Fundamentos de Teoría de la Computación 2021
Examen Parte 1 Fecha 1 - Lunes 3 de mayo

Instrucciones

a) El examen es INDIVIDUAL, no nos defraudes.
b) Ni bien recibas el examen enviame un email con tu nombre, DNI y legajo, confirmándome que lo recibiste.
c) Tenés 2 horas para responder el máximo de preguntas que puedas, no te detengas mucho en ninguna, y respondé claro y breve. Hay que responder preguntas de las 2 partes, Computabilidad y Complejidad Computacional. Respondé sobre el email o el Word, debajo de cada pregunta.
d) Cuando termines, enviame el examen resuelto por email, repitiendo en el asunto tu nombre, DNI y legajo. Cuando yo lo reciba te enviaré un ok.

¡Mucha suerte!

Parte 1. Computabilidad.

- Vimos en clase que el lenguaje SAT, que representa el problema que plantea si una fórmula booleana sin cuantificadores es satisfactible, es recursivo. ¿Qué significa que el lenguaje SAT es recursivo?
- ¿Por qué el complemento de un lenguaje de (RE – R) no puede pertenecer a RE?
- Describir la función de transición de una Máquina de Turing que, recibiendo sólo cadenas de 1 y 0, las transforma cambiando 1 por 0 y 0 por 1.
- Describir la función de transición de un Autómata Finito con estado inicial q_0 y conjunto de estados finales $\{q_F\}$ que, recibiendo sólo cadenas de 1 y 0, acepta las que tengan al menos una vez dos 0 consecutivos.
- Comentar la idea general de cómo se puede transformar una Máquina de Turing en otra equivalente que no modifique el input.
- Vimos en clase el lenguaje QBF = {φ | φ es una fórmula booleana con cuantificadores (todos al comienzo), cerrada (es decir que no tiene variables libres) y verdadera}. Por ejemplo, $\varphi_1 = \forall x \forall y \exists z\ (x \vee y \vee z) \in QBF$, y en cambio $\varphi_2 = \forall x \forall y\ (x \wedge y) \notin QBF$. Explicar por qué la siguiente Máquina de Turing no determinística no acepta QBF: dado un input φ, si no es correcto sintácticamente rechaza, y si lo es genera no determinísticamente una asignación A de valores de verdad para todas las variables de φ y luego en cada computación evalúa φ con A sin considerar los cuantificadores, aceptando sii la evaluación resulta verdadera. Ayuda: ver por ejemplo qué sucede con la fórmula $\forall x \forall y\ (x \wedge y)$.
- Sean L_1 un lenguaje recursivamente numerable y L_2 un lenguaje recursivo. Describir la idea general de cómo trabajaría una Máquina de Turing que acepte $L_1 \cup L_2$.
- Explicar por qué la siguiente Máquina de Turing no acepta el lenguaje L = {<M> | L(M) ≠ ∅}: la máquina genera una cadena w, ejecuta M sobre w, y si acepta entonces acepta; si no, genera otra cadena w, ejecuta M sobre w, y si acepta entonces acepta; y así sucesivamente.
- Dados k lenguajes recursivamente numerables $L_1, L_2, \ldots, L_k$, disjuntos dos a dos y cuya unión es $\sum^*$, describir la idea general de cómo trabajaría una Máquina de Turing que acepte L_1 y pare siempre. Ayuda: M podría ejecutar "en paralelo" las Máquinas de Turing que aceptan $L_1, L_2, \ldots, L_k$.
- Explicar cómo se puede detectar si una Máquina de Turing entra en loop cuando se mueve en un espacio limitado de celdas.
- Supongamos que una Máquina de Turing M genera un lenguaje L, es decir, imprime en una cinta, una a una, todas las cadenas de L. Explicar la idea general de cómo trabajaría una Máquina de Turing M´ que acepte el lenguaje L, es decir que dada una cadena w la acepte sii w ∈ L. Ayuda: M´ debería invocar a M.
- ¿Por qué no puede existir una reducción del lenguaje HP^C al lenguaje HP?
- Vimos en clase que se puede reducir el lenguaje HP al lenguaje L_u por medio de una función que primero valida la sintaxis del input y luego, si es correcto, es decir si tiene la forma (<M>,w), lo transforma cambiando los estados q_R de <M> por q_A, y manteniendo w. Explicar por qué esta función es total computable.
- Turing demostró que la lógica de predicados de primer orden no es recursiva, encontrando una reducción del halting problem (lenguaje HP) a dicha lógica. Si la misma fuese recursiva, también lo sería HP. ¿Por qué?
- Sea L = $\{w_i \mid M_i \text{ acepta } w_i\}$. Plantear la idea general de una reducción de L a L_U. Ayuda: Usar la definición del lenguaje L_U.

