UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

INGENIERÍA EN COMPUTACIÓN

LENGUAJES FORMALES Y AUTÓMATAS

INVESTIGACIÓN

PROFESORA: JOSEFINA ROSALES GARCÍA

ALUMNOS: Cárdenas Cárdenas Jorge
Murrieta Villegas Alfonso

SEMESTRE: 2019-2

FECHA:

ÍNDICE

▸ Indecibilidad

En la teoría de la computación y en teoría de la complejidad computacional, un problema indecidible es un problema de decisión para el cual es imposible construir un algoritmo que siempre conduzca a una respuesta de sí o no correcta.

Un problema de decisión es cualquier pregunta arbitraria de sí o no en un conjunto infinito de entradas.
Por ello es tradicional definir el problema de decisión como equivalente al conjunto de entradas para las que el problema retorna sí.

Nota: Estas entradas pueden ser números naturales, o bien valores de otro tipo, tales como cadenas de un lenguaje formal.

Un problema de decisión es un subconjunto de los números naturales. El problema informal correspondiente consiste en decidir si un número dado está en el conjunto.
A un problema de decisión A, si A es un conjunto recursivo, se le denomina decidible o efectivamente solucionable.
Si A es un conjunto recursivamente enumerable, el problema es parcialmente decidible, semidecidible, solucionable o demostrable.
A problemas parcialmente decidibles y a los no decidibles se les califica de indecidibles.

▸ Lenguajes recursivos y recursivos enumerables

En matemáticas y ciencias de la computación a un lenguaje formal donde es un subconjunto recursivo del conjunto de todas las secuencias finitas posibles sobre el alfabeto del lenguaje.
Por lo tanto, un lenguaje formal es recursivo si existe una máquina de Turing que siempre se detiene cuando dada una secuencia finita de símbolos del alfabeto del lenguaje (Ya sea cadena de carácteres o palabra) como entrada, acepta solo esas palabras que son parte del lenguaje y rechaza todas las otras palabras.
Los lenguajes recursivos también se denominan lenguajes decidibles.
La clase de todos los lenguajes recursivos es a menudo llamada R, aunque este nombre también es usado para la clase RP

Este tipo de lenguaje no estaba definido en la jerarquía de Chomsky. Todos los lenguajes recursivos son también recursivamente enumerables.
Todos los lenguaje regulares, libres de contexto, y sensibles al contexto son lenguajes recursivos.

▸ Lenguaje Recursivamente Enumerable

Son un tipo de lenguaje formal que es también llamado parcialmente decidible o Turing-Computable, son conocidos como lenguajes tipo 0 en la jerarquía de Chomsky.

Definición | Es aquel para el cual existe una maquina de Turing que acepta y se detiene con cualquier cadena del lenguaje.

Algunas propiedades de estas son:

- El cierre o cerradura estrella L^* de L
- La concatenación $L \circ P$ de L y P
- La unión $L \cup P$ de L y P
- La intersección $L \cap P$ de L y P

Nota: Los lenguajes recursivamente enumarables no son cerrados con la diferencia ni el complementario

- $L \setminus P$ puede no ser recursivamente enumerable
- $\bar{L}$ es recursivamente enumerable si y solo si L es también recursivo.

7.3] Tesis de Church-Turing

Dentro de la Teoría de la Computabilidad, la tesis de Church-Turing fórmula hipotéticamente la equivalencia entre los conceptos de función computable y maquina de Turing, lo cual se refiere a la equivalencia de un algoritmo de la maquina de Turing.

Cabe destacar que esto no es una afirmación matemática formal sino una aceptación prácticamente universal.

- Previo

Como se mencionó anteriormente, los lenguajes formales que son aceptados por una maquina de Turing son todos aquellos que pueden ser generados por una gramática formal, además, pueden ser computados por con el cálculo Lambda de Church son exactamente aquellas que pueden ser computadas por la maquina de Turing.

-Cálculo Lambda

El cálculo lambda es un sistema formal diseñado para investigar la definición de función, la notación de aplicaciones/funciones y la recursión.

Fue introducido por Alonzo Church y Stephen Kleene en 1930. Church usó el cálculo lambda en 1936 para resolver el Entscheidungsproblem, y además sirve para definir que es una función computable.

