciones obtenidas al ejecutar **dos** pasos.
- ⋆ Y así sucesivamente.

Una *dirección* es una cadena del alfabeto $\Gamma_p = \{1, \ldots, p\}$

- ⋆ Cada nodo de T_u tiene asociada unívocamente una dirección (*válidas*).
- ⋆ Existen direcciones que no se corresponden con ningún nodo (*no válidas*).

Las cadenas de Γ_P^* se ordenan lexicográficamente, según su longitud.

Descripción informal de una MTD, M', con **tres** cintas que *simula* a M:

- ⋆ Cinta 1: de entrada.
- ⋆ Cinta 2: de simulaciones.
- ⋆ Cinta 3: de direcciones.

Un algorimo que implementa una MTD, M', es el siguiente:

1. Inicialmente, la cinta 1 contiene a u y las cintas 2 y 3 están vacías
2. Colocar en la cinta 3 la primera cadena de Γ_P^*
3. Borrar la cinta 2, y copiar la cinta 1 en la cinta 2
4. Usar la cinta 2 para simular la parte de la computación *correspondiente* a la cadena de Γ_P^* que aparece en la cinta 3. Para ello:
   - ⋆ Se analiza el primer símbolo de la cadena de la cinta 3.
   - ⋆ (*) Si es *válido*, entonces
     - Realizar la correspondiente elección no determinista
     - Si ha llegado a una configuración de aceptación, entonces **aceptar** y **terminar**
     - Si no,

       si queda algún símbolo por analizar en la cinta 3, elegir el siguiente e ir a (*)

       si no, ir al paso 5

     si no es *válido*, ir al paso 5
5. Reemplazar la cadena de la cinta 3 por la *siguiente* cadena.
   - ⋆ Si no hay siguiente cadena, entonces **terminar**
6. Ir al paso 3.

Espacio versus tiempo

Teorema 1: *Si* $X \in$ **NTIME**(f), *entonces* $X \in$ **TIME**(k^f) (*siendo* $k \geq 2$ *una constante*) *y* $X \in$ **SPACE**(f).

Demostración: Sean $X \in$ **NTIME**(f) y M una MTND que resuelve X en tiempo $O(f(n))$.

Sea p el número máximo de elecciones no deterministas en M).

Sean M' una MTD que simula a M y $u \in E_X$.

- ⋆ Coste en tiempo de $M(u)$: $O(p^{f(|u|)})$, $(n = |u|)$.
  - ∗ M' debe analizar todas las posibles configuraciones tras un paso (p), tras dos pasos (p^2), . . . , tras $f(n)$ pasos $(p^{f(n)})$. En total, $\theta(p^{f(n)})$ pasos.
  - ∗ Luego, el coste en tiempo de $M'(u)$ es del orden $O(p^{f(|u|)})$.
- ⋆ Coste en espacio de $M(u)$: $O(f(|u|))$,
  - ∗ El coste en espacio de $M'(u)$ es del orden $O(f(|u|))$, que es el máximo espacio usado por *cualquier* computación (simulada) de $M(u)$.

Teorema 2: $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{TIME}(k^{\lg n + f(n)})$, *con* $k > 1$.

Demostración: Sean $X \in \mathbf{NSPACE}(f(n))$ y M una MTND que decide X en espacio $O(f(n))$.

- $\star$ El número de posibles config. de M con entrada $u \in \Sigma^n$ es del orden $O(n \cdot c^{f(n)}) = O(d^{\lg n + f(n)})$, para una constante $c > 1$ que depende de M.

Sea T_u el árbol de computación asociado a u (tamaño $d^{\lg n + f(n)}$).

Se tiene que

- $\star$ $u \in L_X$ sii existe un camino desde I_u a una d.i. de parada y aceptación.

Luego se puede simular M por una MTD que resuelve X en tiempo $O(k^{\lg n + f(n)})$.

■

Corolario: $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{SPACE}(k^{lg_{n+f(n)}})$, *para una cierta constante* $k > 1$.

Demostración:

$$\begin{aligned}
\mathbf{NSPACE}(f(n)) &\subseteq \mathbf{TIME}(k^{\lg_{n+f(n)}}) \\
&\subseteq \mathbf{NTIME}(k^{\lg_{n+f(n)}}) \\
&\subseteq \mathbf{SPACE}(k^{\lg_{n+f(n)}})
\end{aligned}$$

■

El problema de la accesibilidad: Reachability

Enunciado: *Dado un grafo dirigido, G, y dos nodos u, v determinar si existe un camino en G de u a v.*

- ⋆ Solución en tiempo cuadrático (DFS), y usando **espacio de orden lineal**.

