8,5

LICENCIATURA EN CS. DE LA COMPUTACION

Introducción a la Inteligencia Artificial

2015 - Parcial 1

1. Dadas las siguientes fórmulas en lógica de primer orden:

$\forall x\,(S(x) \rightarrow \exists y\,(T(y) \wedge E(A, y)))$

$\forall x\,(W(x) \rightarrow S(x))$

$W(A) \wedge D(A)$

$\forall z\,(T(z) \wedge (D(y) \wedge E(x, z)) \rightarrow M(z)))$

* Considerar x, y, z variables y A constante

a- Determinar si se puede probar que la siguiente fórmula es consecuencia del conjunto anterior utilizando resolución: $\exists z\,(T(z) \wedge M(z))$

b- Se pueden encontrar todas las fórmulas que derivan del conjunto de fórmulas dado utilizando resolución? justificar.

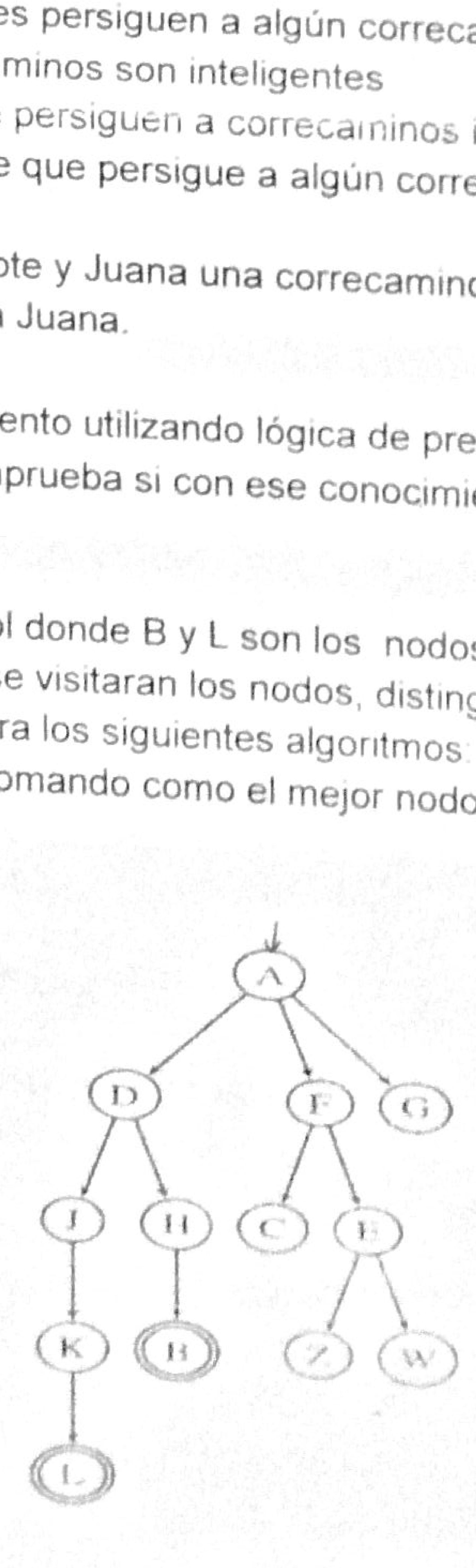
2. Dado el siguiente fragmento de conocimiento:

1- Todos los coyotes persiguen a algún correcaminos
2- Algunos correcaminos son inteligentes
3- Los coyotes que persiguen a correcaminos inteligentes no los atrapan.
4- Cualquier coyote que persigue a algún correcaminos pero no lo atrapa, estará hambriento
5- Pepe es un coyote y Juana una correcaminos y ambos son inteligentes
6- Pepe persigue a Juana.

a- Representa el conocimiento utilizando lógica de predicados

b- Usando resolución comprueba si con ese conocimiento es posible saber si "Pepe está hambriento".