Parte 2. Complejidad Computacional.

- Probar que todo lenguaje finito pertenece a la clase P.
- Sean f una función computable en tiempo determinístico polinomial y A un lenguaje de la clase P. Probar que el lenguaje B = {w | f(w) ∈ A} también pertenece a P.
- Probar que L = {(<M>, w, 1^k) | M es una Máquina de Turing determinística con una sola cinta que acepta w en a lo sumo k pasos} ∈ P.

- Analizamos en clase el problema que plantea si un número N tiene un divisor que termina en 3. El algoritmo propuesto hacía O(N) iteraciones. ¿Esto significa que el problema está en P? Justificar la respuesta.
- Probar que $P \subseteq NP \cap CO\text{-}NP$.
- Sea el lenguaje SI-NO-SAT = {φ | φ es una fórmula booleana sin cuantificadores, satisfactible al menos por una asignación e insatisfactible al menos por una asignación}. Probar que SI-NO-SAT ∈ NP. Ayuda: construir una Máquina de Turing no determinística que trabaje en tiempo poly(n), y recordar que la evaluación de una fórmula booleana con una asignación de valores de verdad consume tiempo determinístico polinomial.
- ¿Qué significa que todo elemento de un lenguaje de la clase NP cuenta con un certificado suscinto? Dar un ejemplo.
- El problema VAL plantea si una formula booleana sin cuantificadores es válida, es decir, si cualquier asignación de valores de verdad la satisfice. Comentar por qué VAL no estaría en la clase NP.
- Se define que un lenguaje L es P-completo si está en P y además todos los lenguajes de P se reducen polinomialmente a L. Probar que el lenguaje {0} es P-completo. Ayuda: (a) La prueba de que {0} está en P es trivial. (b) También es sencillo construir una reducción polinomial de cualquier lenguaje L de P al lenguaje {0}: la función de reducción debería chequear si el input está o no en L y en base a ello generar un output adecuado.
- Probar, dados dos lenguajes cualesquiera NP-completos, que cada uno se reduce polinomialmente al otro.
- Vimos en clase el lenguaje TSP que representa el problema del viajante de comercio. Se prueba que TSP es NP-completo. Explicar por qué, si existe una reducción polinomial de TSP a un lenguaje L de la clase NP, entonces L también es NP-completo.
- El problema FSAT plantea encontrar una asignación de valores de verdad que satisfaga una fórmula booleana sin cuantificadores (si no existe ninguna el output es “no”). Suponiendo que una Máquina de Turing M resuelve FSAT en tiempo poly(n), se prueba fácilmente que SAT ∈ P. Comentar la idea general de la prueba. Ayuda: la Máquina de Turing que decida SAT en tiempo poly(n) debería invocar a M.
- Volviendo al lenguaje QBF: se indicó en clase que pertenece a la clase PSPACE, ¿Qué significa que QBF pertenece a PSPACE?
- ¿Cuál es la clase espacial que se considera como clase de problemas de resolución eficiente en el marco de la complejidad espacial? Justificar la respuesta.
- Explicar por qué la clase temporal P está incluida en la clase espacial PSPACE.