Teorema 3: Reachability $\in$ **SPACE**$(\lg^2 n)$.

Demostración: Sea G un grafo dirigido de tamaño n. Sean u, v nodos de G.

Consideremos el predicado

PATH $(x, y, j) \equiv$ existe un camino de x a y de longitud, a lo sumo, 2^j.

Basta decidir **PATH** $(u, v, \lceil \lg n \rceil)$.

Idea para decidir el predicado **PATH** (x, y, j):

- ∗ `si` $j = 0$, `ver si` x `e` y `son adyacentes.`
  `si` $j > 0$, `entonces`
  `para cada nodo` z `hacer`
  si **PATH**$(x, z, j-1) =$ *T* `y` **PATH**$(z, y, j-1) =$ *T* `entonces`
  devolver `SI`

devolver `NO`

Descripción de una MTD, M, con una cinta de entrada y dos de trabajo, que implementa el algoritmo usando espacio $O(\lg^2 n)$.

- ⋆ Cinta de entrada: contiene G, u y v.
- ⋆ Primera cinta de trabajo: contiene las ternas (x, y, j); se inicializa en $(u, v, \lceil \lg n \rceil)$.
- ⋆ Segunda cinta de trabajo: contiene una enumeración de los nodos del grafo.

En cada instante de la computación de M sobre x:

- ⋆ La primera cinta de trabajo contiene $O(\lceil \lg n \rceil)$ ternas; cada una de longitud $O(3 \cdot \lg n) = O(\lg n)$. Luego usa espacio $O(\lg^2 n)$.
- ⋆ La segunda cinta usa espacio del orden $O(\lg n)$.

Por tanto, el espacio total usado es del orden $O(\lg^2 n)$. ■

Teorema de Savitch: $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{SPACE}(f^2(n))$, *para cada función computable 1–aria* f *tal que* $f(n) \geqslant \lg n$.

Demostración: Sean $X \in \mathbf{NSPACE}(f(n))$ y M una MTND que decide X en espacio $O(f(n))$.

Sea $u \in \Sigma^n$.

- ⋆ El árbol T_u de computaciones de M sobre u tiene tamaño $c^{\lg n+f(n)} = O(k^{f(n)})$, para una cierta constante, k, que depende de M.
- ⋆ Decidir que $u \in L_X$ equivale a ver si existe un camino entre dos nodos del grafo de longitud, a lo sumo, $k^{f(n)}$.

Luego se puede simular M por una MTD que resuelve X y usa espacio del orden $O((\lg k^{f(n)})^2) = O(f^2(n))$.

■

Resumen de relaciones obtenidas

- $\mathbf{SPACE}(f) \subseteq \mathbf{NSPACE}(f)$.
- $\mathbf{NTIME}(f) \subseteq \mathbf{SPACE}(f)$.
- $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{TIME}(k^{\lg n + f(n)})$.
- $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{SPACE}(k^{\lg n + f(n)})$.
- $\mathbf{NSPACE}(f) \subseteq \mathbf{SPACE}(f^2)$.

Las clases PSPACE y NPSPACE

Clases correspondientes a **P** y **NP** en la complejidad en espacio.

Definición: $\mathbf{PSPACE} = \bigcup_{k \in \mathbf{N}} \mathbf{SPACE}(n^k)$

- ⋆ **PSPACE**: clase de los problemas resolubles por una MTD que usa espacio polinomial.

Definición: $\mathbf{NPSPACE} = \bigcup_{k \in \mathbf{N}} \mathbf{NSPACE}(n^k)$

- ⋆ **NPSPACE**: clase de los problemas resolubles por una MTND que usa espacio polinomial.

Definición: $\mathbf{L} = \mathbf{SPACE}(\lg n)$.

- ⋆ **L**: clase de los problemas resolubles por una MTD que usa espacio logarítmico.

Definición: $\mathbf{NL} = \mathbf{NSPACE}(\lg n)$.

- ⋆ **NL**: clase de los problemas resolubles por una MTND que usa espacio logarítmico.

Se verifica:

- ⋆ $\mathbf{PSPACE} \subseteq \mathbf{NPSPACE}$.
- ⋆ $\mathbf{L} \subseteq \mathbf{NL}$.
- ⋆ **Reachability** $\in \mathbf{NL}$ (¿$\in \mathbf{L}$?).

Proposición 1: $\mathbf{NL} \subseteq \mathbf{P}$.