3. Dado el siguiente árbol donde B y L son los nodos meta y A es el nodo inicial. Indicar en qué orden se visitaran los nodos, distinguiendo nodos generados de nodos expandidos, para los siguientes algoritmos: (1) búsqueda a lo ancho y (2) Primero el mejor, tomando como el mejor nodo aquel con menor orden alfabético.



4. Dados una cantidad arbitraria de bloques, N, cada uno de ellos con una altura diferente, ni, se desean construir dos pilas cuya altura sea lo más parecida posible. Por ejemplo, si hubiera cuatro bloques (N = 4) cuyas alturas fueran n1 = 2, n2 = 3, n3 = 4 y n4 = 5, sería posible organizar una pila con los bloques 1 y 4 (con una altura a 7) de modo que la segunda pila contendría los bloques 2 y 3, con una altura también igual a 7.

Se pide:

a- Suponiendo que tenemos 6 bloques de altura que toman valores enteros de 1 a 10. Definir formalmente el problema como un espacio de estados.

b- Definir una heurística que pueda aplicarse para la resolución del problema mediante búsqueda. (no se requiere aplicar el algoritmo).

5. Supongamos que tenemos un algoritmo de búsqueda de tipo el mejor primero que utiliza la siguiente función
de evaluación:

$$f(n) = (2 - w)\, g(n) + w.h(n)$$

donde g es la función costo de ruta y h es una heurística admisible.

a- ¿Qué tipo de búsqueda hace cuándo w=0? ¿cuándo w=1? ¿y para w=2?

b- Analice para los distintos valores de w si el algoritmo puede asegurarse que es óptimo y/o completo

Primer parcial de Introducción a la Inteligencia Artificial

2.a. Cláusulas en lógica de predicados:

1. $\forall x(Coyote(x) \to \exists y(Correcaminos(y) \wedge Persigue(x,y)))$
2. $\exists x(Correcaminos(x) \wedge Inteligente(x))$
3. $\forall x\ \forall y(Coyote(x) \wedge Correcaminos(y) \wedge Inteligente(y) \wedge Persigue(x,y)$
   $\to \neg Atrapa(x,y))$
4. $\forall x(Coyote(x) \wedge \exists y(Correcaminos(y) \wedge \neg Atrapa(x,y)) \to Hambriento(x))$ $\wedge$ Persigue(x, y)
5. $Coyote(Pepe) \wedge Correcaminos(Juana) \wedge Inteligente(Pepe) \wedge Inteligente(Juana)$
6. $Persigue(Pepe, Juana)$

2.b. Agregamos la cláusula de la conclusión:

$Hambriento(Pepe)$

Pasamos todas a forma normal:

1. $\forall x(Coyote(x) \to \exists y(Correcaminos(y) \wedge Persigue(x,y)))$

$\equiv \forall x(\neg Coyote(x) \vee (\exists y(Correcaminos(y) \wedge Persigue(x,y))))$

$\equiv \forall x\ \exists y(\neg Coyote(x) \vee (Correcaminos(y) \wedge Persigue(x,y)))$

$\equiv \exists y(\neg Coyote(x) \vee (Correcaminos(y) \wedge Persigue(x,y)))$

$\equiv \neg Coyote(x) \vee (Correcaminos(S_0(x)) \wedge Persigue(x, S_0(x)))$ función de Skolem

La variable es la misma $\equiv (\neg Coyote(x) \vee Correcaminos(S_0(x))) \wedge (\neg Coyote(x) \vee Persigue(S_0(x)))$

$\equiv (\neg Coyote(x_0) \vee Correcaminos(S_0(x_0))) \wedge (\neg Coyote(x_1) \vee Persigue(S_0(x_1)))$

2. $\exists x(Correcaminos(x) \wedge Inteligente(x))$

$\equiv Correcaminos(S_1) \wedge Inteligente(S_1)$

3. $\forall x\ \forall y(Coyote(x) \wedge Correcaminos(y) \wedge Inteligente(x) \wedge Persigue(x,y)$
   $\to \neg Atrapa(x,y))$

$\equiv \forall x\ \forall y(\neg(Coyote(x) \wedge Correcaminos(y) \wedge Inteligente(x) \wedge Persigue(x,$

