



Pontificia Universidad Católica de Chile

IIC2613 - Inteligencia Artificial

Tarea 2

Septiembre de 2024

1. Parte 1

1.1. # 120 François Chollet

François Chollet considera que el test de turing es una mala forma de medir la inteligencia o capacidad de analizar ya que considera que en el test esta se reduce a la capacidad de juicio de unos jueces arbitrarios y que naturalmente pueden permitir errores, esto produce que no exista confiabilidad en el criterio para aprobar o no el test de turing. De esta forma, considera que el test requiere de algun grado de estandarización para ser considera un buen mecanismo metodologico.

Sumado a esto, argumenta que desde sus inicios el test nunca tuvo intención de ser utilizado como un mecanismo de testeo, sino más bien como un experimento dentro de una discusión filosófica y que por consiguiente carece de una estructura propia de un método para demostrar en base a un caso empírico. De esta forma, considera que el mecanismo del test de turing propicia prácticas que atentan contra el objetivo final, permitiendo que aprobar el test se concentre en mayor medida en burlar o engañar los criterios establecidos por los jueces en lugar de concretamente simular un pensamiento humano con código.

Por otra parte, Lex Fridman está en desacuerdo con la afirmación previa ya que considera que en la práctica este último caso no sucede, puesto que es más fácil intentar aprobar del test de manera 'sensata' o 'real' antes que buscar mecanismo para engañar a los jueces.

A modo personal, comparto parcialmente la perspectiva de François Chollet, considero que si los investigadores en IA se pusieran como objetivo pasar el Test de Turing lograrían un gran avance en una etapa inicial, ya que es una buena práctica para concentrarse en parámetros establecidos y partir un por un área del razonamiento o un área a tratar, de esta forma considero que el método del test de turing es un mecanismo óptimo para plantear el escenario que queremos modelar, sin embargo tiene limitaciones como relata Chollet ya que intenta resolver un problema puntual en lugar de buscar generar una herramienta funcional en la relación humano-computadora.

De esta forma, creo que posterior a pasar una cantidad predefinida de estos test, el camino de la inteligencia artificial debe avanzar hacia un análisis más cualitativo de las respuestas y como perfeccionarlas para que esta sea una herramienta con mayor alcance y potencial.

En ese sentido concuerdo con Chollet donde dice que un buen test, en este caso del de turing, es aquel que te aproxima hacia el primer peldaño de la escalera y no necesariamente la cima. De esta forma concuerdo con que es un buen método para comenzar y avanzar en una solución, mas no para perfeccionar las 'habilidades cognitivas' de la máquina y buscar la semejanza o indistinguibilidad con el humano.

1.2. # 61 Melanie Mitchell

Melanie Mitchell en particular habla de los conceptos y como el trabajo con ellos son de los problemas abiertos más grandes dentro de la IA, en este sentido Melanie habla de como los conceptos se entienden dentro de un mundo lleno de otros conceptos donde unos son combinaciones

de otros complejos conceptos.

De esta forma, Melanie Mitchell habla de las analogías al referirse a la capacidad humana de encontrar similitudes en la esencia de 2 hechos o conceptos, donde teniendo miles de asepciones totalmente distintas, en contextos distintos, de todas formas el ser humano tiene la capacidad de *analizar* y encontrar un patrón común con un concepto completamente fuera del contexto en el que se trató el primer concepto.

En ese sentido, podemos traer esta conversación al ASP, donde trabajamos con conceptos y sus relaciones. Un ejemplo de esto que podemos hacer con un sistema computacional, es entender la relación entre 2 eventos completamente distintos pero que comparten una estructura similar.

Imaginemos que queremos minimizar los costos de un viaje familiar y minimizar el gasto en cables de una empresa eléctrica, como parte de nuestro razonamiento humano sabemos que lo que realmente queremos en ambos casos es simplemente minimizar la ruta para tener que incurrir en el menor gasto de materia prima posible.

Así, lo que deberíamos hacer en ASP es modelar el caso con términos compatibles para ambos escenarios y luego simplemente establecer la relación entre ambos casos de manera manual o eventualmente tener un modelo que reconozca los patrones comunes de estos escenarios y que haga la relación mediante a un previo entrenamiento.

2. Parte 2

2.1.

Parte A

Si existe el modelo M tal que $p \in M$.

- La regla $\leftarrow p$ nos dice que p no puede estar en el modelo M , ya que esta regla hace que p sea falso.
- Si $p \in M$, no se cumpliría la regla $\leftarrow p$, lo cual es una contradicción.
- Si $p \notin M$, la regla p no se satisface, porque exige que p sea verdadero.

De esta forma, asumiendo inicialmente que existe el modelo M tal que $p \in M$, logramos ver que siguiendo una secuencia lógica, nos encontramos frente a una contradicción, por lo tanto el programa no tiene ningún modelo.

