Tópicos Avanzados en Computabilidad

Pablo F. Castro - UNRC

AutoReferencia

Una característica importante de las máquinas de Turing es la posibilidad de autoreferencia:

- Las máquinas de Turing pueden tomar descripciones de otras máquinas de Turing como entrada,
- Podemos pasarle a una MT como entrada la descripción de ella misma,
- Esto permite, por ejemplo, demostrar que hay problemas indecidibles.

De la misma forma las MT pueden producir descripciones de MT

Podrá una MT escribir una descripción de ella misma?

La MT `Self`

Veremos como construir una máquina de Turing que devuelve una descripción de ella misma.

Propiedad: Existe una función computable $q : \Sigma^* \to \Sigma^*$ tal que dado $w \in \Sigma^*$, $q(w)$ es la descripción de una MT P_w que imprime w y luego termina.

La MT M que computa q es la siguiente:

Dada una entrada w:

- Construye una máquina P_w que:
  - borra todo de la cinta, escribe w y termina
  - Escribe en la cinta $\langle P_w \rangle$

La MT `Self`

Utilizando q podemos definir Self:

La idea es que `Self` consta de dos partes A y B:

- A es $P_{\langle B \rangle}$, es decir la MT que imprime $\langle B \rangle$ y termina, esta se puede obtener mediante $q(\langle B \rangle)$

- B hace lo siguiente, después que se ejecutó A quedó $\langle B \rangle$ en la cinta, entonces ejecuta $q(\langle B \rangle)$ y obtiene $P_{\langle B \rangle}$ que es A, y imprime $\langle AB \rangle$ en la cinta.

La MT Self

Lo que hace Self es reproducirse (al estilo de un virus). Lo mismo podemos hacer textualmente con:

Copiar dos veces el siguiente párrafo, una con doble comillas:

“Copiar dos veces el siguiente párrafo, una con doble comillas:”

Aca el B es la primer parte sin comillas, la parte A es la parte con comillas.

A le da a B una copia de ella misma para que la pueda procesar.

El Teorema de la Recursión

La idea del teorema es que nos provee una forma de implementar autoreferencia en cualquier lenguaje de programación

Teorema: Sea $t : \Sigma^* \times \Sigma^* \to \Sigma^*$ una función computable por MT T, entonces existe una función computable (por una MT R) tal que computa $r : \Sigma^* \to \Sigma^*$ tal que $r(w) = t(R, w)$.

Intuición: T es una máquina de Turing que dada una máquina y una entrada computa el resultado de ejecutar esa MT con la entrada. El teorema me garantiza que puedo crear una MT en donde se corre a ella misma $r(T, w)$

Teorema de la Recursión

Para demostrarlo la idea es parecida que la construcción de la MT Self:

Dada T consideremos las MT A y B,

- A es la MT $P_{\langle BT \rangle}$ además A escribe el string pasado como parámetro, es decir, después de ejecutar A en la cinta tenemos $w\langle BT \rangle$,
- B toma $\langle BT \rangle$ y le aplica q, es decir, obtiene la MT $P_{BT} = A$, entonces obtiene $\langle ABT \rangle = R$, escribe $\langle R, w \rangle$ en la cinta y ejecuta T

Teorema de la Recursión

El teorema de la recursión nos dice que podemos obtener la descripción de la MT que estamos usando y ejecutarla

Esto permite que en la descripción de una MT podemos decir que obtenemos su propia descripción y la ejecutamos

Usando el Teorema de la Recursión

Podemos usar el teorema de la recursión para demostrar que A_{TM} es indecidible

Supongamos que tenemos una MT H que decide A_{TM}, entonces podemos construir la siguiente máquina (B):

- Con el input w:
  - se obtiene la descripción de la misma máquina $\langle B \rangle$,
  - corremos H con $\langle B \rangle$ con el input w
  - si H acepta entonces se rechaza, si H rechaza de acepta

Observar que B en el input w hace lo contrario que dice H entonces H no puede existir.

Introspección

La introspección en los lenguajes de programación nos permite ver el mismo código que estamos ejecutando

En Python podemos usar la biblioteca inspect:

```
import inspect

def foo():
    pass

print(inspect.getmembers(foo))
```

Imprime toda la información sobre el método

```
import inspect

def foo(x, y, z):
    pass

print(inspect.getargspec(foo))
```

Imprime información sobre los argumentos del método

El Lenguaje Min

Sea una MT T y $\langle T \rangle$ su descripción, decimos que T es **minimal** si no existe una MT cuya descripción sea más corta, que sea equivalente.

Podemos considerar el siguiente lenguaje:

$$Min = \{\langle M \rangle \mid M \text{ es minimal } \}$$

El lenguaje Min no es reconocible. Supongamos que es reconocible por C. Es decir, existe un enumerador E que lo enumera. Consideremos la MT M

- Ejecutamos E,
- Cuando aparece una MT D cuya descripción es más larga que la de M,
- Ejecutamos M con la entrada.

Entonces D no puede ser minimal!

Oráculos

En general hemos visto que podemos definir máquinas de Turing que utilizan otras máquinas de Turing para resolver tareas. Podemos generalizar esto:

Definición: Un oráculo para el lenguaje B es una máquina que dice si o no $w \in B$.

Definición: Una máquina de Turing con un oráculo, es una máquina que puede consultar a un oráculo para decidir pertenencia a un lenguaje.

Escribimos Notamos M^B cuando la MT M puede utilizar el oráculo B

Oráculos

Si asumimos oráculos que pueden responder correctamente cualquier pregunta, podríamos computar problemas no computables

Suponiendo que tenemos un oráculo para A_{TM}, entonces podemos resolver E_{TM}, veamos:

Construimos la MT E que dada la MT M hace lo siguiente:

Consideramos la MT N que en cualquier entrada:

- Corre M M con string de longitud 1,2,3,4,…
  - Si acepta alguno acepta

Chequeamos con el oráculo si N con ϵ termina o no

Reducibilidad Usando Oráculos

Podemos definir la noción de reducibilidad utilizando oráculos:

Decimos que el lenguaje A es decidible relativo a B, si hay una máquina de Turing M^B que decide A teniendo un oráculo para B

Reducción: $A \leq_R B$ (A es Turing reducible a B) ssi el lenguaje A es decidible relativo a B

Por ejemplo E_{TM} es reducible a A_{TM}.

Ejercicios

1. Escribir en Python la función Self que se reproduce a si misma.

2. Describir dos máquina de Turing M y N tal que cuando comienza en cualquier input, M imprime $\langle N \rangle$ y N imprime $\langle M \rangle$.

3. Usar la propiedad que el conjunto de lenguajes es incontable para demostrar que existen lenguajes que no pueden ser reconocidos con una máquina con un oráculo A_{TM}.

4. Escribir una clase Persona en Python, con los atributos clásicos. Y hacer una función que con introspección:

   - Imprima la lista de atributos de la clase,
   - Imprima el código de alguno de los métodos

5. Usando introspección hacer una función que cree una clase dinámicamente.

Teorías Lógicas

Recordemos el lenguaje de primer orden: