Universidad Veracruzana

Maestría en Inteligencia Artificial

Proyecto Final: Máquinas de turing
Computabilidad

Velazco–Muñoz José David

zS22000515@estudiantes.uv.mx

30 de junio de 2023

Máquina de turing

Una máquina de Turing consta de una cinta infinita dividida en celdas, donde cada celda puede contener un símbolo. Además, la máquina tiene una cabeza de lectura/escritura que puede moverse a lo largo de la cinta y leer o escribir símbolos en cada celda. La máquina también tiene un conjunto de estados y una función de transición que especifica cómo se debe comportar la máquina en función del símbolo que se encuentra debajo de la cabeza y el estado actual. El funcionamiento principal de este proyecto, es generar las reglas universales de la máquina de turing. Utilizando el numero universal de turing:

$$U = 7244855335339317577198395039615711237952360672556559631108144796606505059404 83...$$

El número universal de Turing representa una descripción completa de una máquina de Turing universal. Cada dígito en el número codifica información sobre el estado interno, la cinta de entrada, las transiciones y las acciones de una máquina de Turing específica. Sin embargo, descifrar y entender el significado de cada dígito individual en este número largo es una tarea extremadamente compleja y requiere un análisis detallado.
En términos generales, el número universal de Turing se utiliza para demostrar la existencia de una máquina de Turing que puede simular cualquier otra máquina de Turing. Es una construcción teórica importante en la teoría de la computación para demostrar la equivalencia de diferentes modelos de computación y establecer límites teóricos sobre lo que es computable.
Para poder ver el comportamiento del algotimo generado, el primer paso será convertir el UN+1 a binario.

En este caso se ocupará:

$$UN + 1 = 177642$$

Número binario:

`101011010111101010`

Después ponemos una subcadena 110 al principio y al final de nuestra cadena binaria para representar R

```
11010101101011110101010110
```

Después de agregar estas cadenas, podemos aplicar las reglas para obtener los simbolos correspondientes en las reglas.

Estas reglas de sustitución representan las acciones que una máquina de Turing puede realizar en

Símbolo	**Sustitución**
0	0
1	10
R	110
L	1110
STOP	11110

Tabla 1: Reglas para decodificar reglas de una Máquina de Turing.

función del símbolo que se encuentra actualmente en la cinta de trabajo. Cada símbolo de la cinta se sustituye por una secuencia de símbolos según las reglas establecidas. Aquí tienes una descripción de cómo se utilizan estas reglas[1]:

1. Cuando la máquina de Turing encuentra el símbolo 0 en la cinta, lo deja sin cambios. Es decir, el símbolo 0 se sustituye por sí mismo.
2. Si la máquina de Turing encuentra el símbolo 1 en la cinta, lo sustituye por la secuencia 10. Esto implica que el símbolo 1 es reemplazado por un 1 seguido de un 0.
3. Cuando la máquina de Turing encuentra el símbolo R en la cinta, lo sustituye por la secuencia 110. Esto indica que el símbolo R es reemplazado por un 1, seguido de otro 1 y finalmente un 0.
4. Si la máquina de Turing encuentra el símbolo L en la cinta, lo sustituye por la secuencia 1110. Esto significa que el símbolo L es reemplazado por un 1, seguido de dos 1 y finalmente un 0.
5. Cuando la máquina de Turing encuentra el símbolo $STOP$ en la cinta, lo sustituye por la secuencia 11110. Esto indica que el símbolo $STOP$ es reemplazado por un 1, seguido de tres 1 y finalmente un 0.

Esto es aplicado en el algoritmo para establecer los movimientos en la máquina de turing. Realizando la sustitución de los las reglas, se puede decodificar una serie de secuencias en la representación binaria.

```
['R', 1, 1, 'R', 1, 'STOP', 1, 1, 'R']
```

Reglas generadas por el número $UN + 1$:

```
[[0, 0, 'R'], [1, 1, 'R'], [0, 1, 'STOP'], [1, 1, 'R']]
```

Para complementar el algoritmo realizado se agregará una comprobación más, ahora con $XN+1$

$$XN+1 = 450813704461563958982113775643437908$$

Número binario:

```
1101010110101111010101110
```

Aumentamos subcademas al princio y al final para establecer R

```
11011010101101011111010101110110
```

