ALEJANDRO SANTURUM VARELA 44190946-S Gr 320 (DG)

1ª ENTREGA EJERCICIOS I.A

1

a) Formalización base de conocimiento

I) $\forall m \; \neg Igual(0, suc(m))$

II) $\forall n \left[\neg Igual(n,0) \Rightarrow \exists m \; Igual(n, suc(m)) \right]$

III) $\neg Par(1) \wedge Igual(1, suc(0)) \equiv$ $\neg Par(1)$ / $Igual(1, suc(0))$

IV) $\forall n \left[\neg Par(suc(n)) \Rightarrow Par(n) \right]$

V) $\forall n \left[Par(suc(n)) \Rightarrow \neg Par(n) \right]$

VI) $\forall n \left[Igual(sum(n,0), n) \right]$

VII) $\forall n,m \left[Igual(suc(n,m), sum(n, suc(m))) \right]$

VIII) $\forall m,n \left[(Igual(n,m) \wedge Par(m)) \Rightarrow Par(n) \right]$ (adicional por el enunciado)

b) Conversión a FNC

I) $\forall m \left[\neg Igual(0, suc(m)) \right]$

- Eliminación del $\forall$: $\boxed{\neg Igual(0, suc(m)) \quad [1]}$

II) $\forall n \left[\neg Igual(n,0) \Rightarrow \exists m \; Igual(n, suc(m)) \right]$

- Eliminación del $\Rightarrow$ + reducción ámbito $\neg$: $\forall n \left[Igual(n,0) \vee \exists m \; Igual(n, suc(m)) \right]$
- Skolemnización: $\forall n \left[Igual(n,0) \vee Igual(n, suc(f(n))) \right]$
- Eliminación del $\forall$: $\boxed{Igual(n,0) \vee Igual(n, suc(f(n))) \quad [2]}$

III) $\neg Par(1) \wedge Igual(1, suc(0))$ $\boxed{\neg Par(1) \quad [3]}$ $\equiv$ $\boxed{\neg Par(suc(0)) \quad [3]}$ $\boxed{Igual(1, suc(0)) \quad [4]}$ PEANO

IV) $\forall n \left[\neg Par(suc(n)) \Rightarrow Par(n) \right]$

- Eliminación del $\Rightarrow$ + reducción ámbito $\neg$: $\forall n \left[Par(suc(n)) \vee Par(n) \right]$
- Eliminación del $\forall$: $\boxed{Par(suc(n)) \vee Par(n) \quad [5]}$

V) $\forall n\ [Par(suc(n)) \Rightarrow \neg Par(n)]$

- Eliminación del $\Rightarrow$ $\forall n\ [\neg Par(suc(n)) \vee \neg Par(n)]$
- Eliminación de $\forall$: $\boxed{\neg Par(suc(n)) \vee \neg Par(n) \quad [6]}$

VI) $\forall n\ [Igual(sum(n,0), n)]$

- Eliminación del $\forall$: $\boxed{Igual(sum(n,0), n) \quad [7]}$

VII) $\forall n,m\ [Igual(suc(sum(n,m)), sum(n, suc(m)))]$

- Eliminación del $\forall$: $\boxed{Igual(suc(sum(n,m)), sum(n, suc(m))) \quad [8]}$

VIII) $\forall n,m\ [(Igual(n,m) \wedge Par(m)) \Rightarrow Par(n)]$

- Eliminación del $\Rightarrow$: $\forall n,m\ [\neg(Igual(n,m) \wedge Par(m)) \vee Par(n)]$
- Ley de De Morgan: $\forall n,m\ (\neg Igual(n,m) \vee \neg Par(m) \vee Par(n))$
- Eliminación del $\forall$: $\boxed{\neg Igual(n,m) \vee \neg Par(m) \vee Par(n) \quad [9]}$

c) Demostrar que la suma de un nº natural par con uno es impar.

$\forall n\ [Par(n) \Rightarrow \neg Par(sum(n, suc(0)))]$ (META)

Negamos la META:

$\neg \forall n\ [Par(n) \Rightarrow \neg Par(sum(n, suc(0)))]$

Y la transformamos en FNC:

- Eliminación del $\Rightarrow$: $\neg \forall n\ [\neg Par(n) \vee \neg Par(sum(n, suc(0)))]$
- Ámbito $\neg$: $\exists n\ Par(n) \wedge Par(sum(n, suc(0)))$
- Skolemnización: $Par(sk_1) \wedge Par(sum(sk_1, suc(0)))$

