Ma5201 Complejidad Computacional **23 de Junio de 2012**

Pauta Control 3

Prof. Cátedra: M. Kiwi *Prof. Auxiliar: P. Muñoz, D. Salas*

PROBLEMA 1:

(i).- Como UPath está en log-espacio, existe una máquina de Turing M que en log-espacio decide UPath. Usaremos M como subrutina para decidir, dado $\langle G\rangle$ instancia de Bipartite, si G es un grafo bipartito. De hecho, ejecutaremos M en $|V(G)|$ instancias distintas construidas a partir de $\langle G\rangle$. Primero definimos el grafo H_G tal que por cada nodo v de G hay dos copias v' y v'' en H_G. Por cada arco uv de G, se agregan dos arcos (u',v'') y (u'',v') en H_G. Afirmamos que G tiene cíclos de largo impar si y sólo si para algún nodo v de G existe un camino en H_G entre v' y v''. En efecto, $v_0v_1,\ldots,v_{2k},v_0$ es cíclo de largo impar en G si y sólo si $v'_0,v''_1,v'_2,\ldots,v'_{2k},v''_0$ es camino en H_G de v'_0 a v''_0. Sea B la máquina de Turing que en la instancia $\langle G\rangle$ de Bipartite simula M en $\langle H_G,v',v''\rangle$ para todo $v\in V(G)$ y acepta si M rechaza todas las veces. Sigue que B acepta $\langle G\rangle$ si y solo si G no posee cíclos de largo impar, es decir B decide Bipartite.

Falta ver que B es a log-espacio. Para ello, basta notar que B solo requiere recordar el último valor de $v\in V(G)$ para el que ha simulado M en $\langle H_G,v',v''\rangle$ (lo que requiere espacio $O(\log n)$), y luego simular M en $\langle H_G,v',v''\rangle$. Para simular M, la máquina de Turing B no necesita construir H_G puesto que la codificación de G implícitamente determina H_G (por ejemplo, para determinar que $u'v''$ es un arco de H_G basta verificar que uv es un arco de G). Dado que H_G es el doble de grande que G, la simulación de M requiere espacio $O(\log(|V(H_G)|^2)) = O(\log|V(G)|^2) = O(\log|V(G)|)$, es decir es a log-espacio en el tamaño de $\langle G\rangle$. En resumen B es a log-espacio.

(ii.1).- Notar que:

$$\langle G,d,k\rangle\in \text{MinServLoc} \iff (\exists S\subseteq V(G), |S|\le k, \max_{v\in V(G)} \text{dist}(v,S)\le d) \qquad (1)$$
$$\wedge\,(\forall U\subseteq V(G), |U|\le k-1 \Longrightarrow \max_{v\in V(G)} \text{dist}(v,U)\ge d+1)\,,$$

donde la distancia $\text{dist}(u,C)$ denota el largo mínimo de un camino en G de u a un nodo de $C\subseteq V(G)$. Observar que cualquier $C\subseteq V(G)$ puede codificarse usando $O(|V(G)|\log|V(G)|)$ bits. Además, para todo $C\subseteq V(G)$ el valor de

$$\max_{v\in V(G)} \min_{c\in C} \text{dist}(v,c)$$

puede ser calculado en tiempo polinomial — por ejemplo, calculando primero la distancia entre cualquier par de nodos en G con un algoritmo de *all-pair-shortest path* (lo que puede hacerse en tiempo $O(|V(G)|^3)$ en el modelo RAM y posteriormente calculando los máximos y mínimos correspondientes en tiempo $O(|V(G)|^2)$ (nuevamente en el modelo RAM), Sigue que una máquina de Turing que realiza los siguientes pasos decide MinServLoc en espacio polinomial:

- La máquina cícla sobre todos los $S\subseteq V(G)$, $|S|\le k$, y verifica que existe alguno tal que $\max_{v\in V(G)}\text{dist}(v,S)\le d$).

- La máquina cícla sobre todos los $U \subseteq V(G)$, $|U| \leq k-1$, y verifica que para todos ellos se cumple que $\text{máx}_{v \in V(G)} \text{dist}(v,U) \geq d+1$).

(ii.2).- Afirmamos que MinServLoc está en Σ_2^P, la que se conjetura estrictamente contenida en PESPACIO (lo contrario implicaría que PESPACIO $= \Sigma_2^P \cap \Pi_2^P$ y que la jerarquía polinomial colapsa a su segundo nivel). Para establecer la afirmación basta notar que el valor de verdad del lado derecho de (1) puede ser evaluado por una máquina de Turing alternante del tipo Σ_2^P dado que, como se observo en el punto anterior, cualquiera sea $C \subseteq V(G)$, el valor de $\text{máx}_{v \in V(G)} \text{mín}_{c \in C} \text{dist}(v,c)$ puede ser calculado en tiempo polinomial en el modelo RAM. Sigue que MinServLoc $\in \Sigma_2^P$.