Realizamos la sustitución de la subcadena a su simbolo correspondiente:

```
['R', 1, 1, 'R', 'R', 1, 0, 1, 'R', 1, 1, 0, 'L', 1, 0, 1, 'R', 1, 'STOP', 1, 0, 0, 0,
'L', 1, 0, 1, 1, 'L', 1, 0, 0, 1, 'L', 1, 1, 0, 0, 'R', 1, 0, 1, 'R', 'R', 1, 1, 1, 1,
'R', 1, 1, 1, 'R', 1, 1, 1, 0, 'R']
```

Reglas generadas con $XN+1$

```
[[0, 0, 'R'], [1, 1, 'R'], [0, 0, 'R'], [10, 1, 'R'], [11, 0, 'L'], [10, 1, 'R'],
[0, 1,'STOP'],[100, 0, 'L'], [101, 1, 'L'], [100, 1, 'L'], [110, 0, 'R'],
[10, 1, 'R'], [0, 0, 'R'], [111, 1, 'R'], [11, 1, 'R'], [111, 0, 'R']]
```

Estas reglas generadas corresponden a sus respectuvas máquina. Dandonos a la idea de como funciona una máquina de turing, dandole alguna cadena binaria y obtener una serie de instrucciones.

La siguiente parte se puede determinar mediante $U(n, m)$, que tiene dos partes importantes, donde n es el número binario en la máquina que se va a imitar. y m será el tipo de máquina que se ocupará, en este caso se ocuparán las instrucciones dadad por el pasádo algoritmo del número universal U

Utulizando el algoritmo anterior, determinamos m en la máquina universal de turing:
Reglas del número U:

```
[[0, 0, 'R'], [10000000, 1, 'L'], [1, 0, 'R'], [1001011, 1, 'R'], [10, 0, 'R'], [10011000,
0, 'R'], [11, 0, 'R'], [1001110, 1, 'R'], [100, 0, 'R'], [1001101, 1, 'R'], [101, 0, 'L'],
[1011010, 1, 'L'], [110, 0, 'L'], [1110101, 1, 'L'], [111, 0, 'R'], [10011001, 1, 'R'],
[1000, 0, 'R'], [10100001, 1, 'R'], [1001, 0, 'L'], [10110001, 1, 'L'], [1010, 0, 'L'],
[10111000, 1, 'L'], [1011, 0, 'R'], [1000100, 0, 'L'], [1100, 0, 'L'], [11000101, 0, 'L'],
[10011, 1, 'R'], [1101, 1, 'R'], [1000, 0, 'R'], [1110, 1, 'R'], [101010, 0, 'R'],
[1000000, 0, 'R'], [10000, 0, 'L'], [110111, 1, 'L'], [10001, 0, 'L'], [10111111, 1, 'L'],
[100011, 1, 'L'], [10001100, 1, 'L'], [10011, 0, 'R'], [10100011, 0, 'R'], [101101, 0,
'R'], [10100, 1, 'R'], [1100, 1, 'L'], [10101, 1, 'L'], [10110, 0, 'L'], [1101101, 0, 'L'],
[111100, 0, 'R'], [10111, 1, 'L'], [110100, 1, 'R'], [11000, 1, 'R']...]
```

Ahora generamos un tape, que será nuestro n donde es el número binario que se va a imitar en la máquina universal $U(n, m)$ Para poder determinar el tape, tomaremos a $UN + 1$ a imitar, donde lo concatenaremos con un separador, seguido del binario expandido de 2.

```
000001101010110101111010101101111010011010000
```

Pasando estos valores en $U(n, m)$ obtenemos:

```
01010000000000000000000000
```

Sabiendo que el número expandido de 3 es:

```
1010
```

Podemos notar que está el número binario expandido 3 después del primer 0, esto se da por los ceros que se pusieron al principio para identificar como inicio de la cadena. Pomeos extablecer que serie de instrucciones son correctas.
Para asegurar más el resultado, utilizaremos otro ejemplo ahora con el número 149, para probar $XN + 1$. Generamos el tape:

```
000001010110110100101101010011101001011010111101000011101001010111010001011101010001101001011011010101011010101011010100111110100010010010
```