$\boxed{Par(sk_1) \quad [10]}$

$\boxed{Par(sum(sk_1, suc(0))) \quad [11]}$

Utilizamos ahora resolución:

$$\begin{matrix}[10]\\ [6]\end{matrix} \left| \frac{n := SK_1}{RES_{\neg Par(SK_1)}} \right. \quad \neg Par(suc(SK_1)) \quad [12]$$

$$\begin{matrix}[11]\\ [8]\end{matrix} \left| \frac{}{equiv} \right. \quad Par(suc(sum(SK_1, 0))) \quad [13]$$

$$\begin{matrix}[13]\\ [7]\end{matrix} \left| \frac{}{equiv.} \right. \quad Par(suc(SK_1)) \quad [14]$$

$$\begin{matrix}[12]\\ [14]\end{matrix} \left| \frac{}{RES_{Par(suc(SK_1))}} \right. \quad \square \quad \text{Falso} \Rightarrow \text{demostrado} \quad \blacksquare$$

d) TRUCO DE GREEN determinar si existe algún número natural par que no sea sucesor de un número impar y cuál es.

META: $\exists n \left[Par(n) \wedge \left(\neg \exists m \left(\neg Par(m) \wedge Igual(n, suc(m)) \right) \right) \right]$

NEG META: $\neg \exists n \left[Par(n) \wedge \left(\neg \exists m \left(\neg Par(m) \wedge Igual(n, suc(m)) \right) \right) \right]$
+ FNC

- Ámbito negaciones + distributiva:
  $\forall n \left[\neg Par(n) \vee \left(\neg Par(f(n)) \wedge Igual(n, suc(f(n))) \right) \right]$
- Elim $\forall$ + Respuesta(n) truco de Green:

$$\boxed{\neg Par(n) \vee \neg Par(f(n)) \vee Res(n) \quad [10]}$$

$$\boxed{\neg Par(n) \vee Igual(n, suc(f(n))) \vee Res(n) \quad [11]}$$

$$\begin{matrix}[5]\\ [10]\end{matrix} \left| \frac{}{RES_{Par(n)}} \right. \quad Par(suc(n)) \vee \neg Par(f(n)) \vee Res(n) \quad [12]$$

$$\begin{matrix}[5]\\ [11]\end{matrix} \left| \frac{}{RES_{Par(n)}} \right. \quad Par(suc(n)) \vee Igual(n, suc(f(n))) \vee Res(n) \quad [13]$$

$$\begin{matrix}[12]\\ [3]\end{matrix} \left| \frac{n := 0}{RES_{Par(suc(0))}} \right. \quad \neg Par(f(0)) \vee Res(0) \quad [14]$$

$$\begin{matrix}[13]\\ [3]\end{matrix} \left| \frac{n := 0}{RES_{Par(suc(0))}} \right. \quad Igual(0, suc(f(0))) \vee Res(0) \quad [15]$$

$$\begin{matrix}[1]\\ [15]\end{matrix} \left| \frac{}{RES_{Igual(..)}} \right. \quad Res(0)$$

$\Downarrow$

Según el truco de Green, lo cumple el 0.

Añadimos a la base de conocimiento inicial la negación de la siguiente meta:

$$\forall x,y \left[\left(P(x) \land P(y) \land \neg \exists z \ suc(x,z) \land \neg \exists w (y,w) \right) \Rightarrow (x = y) \right]$$

Llegamos a que el 0 es el único neutro.

2. Resolución de un problema con resolución + refutación:

1er paso: Indicar constantes, predicados y funciones usadas

CONSTANTES: Groucho (persona)

VARIABLES: $x_1, x_2, \ldots$ (personas)

$c_1, c_2, \ldots$ (clubs)

PREDICADOS

$SC(x_1, x_2)$: evalúa a verdadero si x_1 es como x_2

$SM(x_1, c_1)$: evalúa a verdadero si x_1 es miembro de c_1.

$QSM(x_1, c_1)$: evalúa a verdadero si x_1 quiere ser miembro de c_1.

$A(c_1, x_1)$: evalúa a verdadero si c_1 ha aceptado/acepta a x_1.

2º paso: Formalización de la base de conocimiento

I) $\forall x \; SC(x,x)$

II) $\forall x,c \left[SM(x,c) \Longleftrightarrow (QSM(x,c) \wedge A(c,x)) \right]$

III) Groucho ya es una constante en el ámbito de personas

IV) $\forall x,c \left[(SC(x, Groucho) \wedge A(c,x)) \Longrightarrow \neg QSM(Groucho, c) \right]$

3er paso: Conversión a FNC

I) $\boxed{SC(x,x) \quad [1]}$

II) $\forall x,c \left\{ \left[SM(x,c) \Rightarrow (QSM(x,c) \wedge A(c,x)) \right] \wedge \left[(QSM(x,c) \wedge A(c,x)) \Rightarrow SM(x,c) \right] \right\}$

$\forall x,c \left\{ \left[\neg SM(x,c) \vee (QSM(x,c) \wedge A(c,x) \right] \wedge \left[\neg QSM(x,c) \vee \neg A(c,x) \vee SM(x,c) \right] \right\}$

