Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2213 — Lógica para Ciencia de la Computación — 2' 2022
Alumno: Diego Iruretagoyena

Tarea 2

Considerando alfabeto

$$A = \{0, 1\}$$

tenemos

$$L_{0y00} = \{w \in \{0,1\}^* | \ \exists \text{ MTM tq. } w = C(M) \wedge \text{M acepta al string 0} \wedge \text{M acepta al string 00 } \}$$

$$L_{0ssi00} = \{w \in \{0,1\}^* | \ \exists \text{ MTM tq. } w = C(M) \wedge \text{M acepta al string 0 si y solo si M acepta al string 00 } \}$$

Parte A: Muestra que ambos lenguajes son indecidibles

Para resolver esta pregunta, enunciamos algunas deficiones en las que basaremos nuestra respuesta

Definición 1: Decibilidad

Un lenguaje L sobre un alfabeto A es decidible si

- $\exists$ máquina de turing M tal que $L(M) = L$, es decir, el lenguaje de la maquina es precisamente L.
- M para en todas las entradas.

Si L no es decidible, entonces decimos que L es indecidible. Además, enunciamos definición de reducción

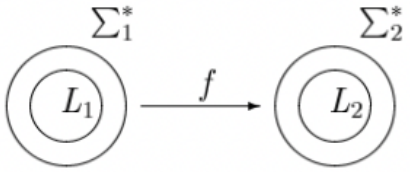
Definición 2: Reducción

Dados lenguajes L1 y L2 sobre un alfabeto A, decimos que $L1$ es Turing-reducible a $L2$, o igualmente que existe una reducción (de Turing) de $L1$ a $L2$, si existe una función computable f tal que para todo $w \in A, w \in L1$ si y solo si $f(w) \in L2$. El concepto de reduccion captura la nocion de que L2 es al menos tan dificil como L1, o que, equivalentemente, L1 no es mas dificil que L2. Esto lo podemos formalizar como:

Teorema 1: Si L1 es reducible a L2, entonces:

- Si L2 es decidible, entonces L1 es decidible.
- Si L1 es indecidible, entonces L2 es indecidible

Sean $L_1 \subseteq \Sigma_1^\star$ y $L_2 \subseteq \Sigma_2^\star$. Decimos que L_1 es **reducible** a L_2, y escribimos $L_1 \leq L_2$, si existe una función f de $\Sigma_1^\star$ en $\Sigma_2^\star$



tal, que:

- para todo $w \in \Sigma_1^* : w \in L_1 \Leftrightarrow f(w) \in L_2$
- $f : \sum_1^* \to \sum_2^*$ es computable

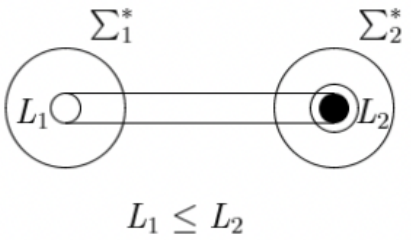


Figure 1: Reducibilidad - Dr. Leopoldo Bertossi

La demostración de este teorema la vimos en clases y está en el material del curso, semana 3. Entonces, necesitamos tomar un L1 que sepamos indecidible, demostrar que L0y00 es al menos tan dificil como L1, y generar una reducción.

Entonces, buscamos una función computable $f(w)$ tal que para todo $w \in A, w \in L1$ ssi $f(w) \in L2$.

...función computable?

Definición 3.
Sea A un alfabeto, y $f : A \to A$ una funcion que transforma algunas palabras en otras. Decimos que f es computable si existe una máquina de turing M con un unico estado final tal que para cada $w \in A$ la maquina se detiene en el estado final, y donde la cinta de trabajo consiste en una secuencia infinita de blancos, seguidos de $f(w)$, seguidos de otra secuencia infinita de blancos.

Algo importante de mencionar es que L_{0y00} no acepta ni a 0 ni a 00, si no que L_{0y00} acepta a las codificaciones de máquinas de turing $C(M)$ tales que M acepta al 0 y al 00. Es decir, si una máquina acepta

el 0, pero rechaza el 00, no pertenece a este lenguaje. Y si acepta el 00 pero rechaza el 0, no pertenece al lenguaje. Si acepta ambos strings, pertenece al lenguaje, y si rechaza ambos strings, pertenece al lenguaje. Ahora, con "aceptar ambos strings" no se refiere a aceptar al mismo tiempo, si no que recibe un único string y decide si aceptar o rechazar. Entonces, L_{0y00} acepta algunas máquinas y rechaza otras, no es que reciba strings 0s o 00s.

Ahora que ya tenemos todo lo necesario, pensamos en cómo demostrar que estos lenguajes son indecidibles. Las técnicas usuales suelen ser 1. por contradicción, 2. por reducción o 3. por Rice. Propongo generar la reducción desde un lenguaje que sepamos indecidible hacia L_{0y00}. No podemos usar una máquina de Turing universal ya que podría nunca retornar, y no sabríamos qué hacer. Como vimos en clases, lograr una función computable desde un lenguaje L1 a otro L2, sabiendo L1 indecidible, es suficiente para demostrar que L2 es indecidible.

Entonces, como queremos realizar una reducción, nos faltan dos ingredientes. Uno, elegir un lenguaje indecidible para reducir desde. El otro, una función computable. Como lenguaje indecidible utilizaremos el clásico problema de **HALTING**.

(Ahora si última) Definición: HALTING El problema de la parada o problema de la detención para máquinas de Turing consiste en lo siguiente: dada una Máquina de Turing M y una palabra w, determinar si M terminará en un número finito de pasos cuando es ejecutada usando w como dato de entrada. Alan Turing, en su famoso artículo ≪On Computable Numbers, with an Application to the Entscheidungsproblem≫ (1936), demostró que el problema de la parada de la Máquina de Turing es indecidible (no computable o no recursivo), en el sentido de que ninguna máquina de Turing lo puede resolver.