Demostración:

$$\mathbf{NL} = \mathbf{NSPACE}(\lg n) \subseteq \bigcup_{k \in \mathbf{N}} \mathbf{TIME}(k^{\lg n + \lg n}) \subseteq \bigcup_{k \in \mathbf{N}} \mathbf{TIME}(n^k)$$

■

Proposición 2: $\mathbf{NP} \subseteq \mathbf{PSPACE}$.

Demostración:

$$\mathbf{NP} = \bigcup_{k \in \mathbf{N}} \mathbf{NTIME}(n^k) \subseteq \bigcup_{k \in \mathbf{N}} \mathbf{SPACE}(n^k) = \mathbf{PSPACE}$$

■

Proposición 3: $\mathbf{NPSPACE} \subseteq \mathbf{EXP} \stackrel{def}{=} \bigcup_{k \in \mathbf{N}} \mathbf{TIME}(k^n)$.

Demostración:

$$\mathbf{NPSPACE} = \bigcup_{k \in \mathbf{N}} \mathbf{NSPACE}(n^k) \subseteq \bigcup_{k, k_1 \in \mathbf{N}} \mathbf{TIME}({k_1}^{\lg n + n^k})$$

$$\subseteq \bigcup_{k_2 \in \mathbf{N}} \mathbf{TIME}({k_2}^n) = \mathbf{EXP}$$

■

Proposición 4: $\mathbf{PSPACE} = \mathbf{NPSPACE}$.

Demostración:

$$\mathbf{NPSPACE} = \bigcup_{k \in \mathbf{N}} \mathbf{NSPACE}(n^k) \overset{Savitch}{\subseteq} \bigcup_{k \in \mathbf{N}} \mathbf{SPACE}(n^{2k}) = \mathbf{PSPACE}$$

■

Corolario: $\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$.

Consideraciones:

- ⋆ El no determinismo es *menos potente* respecto al espacio que respecto al tiempo.
- ⋆ Se prueba que $\mathbf{NPSPACE} = \mathbf{co} - \mathbf{NPSPACE}$.
- ⋆ Se desconoce si algunas de las inclusiones anteriores son estrictas.