$y)) \vee \neg Atrapa(x, y))$

$\equiv \forall x \forall y (\neg Coyote(x) \vee \neg Correcaminos(y) \vee \neg Inteligente(y) \vee$
$\vee \neg Persigue(x, y) \vee \neg Atrapa(x, y))$

$\equiv \neg Coyote(x_2) \vee \neg Correcaminos(x_3) \vee \neg Inteligente(x_3) \vee \neg Persigue(x_2, x_3) \vee$
$\vee \neg Atrapa(x_2, x_3)$

4. $\forall x (Coyote(x) \wedge \exists y (Correcaminos(y) \wedge \neg Atrapa(x, y)) \rightarrow Hambriento(x))$

$\equiv \forall x (\neg(Coyote(x) \wedge \exists y (Correcaminos(y) \wedge \neg Atrapa(x, y))) \vee$
$\vee Hambriento(x))$

$\equiv \forall x (\neg Coyote(x) \vee \neg \exists y (Correcaminos(y) \wedge \neg Atrapa(x, y)) \vee$
$\vee Hambriento(x))$

$\equiv \forall x (\neg Coyote(x) \vee \forall y (\neg(Correcaminos(y) \wedge \neg Atrapa(x, y))) \vee$
$\vee Hambriento(x))$

$\equiv \forall x (\neg Coyote(x) \vee \forall y (\neg Correcaminos(y) \vee Atrapa(x, y)) \vee$
$\vee Hambriento(x))$

$\equiv \forall x \forall y (\neg Coyote(x) \vee \neg Correcaminos(y) \vee Atrapa(x, y) \vee$
$\vee Hambriento(x))$

$\equiv \neg Coyote(x) \vee \neg Correcaminos(y) \vee Atrapa(x, y) \vee Hambriento(x)$

$\equiv \neg Coyote(x_4) \vee \neg Correcaminos(x_5) \vee Atrapa(x_4, x_5) \vee Hambriento(x_4)$

Queda entonces lo siguiente:

(C1) $\neg Coyote(x_0) \vee Correcaminos(S_0(x_0))$

(C2) $\neg Coyote(x_1) \vee Persigue(S(x_1))$

(C3) $Correcaminos(S_1)$

(C4) $Inteligente(S_1)$

(C5) $\neg Coyote(x_2) \vee \neg Correcaminos(x_3) \vee \neg Inteligente(x_3) \vee \neg Persigue(x_2, x_3) \vee$
$\vee \neg Atrapa(x_2, x_3)$

(C6) $\neg Coyote(x_4) \vee \neg Correcaminos(x_5) \vee Atrapa(x_4, x_5) \vee Hambriento(x_4)$

(C7) Coyote(Pepe)

(C8) Correcaminos(Juana)

(C9) Inteligente(Pepe)

(C10) Inteligente(Juana)

(C11) Persigue(Pepe, Juana)

Agregamos la negación de la conclusión como una premisa más.

(C12) ¬Hambriento(Pepe)

Ahora, aplicando resolución, ¿se llega a una contradicción? Sí:

(C13) ¬Coyote(Pepe) ∨ ¬Correcaminos(x_5) ∨ Atrapo(Pepe, x_5) {C12, C6}

(C14) ¬Correcaminos(x_5) ∨ Atrapo(Pepe, x_5) {C13, C7}

(C15) Atrapo(Pepe, Juana) {C14, C8}

(C16) ¬Coyote(Pepe) ∨ ¬Correcaminos(Juana) ∨ ¬Inteligente(Juana) ∨ ¬Persigue(Pepe, Juana) {C15, C5}

(C17) ¬Correcaminos(Juana) ∨ ¬Inteligente(Juana) ∨ ¬Persigue(Pepe, Juana) {C16, C7}

(C18) ¬Inteligente(Juana) ∨ ¬Persigue(Pepe, Juana) {C17, C8}

(C19) ¬Persigue(Pepe, Juana) {C18, C10}

(C20) ∅ {C19, C11}

Por lo tanto con el conocimiento dado se puede saber que, en efecto, Pepe está hambriento.