(iii).- Afirmamos que Pebble es P-completo. Hay dos formas de establecer que Pebble es P-duro. Una forma es tomar un lenguaje cualquiera en P y recordar que dado que P es igual a AL (la clase de lenguajes decididos por máquinas de Turing alternantes a log-espacio), entonces existe una máquina alternante a log-espacio, digamos A, que decide L. Luego, dado ω instancia de L uno construye una instancia de Pebble que pertenece a dicho lenguaje si y sólo si A acepta ω. Otro forma es mostrar que un lenguaje P-duro log-espacio reduce a Pebble. Elaboraremos la segunda manera de abordar el problema dado que es más fácil de describir.

Consideremos el lenguaje MonCVAL, es decir la colección de instancias $\langle C,x \rangle$ en que C es un circuito Booleano monótono en n entradas y $x \in \{0,1\}^n$ son tales que $C(x) = 1$.

Podemos asumir que la salida de los circuitos Booleanos monótonos C considerados son puerta lógicas del tipo $\vee$, y que C tiene una cantidad par de niveles de puertas lógicas, donde todas aquellas que están a profundidad par (respectivamente impar) son del tipo $\vee$ (respectivamente $\wedge$). En efecto, para transformar en log-espacio un circuito Booleana monótono a otro con las propiedades descritas basta intercalar adecuadamente puertas lógicas del tipo $\vee$ y $\wedge$ con un único arco entrante y un único arco saliente.

Sea entonces la reducción que a $\langle C,x \rangle$ le asocia $f(\langle C,x \rangle) = \langle G, V_0, V_1, s, t \rangle$ tal que

- Los nodos de G son los nodos de C más dos nodos especiales u_0 y u_1 (representando 0 y 1, respectivamente). Denotaremos por $v_{x_1}, \ldots, v_{x_n}$ los nodos de G asociados a las entradas $x_1, \ldots, x_n$ de C.
- Los arcos de G son el reverso de los arcos de C más arcos $v_{x_i} u_b$ si $x_i = b$, donde $i = 1, \ldots, n$ y $b \in \{0,1\}$.
- Definimos V_0 (respectivamente V_1) como la colección de los nodos v_{x_i} tales que $x_i = 1$ (respectivamente $x_i = 0$) unión todos los nodos de C correspondientes a puertas lógicas del tipo $\vee$ (respectivamente $\wedge$). Los nodos especiales u_0 y u_1 están ambos en V_0.
- Por último, hacemos s igual al nodo de salida de C y $t = u_1$.

Dado que la construcción de $f(\langle C,x \rangle)$ se puede hacer a partir de $\langle C,x \rangle$ con operaciones locales, es fácil concluir que f es una reducción a log-espacio.

Afirmamos que $\langle C,x \rangle \in$ MonCVAL si y sólo si $\langle G, V_0, V_1, s, t \rangle \in$ Pebble. Como ya se observo, podemos asumir que C tiene $2k$ niveles de puertas lógicas (las de nivel par del tipo $\vee$ y las de nivel impar del tipo $\wedge$, en particular la salida es del tipo $\vee$). Para un nodo interno v de C, denotaremos por $\text{tp}(v) \in \{\vee, \wedge\}$ el tipo de puerta lógica de v y por $v(x) \in \{0,1\}$ el valor asignado a la puerta lógica v al evaluar C en x.

Notar entonces que $C(x) = 1$ si y sólo si

$$\exists v_1, v_1 s \in E(C), \mathrm{tp}(v_1) = \wedge, v_1(x) = 1, \forall v_2, v_2 v_1 \in E(C), \mathrm{tp}(v_2) = \vee, v_2(x) = 1,$$
$$\exists v_3, v_3 v_2 \in E(C), \mathrm{tp}(v_3) = \wedge, v_3(x) = 1, \ldots \exists v_{2k-1}, v_{2k-1} v_{2k-2} \in E(C), \mathrm{tp}(v_{2k-1}) = \vee, v_{2k-1} = 1,$$
$$\forall v_{x_i}, v_{x_i} v_{2k-1} \in E(C), x_i = 1\,.$$

Lo anterior equivale a

$$\exists v_1, s v_1 \in E(G), \forall v_2, v_1 v_2 \in E(G), \exists v_3, v_2 v_3 \in E(G), \ldots$$
$$\ldots \exists v_{2k-1}, v_{2k-2} v_{2k-1} \in E(G), \forall v_{x_i}, v_{2k-1} v_{x_i} \in E(G), v_{x_i} t \in E(G)\,,$$

que a su vez es equivalente a decir que J_0 tiene una estrategia ganadora en la instancia $f(\langle C, x\rangle)$. Sigue que f es una reducción de MonCVAL a Pebble. En resumen, MonCVAL log-espacio reduce a Pebble. Como se vió en auxiliar, MonCVAL es P-completo. Sigue que Pebble es P-duro.