Parte B

- Si p es verdadero, entonces por la regla $p \leftarrow \text{not } q$, q debe ser falso.
- Si q es verdadero, entonces por la regla $q \leftarrow \text{not } p$, p debe ser falso.

De esta forma, podemos ver que tenemos 2 posibles modelos:

- $M_1 = \{p\}$. En este caso, p es verdadero y q es falso.
 - La regla $p \leftarrow \text{not } q$ se satisface, ya que q es falso.
 - La regla $q \leftarrow \text{not } p$ no aplica, porque q es falso.
- $M_2 = \{q\}$. En este caso, q es verdadero y p es falso.
 - La regla $q \leftarrow \text{not } p$ se satisface, ya que p es falso.
 - La regla $p \leftarrow \text{not } q$ no aplica, porque p es falso.

2.2.

Consideramos que las reglas en los programas lógicos son de la forma $\text{Head} \leftarrow \text{Body}$, con $|\text{Head}| = 1$, es decir, la cabeza de la regla contiene un átomo, según lo planteado en el enunciado.

- Sea Π un programa, dado que M es un modelo de Π , este satisface las reglas de Π . Por lo tanto para cada regla $\text{Head} \leftarrow \text{Body}$, si Body es verdadero en M , entonces Head también debe ser verdadero en M .

- Si agregamos una nueva regla a Π , tenemos un programa Π' de modo tal que $\Pi \subseteq \Pi'$. En ese sentido hay que demostrar que el modelo M puede ampliarse a un modelo M' que $\in \Pi'$.
- Supongamos que el programa Π tiene n_i reglas. Agregando una nueva regla n_{i+1} a Π , tenemos $\Pi' = \Pi \cup \{n_{i+1}\}$.
- Ahora debemos demostrar que existe un modelo M' tal que $M \subseteq M'$.
- Dado que n_{i+1} es una regla de la forma $\text{Head} \leftarrow \text{Body}$, si Body es verdadero en M , entonces Head debe añadirse al conjunto, de este modo $M' = M \cup \{\text{Head}\}$.
- Por otro lado, como no hay negación en las reglas, agregar una nueva regla no elimina otras soluciones de reglas previamente definidas. Por lo tanto, M' sigue siendo un modelo de Π , y además cumple con la regla n_{i+1} , por lo que es un modelo de Π' .

De esta forma, por inducción se demuestra que si $\Pi \subseteq \Pi'$ y M es un modelo de Π , entonces existe un modelo M' de Π' tal que $M \subseteq M'$.

2.3.

En este caso queremos demostrar que todo programa que tiene reglas de la forma $\text{Head} \leftarrow \text{Body}$, sin negación, y con $|\text{Head}| \leq 1$, tiene a lo más un modelo.

Para esto nos basaremos en una demostración según el número de reglas de un programa.

- Sea el caso de que tenemos un programa que contiene una única regla de la forma $\text{Head} \leftarrow \text{Body}$.
- Por definición si Body es verdadero, entonces Head es verdadero, por lo tanto solo tenemos un modelo, el cual es el que contiene Head .
- Si Body es falso, Head no tenemos información de este, por lo que tenemos un modelo vacío. De esta forma vemos que para el caso base existe a lo más un modelo.
- Por ahora vamos a suponer debido a la inducción que todo programa con n reglas tiene a lo más un modelo.
- Sea un programa con $n + 1$ reglas, si nos enfocamos en ver el subconjunto de n reglas, podemos decir por el paso anterior que tiene a lo más un modelo, de esta forma nos enfocaremos en el subconjunto $n + 1$.
- Si Body_{n+1} es falso, Head_{n+1} no se activa por lo que seguimos con el mismo modelo del caso anterior.
- En cambio, si Body_{n+1} es verdadero en el modelo, entonces Head_{n+1} se activa e incorpora al modelo, de esta forma tenemos un nuevo modelo M' el cual de todas formas sigue siendo solo uno.

De esta forma, por inducción podemos demostrar que todo programa que tiene reglas de la forma $\text{Head} \leftarrow \text{Body}$, sin negación, y con $|\text{Head}| \leq 1$, tiene a lo más un modelo.

3. Parte 3

3.1.