- Contenido empírico de Teorema de Normalizacion

I] Sea Δ una clase de funciones diseñada con el fin de capturar sólo funciones numéricas efectivamente calculables

"Δ es correctamente en relación al concepto informal de función efectivamente calculable"

II] Supongamos que se ha demostrado que la clase Γ por todas las funciones Turing-computables guarda con Δ la siguiente relación

$$\Gamma \subseteq \Delta$$

Entonces

Existe una demostración de la afirmación según la cual $\Delta \subseteq \Gamma$

▶Teorema de Rice

En la teria de la computación, el teorema de Rice es un teorema enonciado por Henry Gordon Rice y luegoo generalizado por John Myhill y Norman Shapiro donde menciona que:

"Dadu una propiedad no trivial de las funciones parciales, no es computable determinar si una función arbitraria la posee o no"

Formalmente se menciona que:

Sea S un conjunto de lenguajes no triaviales, es decir:

1] existe una maquina de Turing que reconoce un lenguage en S

2] Existe una maquina de Turing que reconoce un lenguage ni en S

Entonces, es indecidible determinar si un lenguaje decidido por una máquina de Turing arbitraria se encuentra en S.

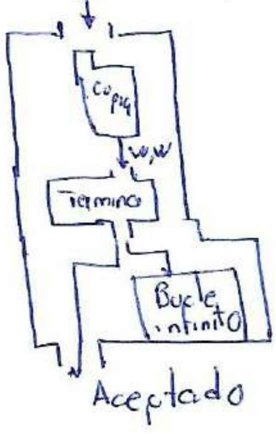
Nota: El teorema de Rice no dice nada acerca de las propiedades de las maquinas o programas que no son propiedades de las funciones y los lenguajes. Por ejemplo si una maquina tiene más de 5 estados es una propiedad decidible de la maquina, ya que el número de pasos puede ser contado simplemente

▶ Problemas Indecidibles

A continuación se muestran algunos ejemplos de problemas indecidibles;

1] Problema de la parada

Dada una Máquina de Turing M y una palabra w, determinar si M terminará en un número finitos de pasos cuando es ejecutada usando w como dato de entrada



En el artículo " On computable numbers, with an application to the Entscheidungsproblem", Alan Turing demostró que no es posible y por ello es un problema indecidible.

2] Problema de la matriz mortal

Determinar, dado un conjunto finito de nxn matrices con entradas enteras, si se pueden multiplicar en algún orden, posiblemente con repetición para obtener la matriz cero.

- En este caso se vuelve indecidible para un conjunto de 6 o más matrices 3 x 3 o un conjunto de 2 matrices de 15 x 15.

PROBLEMA DE CORRESPONDENCIA DE POST

Una instancia del Problema de la Correspondencia de Post (PCP) consiste de dos listas de cadenas, $A = w_1, \dots w_K$ y $B = x_1, \dots x_n$, sobre algún alfabeto Σ. Esta instancia del PCP tiene una solución si existe una sucesión de enteros $i_1, i_2, \dots i_m$, con $m \geq 1$ tal que:

$$w_{i_1}, w_{i_2}, \dots w_{i_m} = x_{i_1}, x_{i_2}, \dots x_{i_m}$$

Ejemplo: Sea $\Sigma = \{0, 1\}$. Sean A y B listas de tres cadenas cada una, tal como se definen a continuación:

	Lista A	Lista B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

En este caso, el PCP tiene una solución. Sea $m = 4$, $i_1 = 2$, $i_2 = 1$, $i_3 = 1$ e $i_4 = 3$. Entonces

$$w_2 w_1 w_1 w_3 = x_2 x_1 x_1 x_3 = 101111110$$

A continuación se probará que el PCP es indecidible, mostrando que si fuera decidible, entonces se tendría un algoritmo L_u. Primero se probará que si el PCP fuese decidible, entonces una versión modificada del PCP sería también decidible. El Problema de correspondencia de Post Modificado (PCPM) es el siguiente:

Dadas las listas A y B, de K cadena cada una, que estan en Σ^*, por decir,

$$A = w_1, w_2, \dots w_K \text{ y } B = x_1, x_2, \dots, x_K,$$

¿existe una sucesion de enteros, $i_1, i_2, \dots, i_r$, tal que

$$w_1 w_{i_1} \dots w_{i_r} = x_1 x_{i_1} x_{i_2} \dots x_{i_r}?$$

la diferencia entre el PCPM y el PCP es que el PCPM requiere una solución para empezar con la primer cadena de la lista.

Si el PCP fuera indecidible, entonces el PCPM sería indecidible. Esto es, el PCPM se puede reducir al PCP.

Demostración: Sean

$$A = w_1, w_2, \dots, w_K \quad y \quad B = x_1, x_2, \dots x_n$$

una instancia del PCPM. Esta instancia del PCPM se puede convertir a una instancia del PCP que tenga solución si y sólo si la instancia del PCPM tiene solución. Si el PCP fuera decidible, entonces se podría resolver el PCPM, lo que probaría el lema.