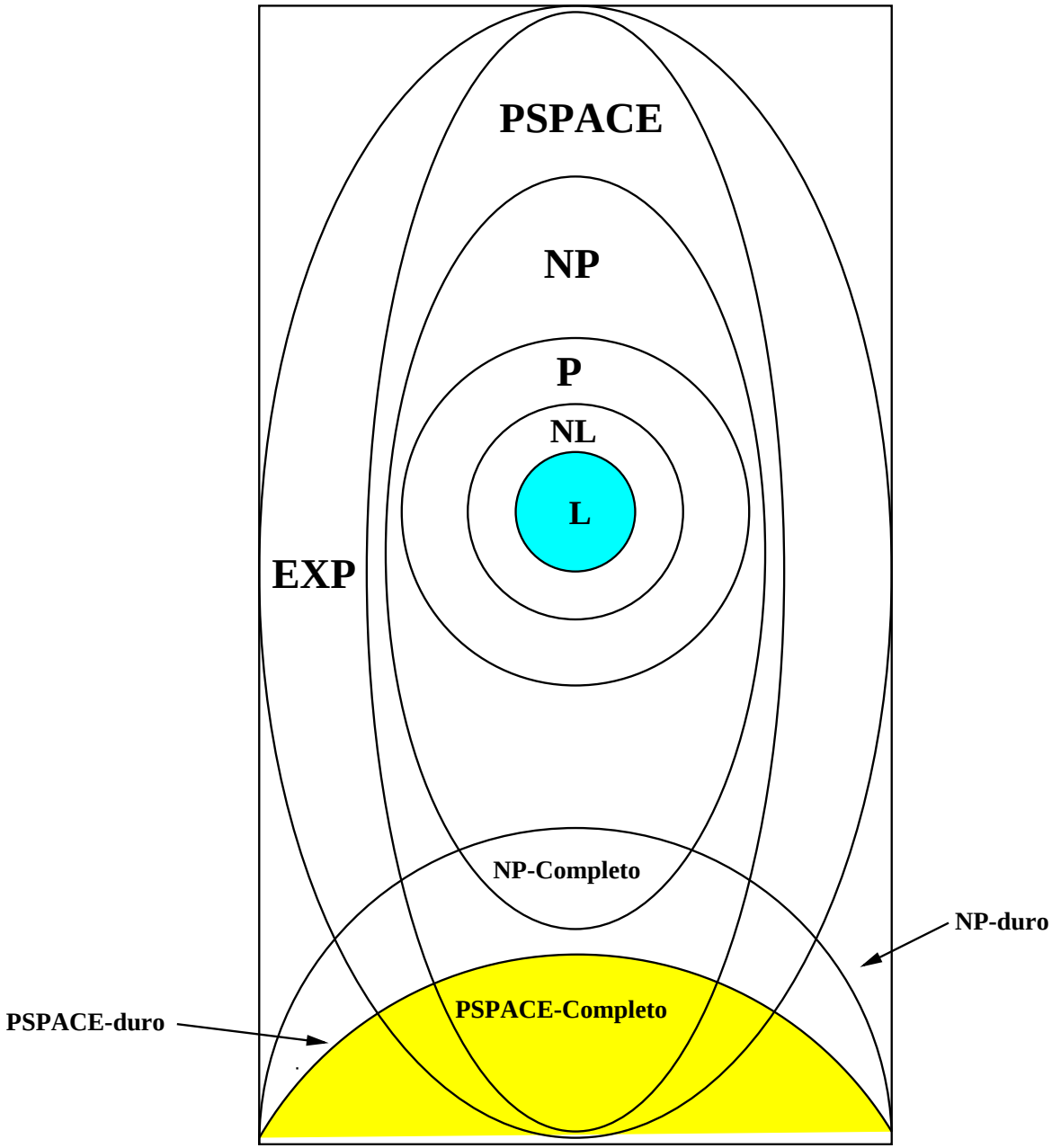
PSPACE-completitud

Definición: *Un problema X es* **PSPACE–duro** *sii para cada $Y \in$* **PSPACE** *se tiene que $Y \leqslant^{p}_{log} X$* (reducibilidad en tiempo polinomial y espacio logarítmico).

Proposición 1: *Si X es* **PSPACE***–duro y $X \leqslant^{p}_{log} Y$, entonces Y es* **PSPACE***–duro.*

Definición: *Un problema X es* **PSPACE–completo** *sii $X \in$* **PSPACE** *y X es* **PSPACE***–duro.*

- Los problemas **PSPACE**–completos son los más difíciles de la clase **PSPACE**
  - Candidatos a testigos de que **P** $\subsetneq$ **PSPACE** y de que **NP** $\subsetneq$ **PSPACE**.


PSPACE
NP
P
NL
L
EXP
NP-Completo
NP-duro
PSPACE-Completo
PSPACE-duro

Proposición 2: *Si X, Y son problemas* **PSPACE***–completos, entonces* $X \equiv^{p}_{log} Y$.

Proposición 3: (Generación de problemas **PSPACE**–completos) *Sea X un problema tal que*

- $\star$ $X \in$ **PSPACE**.
- $\star$ *Existe* Y (Y **PSPACE***–completo* $\wedge$ $Y \leqslant^{p}_{log} X$).

Entonces X es **PSPACE***–completo.*

La búsqueda de **un** adecuado problema **PSPACE**–completo puede responder en sentido afirmativo o en sentido negativo a la cuestión $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ o a la cuestión $\mathbf{NP} \stackrel{?}{=} \mathbf{PSPACE}$.

Proposición 4: *Si X es un problema* **PSPACE***–completo tal que* $X \in \mathbf{P}$, *entonces* $\mathbf{P} = \mathbf{PSPACE}$.

Proposición 5: *Si X es un problema* **PSPACE***–completo tal que* $X \in \mathbf{NP}$, *entonces* $\mathbf{NP} = \mathbf{PSPACE}$.

Fórmulas booleanas cuantificadas

Definición: (Definición recursiva de **FBC**)

- ⋆ *Toda fórmula proposicional es una FBC.*
- ⋆ *Si φ es una FBC y x es una variable que aparece en φ, entonces $\exists x\varphi$ y $\forall x\varphi$ son FBC.*

Definición: *Una FBC φ está en* **forma normal conjuntiva sii** *es de la forma* $Q_1x_1\,Q_2x_2\ldots Q_rx_r\,\psi(x_1,\ldots,x_r,x_{r+1},x_n)$, en donde $Q_i\in\{\exists,\forall\}$ y ψ es una fórmula proposicional en FNC. Diremos que $\mathrm{Var}(\varphi)=\mathrm{Var}(\psi)$ y que ψ es el **núcleo** de φ.