Al ejecutar código que se nos entrega,

```
amistad(juan,maria).
amistad(maria,elena).
amistad(elena,fernando).
```

```
usuario(X) :- amistad(X,Y).
usuario(Y) :- amistad(X,Y).
```

```
cadena(X,Y,Largo) :- amistad(X,Y), Largo=1. %Caso base
cadena(X,Y,Largo) :- cadena(X,Z,L1), cadena(Z,Y,L2), Largo=L1+L2.
```

Vemos que se ejecuta correctamente, generando un modelo de la siguiente manera:

```
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main)> clingo aux.lp
clingo version 5.7.1
Reading from aux.lp
Solving...
Answer: 1
amistad(juan,maria) amistad(maria,elena) amistad(elena,fernando) usuario(maria) usuario(elena) usuario(fernando) usuario(juan) cadena(juan,maria,1) cadena(maria,elena,1)
cadena(elena,fernando,1) cadena(maria,fernando,2) cadena(juan,elena,2) cadena(juan,fernando,3)
SATISFIABLE

Models      : 1
Calls       : 1
Time        : 0.002s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time    : 0.002s
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main) [30]>
```

Sin embargo, si agregamos `amistad(fernando,juan)` al código, vemos que se genera un ciclo, donde tenemos que interrumpir el código ya que no llega nunca a una solución.

```
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main) [30]> clingo aux.lp
clingo version 5.7.1
Reading from aux.lp
^C*** Info : (clingo): INTERRUPTED by signal!
UNKNOWN

INTERRUPTED : 1
Models      : 0+
Calls       : 1
Time        : 20.124s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time    : 20.106s
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main) [1]>
```

De esta forma, tenemos que buscar una forma de plantear el código para que evitemos entrar en un ciclo y por resolver el problema.

Con todo esto en marcha, ahora ahora podemos considerar replantear el problema, eliminando las últimas 2 líneas de código, que son las que traen problemas.

Primero agregaremos una línea para asegurar que no importe la posición en cuanto a la amistad.

```
amistad(X,Y) :- amistad(Y,X).
```

Luego utilizando count reformularemos las 2 líneas que traen problemas. De esta forma el código actualizado y funcional queda de la siguiente manera:

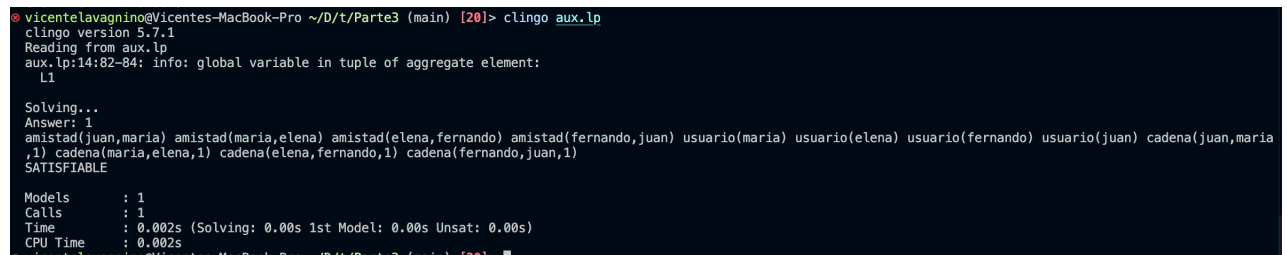
```
amistad(juan,maria).
amistad(maria,elena).
amistad(elena,fernando).
amistad(fernando,juan). % Caso que produce un ciclo

usuario(X) :- amistad(X,Y).
usuario(Y) :- amistad(X,Y).

% cadena(X,Y,Largo) :- amistad(X,Y), Largo=1. % NO USADO
% cadena(X,Y,Largo) :- cadena(X,Z,L1), amistad(Z,Y,L2), Largo=L1+L2. % NO
⇨ USADO

% uso de #count para evitar ciclos en cadena en caso base
cadena(X, Y, Largo) :- amistad(X, Y), Largo = 1.
cadena(X, Y, Largo) :- cadena(X, Z, L1), amistad(Z, Y), Largo = L1 + 1, #
⇨ count { L1 : cadena(X, N, L1) } = Largo, X != Y, Z != X, Z != Y.
```

En particular lo que se hizo fue replicar la estructura existente en la primera línea de `cadena(X,Y,Largo)` y luego modificar la segunda que genera el ciclo para ver, en los casos que no sean iguales las variables, que no se cuantifique recursivamente los casos de unión.



```
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main) [20]> clingo aux.lp
clingo version 5.7.1
Reading from aux.lp
aux.lp:14:82-84: info: global variable in tuple of aggregate element:
  L1

Solving...
Answer: 1
amistad(juan,maria) amistad(maria,elena) amistad(elena,fernando) amistad(fernando,juan) usuario(maria) usuario(elena) usuario(fernando) usuario(juan) cadena(juan,maria,1)
cadena(maria,elena,1) cadena(elena,fernando,1) cadena(fernando,juan,1)
SATISFIABLE

Models      : 1
Calls       : 1
Time        : 0.002s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time    : 0.002s
vicentelavagnino@Vicentes-MacBook-Pro ~/D/t/Parte3 (main) [20]>
```