Control 1
Inteligencia Artificial (CC52A)

9 de octubre de 2007

1.
- (a) Mencione (y explique en una linea) dos métodos distintos a resolución para verificar consecuencia lógica en lógica proposicional.

  Respuesta:

  Para respoinder si $KB \models \alpha$: (i) Fuerza bruta (tablas de verdad): enumerar todas las interpretaciones posibles y verificar si los modelo de KB también son modelos de α. (ii) Búsqueda de modelos: buscar un modelo para $KB \wedge \neg\alpha$, si existe entonces no se cumple la consecuencia lógica.

- (b) Diga la complejidad de verificar validez y satisfacibilidad de una fórmula proposicional en (a) CNF, (b) DNF y (c) general.

  Respuesta:

	CNF	DNF	General
Satisfacibilidad	$\Theta(2^n)$ (NPC)	$\Theta(n)$	$\Theta(2^n)$ (NPC)
Validez	$\Theta(n)$	$\Theta(2^n)$ (NPC)	$\Theta(2^n)$ (NPC)

2.
- (a) Dé un ejemplo de un conjunto de cláusulas en lógica de predicado para las cuales el procedimiento de refutación mediante resolución no termina.

  Respuesta:

  Ver ejemplo dado los apuntes (lógica de predicados 3).

- (b) Dada la siguiente sentencia: $\forall X \exists Y (P(X,Y)) \vee \forall W \exists Z (Q(W,Z)$. Diga (y justifique) cuáles de las siguientes cláusulas preservan satisfacibilidad:

  - (i) $\forall X \forall W (P(X, f(X)) \vee Q(W, g(X,W)))$,
  - (ii) $\forall X \forall W (P(X, f(X,W)) \vee Q(W, g(W)))$,
  - (iii) $\forall X \forall W (P(X, f(X)) \vee Q(W, g(W)))$.

  Respuesta:

  Las tres preservan satisfacibilidad. Las tres resultan de distintos procesos de eliminación de cuantificadores existenciales usando funciones de Skolem.

(i) En este caso transformamos a prenex, quedando:
$\forall X \exists Y \forall W \exists Z (P(X, Y)) \vee (Q(W, Z)$. Y luego introducimos funciones de Skolem.

(ii) En este caso transformamos a prenex y movemos hacia afuera los cuantificadores de W y Z (esto preserva equivalencia debido a que el alcanze de W, Z es distinto al alcance de X, Y), quedando:
$\forall W \exists Z \forall X \exists Y (P(X, Y)) \vee (Q(W, Z)$. Y luego introducimos funciones de Skolem.

(iii) En este caso Skolemizamos directamente (sin pasar a prenex). Esta es la Skolemización más eficiente.

3. - (a) Use resolución para demostrar que el siguiente conjunto de cláusulas no es satisfacible
   $\neg R(X, Y) \vee \neg R(Y, Z) \vee R(X, Z)$
   $\neg R(f(X), f(f(f(X))))$
   $R(X, f(X))$
   Respuesta:
   Primero, para evitar conflictos de variables renombramos las variables de la segunda y tercera cláusula y obtenemos:
   (1)$\neg R(X, Y) \vee \neg R(Y, Z) \vee R(X, Z)$
   (2)$\neg R(f(U), f(f(f(U))))$
   (3)$R(V, f(V))$
   Resolvemos 1 y 2 con el unificador $\{X/f(U), Z/f(f(f(U)))\}$ obteniendo:
   (4)$\neg R(f(U), Y) \vee \neg R(Y, f(f(f(U))))$
   Resolvemos 3 y 4 con el unificador $\{V/f(U), Y/f(f(U))\}$ obteniendo:
   (5)$\neg R(f(f(U)), f(f(f(U))))$
   Finalmente, resolvemos 3 y 5 con el unificador $\{V/f(f(U))\}$ obteniendo la cláusula vacía.

   - (b) Considere el siguiente enunciado: "Toda persona que pasa el examen de IA y gana la loteria es feliz. Pero todo el que estudia o es afortunado puede pasar todos sus examenes. Juan no estudia pero es afortunado. Todo el que es afortunado gana la lotería.". Aplique resolución para verificar si Juan es feliz.
   Respuesta:
   Primero, escribimos los enunciados en lógica de predicados:
   1. $\forall X((PasaE(X, IA) \wedge GanaLot(X)) \Rightarrow Feliz(X))$
   2. $\forall X, Y((Estudia(X) \vee Afortunado(X)) \Rightarrow PasaE(X, Y))$
   3. $\neg Estudia(Juan) \wedge Afortunado(Juan)$
   4. $\forall X(Afortunado(X) \Rightarrow GanaLot(X)$
   Objetivo negado:
   5. $\neg Feliz(Juan)$

Luego, transformamos a cláusulas:
1. $(\neg(PasaE(X, IA) \vee \neg GanaLot(X)) \vee Feliz(X))$
2.1 $(\neg(Estudia(X) \vee PasaE(X, Y))$
2.2 $(\neg Afortunado(X)) \vee PasaE(X, Y))$
3.1 $\neg Estudia(Juan)$
3.2 $Afortunado(Juan)$
4. $(\neg Afortunado(X) \vee GanaLot(X))$
5. $\neg Feliz(Juan)$
Luego, aplicamos resolución:
Res. 1 y 5, obtenemos 6. $(\neg(PasaE(Juan, IA) \vee \neg GanaLot(Juan))$
Res. 6 y 4, obtenemos 7. $(\neg(PasaE(Juan, IA) \vee \neg Afortunado(Juan))$
Res 7 y 2.2, obtenemos 8 $\neg Afortunado(Juan)$
Res 8 y 3.2 obtenemos ().

4. Escribe el predicado *sort* en Prolog que ordena una lista. Por ejemplo, la llamada a *sort*([1,6,3,4],L) retorna L=[1,3,4,6].

   Respuesta:

   Se puede implementar cualquier algoritmo de ordenamiento (naive, insert, bubble, merge, quick, etc). Todos ellos están implementados en prolog en:

   http://ktiml.mff.cuni.cz/ bartak/prolog/sorting.html