

Modelos de Computación y Complejidad

(4º curso del Grado en Ingeniería Informática. Tecnologías Informáticas)

Algunos ejemplos de “**Justificar**” para el EXAMEN ONLINE INDIVIDUAL

PARTE SEGUNDA : Justificar la veracidad o falsedad de los siguientes asertos

1. Existe **un** programa GOTO que **no** es el programa **vacío** y tal que posee una **configuración de parada** cuyo estado es el estado inicial asociado a un dato de entrada.
2. Existe un programa GOTO P que calcula, a la vez, el **predicado** “mayor o igual” y la **función** constante e igual a 1, de aridad 1.
3. La función $f : \mathbb{N} \rightarrow \mathbb{N}$ definida por $f(n) = n$, si n es primo, y $f(n) \uparrow$, si n no es primo, es GOTO-computable.
4. Si P y Q son programa GOTOs calculan la **misma** función de aridad 1, entonces calculan la **misma** función de aridad 2.
5. Si P es un programa GOTO que sólo contiene las variables X_1 , X_2 e Y entonces **sólo** puede calcular una función de aridad 1 y otra de aridad 2.
6. La función vacía, la función constante igual a 0 y la función identidad en $\mathbb{N}$, se pueden calcular con un programa GOTO que posea exactamente **tres** instrucciones.
7. La función vacía, la función constante igual a 0 y la función identidad en $\mathbb{N}$, se pueden calcular mediante programas GOTO que contengan únicamente instrucciones del tipo **incremento** y del tipo **condicional**.
8. Sean A y B conjuntos de números naturales tales que los conjuntos $A - B$, $B - A$ y $A \cap B$ son GOTO-computable. Entonces los conjuntos A y B **también** son GOTO-computables.
9. Sean A y B conjuntos de números naturales tales que $A \subseteq B$ y B es un conjunto GOTO-computable. Entonces, A **también** es un conjunto GOTO-computable.
10. Si A y B son conjuntos GOTO-computables de números naturales entonces el conjunto $(A - B) \cup (B - A)$ es, **también**, GOTO-computable.
11. Si un programa GOTO carece de instrucciones condicionales entonces todas sus computaciones son de parada.
12. Si P es un programa GOTO que consta de una **única instrucción** entonces el código de P **no puede coincidir** con el código de una **configuración** de ese programa.
13. **Toda configuración** del **programa vacío** es, asimismo, una **configuración inicial** de dicho programa.
14. Si A es un conjunto GOTO-computable entonces la función $f : \mathbb{N} \rightarrow \mathbb{N}$ definida por $f(x) \uparrow$ si $x \in A$ y $f(x) = x + 2$ si $x \notin A$ **es GOTO-computable**.
15. Si f y g son funciones GOTO-computables de la misma aridad y con el mismo dominio, entonces la función h definida por $h(x) = f(x) + g(x)$, si x es par, y $h(x) = f(x) \cdot g(x)$, si x es impar, **también** es GOTO-computable.
16. El código de un estado **siempre** es un número estrictamente mayor que cero.
17. Si el código de un programa P **coincide** con el código de una **configuración de parada** de P , entonces el programa P posee más de una instrucción.

18. Si P es un programa con, **exactamente dos** instrucciones y tal que su código **coincide** con el de una **configuración de parada** σ de P , entonces el estado s de esa configuración es un **estado inicial**.
19. La función $f : \mathbb{N}^- \rightarrow \mathbb{N}$ definida por $f(x) = \mathbf{U}_1(3x, 2x)$ es GOTO-computable.
20. Si $\mathcal{K} = \{n \in \mathbb{N} \mid \varphi_n^{(1)}(n) \downarrow\}$ es el conjunto del problema de la parada, entonces la función f definida por $f(x) = x$ si $x \in \mathcal{K}$ y $f(x) \uparrow$ si $x \notin \mathcal{K}$, **no es** GOTO-computable.
21. Para cualquier tupla de entrada, el programa universal $\mathbf{U}_n$ de orden $n \geq 1$, **simula una computación** de un cierto programa GOTO.
22. El programa universal $\mathbf{U}_n$ de orden n **única**mente **simula** computaciones de otros programas GOTO cuando se le suministra como dato de entrada una n -tupla arbitraria.
23. Sea P un programa con, **exactamente cuatro** instrucciones y tal que su código **coincide** con el de una **configuración de parada** σ de P . Entonces el estado s de la configuración σ es un **estado inicial** asociado a un dato de entrada.
24. Si un programa GOTO carece de instrucciones que estén **etiquetadas a la izquierda** entonces **toda computación** del programa es de **parada**.
25. La función **total** f obtenida por **minimización acotada** a partir de un **predicado no tiene porqué** ser un predicado.
26. La cuantificación existencial de un predicado GOTO-computable **no** tiene por qué ser un predicado GOTO-computable.
27. La cuantificación universal de un predicado GOTO-computable **no** tiene por qué ser un predicado GOTO-computable.
28. La negación de un predicado parcialmente decidible **también** es un predicado parcialmente decidible.
29. La conjunción de dos predicados parcialmente decidible de la misma aridad es, también, un predicado parcialmente decidible.
30. **No existe** un programa GOTO cuyo código coincida exactamente con el código de alguna configuración del propio programa.
31. El código de un **estado inicial** (correspondiente a una configuración inicial) de un programa GOTO **nunca puede** ser un número **par**.
32. Si P es un programa GOTO cuyo código es un número primo distinto de 2, entonces la primera instrucción de P **no puede ser** $Y \leftarrow Y$.
33. Si P es un programa GOTO cuyo código es $1 + 2 + \dots + 2^n$ (para un cierto número natural $n \geq 1$) entonces **no** pueden existir configuraciones del programa cuyo código coincida con el del propio programa.
34. En un programa GOTO, **no puede existir** una **configuración inicial** cuyo código coincida exactamente con el código del propio programa.
35. El código de cualquier **configuración inicial** de un programa GOTO **siempre** es un número **impar**.
36. El código de una **configuración** de **parada** de un programa GOTO **puede** ser un **número primo**.
37. **Todo** número natural codifica un **único** programa GOTO.