Definición: *Una* **variable** x_j *de φ (FBC en FNC) es* **libre sii** *no hay una subfórmula del tipo $\exists x_j\psi(x_1,\ldots,x_j,\ldots x_n)$ ó $\forall x_j\psi(x_1,\ldots,x_j,\ldots x_n)$.*

Definición: *Diremos que una FBC en FNC es* **cerrada sii** *carece de variables libres.*

Ejemplo: la fórmula $\varphi(x_1,x_2,x_3)\equiv\exists\,x_1\forall\,x_2\exists x_3\,[(x_1\vee\overline{x}_2\vee x_3)\wedge(\overline{x}_1\vee\overline{x}_2\vee x_3)]$

Definición: (satisfactibilidad) *Sea φ una FBC en FNC:*

- ⋆ *Si φ es proposicional, φ es satisfactible* **sii** *existe una valoración, σ, tal que $\sigma(\varphi) = 1$.*
- ⋆ *$\varphi \equiv \exists x_j \psi$ es satisfactible* **sii** *existe una valoración, σ, relevante para ψ tal que $\sigma(\psi) = 1$.*
- ⋆ *$\varphi \equiv \forall x_j \psi$ es satisfactible* **sii** *existen dos valoraciones, σ_1 y σ_2, relevantes para ψ tales que $\sigma_1(x_j) = 1$, $\sigma_1(\psi) = 1$, $\sigma_2(x_j) = 0$, $\sigma_2(\psi) = 1$*

Ejemplo: $\varphi(x_1, x_2, x_3) \equiv \exists\, x_1 \forall\, x_2 \exists x_3\, [(x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3)]$ es satisfactible ya que para $x_1 = x_3 = 1$, la fórmula $\psi \equiv (x_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3)$ será verdadera, con independencia del valor de x_2.

El problema **Q–SAT**: Dada una FBC en FNC y cerrada, determinar si es satisfactible.

Teorema 1: Q–SAT $\in$ PSPACE.

Demostración: Un algoritmo recursivo, A, que resuelve **Q–SAT**:

```
Entrada:  φ (una FBC en FNC y cerrada)
    si φ no contiene cuantificadores entonces
        si φ es satisfactible entonces
            devolver 1
        si no
            devolver 0
    si φ es de la forma ∃xψ entonces
        si A(ψ(x|1)) = 1 ∨ A(ψ(x|0)) = 1 entonces
            devolver 1
        si no
            devolver 0
    si φ es de la forma ∀xψ entonces
        si A(ψ(x|1)) = 1 ∧ A(ψ(x|0)) = 1 entonces
            devolver 1
        si no
            devolver 0
```

En donde $\psi(x|k)$ es la fórmula obtenida a partir de ψ sustituyendo x por k y eliminando el posible cuantificador asociado a x.

El algoritmo A resuelve el problema **Q–SAT**.

Veamos que una MT determinista que implemente este algoritmo puede usar espacio polinomial.

- ⋆ La profundidad de cada llamada recursiva es, a lo sumo, el número de variables, n, de la fórmula.
- ⋆ Cada nivel de la recursión sólo almacena el valor de una variable.

Reutilizando el espacio en cada llamada recursiva, la cantidad de espacio usada será del orden $O(n)$.

■

Teorema 2: *El problema* **Q–SAT** *es* **PSPACE***–duro.*

Idea de la demostración:

Sea $X \in$ **PSPACE**. Veamos que $X \leqslant^{p}_{log}$ **Q–SAT**.

Sea M una MTD que resuelve X y usa espacio $O(n^k)$, para un cierto k.

Para cada $u \in \Sigma^*$ se representan todas las config. de la computación de M sobre u mediante fórmulas proposicionales en FNC (como en el teor. de Cook).

Para cada par de config. c_1, c_2 y cada $t > 0$, se define una FBC en FNC cerrada, $\varphi_{c_1,c_2,t}$, de manera que es satisfactible sii *en la computación de M sobre x se puede pasar de* c_1 *a* c_2 *en, a lo sumo,* t *pasos.*

Construcción de la fórmula:

- ⋆ Caso 1: $t = 1$.

  Basta que la fórmula sea verdadera sii $(c_1 = c_2) \vee (c_1 \vdash_M c_2)$.

- ⋆ Caso 2: $t > 1$.

  - ∗ Primera idea:

    $\varphi_{c_1,c_2,t} \equiv \exists m(\varphi_{c_1,m,\lceil \frac{t}{2} \rceil} \wedge \varphi_{m,c_2,\lceil \frac{t}{2} \rceil})$

    en donde m se expresa a partir de una serie de variables $x_1, \ldots, x_l$ tal que $l \in O(n^k)$.

    La fórmula así construida tendría un tamaño del orden $O(t)$ (y se aplicará al caso $t \in O(2^{n^k})$, número de config. de la computación de M sobre u).