Para concluir, veamos que Pebble se puede decidir en tiempo polinomial. Para ello consideramos el algoritmo recursivo $\mathcal{A}$ descrito en la Figura 1

Algorithm 1: Algoritmo para decidir Pebble.

input : $\langle G, V_0, V_1, s, t\rangle$ instancia de Pebble.
output: Aceptar si el jugador J_0 tiene una estrategia ganadora.

```
/* Inicializar nodos marcados                                   */
M_0 ← {t};
M_1 ← ∅;

/* Re-marcar nodos                                              */
for i = 1, ..., |V(G)| do
    for v ∈ V(G) \ (M_0 ∪ M_1) do
        /* Determinar vecindad de v                             */
        U ← {u ∈ V(G) : vu ∈ E(G)};
        /* Si los vecinos de v están marcados, marcar v         */
        case v ∈ V_0 y U ∩ M_0 ≠ ∅
            M_0 ← M_0 ∪ {v};
        case v ∈ V_0 y U ⊆ M_1
            M_1 ← M_1 ∪ {v};
        case v ∈ V_1 y U ∩ M_1 ≠ ∅
            M_1 ← M_1 ∪ {v};
        case v ∈ V_1 y U ⊆ M_0
            M_0 ← M_0 ∪ {v};

/* Aceptar si y sólo si s fue marcado con un 0                  */
if s ∈ M_0 then
    return(Aceptar);
else
    return(Rechazar);
```

Observar que en el algoritmo $\mathcal{A}$ la parte interna del doble **for** se ejecuta $O(|V(G)|^2)$ veces y que cada ejecución requiere realizar $O(|V(G)|)$ operaciones (en el modelo RAM) – $O(|V(G)|)$ para determinar la vecindad de de

cada nodo v y $O(|V(G)|)$ para determinar si se cumple alguno de los casos. Luego, el algoritmo $\mathcal{A}$ se puede implementar en tiempo $O(|V(G)|^3)$ en el modelo RAM, y por lo tanto es a tiempo polinomial. En resumen, Pebble se puede decidir en tiempo polinomial.

(iv).- Por contradicción, supongamos que DESPACIO$(n) =$ P.

Veamos que se tiene que PESPACIO $\subseteq$ P. En efecto, consideremos $L \in$ PESPACIO. Se tiene que existe una máquina de Turing M que decide L a espacio $p(n)$, donde p es un polinomio. Sin pérdida de generalidad podemos asumir que $p(n) \geq n$ para todo $n \in \mathbb{N}$. Sea $f(n) = p(n) - n$ y definamos padL como se sugiere en la indicación. Afirmamos que padL $\in$ DESPACIO(n). En efecto, sea $\widetilde{M}$ la máquina de Turing que en la entrada $\sigma = \omega\#^m$, con $\omega \in \Sigma_M^*$, realiza los siguientes pasos:

1. Verifica que $m = p(|\omega|) - |\omega|$ (observar que podemos asumir que $\widetilde{M}$ conoce p, y que el cáculo de $p(|\omega|)$ puede hacerse en espacio $p(|\omega|) = |\sigma|$).
2. Simula M en ω. La máquina $\widetilde{M}$ acepta σ si y sólo si M acepta ω (observar que la simulación de M requiere espacio $p(|\omega|) = |\sigma|$).

En resumen se cumple la afirmación enunciada.

Como padL $\in$ DESPACIO(n), por hipótesis, se tiene que padL $\in$ P. Luego, existe una máquina de Turing T a tiempo polinomial $q(n)$ que decide padL. Afirmamos ahora que $L \in$ P. En efecto, sea $\widetilde{T}$ la máquina de Turing que en la entrada ω calcula $m = p(|\omega|) - |\omega|$, construye $\sigma = \omega\#^m$ y simula T en σ. La máquina $\widetilde{T}$ acepta ω si y sólo si T acepta σ. Es fácil ver que $\widetilde{T}$ decide L en tiempo polinomial $q(p(n))$. Es resumen se cumple la afirmación enunciada.