38. La configuración de código 5293 es una configuración inicial.
39. Un programa universal **sólo** es capaz de simular **algunas** de sus propias computaciones.
40. Existen conjuntos de números naturales que ni son GOTO-computables ni son recursivamente enumerables.
41. El dominio, el rango y la gráfica de una función GOTO-computable (**parcial** o **total**) son conjuntos recursivamente enumerables.
42. El conjunto $A = \{n \mid \text{el programa de código } 5n \text{ para sobre } 5n\}$ es recursivamente enumerable.
43. El conjunto $A = \{\langle x, y \rangle \in \mathbb{N} \mid \text{el programa de código } x^2 \text{ para sobre } y^3 \text{ en, exactamente, } x^3 + y^2 \text{ pasos}\}$ es recursivamente enumerable.
44. Dos funciones GOTO-computables distintas, pueden tener asociadas el **mismo** índice.
45. El complementario de un conjunto GOTO-computable es, también, un conjunto GOTO-computable.
46. El complementario de un conjunto recursivamente enumerable es, también, un conjunto recursivamente enumerable.
47. Para afirmar que un conjunto **no** es recursivamente enumerable es suficiente probar que **no** es GOTO-computable y que, en cambio, su complementario **sí** es recursivamente enumerable.
48. El teorema de Rice se puede usar para establece que un conjunto **no** es GOTO-computable.
49. El conjunto de índices de **una** función GOTO-computable de aridad 1, es un conjunto GOTO-computable.
50. El conjunto $A = \{n \in \mathbb{N} \mid \text{el programa GOTO de código } 9n \text{ carece de instrucciones del tipo } \text{CONDICIONAL}\}$ es GOTO-computable.
51. El conjunto $A = \{n \in \mathbb{N} \mid \text{el programa GOTO de código } 4n + 8 \text{ posee, a lo sumo, } 10 \text{ instrucciones y alguna de ellas es un } \text{DECREMENTO}\}$ es GOTO-computable.
52. El conjunto $A = \{n \in \mathbb{N} \mid \text{el programa GOTO de código } n \text{ posee alguna instrucción etiquetada a la izquierda}\}$ es GOTO-computable.
53. El conjunto $A = \{n \in \mathbb{N} \mid \text{dom}(\varphi_n^{(1)}) \text{ es un conjunto finito}\}$ **no** es GOTO-computable.
54. Las computaciones de una MTD con un dato de entrada realizan, al menos, un paso de transición (o computación)
55. Todas las computaciones de una MTD cuyo alfabeto de trabajo sea $\{0, 1, B, \triangleright\}$ y calcula la **función vacía**, ha de realizar, al menos, 2222 pasos de transición.
56. Si M es una MTND de decisión y el coste en tiempo verifica $t_M(2) = 345$ entonces, ejecutando 345 pasos de cualquier computación de M sobre un dato de entrada de tamaño 2, sabremos **con toda seguridad** si aceptamos o no dicho dato.
57. La familia $\{\varphi_e^{(1)} \mid e \in \mathbb{N}\}$ de todas las funciones computables 1-arias sobre $\mathbb{N}$, es una medida de complejidad.
58. Si X_1 y X_2 son problemas **NP**-completos tales que $X_1 \in \mathbf{P}$ entonces $X_2 \in \mathbf{P}$.
59. Si X_1 y X_2 son problemas de la clase **P**, entonces se tiene que $X_1 \leq^P X_2$ o bien que $X_2 \leq^P X_1$.
60. Si X_1 y X_2 son problemas **NP**-completos entonces **cualquiera de ellos** es *reducible en tiempo polinomial* al otro.