  - ∗ Segunda idea:

    $\varphi_{c_1,c_2,t} \equiv \exists m \forall (c_3, c_4)[((c_3, c_4) = (c_1, m) \vee (c_3, c_4) = (m, c_2) \rightarrow \varphi_{c_3,c_4,\lceil \frac{t}{2} \rceil})]$

Se tiene que $u \in L_X \Longleftrightarrow \varphi_{c_{inic}, c_{acept}, 2^{n^k}}$ es satisfactible.

El tamaño de la fórmula construida es del orden $O((n^k)^2)$ ya que

- ⋆ La profundidad de la recursión es del orden $O(\lg 2^{n^k}) = O(n^k)$.
- ⋆ En cada nivel de la recursión se añaden tres nuevas variables que se describen por $O(n^k)$ variables booleanas.

■

Cuantificadores y Juegos

Juego: varios jugadores tratan de alcanzar un objetivo de acuerdo con unas reglas.

Un jugador tiene una **estrategia ganadora** si vence cuando todos juegan de manera óptima.

El juego de la fórmula: *Dada una FBC en FNC y cerrada:*

- ⋆ *El jugador A selecciona valores de las variables cuantificadas* **universalmente** *y el jugador B de las variables cuantificadas* **existencialmente**.
- ⋆ *El turno de jugada lo da el tipo de cuantificador de la fórmula (de izquierda a derecha).*
- ⋆ *El jugador B gana si la valoración generada hace verdadera el núcleo de la fórmula. Caso contrario, gana A.*

El problema del juego de la fórmula: *Dada una FBC en FNC y cerrada, determinar si el jugador B posee una estrategia ganadora.*

Teorema: *El problema del juego de la fórmula es* **PSPACE***–completo.*

Demostración: Dada una FBC, φ, en FNC y cerrada, son equivalentes:

- $\star$ Determinar si el jugador B posee una estrategia ganadora en el juego asociado a esa fórmula.
- $\star$ La formula φ es satisfactible (como FBC).

Por tanto, la aplicación identidad es una *equivalencia en tiempo polinomial* entre el problema del juego de la fórmula y el problema **Q–SAT**.

■

El juego de Geografía

Se parte de un conjunto finito de ciudades del mundo.

- ⋆ Dos jugadores van nombrando alternativamente ciudades, de manera que:
  - ∗ Un jugador no puede decir una ciudad que ya haya sido nombrada anteriormente por alguno de ellos.
  - ∗ La primera letra de la ciudad que dice un jugador debe coincidir con la última letra de la nombrada anteriormente por el otro.
- ⋆ Pierde aquél jugador que no sepa decir una nueva ciudad verificando la condición citada.

Formalización: *Sea $G = (V, E)$ un grafo dirigido y x_0 un nodo distinguido que inicialmente está marcado. Intervienen dos jugadores I y II.*

- ⋆ *Los jugadores realizan movientos de forma alternativa, marcando un nodo.*
- ⋆ *Comienza el juego el jugador I, marcando un nodo u tal que $(x_0, u) \in E$.*
- ⋆ *Movimiento válido de un jugador: si u es el nodo marcado en el último movimiento de un jugador, el otro debe marcar un nodo v no marcado y tal que $(u, v) \in E$.*
- ⋆ *Pierde el jugador que no pueda marcar ningún nodo más.*

El problema del juego de Geografía: *Dado un grafo dirigido, G, y un nodo distinguido, x_0, determinar si el jugador I posee una estrategia ganadora para el juego de Geografía con entrada (G, x_0).*

Teorema 1: *El problema del juego de Geografía pertenece a la clase* **PSPACE**.

Demostración: Un algoritmo recursivo, A, que resuelve dicho problema:

```
procedimiento A(G, x_0)   (G = (V, E))
    si grado-out (x_0) = 0 entonces
        devolver 0
    si no
        Sea G_1 el subgrafo de G inducido por V − {x_0}
        para cada v_1, ..., v_r tal que (x_0, v_i) ∈ E hacer
            si A(G_1, v_1) = 1 ∧ ··· ∧ A(G_1, v_r) = 1 entonces
                devolver 0
            si no
                devolver 1
```

Veamos que una MT determinista que implemente este algoritmo puede usar espacio polinomial.

- ⋆ La profundidad de cada llamada recursiva es, a lo sumo, el número de nodos, n del grafo.
- ⋆ En cada nivel de la recursión sólo se añade un nuevo nodo.

Reutilizando el espacio en cada llamada recursiva, la cantidad de espacio usada será del orden $O(n)$.