$(\neg SM(x,c) \vee QSM(x,c)) \wedge (\neg SM(x,c) \vee A(c,x)) \wedge (\neg QSM(x,c) \vee \neg A(c,x) \vee SM(x,c))$

Entonces:

$\boxed{\neg SM(x,c) \vee QSM(x,c) \quad [2]}$

$\boxed{\neg SM(x,c) \vee A(c,x) \quad [3]}$

$\boxed{\neg QSM(x,c) \vee \neg A(c,x) \vee SM(x,c) \quad [4]}$

III) $\forall x,c\ [\neg SC(x, Groucho) \vee \neg A(c,x) \vee \neg QSM(Groucho, c)]$

$\boxed{\neg SC(x, Groucho) \vee \neg A(c,x) \vee \neg QSM(Groucho, c) \quad [5]}$

4º paso: Formalizamos meta, la negamos y la convertimos a FNC

META: $\forall c\ \neg SM(Groucho, c)$

META NEGADA: $\exists c\ SM(Groucho, c)$

FNC: $\boxed{SM(Groucho, SK_c) \quad [6]}$

5º paso: Resolución + refutación buscando la cláusula vacía

$[3], [6] \xrightarrow[RES_{SM}]{x := Groucho,\ c := SK_c}$ $A(SK_c, Groucho)$ [7]

$[2], [6] \xrightarrow[RES_{SM}]{x := Groucho,\ c := SK_c}$ $QSM(Groucho, SK_c)$ [8]

$[4], [8] \xrightarrow[RES_{QSM}]{x := Groucho,\ c := SK_c}$ $\neg A(SK_c, Groucho) \vee SM(x, c)$ [9]

$[7], [9] \xrightarrow[RES_{A}]{x := Groucho,\ c := SK_c}$ $SM(Groucho, SK_c)$ [10] (inútil, ya la tenía)

$[1], [5] \xrightarrow[RES_{SC}]{x := Groucho}$ $\neg A(c, Groucho) \vee \neg QSM(Groucho, c)$ [11]

$[8], [11] \xrightarrow[RES_{QSM}]{c := SK_c}$ $\neg A(SK_c, Groucho)$ [12]

$[7], [12] \xrightarrow[RES_{A}]{}$ $\square$ $\Longrightarrow$ acabamos de demostrar que Groucho NO es miembro de ningún club.

3 Resolución de un problema con resolución + refutación:

1er paso: Indicar constantes, variables, predicados y funciones

CONSTANTES: 0

VARIABLES: x, y, z (números reales)

PREDICADOS: Igual(x,y) : evalúa a T si x = y

Neutro(x) : evalúa a T si x es el elem neutro de la suma

FUNCIONES: Resta(x,y) : evalúa al número (x − y).

2º paso: Formalización de la base de conocimiento

I) $\forall x,y \left[Igual(Resta(x,y), 0) \Leftrightarrow Igual(x,y) \right]$

II) $\forall x \left[Neutro(x) \Leftrightarrow \left(\forall y \; Igual(Resta(y,x), y) \right) \right] \equiv$

$\equiv \forall x,y \left[Neutro(x) \Leftrightarrow Igual(Resta(y,x), y) \right]$

3er paso: Conversión a FNC

I) $\forall x,y \left[(Igual(Resta(x,y),0) \Rightarrow Igual(x,y)) \wedge (Igual(x,y) \Rightarrow Igual(Resta(x,y),0)) \right]$

$\forall x,y \left\{ \left[\neg Igual(Resta(x,y),0) \vee Igual(x,y) \right] \wedge \left[\neg Igual(x,y) \vee Igual(Resta(x,y),0) \right] \right\}$

$\boxed{\neg Igual(Resta(x,y),0) \vee Igual(x,y) \quad [1]}$

$\boxed{\neg Igual(x,y) \vee Igual(Resta(x,y),0) \quad [2]}$

II) $\forall x,y \left[(Neutro(x) \Rightarrow Igual(Resta(y,x),y)) \wedge (Igual(Resta(y,x),y) \Rightarrow Neutro(x)) \right]$

$\forall x,y \left\{ \left[\neg Neutro(x) \vee Igual(Resta(y,x),y) \right] \wedge \left[\neg Igual(Resta(y,x),y) \vee Neutro(x) \right] \right\}$

$\boxed{\neg Neutro(x) \vee Igual(Resta(y,x),y) \quad [3]}$

$\boxed{\neg Igual(Resta(y,x),y) \vee Neutro(x) \quad [4]}$

4° paso Formalizamos la meta, la negamos y la convertimos a FNC

META: $\forall x \left[Neutro(x) \Rightarrow Igual(x,0) \right]$

META NEGADA $\exists x \left[\neg \left(Neutro(x) \Rightarrow Igual(x,0) \right) \right]$