Pasando estos valores en $U(n, m)$ obtenemos:

```
0100010010100000000000000000000000000111101010001
```

El resultado del 150 en binario espandido es:

```
100010010100
```

Podemos notar que tambien funciona con la máquina $XN + 1$, de la misma manra que el unario, se muesta el resultado de binario espandido del siguiente numero que se ingreso en el tape, lo que ponemos concluir que funciona correctamente. Todos los restltados mostados fueron tomados en el REPL del código realizado para la obtención de las cadenas binarias.

Referencias

[1] R. Penrose, *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*. Oxford University Press, 1990.

[2] S. Russell y P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 2016.

[3] N. Bostrom, *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, 2014.

[4] D. R. Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books, 1979.

Código

```
''' Turing Machine'''
'''
Autor: José David Velazco Muñoz
'''

''' Numero universal de Turing'''
universal_num = 72448553353393175771983950396157112379523606725565596311108...
'''Maquina universal de turing XN+1'''
num_xn_1 =  450813704461563958982113775643437908

'''Maquina universal de turing UN+1'''
num_un_1 =  177642

num =universal_num
print(bin(num_un_1))
''' Convertimos el número a binario'''
bin_num = bin(num)
len_num = len(bin_num)
#print('Numero binario: ',bin_num)

'''Agregamos la subcadena '110' al principio y al final de la cadena binaria
   del número ingresado'''

num_binario = [1,1,0]
# Agregamos la cadena binaria
for i in range(2, len_num):
    num_binario.append(int(bin_num[i]))
# Agregamos al final
num_binario.append(1)
num_binario.append(1)
num_binario.append(0)


#print(num_binario)

''' Establecemos las regls en un diccionario'''

reglas = {
      0: [0],
      1: [1, 0],
    "R": [1, 1, 0],
    "L": [1, 1, 1, 0],
 "STOP": [1, 1, 1, 1, 0]}

''' Convertimos las subcadenas de la cadena binaria en las respectivas reglas'''
'''
Reconoce una secuencia binaria y busca coincidencias con las reglas.
Cada regla busca coincidencias entre un patrón y una proporción de la
secuencia binaria, se agrega el resultado correspondiente en una lista.
'''

def convert_bin_maq(bin_maq, rules):
```

```
    pre_maq = []
    len_bin_maq = len(bin_maq)
    i = 0
    while i < len_bin_maq:
        binary_rule = []
        # Obtaining the regla until zero from bin_maq
        while bin_maq[i] == 1:
            binary_rule.append(bin_maq[i])
            i += 1

        # Append a zero since a zero wasn't appended before
        binary_rule.append(0)
        i += 1

        # Search for the regla
        for result, binary_pattern in rules.items():
            if binary_rule == binary_pattern:
                pre_maq.append(result)
                break

    return pre_maq

'''
    La función tiene como objetivo tomar una lista con reglas. El proceso
    implica separar cada regla en movimiento, sobrescrituras y estados
    siguientes, luego generar una nueva estructura de instrucciones. Al
    final se devuelve una lista con las instrucciones en listas dentro
    de una lista.
'''

def obtener_reglas(pre_maq_rules):
    len_pmr = len(pre_maq_rules)
    spr = []
    i = 0

    # Separar cada regla de las demás
    while i < len_pmr:
        r = []

        # Hasta llegar a un movimiento: 'R', 'L' o 'STOP'
        while pre_maq_rules[i] == 1 or pre_maq_rules[i] == 0:
            r.append(pre_maq_rules[i])
            i += 1

        # Agregar el movimiento
        r.append(pre_maq_rules[i])
        i += 1

        # Agregar la nueva regla a la lista de reglas
        spr.append(r)

    rspr = []

    # Separar movimiento, sobrescritura y ns
```

```
    for r in spr:
        lr = len(r)

        # Si la regla tiene solo el movimiento 'R', 'L' o 'STOP'
        if lr == 1:
            rspr.append([0, 0, r[0]])
        else:
            # De lo contrario, obtener el movimiento, sobrescritura y ns
            m = r[lr - 1]