■

Teorema 2: *El problema del juego de Geografía es* **PSPACE**–*completo.*

Demostración: Veamos que el problema del juego de la fórmula es reducible en tiempo polinomial al problema del juego de Geografía.

Sea φ una instancia del problema del juego de la fórmula.

- ⋆ Se puede suponer que $\varphi \equiv \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_{n-1} \exists x_n \psi$, en donde ψ es una fórmula proposicional en FNC.

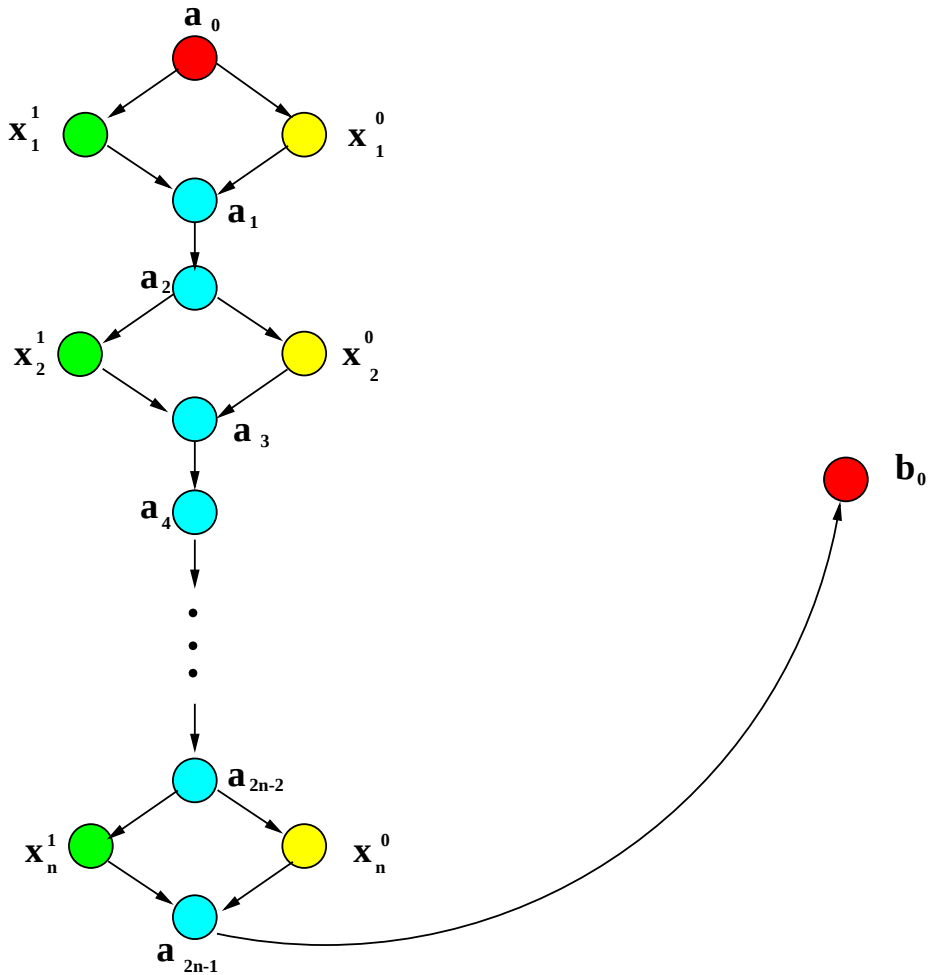
Se asociará a φ un grafo dirigido G_φ y un nodo distinguido a_0 de manera que

- ⋆ El jugador B tiene una estrategia ganadora para φ sii el jugador I tiene una estrategia ganadora para (G_φ, a_0).

Para ello se construirá (G_φ, a_0) de manera que el jugador I simule el comportamiento del jugador B en el juego de la fórmula para φ.

La construcción de G_φ se realiza en dos fases.