Sea Σ el alfabeto más pequeño que contiene todos los símbolos de las listas A y B y sean ¢ y \$ símbolos que no estan en Σ. Sea y_i la cadena que se forma a partir de w_i al insertar el símbolo ¢ después de cada caracter de w_i y sea z_i la cadena arbitraria partir de x_i al insertar el símbolo ¢ adelante de cada caracter x_i. Creandose las nuevas palabras:

$$y_0 = ¢\, y_1 \quad , \quad z_0 = z_1$$
$$y_{K+1} = \$ \qquad z_{K+1} = ¢\,\$$$

Sean $C = y_0, y_1, \dots y_{K+1}$ y $D = z_0, z_1, \dots z_{K+1}$
las listas C y D representan una instancia del PCP. Se afirma que esta instancia del PCP tiene una solución si y sólo si la instancia del PCPM representada por las listas A y B tiene solución. Para ver esto, nótese que si $1, i_1, i_2, \dots i_r$ es una solución al PCPM con listas A y B, entonces

$$0, i_1, i_2, \dots, i_r, K+1$$

es una solución al PCP con listas C y D.

Si existe un algoritmo para decidir el PCP, se puede construir un algoritmo para decidir el PCPM convirtiendo cualquier instancia del PCPM a una instancia del PCP como se hizo previamente.

Basta mostrar que si el PCPM fuera decidible, entonces se podría decidir si una MT acepta una palabra dada. Esto es, Lu se reduce al PCPM, el cual, como se vio anteriormente se convierte al PCP. Para cada M y w se construye una instancia del PCPM que tiene una solución si y sólo si M acepta a w.

Esto se hace construyendo una instancia del PCPM que, si tiene una solución, dicha solución comienza con $\# q_0 w \# \alpha_1 q_1 \beta_1 \# \ldots \# \alpha_k q_k \beta_k \#$, donde las cadenas entre #'s consecutivos son descripciones instantaneas consecutivas en un cálculo de M con entrada w, y q_k es un estado final.

DEMOSTRACION DE LA IRRESOLUBILIDAD DEL PCPM

Supongamos que el PCPM es resoluble, que existe un algoritmo que determina si tiene solución un PCPM cualquiera. Pretendemos demostrar que hay un algoritmo que puede determinar si una maquina de Turing arbitraria M parara con una entrada w.

Cualquier maquina de Turing puede convertirse en una maquina que solo para en un estado de aceptación, haciendo que si para en algún estado de no aceptación se introduzca un bucle o algo así. Esta modificación no afecta al lenguaje que acepta la maquina.

Sea $M = (Q, \Sigma, \Gamma, s, b, F, \delta)$ y una cadena w sobre Σ. Construiremos una muestra del PCPM para el que la determinación de si tiene solución (que suponemos que podemos hacer) sea equivalente a determinar si M para sobre la entrada w.

El razonamiento realizado, pues, tiene el siguiente esquema:

1. Demostramos que el problema de la parada es irresoluble.
2. Argumentamos que si el PCP es resoluble $\Rightarrow$ el PCPM es resoluble.
3. Asimismo, si el PCMP es resoluble $\Rightarrow$ el problema de la parada tambien lo sería
4. Pero el problema de la parada es irresoluble, de modo que el PCPM se deduce irresoluble.
5. Y puesto que el PCPM es irresoluble $\Rightarrow$ el PCP es irresoluble.

El PCP se puede usar para demostrar que una gran variedad de problemas son indecidibles. A continuacion, se da sólo una aplicación: la indecibilidad de la ambiguedad para gramáticas libres de contexto.

Teorema: Es indecidible saber si una GLC arbitraria es ambigua.

Demostración: Sean

$$A = w_1, w_2, \cdots w_n \text{ y } B = x_1, x_2, x_{\cdots} \cdots x_n$$

dos listas de palabras

$$L_A = \{ w_{i_1} w_{i_2} \cdots w_{i_m} a_{i_m} a_{i_{m-1}} \cdots a_{i_1} \mid m \geq 1 \}$$

sobre un alfabeto finito Σ_1. Sean $a_1, a_2, a_3, \cdots, a_n$ nuevos simbolos. Sean

$$L_A = \{ w_{i_1} w_{i_2} \cdots w_{i_m} a_{i_m} a_{i_{m-1}} \cdots a_{i_1} \mid m \geq 1 \}$$

y

$$L_B = \{ x_{i_1} x_{i_2} \cdots x_{i_m} a_{i_m} a_{i_{m-1}} \cdots a_{i_1} \mid m \geq 1 \}$$

Sea G la GLC

$$(\{S, S_1, S_2\}, \Sigma \cup \{a_1, \ldots a_n\}, P, S),$$

donde P contiene las producciones $S \to S_A$, $S \to S_B$ y para $1 \leq i \leq n$, $S_A \to w_i S_A a_i$, $S_A \to w_i a_i$, $S_B \to x_i S_B a_i$, $S_B \to x_i a_i$. La gramatica genera $L_A \cup L_B$.