En cada uso de la regla de resolución se realizó unificación de variables; no detallé las sustituciones hechas en cada paso pero en todos se trata de a lo sumo una sustitución fácil de deducir.

1.a. Tras pasar a forma normal, quedan las siguientes cláusulas

(C1) $\neg S(x_0) \vee T(Sk_0(x_0))$

(C2) $\neg S(x_1) \vee E(A, Sk_0(x_1))$

función de Skolem

(C3) $\neg W(x_2) \vee S(x_2)$

(C4) $W(A)$

(C5) $D(A)$

(C6) $T(x_3)$

(C7) $\neg D(x_4) \vee \neg E(x_5, x_6) \vee M(x_6)$

Más la negación de la conclusión:

(C8) $\neg T(Sk_1(x_7)) \vee \neg M(Sk_1(x_7))$ — función de Skolem ~~X/A~~

Al negarla queda un ∀. Al no haber escrito los pasos que existe ~~[illegible]~~ no puedo juzgar lo siguiente

Intentamos llegar a una contradicción por resolución:

(C9) $\neg M(Sk_1(x_7))$ $\{C8, C6\}$

(C10) $\neg D(x_4) \vee \neg E(x_5, Sk_1(x_7))$ $\{C9, C7[Sk_1(x_7)/x_6]\}$

(C11) $\neg E(x_5, Sk_1(x_7))$ $\{C10[A/x_4], C5\}$

[illegible] φ

Acá ya no se puede seguir. Podría pensarse en aplicar resolución con C11 y C2, pero no se pueden unificar los términos $Sk_1(x_7)$ y $Sk_0(x_1)$. Y todas las otras cláusulas no son relevantes porque no involucran E, que necesariamente tiene que ser contemplado para llegar a una solución.

1.b. El método de resolución se emplea con encadenamiento hacia atrás ("backward chaining") para probar si una fórmula deriva de premisas dadas. Con encadenamiento hacia adelante ("forward chaining") se puede, en cambio, generar un conjunto de fórmulas derivables de las premisas.

Sin embargo, no se puede encontrar el conjunto de todas las fórmulas derivables, en general. Esto implicaría que tal conjunto sea decidible, es decir, que se pueda determinar para cualquier fórmula si la misma es derivable o no, y por tanto (dado que la lógica de predicados es consistente), si la fórmula es verdadera o falsa.

3. Búsqueda a lo ancho:

Iter.	Nodo visitado	Nodos a visitar
0	A	D, F, G
1	D	F, G, J, H
2	F	G, J, H, C, E
3	G	J, H, C, E
4	J	H, C, E, K
5	H	C, E, K, B ✓
6	C	E, K, B
7	E	K, B, Z, W
8	K	B, Z, W, L
9	B	Z, W, L

Búsqueda avara (un tipo de búsqueda primero el mejor):

Iter.	Nodo visitado	Nodos a visitar
0	A	D, F, G
1	D	F, G, J, H
2	F	C, G, J, H
3	C	G, J, H ✓
4	G	J, H
5	J	H, K
6	H	B, K
7	B	K

5.a. $w = 0 \rightarrow f(n) = 2\,g(n)$

Cuando $w = 0$, f representa solo el costo real del recorrido ya hecho. Esto corresponde a la búsqueda de costo uniforme. ✓

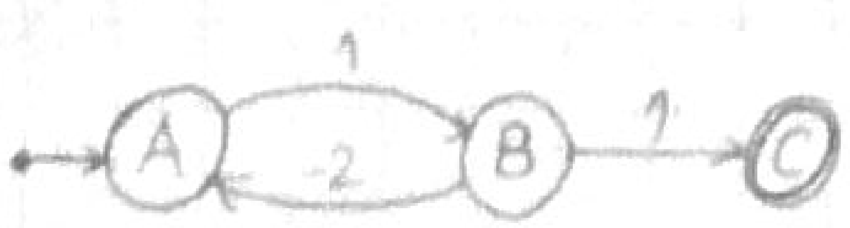
$w = 1 \rightarrow f(n) = g(n) + h(n)$

Acá f contempla tanto el costo real de lo ya recorrido como el costo estimado de lo que falte recorrer. Corresponde a la búsqueda A^* ✓