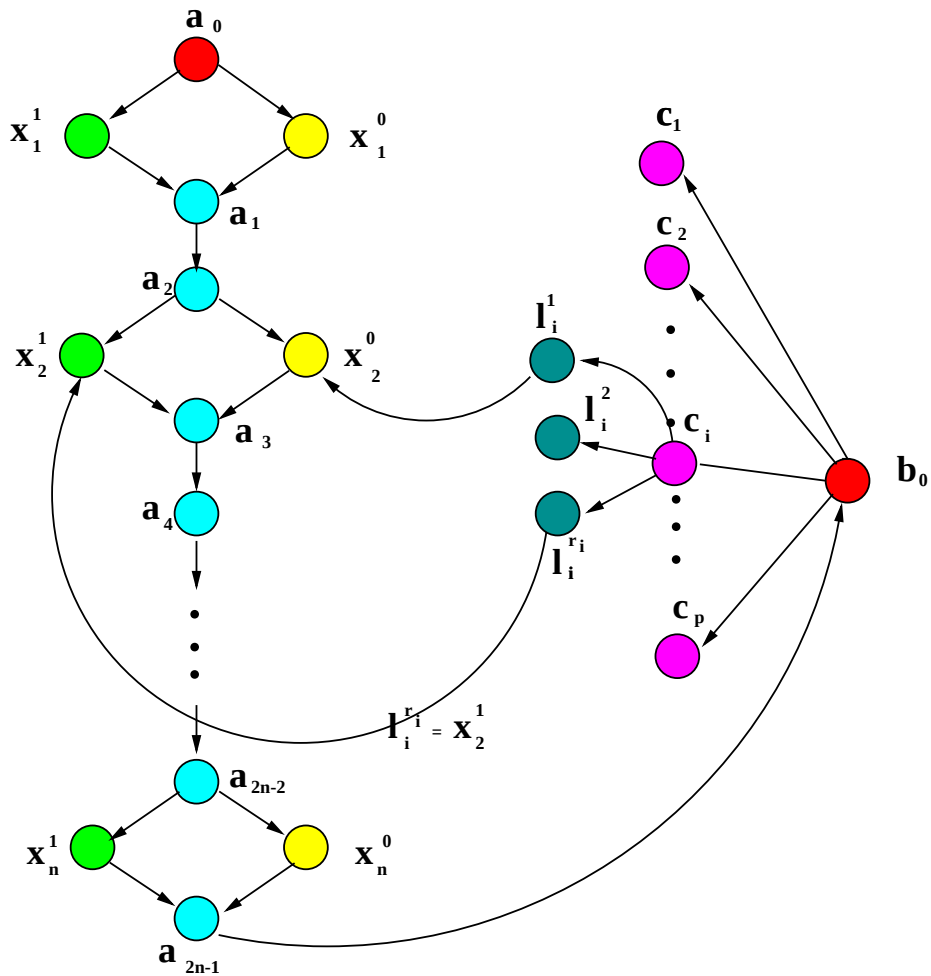
- ⋆ En la primera se considera una sucesión de diamantes (uno por cada variable $x_1, \ldots, x_n$ de φ), con "primer nodo" a_0 y un nodo adicional, b_0, conectados como sigue:



Cada camino $(a_0, x_1^{i_1}, a_1, a_2, x_2^{i_2}, \ldots, x_n^{i_n}, a_{2n-1}, b_0)$ desde a_0 a b_0 se identifica con la valoración, σ, caracterizada por $\sigma(x_j) = i_j$.

- El jugador I marca $x_1^{i_1}$.
- El jugador II está forzado a marcar a_1
- El jugador I está forzado a marcar a_2.
- El jugador II marca $x_2^{i_2}$.
- El jugador I está forzado a marcar a_3.
- El jugador II forzado a marcar a_4.
- Tras las $3n$ primeras jugadas del juego de Geografía, se llega a que el jugador I marca b_0.

- ⋆ En la segunda fase añadimos un nodo por cada cláusula y por cada literal de cada cláusula, conectados como sigue:



Si el literal l_i^j de la cláusula c_i es x_k, entonces (l_i^j, x_k^1) es un arco.

Si el literal l_i^j de la cláusula c_i es $\overline{x}_k$, entonces (l_i^j, x_k^0) es un arco.

Aserto: *La fórmula φ es satisfactible sii el jugador I tiene una estrategia ganadora para (G_φ, a_0).*

- ⋆ Supongamos que φ es satisfactible. En la primera fase del juego, el jugador I simula al jugador B para φ (que construye una valoración σ que hace verdadera φ).
  - ∗ El jugador II marcará un nodo c_t (cláusula verdadera).
  - ∗ El jugador I marcará un literal $l_t^{i_t}$ que sea verdadero por σ.
  - ∗ El jugador II pierde.

- ⋆ Supongamos que φ no es satisfactible. Tras la primera fase del juego, se habrá seleccionado una valoración σ que hará falsa φ.
  - ∗ El jugador II marcará un nodo c_t tal que corresponda a una cláusula falsa.
  - ∗ El jugador I marcará un literal $l_t^{i_t}$ (que ha de ser falso).
  - ∗ Si $l_t^{i_t} = x_k$, el jugador II marcará x_k^1. Si $l_t^{i_t} = \overline{x}_k$ el jugador II marcará x_k^0.
  - ∗ El jugador I pierde.

■