Si la instancia (A, B) del PCP tiene una solución, por decir, $i_1, i_2, \ldots i_m$, entonces existe una palabra $x_{i_1} x_{i_2} \cdots x_{i_m} a_{i_m} a_{i_{m-1}} \cdots a_{i_1}$ en L_A que es igual a la palabra $w_{i_1} w_{i_2} \cdots w_{i_m} a_{i_m} a_{i_{m-1}} \cdots a_{i_1}$ en L_B. Esta palabra tiene una derivación más a la izquierda que comienza con $S \to S_A$, y otra que comienza con $S \to S_B$. Por lo tanto, en este caso G es ambigua.

El PROBLEMA DE LA PARADA (HALTING PROBLEM)

El problema irresoluble más conocido es el problema de la parada para Máquinas de Turing, que se enuncia como sigue:

Sea M una máquina de Turing arbitraria con un alfabeto de entrada Σ, y sea $w \in \Sigma^*$. ¿Parará M con la cadena w como cadena de entrada?

El problema de la parada para las máquinas de Turing es irresoluble.
la irresolubilidad del problema de la parada se usa para demostrar que otros problemas son irresolubles. la estrategia consiste en demostrar que si un determinado problema se puede resolver ello implicaría que el problema de la parada tambien es resoluble.

PROBLEMA DE LA CINTA EN BLANCO.

Este problema consiste en decidir si una Maquina de Turing parará cuando comience su ejecución con una cinta en blanco. Este problema tambien es irresoluble.

DEMOSTRACIÓN

Sea una Máquina de Turing M y una cadena cualquiera $w \in \Sigma$. Sea M' la Maquina de Turing que comienza con la cintan en blanco, escribe en ella w y posteriormente dse posiciona en la primera celda de w en el estado q_1 (primer estado de M) y la emula a partir de ahí. Si tuviésemos un algoritmo que determinaese si una Máquina de Turing arbitraria que comienza con una cinta en blanco para, podriamos determinar si M' para. Claro que M' para si y sólo si la Maquina de Turing M original para con la cadena w como entrada, de modo que tendriamos la solución para el problema de la parada, algo que sabemos que no es cierto y que nos lleva a afirmar, consecuentemente, que el problema de la cinta en blanco es irresoluble.

Está tecnica de demostración (reducir un problema al problema de la parada) se denomina reducción.

CONCLUSIONES

los problemas a los que nos enfrentamos hoy en día y más aún, a los que de manera general la humanidad se enfrentara el día de mañana, requieren de personas competentes y con amplios conocimientos. Nosotros como ingenieros en computación, estaremos constantemente ante la necesidad de crear, diseñar, innovar o modelar la solución a tales problemas, no solo con un algoritmo eficiente, sino programable, para de esta manera automatizar la resolución de dichos problemas.

Al realizar esta investigación pudimos constatar que la indecibilidad de un algoritmo, es aquello que nos indica si el problema que tal algoritmo modela es computable o no. Analizamos también las aportaciones de grandes matemáticos. tales como Alan Turing o Rice, que a lo largo de su vida sentarón las bases de todo lo que conocemos relacionado con la computación; al emprender dicha acción, se enfrentaron ante la necesidad de definir lo computable y lo no computable, para en base a ello marcar un punto de referencia entre lo que se debe emprender, sabiendo que los resultados seran favorables, y lo que simplemente es imposible de resolver, formulando con ello los problemas mas famosos de la teoría computacional, tal como el problema de la parada, o el problema de correspondencia de Post (abordados en esta investigación), entre algunos otros.

REFERENCIAS

- CASES MUÑOZ, Rafael, MARQEZ VILLODRE, Lluis. lenguajes, gramaticas y automatas, Mexico, Alfaomega. 2012
- GARCIA, Pedro. PERES, Tomas. et al. Teoria de automatas y lenguajes formales, Mexico, Alfaomega. 2001
- Teoría de Autómatas y lenguajes Formales, laura M. Castro Souto. Consultado en: http://quegrande.org/apuntes/EI/3/TALF/teoria/00-01/apuntes_completos.pdf