IIC2213 — Lógica para Ciencia de la Computación

Semana 10: complejidad computacional

Teorema de Cook-Levin

La meta es demostrar lo siguiente (es lo que nos falta para mostrar que SAT es NP-completo):

Teorema. SAT es NP-hard

Sea L un lenguaje cualquiera en NP. Tenemos que demostrar que hay una reducción polinomial desde L a SAT. Para la demostración, necesitamos definir una función f tal que para toda palabra $w \in \{0,1\}^*$ se tiene que $w \in L$ si y solo si $f(w)$ pertenece a SAT. Para no lidiar con codificaciones binarias, vamos a asumir que nuestra función construye directamente una fórmula φ_w a partir de cada palabra w, es decir, $f(w) = \varphi_w$.

La demostración funciona de la siguiente forma.

- Usamos lo único que sabemos de L: como L está en NP, entonces existe una máquina de turing no determinista M y un entero k tal que $t_M(n)$ es una función en $O(n^k)$.
- La función f crea una fórmula φ_w que representa en funcionamiento de M con input w.

Vamos a utilizar algunas suposiciones sobre M y w, sin pérdida de generalidad, que nos van a permitir trabajar más fácilmente. Asumimos que $M = (Q, \{0,1\}, q_0, \delta, F)$, y $w = a_0, a_1, \ldots, a_{n-1}$, con cada $a_i \in \{0,1\}$, y donde:

- $F = \{q_f\}$ y δ no tiene transiciones definidas para q_f.
- Para cada estado $q \in Q \setminus \{q_f\}$ y símbolo $a \in \{0,1,B\}$, δ tiene al menos una transición definida para (q, a).
- Cuando la máquina acepta, se demora siempre $t_M(n)$ pasos.

Por simplicidad, digamos que la i-ésima celda de la máquina, para $i \geq 0$, corresponde a i celdas más a la derecha de la celda en la que partió la cabeza de la máquina, y para $i < 0$ corresponde a i celdas más a la izquierda de la celda en que partió la cabeza de la máquina.

Intuitivamente, la fórmula a construir se encargará de *adivinar* una ejecución de la máquina que termine en un estado final.

Conjunto de variables proposicionales. Tu primer ejercicio será definir un conjunto de variables proposicionales que te permitan saber, en cualquiera de los $t_M(n)$ pasos que demora la ejecución de la máquina:

- Qué símbolo hay en la i-ésima celda de la cinta lectora de M en el paso j de la ejecución, para $i \in [-t_M(n), t_M(n)]$ y $j \in [0, t_M(n)]$).
- En qué estado está la máquina en el paso j de la ejecución.

- Dónde está la cabeza de la máquina en el paso j de la ejecucion.

Con estas varables tenemos todo el espectro de lo que podría modificar la máquina M en una ejecución aceptando a w. ¿Puedes ver por qué la máquina nunca va a escribir en una celda a más de $t_M(n)$ celdas de donde partió la cabeza lectora?

Estado Inicial y Final. Define φ_F como una fórmula que se hace verdad cuando el estado de la máquina en el paso $t_M(n)$ es q_f. Con la misma idea, define una fórmula φ_S que se haga verdad cuando las celdas, el estado y la cabeza representan lo que había antes de empezar la ejecución de M en w (es decir, el paso 0).

Correcta configuración de las transiciones. Hasta el momento, tenemos que toda valuación que hace verdad a $\varphi_S \wedge \varphi_F$ representa una ejecución de M que parte bien y termina en un estado final. Pero nos falta especificar que todos los pasos intermedios son válidos! Define φ_C como la conjunción de las siguientes cuatro fórmulas

- Una fórmula que obligue a que siempre haya un solo símbolo en cada celda en cada paso,
- Una fórmula que obligue a que la máquina esté siempre en un solo estado en cada paso,
- Una fórmula que obligue a que la cabeza de la máquina esté siempre en una sola posición en cada paso, y
- Una fórmula que indique que el único valor que cambia entre el paso j y el $j+1$ es el que está apuntado por la cabeza lectora en j.

Ya estamos casi listos. Ahora $\varphi_S \wedge \varphi_F \wedge \varphi_C$ son satisfechas por valuaciones en las que se parte bien, cada transición de un paso a otro se hace según las reglas de la máquina de turing, y se termina en un estado final.

- Nota que todavía no es verdad que M acepta a w si y solo si $\varphi_S \wedge \varphi_F \wedge \varphi_C$ es satisfacible.
- Muestra como construir una fórmula restante φ_T, de manera que se cumpla que M acepte a w si y solo si $\varphi_w = \varphi_S \wedge \varphi_F \wedge \varphi_C \wedge \varphi_T$ es satisfacible.

Ejercicio adicional: 3CNFSAT

Acabamos de mostrar que SAT es NP-hard. Sin embargo, para muchas reducciones (como la de vertex cover que vimos en clases) usamos una versión de SAT que llamamos 3CNFSAT, o simplemente 3SAT, que corresponde al siguiente lenguaje:

$$\begin{gathered}\text{3SAT} = \{\varphi \mid \varphi \text{ es una fórmula en CNF} \\ \text{con exactamente tres literales por cláusula y } \varphi \text{ es satisfacible}\}.\end{gathered}$$

Notar que no podemos establecer una reducción directa desde SAT a 3SAT, por que al transformar una fórmula ψ en otra en CNF nos puede quedar una fórmula exponencialmente más grande que ψ.

Tu deber es demostrar que 3SAT es NP-hard. No puedes asumir que ningún problema es NP-hard, todo lo que muestres debe ser via una adaptación del Teorema de Cook-Levin.