Notar que hemos demostrado que si DESPACIO$(n) =$ P, entonces PESPACIO $\subseteq$ P $=$ DESPACIO(n), lo que contradice el Teorema de la Jerarquía del Espacio.

PROBLEMA 2:

(i).- El punto clave es observar que el hecho que L sea unario implica que contiene o una o ninguna palabra de largo n, cualquiera sea $n \in \mathbb{N}$ — en el primer caso, la palabra en L sería 1^n. Sea T_n y V_n los circuitos Booleanos en n entradas ilustrados en la Figura 1.

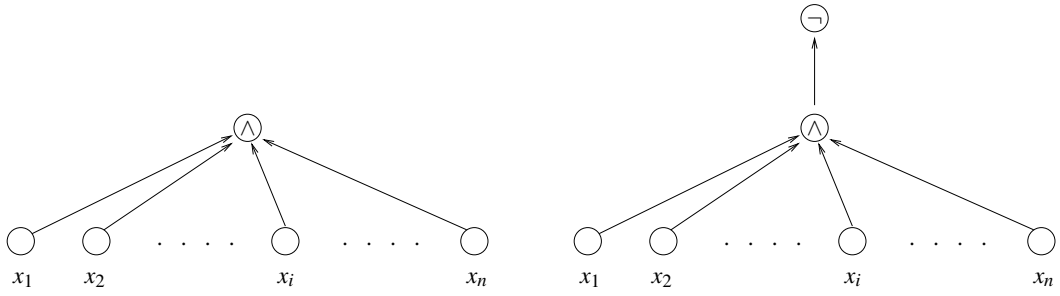

Figura 1: T_n (izquierda) y V_n (derecha).

Sea $(C_n)_{n\in\mathbb{N}}$ la familia de circuitos Booleanos tal que $C_n = T_n$ si $1^n \in L$ y $C_n = V_n$ en caso contrario. Clara-

mente, $\omega \in L$ si y solo si $C_n(\omega) = 1$ donde $n = |\omega|$. Además, $|C_n| \leq 2$ para todo $n \in \mathbb{N}$. Luego, por definición de P/poli, se tiene que $L \in$ P/poli.

(ii).- Basta ver que Halt mucho-a-uno reduce a UHalt. En efecto dada una instancia $\langle M, \omega \rangle$ de Halt (es decir tal que M es máquina de Turing y ω una entrada de M), definimos $f(\langle M, \omega \rangle) = 1^n$ donde $n \in \mathbb{N}$ es tal que su representación en binario es $\langle M, \omega \rangle$. Claramente, f es totalmente recursiva. Además, $\langle M, \omega \rangle \in$ Halt si y solo si $f(\langle M, \omega \rangle) \in$ UHalt. En resumen, Halt mucho-a-uno reduce a UHalt.

(iii).- Veamos primero que P $\subseteq$ P/poli. Del Teorema de Cook-Levin sabemos que $L \in$ P si y solo si existe una familia de circuitos Booleanos $(C_n)_{n \in \mathbb{N}}$ y un polinomio $p(n)$ tales que $|C_n| \leq p(n)$, C_n tiene n entradas y $\omega \in L$ si y sólo si $C_n(\omega) = 1$, donde $n = |\omega|$. Por definición de P/poli sigue que $L \in$ P/poli.

Como todo lenguaje en P es decidible, por (ii), se tiene que UHalt $\notin$ P. Además, por (i), se tiene que UHalt $\in$ P/poli.

En resumen, P esta estrictamente contenida en P/poli.

(iv).- Si NP $\subseteq$ P/poli, entonces SAT $\in$ P/poli. Luego, por definición de P/poli, existe una familia de circuitos Booleanos $(\widetilde{C}_m)_{m \in \mathbb{N}}$ y un polinomio $\widetilde{p}(\cdot)$ tales que $|\widetilde{C}_m| \leq \widetilde{p}(m)$, $\widetilde{C}_m$ tiene m entradas, y cualquiera que sea ϕ fórmula Booleana en n variables, se tiene que

$$\exists v \in \{0,1\}^n \phi(v) = 1 \iff \widetilde{C}_m(\langle \phi \rangle) = 1, \text{ donde } m = |\langle \phi \rangle|.$$

