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

Interrogación 3

IIC2613 - Inteligencia Artificial

Primer Semestre, 2019

Tiempo: 1:45 hrs.

1. *a*) Mencione una crítica que se le hace al *Test de Turing* que haya sido 'solucionada' por los *Esquemas de Winograd*. Justifique.

   Respuesta: Una crítica es que para contestar el Test de Turing es necesario ser capaz de engañar o simular una identidad no propia. En los esquemas de Winograd no hay engaño sino que solo se concentra en preguntas que requieren sentido común para ser respondidas.

   b) Invente (y escriba) **dos** instancias del mismo esquema de Winograd que usen las palabras *adelantó*, *lento*, *rápido*, *micro*, *bicicleta*. Indiqué qué conocimiento de sentido común es necesario para responder estas preguntas.

   Respuesta:

   La bicicleta adelantó a la micro porque iba muy lento. ¿Qué vehículo iba muy lento?

   a) La micro
   b) La bicicleta

   La bicicleta adelantó a la micro porque iba muy rápido. ¿Qué vehículo iba muy lento?

   a) La micro
   b) La bicicleta

   El conocimiento de sentido común necesario para responder esta pregunta es que, frente a un adelantamiento, el vehículo que adelanta va a mayor velocidad (es decir más rápido que) que el vehículo que es adelantado.

   c) ¿Cuántos nodos, aproximadamente, se almacenan en la lista Open del algoritmo DFS en el momento que se expande un nodo a profundidad p? Haga todos los supuestos que permitan simplificar (sin trivializar) su respuesta.

   Respuesta: Suponiendo que el factor de ramificación del árbol de DFS es b, entonces, por cada nivel del árbol hay $(b-1)$ o menos nodos. Entonces podemos aproximar el número por $p(b-1)$ (o bp).

2. *a*) Describa un programa que tenga $\binom{n}{5}$ modelos. **Respuesta:** El programa contiene las reglas $p(a_1), p(a_2), \ldots, p(a_n)$. Más la regla $5\,\{q(X) : p(X)\}\,5$.

   b) Describa un programa que tenga 5^n modelos. **Respuesta:** Acá usamos n predicados: $p_1, \ldots, p_n$. El programa contiene, para cada p_i $(i \in \{1, \ldots, n\})$, la restricción de cardinalidad:

   $$1\,\{p_i(1); p_i(2); p_i(3); p_i(4); p_i(5)\}\,1.$$

   c) Suponga que usa el predicado *nodo* y *arco* para definir los nodos y arcos de un grafo dirigido. Escriba reglas que permitan deducir cuáles son los nodos *superalcanzables*, que son aquellos que son alcanzables desde todo nodo del grafo.

   Respuesta:

```
alcanzable(X,Y) :- arco(X,Y).
alcanzable(X,Y) :- alcanzable(X,Z),alcanzable(Z,Y).

% un nodo no es superalcanzable si desde algún nodo Y no es posible alcanzar a X
nosuperalcanzable(X) :- nodo(Y),not alcanzable(Y,X).

% Un nodo es super alcanzable cuando no es posible demostrar que no es nosuperalcanzable
superalcanzable(X) :- not nosuperalcanzable(X).
```

3. Los circuitos digitales tienen muchísimas aplicaciones prácticas. En esta pregunta exploraremos cómo modelar un circuito digital compuesto por componentes lógicas Booleanas. En nuestra modelación supondremos que cada cable puede transmitir una señal binaria (0 o 1), y utilizaremos solamente los siguientes predicados:

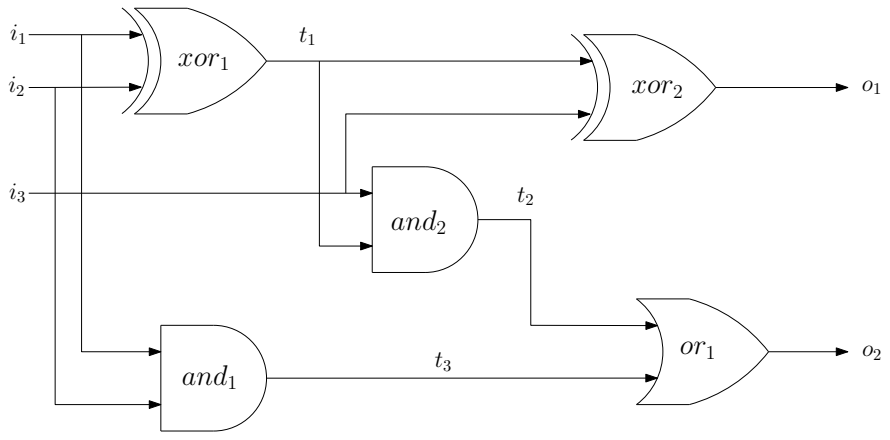
   - $salida(t, x, y, z)$: expresa que la componente de tipo t, cuando sus entradas son x e y, tiene como salida z.
   - $cable(c)$: expresa el hecho que c es un cable.
   - $componente(m)$: expresa el hecho que m es una componente.
   - $val(c, v)$: expresa el hecho la señal que transmite el cable c es v.
   - $defectuosa(m)$: expresa que la componente m está defectuosa.

