Teoría de la Computación | **Segundo Parcial**
Licenciatura en Informática, Universidad Nacional de Quilmes | 1er cuatrimestre de 2022

Ejercicio 1. Sea $\ell = [x_1, x_2, \dots, x_n]$ una lista de 0s y 1s, es decir $x_i \in \{0, 1\}$ para cada $i \in 1..n$. Un *salto* es un índice i tal que $1 \le i < n$ y además $x_i \ne x_{i+1}$. Por ejemplo, la lista $[0, 0, 0, 0, 0]$ no tiene saltos, la lista $[0, 0, 1, 1, 1, 1]$ tiene un salto, la lista $[1, 1, 0, 0, 1, 1]$ tiene dos saltos y la lista $[0, 1, 0, 1, 0, 1]$ tiene cinco saltos.

Sea φ una fórmula de la lógica proposicional arbitraria[1] que usa n variables $x_1, x_2, \dots, x_n$. Decimos que φ es satisfactible con k saltos si existe una asignación de variables que hace verdadera a la fórmula y tiene k saltos. Por ejemplo, la fórmula $((x_1 \vee \neg x_2) \wedge x_3 \wedge \neg x_4 \wedge \neg x_5)$ es satisfactible con un salto, asignándole los valores $[1, 1, 1, 0, 0]$ a las variables x_1, x_2, x_3, x_4, x_5 respectivamente. Considerar el lenguaje:

$$\textsf{3-SALTO} = \{\langle \varphi \rangle \mid \varphi \text{ es satisfactible con 3 saltos}\}$$

Elegir **exactamente una** de las cuatro afirmaciones siguientes y demostrarla[2]:

1. $\textsf{3-SALTO} \in (\textsf{P} \cap \textsf{NP})$, es decir, está en las clases $\textsf{P}$ y $\textsf{NP}$.
2. $\textsf{3-SALTO} \in (\textsf{P} \setminus \textsf{NP})$, es decir, está en la clase $\textsf{P}$ pero no en la clase $\textsf{NP}$.
3. $\textsf{3-SALTO} \in (\textsf{NP} \setminus \textsf{P})$, es decir, está en la clase $\textsf{NP}$ pero no en la clase $\textsf{P}$.
4. $\textsf{3-SALTO} \notin (\textsf{NP} \cup \textsf{P})$, es decir, no está en la clase $\textsf{P}$ ni en la clase $\textsf{NP}$.

Ejercicio 2. Dados lenguajes $A, B \subseteq \Sigma^*$ notamos $A \uplus B$ al lenguaje $\{0.w \mid w \in A\} \cup \{1.w \mid w \in B\}$. Decidir si las siguientes afirmaciones son verdaderas o falsas y demostrar:

1. Si $A \le_P A'$ y $B \le_P B'$, entonces $(A \uplus B) \le_P (A' \uplus B')$.
2. Si A es $\textsf{NP}$-completo y B es $\textsf{NP}$, entonces $(A \uplus B) \le_P A$.
3. Si A es $\textsf{NP}$-completo y B es $\textsf{NP}$, entonces $(A \uplus B)$ es $\textsf{NP}$-completo.

Ejercicio 3. Dadas dos listas de números naturales $\ell_1 = [a_1, a_2, \dots, a_n]$ y $\ell_2 = [b_1, b_2, \dots, b_n]$, decimos que una lista $\ell_3 = [x_1, x_2, \dots, x_n]$ es una combinación de ℓ_1 y ℓ_2 si para todo $i \in 1..n$ se tiene que o bien $x_i = a_i$ o bien $x_i = b_i$. Por ejemplo, la lista $\ell_3 = [1, 8, 7, 4, 5]$ es una combinación de $\ell_1 = [1, 2, 3, 4, 5]$ y $\ell_2 = [9, 8, 7, 6, 0]$. La lista $\ell_3' = [9, 8, 3, 6, 0]$ es otra posible combinación.

Definimos el lenguaje $\textsf{COMB-SUM}$ del siguiente modo:

$$\textsf{COMB-SUM} = \{\langle \ell_1, \ell_2, k \rangle \mid \text{hay una combinación de } \ell_1 \text{ y } \ell_2 \text{ que suma } k\}$$

Por ejemplo, $\langle [1, 2, 3], [50, 50, 50], 101 \rangle \in \textsf{COMB-SUM}$, ya que $[1, 50, 50]$ suma 101 y es una combinación de las listas $[1, 2, 3]$ y $[50, 50, 50]$.

Demostrar que $\textsf{COMB-SUM}$ es $\textsf{NP}$-completo.

Ejercicio 4. Sea $F : \Sigma^* \times \Sigma^* \to \Sigma^*$ una función computable en tiempo polinomial que dadas dos palabras $w_1, w_2 \in \Sigma^*$ devuelve una palabra $F(w_1, w_2) \in \Sigma^*$. Definimos la noción de palabra **buena** recursivamente del siguiente modo:

- **Caso base.** La palabra vacía (λ) es buena.
- **Caso recursivo.** Una palabra w no vacía es buena si y sólo si existen palabras $w_1, w_2 \in \Sigma^*$ tales que (1) w_1 y w_2 son buenas, (2) w_1 y w_2 son **subcadenas** de w (estrictamente más cortas que w), y (3) $F(w_1, w_2) = w$.

Demostrar que el lenguaje $\textsf{BUENA} = \{w \in \Sigma^* \mid w \text{ es una palabra buena}\}$ está en la clase $\textsf{PSPACE}$.

Justificar todas las respuestas.

[1] Usando los conectivos $\neg, \wedge, \vee$ pero no necesariamente en 3-CNF.
[2] Puede haber varias afirmaciones verdaderas, pero alcanza con elegir una y demostrarla.