HALTING = $\{w \in \{0,1\} | w = u0000v$, existe una maquina M tal que u = C(M) y M se detiene con entrada $v\}$

Dada una máquina M y una palabra v, vemos que el problema de Halting es equivalente a decidir si C(M)0000v pertenece a H. Este es el marco de trabajo en el que vamos a apoyarnos para mostrar que Halting es indecidible. **Ahora, para efectos prácticos, asumiremos que ya sabemos que HALTING es indecidible.** Dadas las características de Halting, en donde buscamos saber si la máquina se detiene con determinado input, suena natural utilizarlo para la reducción. Podríamos incluso asumir que $L0y00$ es decidible y mostrar como esto nos lleva a una contradicción.

Utilicemos una función f(w) de reducción desde HALTING hacia $L0y00$. Nuestra función toma un par $< M, w >$ con $w \in \{0,1\}$ y una Maquina M. Construimos una máquina M' que tome cualquier palabra p, la ignora, y corre M (original) sobre w. Si M rechaza, rechazamos o quedamos en un loop infinito, y si M acepta, aceptamos. Esta máquina siempre responde y te da la respuesta correcta. Ahora, sabemos que esto es similar al problema de entrada de HALTING, por lo que sabemos que no puede existir.

Por intuición f(w) es computable dada la forma en que la construimos. Por ende hemos logramos crear un mapeo, sabemos que $w \in HALT \leftrightarrow f(w) \in L0y00$. **Como sabemos que HALTING es indecidible, entonces $L0y00$ también lo es, siendo halting al menos tan difícil como $L0y00$.** La implicancia también puede demostrarse para cada lado. Si el par w, M pertenecen a Halt, entonces sabemos que nuestra función f respondera que si, y llegaremos a un estado final. Para la izquierda, si tenemos un estado final para M', significa que f(w) retornó verdadero, es decir terminó, por lo que sabemos que el par pertenece a Halt. **Traducir esto a $L0ssi00$ es intuitivo**, en donde podemos cambiar las condiciones del problema para representar la relación "si solo si" y así demostrar, nuevamente, que es un lenguaje indecidible y reducible desde HALT.

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2213 — Lógica para Ciencia de la Computación — 2' 2022
Alumno: Diego Iruretagoyena

Tarea 2

Parte B: Muestra que L0y00 es recursivamente enumerable

Definición. Un lenguaje L sobre un alfabeto A es recursivamente enumerable si existe una máquina de turing M tal que L(M) = L, es decir, el lenguaje de la máquina es precisamente L.

Recordamos ciertas propiedades.

- Si L1 y L2 son decidibles, entonces L1 ∪ L2 es decidibles.
- Si L1 y L2 son decidibles, entonces L1 ∩ L2 es decidibles.
- Si L1 es decidible, entonces L 1 es decidible.

Algo importante de mencionar es que L_{0y00} no acepta ni a 0 ni a 00, si no que L_{0y00} acepta a las codificaciones de máquinas de turing $C(M)$ tales que M acepta al 0 y al 00. Es decir, si una máquina acepta el 0, pero rechaza el 00, no pertenece a este lenguaje. Y si acepta el 00 pero rechaza el 0, no pertenece al lenguaje. Si acepta ambos strings, pertenece al lenguaje, y si rechaza ambos strings, pertenece al lenguaje. Ahora, con "aceptar ambos strings" no se refiere a aceptar al mismo tiempo, si no que recibe un único string y decide si aceptar o rechazar. Entonces, L_{0y00} acepta algunas máquinas y rechaza otras, no es que reciba strings 0s o 00s.

Ahora que ya tenemos lo necesario, demostraremos que L_{0y00} es RE. Aquí usaremos una máquina de turing universal, ya que no nos interesa que pasa cuando no acepta. La máquina de turing nos permite simular una máquina. Lo que podemos usar aqui es paralelización de comportamientos. Es decir, separaremos en M_0 = M acepta al string 0 y M_{00} = M acepta al string 00.

Creamos una MT $M_{universal}$ que paralelice ambas ejecuciones de las maquinas al utilizar los indices pares de su cinta para M_0 e impares para M_{00}. Además, se pueden utilizar indicadores de posición tipo escribir palabra vacía para lograr retornar correctamente a la ejecución previa. Nuestra máquina universal puede ejecutar "en paralelo" a pesar de ser realmente en serie. Una máquina universal tiene como input un C(M)0000w, para $w \in A$, entonces U acepta al input si y solo si M acepta a w. Si su input no es de la forma C(M)0000w, entonces U no acepta al input.

Sean Q_{f0} y Q_{f00} los estados finales de cada máquina y Q_{f0y00} la intersección de esos conjuntos. Sabemos que la intersección de ambos estados representa la correcta ejecución de ambas. Es decir, si ambas llegan a un estado final (aceptan) entonces podemos aceptar en $M_{universal}$. Con esto logramos crear una MT que represente el lenguaje.

Entonces, $L(M_{universal}) = L_{0y00}$ **por lo que existe una Máquina de Turing que logra representar el lenguaje**, y $L_{0y00} \in RE$. En esta misma linea, es fácil ver que L_{0ssi00} no es RE ya que no podemos encontrar un lenguaje L(M') = L_{0ssi00}. Es decir, no tenemos una MT que con dicho lenguaje ya que no sabremos cuando retornar y quedará un loop infinito y por eso L_{0ssi00} no es Recursivamente Enumerable. Dadas las características del problema, en donde no podemos aceptar el input hasta saber si la máquina acepta 00, no podemos encontrar un lenguaje de máquina.