   Parte de las reglas del programa para modelar el circuito se encuentran ya escritas en el reverso de esta hoja.

   a) Escriba 5 reglas adicionales que permitan deducir qué valor está asociado a los cables $t1$, $t2$, $t3$, $o1$, y $o2$, en el caso que las componentes **no sean defectuosas**. Cada regla debe mencionar (correctamente) al predicado *defectuosa* en el lado derecho. En particular, si a su programa se agregan las tres sentencias $val(i1, 1)$, $val(i2, 0)$, $val(i3, 1)$, entonces se debe deducir que $val(o1, 0)$ y $val(o2, 1)$.

   b) Ahora supondremos que cuando una componente m falla, entonces m podría dar cualquier salida, tanto 0 como 1, independiente de cuál es la entrada. Escriba 5 reglas adicionales—una por componente—que modelen el comportamiento del circuito cuando las componentes puedan fallar.

   c) ¿Cómo es posible usar el sistema de programación en lógica para descubrir cuál componente está fallando si ante la entrada descrita por $val(i1, 1)$, $val(i2, 0)$, $val(i3, 1)$, se obtiene un 1 en $o1$ y un 0 en $o2$? Diga qué agregar al programa y luego cómo obtener la respuesta.



```
salida(xor,0,0,0).
salida(xor,0,1,1).
salida(xor,1,0,1).
salida(xor,1,1,0).

salida(or,0,0,0).
salida(or,0,1,1).
salida(or,1,0,1).
salida(or,1,1,1).

salida(and,0,0,0).
salida(and,0,1,0).
salida(and,1,0,0).
salida(and,1,1,1).

cable(i1).
cable(i2).
cable(i3).
cable(t1).
cable(t2).
cable(t3).

componente(xor1).
componente(xor2).
componente(and1).
componente(and2).
componente(or1).
```

Respuesta:

```
salida(xor,0,0,0).
salida(xor,0,1,1).
salida(xor,1,0,1).
salida(xor,1,1,0).

salida(or,0,0,0).
salida(or,0,1,1).
salida(or,1,0,1).
salida(or,1,1,1).

salida(and,0,0,0).
salida(and,0,1,0).
salida(and,1,0,0).
salida(and,1,1,1).

cable(i1).
cable(i2).
cable(i3).
cable(t1).
cable(t2).
cable(t3).
```

```
componente(xor1).
componente(xor2).
componente(and1).
componente(and2).
componente(or1).

val(i1,1).
val(i2,0).
val(i3,1).

%% PARTE a)
val(t1,V) :- salida(xor,V1,V2,V),val(i1,V1),val(i2,V2),not defectuosa(xor1).
val(o1,V) :- salida(xor,V1,V2,V),val(t1,V1),val(i3,V2),not defectuosa(xor2).
val(t3,V) :- salida(and,V1,V2,V),val(i1,V1),val(i2,V2),not defectuosa(and1).
val(t2,V) :- salida(and,V1,V2,V),val(i3,V1),val(t1,V2),not defectuosa(and2).
val(o2,V) :- salida(or,V1,V2,V),val(t2,V1),val(t3,V2),not defectuosa(or1).

%% PARTE b)
val(t1,0); val(t1,1) :- val(i1,V1),val(i2,V2),defectuosa(xor1).
val(o1,0); val(o1,1) :- val(t1,V1),val(i3,V2),defectuosa(xor2).
val(t3,0); val(t3,1) :- val(i1,V1),val(i2,V2),defectuosa(and1).
val(t2,0); val(t2,1) :- val(i3,V1),val(t1,V2),defectuosa(and2).
val(o2,0); val(o2,1) :- val(t2,V1),val(t3,V2),defectuosa(or1).

%% PARTE c)
1 {defectuosa(C) : componente(C)} 1.
observacion :- val(o1,1),val(o2,0).
:- not observacion.
```