Consideremos ahora φ fórmula Booleana en $2n$ variables tal que $|\langle \varphi \rangle| \leq r(n)$ donde $r(\cdot)$ es un polinomio. Definamos $\phi_u(\cdot) = \varphi(u, \cdot)$. Sin pérdida de generalidad podemos asumir que todo ϕ_u con las características recién mencionadas tienen un largo de codificación exactamente igual a $\widetilde{r}(n) = 2r(n)$ (esto se puede lograr vía recodificación de 0 y 1 por 00 y 10 respectivamente, y padding con 11's). Sigue que, para todo $u \in \{0,1\}^n$,

$$\exists v \in \{0,1\}^n, \varphi(u,v) = 1 \Longleftrightarrow \exists v \in \{0,1\}^n, \phi_u(v) = 1 \Longleftrightarrow \widetilde{C}_{\widetilde{r}(n)}(\langle \phi_u \rangle) = 1.$$

Definimos entonces $C_n(\langle \varphi \rangle, u) = \widetilde{C}_{\widetilde{r}(n)}(\langle \phi_u \rangle)$. Sigue que la familia de circuitos Booleanos $(C_n)_{n \in \mathbb{N}}$ satisface la condición deseada para el polinomio $p(\cdot)$ tal que $p(n) = \widetilde{p}(\widetilde{r}(n))$.

(v).- Recordemos que dado un algoritmo a tiempo polinomial para decidir SAT uno podía construir un algoritmo también a tiempo polinomial para, dada una (codificación de) una fórmula Booleana ϕ, encontrar una asignación x tal que $\phi(x) = 1$, si tal asignación existe. De manera similar, se tiene que a partir de la familia de circuitos $(C_n)_{n \in \mathbb{N}}$ de la parte (iv) uno puede construir otra familia de circuitos Booleanos en n salidas $(C'_n)_n$, también a tamaño polinomial digamos $q(n)$, tal que dada una codificación de largo $\widetilde{r}(n)$ de una fórmula Booleana ϕ en n variables, determina una asignación $x = C'_n(\langle \phi \rangle)$ tal que $\phi(x) = 1$, si tal asignación existe. Sigue que, de la parte (iv), que

$$\exists v \in \{0,1\}^n, \varphi(u,v) = 1 \implies \varphi(u, C'_n(\langle \varphi \rangle, u)) = 1\,.$$

Por otro lado, el converso se tiene trivialmente.

(vi).- Por (v), se tiene que si NP $\subseteq$ P/poli, entonces existe una familia de circuitos Booleanos con múltiples entradas y n-salidas $(C'_n)_{n \in \mathbb{N}}$ y un polinomio $q(\cdot)$ tales que $|C'_n| \leq q(n)$ y cualquiera sea φ fórmula Booleana en $2n$ variables tal que $|\langle \varphi \rangle| \leq r(n)$ donde $r(n)$ es un polinomio, y cualquiera sea $u \in \{0,1\}^n$, se satisface que:

$$\forall u \in \{0,1\}^n, \exists v \in \{0,1\}^n, \varphi(u,v) = 1 \iff \forall u \in \{0,1\}^n, \varphi(u, C'_n(\langle \varphi \rangle, u)) = 1\,.$$

Luego,

$$\forall u \in \{0,1\}^n, \exists v \in \{0,1\}^n, \varphi(u,v) = 1 \implies \exists C' \text{ circuito Booleano con } n\text{-salidas tal que } |C'| \leq q(n), \quad \forall u \in \{0,1\}^n, \varphi(u, C'_n(\langle\varphi\rangle, u)) = 1\,. \tag{2}$$

Por otro lado, el converso es trivialmente cierto.

Finalmente, recordemos que $\Pi_2$3SAT es el conjunto de instancias $\psi = \forall u, \exists v, \varphi(u,v)$ donde φ es una fórmula Booleana en forma 3-conjuntiva normal y ψ es verdadera. Cualquier de estas instancias de $\Pi_2$3SAT puede ser fácilmente transformada en una fórmula del tipo que aparece a la izquierda de la equivalencia en (2) — es decir tal que φ sea sobre $2n$ variables (considerando/agregando variables que no se utilizan) y tal que $|\langle\varphi\rangle| \leq r(n)$ con $r(\cdot)$ polinomio (tomando $r(n) = O(n^4)$ y observando que toda fórmula Booleana en forma 3-conjuntiva normal sobre m variables tiene a lo más $O(m^3)$ claúsulas distintas, las que se pueden codificar en $O(m^4)$ bits). Además, el predicado de la derecha de la equivalencia en (2) es fácilmente decidible por una máquina de Turing del tipo Σ_2^P. Sigue que $\Pi_2$3SAT $\in \Sigma_2^P$. Como $\Pi_2$3SAT es Π_2^P-completo, sigue que $\Pi_2$3SAT $\in \Sigma_2^P$, luego la jerarquía polinomial colapsa a su segundo nivel.