            o = r[lr - 2]
            ns = []

            # Recuperar el siguiente estado hasta la sobrescritura
            for j in range(lr - 2):
                ns.append(r[j])

            # Si ns es una lista vacía
            if not ns:
                ns = 0
            else:
                # De lo contrario, convertirlo a entero
                ns = int("".join(map(str, ns)))

            rspr.append([ns, o, m])

    lrspr = len(rspr)
    instr = []
    j = 0

    for i in range(lrspr):
        sp = rspr[i]
        es_bin = bin(j)[2:]
        es = int(es_bin)

        if i % 2 == 0:
            instr.append([[es, 0], sp])
        else:
            instr.append([[es, 1], sp])
            j += 1
    return  [r[1] for r in instr]

def forma_expandida(numero):
    # Convertir el número a su representación binaria sin el prefijo '0b'
    binario_str = bin(numero)[2:]
    # Inicializar la cadena expandida
    expandido_str = ""
    # Recorrer cada dígito del número binario
    for digito in binario_str:
        # Si el dígito es '0', agregar un '0' a la cadena expandida
        if digito == '0':
            expandido_str += '0'
            continue
        # Si el dígito es '1', agregar '10' a la cadena expandida
        expandido_str += "10"
```

```
131     return expandido_str
132 def MUT(cinta_de_entrada,reglas):
133     estado_actual = 0  # Estado actual
134     caja_actual = -1  # Caja actual en la cinta
135     movimiento_actual = 'R'  # Movimiento actual
136     while movimiento_actual != 'STOP':
137         if movimiento_actual == 'R':
138             caja_actual += 1  # Si es DERECHA, moverse a la siguiente caja
139         elif movimiento_actual == 'L':
140             caja_actual -= 1  # Si es IZQUIERDA, moverse a la caja anterior
141         simbolo_actual = cinta_de_entrada[caja_actual]  # Leer el contenido de la
            ↪  caja actual en la entrada
142         # Buscar en las reglas correspondientes
143         regla_encontrada = -1
144         for regla in reglas:
145             estado = regla[0][0]  # Estado
146             simbolo = regla[0][1]  # Símbolo leído
147             if estado_actual == estado and simbolo_actual == simbolo:
148                 regla_encontrada = regla[1]  # Devolver resultado
149         siguiente_estado = regla_encontrada[0]  # Siguiente estado
150         simbolo_sobrescribir = regla_encontrada[1]  # Símbolo a sobrescribir
151         movimiento_actual = regla_encontrada[2]  # Siguiente movimiento
152         # Actualizar el estado actual y sobrescribir el símbolo
153         estado_actual = siguiente_estado
154         cinta_de_entrada[caja_actual] = simbolo_sobrescribir
155     return cinta_de_entrada
156 def tape(un,num):
157     ceros = '00000'
158     separador = '111110'
159     tape = ceros + bin(un)[2:] + separador + forma_expandida(num)
160     return tape
161 def convertir_numeros(lista_original):
162     lista_nueva = []
163     for elemento in lista_original:
164         nuevo_elemento = []
165         for item in elemento:
166             if isinstance(item, int):
167                 nuevo_elemento.extend([int(digito) for digito in str(item)])
168             else:
169                 nuevo_elemento.append(item)
170         lista_nueva.append(nuevo_elemento)
171     return lista_nueva
172 ''' Conversion de las subcadenas a simbolos de regla'''
173 pre_maq = convert_bin_maq(num_binario,reglas)
```

```
174 print('Simbolos de subcadenas del numero binario: ',pre_maq)
175 '''Obtenemos la lista de instrucciones'''
176 instrucciones = obtener_reglas(pre_maq)
177 print(instrucciones)
178 ''' Máquina universal de turing U(n,m)'''
179 termino = convertir_numeros(instrucciones)
180 print('Terminos: ',termino)
181 tape = tape(num_xn_1,149)
182 print('Tape: ',tape)
183 print('Resultado: ',termino)
184 next_binary_number = MUT(tape,instrucciones)
185 print(next_binary_number)
186 expandido = forma_expandida(150)
187 print(expandido)
```