$w = 2 \rightarrow f(n) = 2\,h(n)$

En este caso, se considera para f solo la estimación de lo que falta. Es decir, se tiene la búsqueda avara. ✓

5.b. La búsqueda de costo uniforme es completa si g es creciente, es decir, si el costo de cada arista es no negativo. De lo contrario, se puede entrar en un bucle infinito; basta con ver el siguiente ejemplo:



Esta búsqueda es óptima porque siempre que va por un camino, se sabe que antes recorrió todos los caminos de costo menor al actual y no encontró solución, así que la primera solución que encuentre (y para la cual se detendrá) será la óptima. ✓

La búsqueda A^* es completa si se cumple la siguiente desigualdad para todo par de nodos n, n' tal que n sea padre de n' en el árbol de búsqueda:

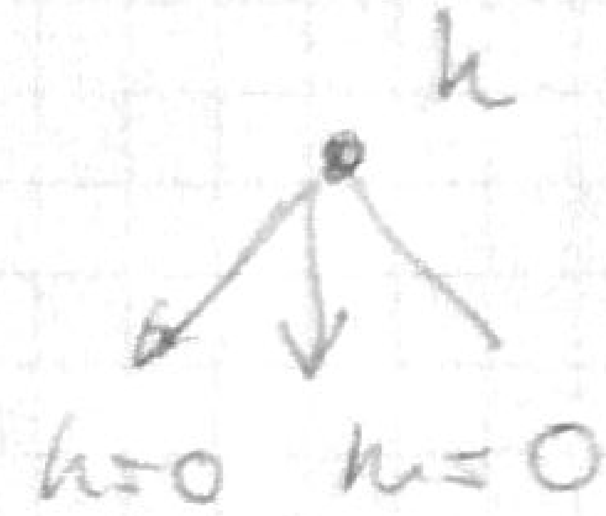
$$h(n) \leq h(n') + c(n,n') \quad \text{donde} \quad c(n,n') = g(n') - g(n)$$

Y es óptima si h es admisible, es decir, si nunca sobreestima el costo real. ✓

$$h(n) \leq c(n,m) \quad \text{donde } n \text{ es nodo cualquiera y } m \text{ es nodo meta}$$

La búsqueda avara no es óptima, salvo que h sea una estimación perfecta. Y no es completa, porque puede caer en bucles infinitos. ✓

4.b. Definimos h así:



$h : S \to S$

$h(\{(x,2)\}) = 0$

$h(\{(x,1),(\overset{y}{x},1)\}) = 0$

$h(s) = \text{máx dom } s - \text{mín dom } s \quad \text{si } \sum_{x \in \text{ran } s} x > 2$

O sea, mientras haya más de 2 pilas, h es la diferencia entre la altura de la más grande y la de la más chica.

Hay que destacar que h no es admisible. Que representen?

4.a. Espacio de estado: $S = \{1 \ldots 60\} \to \{1 \ldots 6\}$ alturas posibles de pilas?

Estados iniciales: s_i tales que dom $s_i \subseteq \{1 \ldots 10\} \wedge \#\text{ran } s = \#s$
es un estado

Estados finales: s_f tales que $\sum_{x \in \text{ran } s_f} x = 2$

Operaciones: juntar : $S \times \{1 \ldots 60\} \times \{1 \ldots 60\} \to S$

juntar$(s, n, m) = s'$

s' es similar a s salvo porque si s tiene (n, x), (m, y) y $(n+m, z)$, entonces s' tiene en cambio $(n, x-1)$, $(m, y-1)$ y $(n+m, z+1)$, y si $x-1=0$ o $y-1=0$, entonces el par respectivo se elimina.