FNC: $\exists x \left[\neg \left(\neg Neutro(x) \vee Igual(x,0) \right) \right]$

$\exists x \left(Neutro(x) \wedge \neg Igual(x,0) \right)$

$\boxed{Neutro(SK) \quad [5]}$ $\quad$ $\boxed{\neg Igual(SK,0) \quad [6]}$

[3] [5] | $x := SK$, RES_{Neutro} $\quad Igual(Resta(y,x), y) \quad [7]$

[4] [6] | $x := SK$, $y := 0$, RES_{Igual} $\quad \neg Igual(Resta(SK,0), 0) \quad [8]$

[4] [7] | $y := 0$, $RES_{Igual(Resta\ldots)}$ $\quad Igual(x,0) \quad [9]$

$\neg Igual(Resta(x,y), 0) \vee Igual(x,y)$
$Igual(Resta(x,y), y)$
¿con sustitución $y = 0$ podemos hacer resolución y obtener $\neg Igual(Resta(x,0),0)$?

[6] [9] | $x := SK$, RES_{Igual} $\quad \square \Rightarrow$ acabamos de demostrar que no hay ningún elemento neutro aparte del cero.

4

Primero programaremos un predicado que evalúe a verdadero cuando se le pase el mínimo de la lista, e.d list_min(L, A) evalúa a verdadero sii A es el mínimo elemento de L:

```
list_min([L|Ls], Min) :- list_min(Ls, L, Min).
list_min([], Min, Min).
list_min([L|Ls], Min0, Min) :-
        Min1 is min(L, Min0),
        list_min(Ls, Min1, Min).
```

Otro predicado auxiliar delete_one(A, L, Ls) evalúa a verdadero sii A es el elemento eliminado de la lista L, resultando en Ls.

```
delete_one(_, [], []).
delete_one(Term, [Term|Tail], Tail).
delete_one(Term, [Head|Tail], [Head|Result]) :-
        delete_one(Term, Tail, Result).
```

Ahora ya podemos hacer los predicados propuestos.

```
minimo(Ls, A, Ys) :- list_min(Ys, A), delete_one(A, Ls, Ys).

ordenar([], []).
ordenar(L, [M|Rs]) :- minimo(L, M, Rest), ordenar(Rest, Rs).
```

5. Vamos a crear dos predicados auxiliares antes de find: indexOf(Ls, Elem, Index) evalúa a verdadero si la primera aparición de Elem en Ls es Index (empezando en 1). Por otro lado, checkFirstOccur(Ls, E, N, Rest) evalúa a verdadero si la primera aparición del elemento E en Ls es en el índice N y la lista restante es Rest

```
indexOf([Element|_], Element, 1) :- !.
indexOf([_|Tail], Element, Index) :-
    indexOf(Tail, Element, Index1),
    Index is Index1 + 1.

checkFirstOccur(WholeList, Elem, Pos, Rest) :-
    checkFirstOccur(WholeList, Elem, Pos, Rest, WholeList).
checkFirstOccur(WholeList, Elem, Pos, R, [Elem|R]) :-
    indexOf(WholeList, Elem, Pos).
checkFirstOccur(WholeList, Elem, Pos, Rest, [_|R2]) :-
    checkFirstOccur(WholeList, Elem, Pos, Rest, R2).

find(Ls, E, Rs) :- find(Ls, E, Rs, 0).
find(Ls, E, [], _) :- \+ member(E, Ls).
find(Ls, E, [N|M], Acum) :-
    checkFirstOccur(Ls, E, Naux, Rest),
    N is Naux + Acum,
    find(Rest, E, M, N).
```

6. Para este ejercicio vamos a crear un predicado auxiliar llamado checkElem(Ls, I, Elem) que evalúa a verdadero sii I está presente en Ls en el índice I.

```
checkElem([Elem|_], 1, Elem).
checkElem([_|Rest], I, Elem) :-
    NewI is I-1,
    checkElem(Rest, NewI, Elem).

seleccionar(_, [], []).
seleccionar(Ls, [I|Is], [R|Rs]) :-
    checkElem(Ls, I, R),
    seleccionar(Ls, Is, Rs).
```

7. Como no podía ser de otra forma, para este ejercicio vamos a implementar un predicado auxiliar checkRep(L, R, Rest) que evaluará a T sii L aparece al principio de R y Rest es el resto de R, sin L.

```
checkRep([], R, R).
checkRep([L|Ls], [L|Rs], Rest) :- checkRep(Ls, Rs, Rest).

replicar(_, 0, []) :- !.
replicar(Ls, N, Rs) :-
    checkRep(Ls, Rs, Rest),
    NewN is N-1,
    replicar(Ls, NewN, Rest).
```