PLP - Segundo Parcial - 2do cuatrimestre de 2024

#Orden	Nro. Libreta	Apellido y Nombre	Ej1	Ej2	Ej3	Nota Final

Este examen se aprueba obteniendo al menos dos ejercicios bien (`B`) y uno regular (`R`), y se promociona con al menos dos ejercicios muy bien (`MB`) y uno bien (`B`). Es posible obtener una aprobación condicional con un ejercicio muy bien (`MB`), uno bien (`B`) y uno insuficiente (`I`), pero habiendo entregado algo que contribuya a la solución del ejercicio. Las notas para cada ejercicio son: `-`, `I`, `R`, `B`, `MB`, `E`. Entregar cada ejercicio en hojas separadas. Poner nombre, apellido y número de orden en todas las hojas, y numerarlas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras. El orden de los ejercicios es arbitrario. Recomendamos leer el parcial completo antes de empezar a resolverlo.

Ejercicio 1 - Programación Lógica

Implementar los predicados respetando en cada caso la instanciación pedida. Los generadores deben cubrir todas las instancias válidas de aquello que generan sin repetir dos veces la misma. Se deben indicar los patrones de instanciación de todos los predicados auxiliares. No usar cut (`!`) ni predicados de alto orden como `setof`, con la única excepción de `not`.

a) Definir el predicado `subsecuenciaCreciente(+L,-S)` que es verdadero cuando `S` es una subsecuencia estrictamente creciente de elementos de `L`. Notar que la secuencia respeta el orden de aparición en `L`. Por ejemplo:

```
?- subsecuenciaCreciente([4,8,1,9],S).
 S = [4, 8, 9] ;                          S = [8] ;
 S = [4, 8] ;                             S = [1, 9] ;
 S = [4, 9] ;                             S = [1] ;
 S = [4] ;                                S = [9] ;
 S = [8, 9] ;                             S = [].
```

b) Definir el predicado `subsecuenciaCrecienteMasLarga(+L,-S)` que es verdadero cuando `S` es la subsecuencia estrictamente creciente de mayor longitud de `L`. Puede haber más de un resultado. Por ejemplo:

```
?- subsecuenciaCrecienteMasLarga([5,6,7,2,8,1,2,3,4,5,7],S).
 S = [1,2,3,4,5,7] ;
 false;

?- subsecuenciaCrecienteMasLarga([5,6,7,2,8,0,2,3,4],S).
 S = [5,6,7,8] ;
 S = [0,2,3,4] ;
 false.
```

c) Definir el predicado `fibonacci(-X)` que instancia en `X` los números pertenecientes a la secuencia de Fibonacci. Por ejemplo:

```
?- fibonacci(X).
 X = 1;                                   X = 3;
 X = 1;                                   X = 5;
 X = 2;                                   ...
```

d) ¿Es reversible el predicado anterior? Justificar.

Ejercicio 2 - Resolución

a) Representar en forma clausal las siguientes fórmulas de lógica de primer orden, que tratan acerca de términos cerrados del cálculo lambda:

i) $\forall T_1.\forall T_2.\forall M.\forall N.((\mathsf{Tipo}(M, T_1 \to T_2) \land \mathsf{Tipo}(N, T_1)) \implies \mathsf{Tipo}(\mathsf{app}(M, N), T_2))$
Si el tipo de un término es $T_1 \to T_2$, y el tipo de otro término es T_1, entonces el tipo de la aplicación es T_2.

ii) $\exists M.\mathsf{Tipo}(M, \alpha \to (\beta \to \gamma))$
Existe un término de tipo $\alpha \to (\beta \to \gamma)$.

iii) $\exists M.\mathsf{Tipo}(M, \alpha \to \beta)$
Existe un término de tipo $\alpha \to \beta$.

iv) $\exists M.\mathsf{Tipo}(M, \alpha)$
Existe un término de tipo α.

Ayuda: se puede pensar en el constructor de tipos $\rightarrow$ como una función binaria, por ejemplo el término $T_1 \rightarrow T_2$ se puede pensar como $\mathsf{flecha}(T_1, T_2)$. Además, α, β y γ son constantes distintas entre sí.

b) Utilizando resolución, determinar si la siguiente fórmula es consecuencia del conjunto anterior: $\exists M.\mathsf{Tipo}(M, \gamma)$ *(Existe un término de tipo γ).*

Indicar la sustitución utilizada en cada paso. Es importante tener un plan (escrito o en la cabeza).

c) ¿Fue SLD la resolución utilizada en el punto anterior? Justificar.

Ejercicio 3 - Inferencia y Deducción Natural

a) Consideremos el Cálculo Lambda tipado extendido con árboles ternarios:

$\tau ::= \cdots \mid \mathtt{AT}(\tau)$

$M ::= \cdots \mid \mathtt{TNil}_\tau \mid \mathtt{Tern}(M, M, M, M) \mid \mathsf{foldAT}\ M \triangleright\ \mathsf{TNil}\ \leadsto M;\ \mathsf{Tern}(x, ri, rm, rd) \leadsto M$

Se extiende también el algoritmo de inferencia para árboles de la siguiente manera:

$\mathbb{W}(\mathtt{TNil}) \stackrel{def}{=} \emptyset \vdash \mathtt{TNil}_X : \mathtt{AT}(X)$ con X variable fresca

$\mathbb{W}(\mathtt{Tern}(U_1, U_2, U_3, U_4)) \stackrel{def}{=} S\Gamma_1 \cup S\Gamma_2 \cup S\Gamma_3 \cup S\Gamma_4 \vdash S(\mathtt{Tern}(M_1, M_2, M_3, M_4)) : S\sigma_4$
donde:

- $\mathbb{W}(U_1) = \Gamma_1 \vdash M_1 : \sigma_1$
- $\mathbb{W}(U_2) = \Gamma_2 \vdash M_2 : \sigma_2$
- $\mathbb{W}(U_3) = \Gamma_3 \vdash M_3 : \sigma_3$
- $\mathbb{W}(U_4) = \Gamma_4 \vdash M_4 : \sigma_4$
- $S = \mathrm{mgu}\ (\{\sigma_2 \stackrel{?}{=} \mathtt{AT}(\sigma_1),\ \sigma_3 \stackrel{?}{=} \sigma_2,\ \sigma_4 \stackrel{?}{=} \sigma_2\} \cup$
  $\{\tau \stackrel{?}{=} \rho \mid x : \tau \in \Gamma_i\ \wedge\ x : \rho \in \Gamma_j\ \wedge\ i, j \in \{1, 2, 3, 4\}\})$

$\mathbb{W}(\mathsf{foldAT}\ U_1 \triangleright\ \mathsf{TNil}\ \leadsto U_2;\ \mathsf{Tern}(x, ri, rm, rd) \leadsto U_3) \stackrel{def}{=} S\Gamma_1 \cup S\Gamma_2 \cup S\Gamma_{3'} \vdash S(\mathsf{foldAT}\ M_1 \triangleright\ \mathsf{TNil}\ \leadsto M_2;\ \mathsf{Tern}(x, ri, rm, rd) \leadsto M_3) : S\sigma_2$
donde:

- $\mathbb{W}(U_1) = \Gamma_1 \vdash M_1 : \sigma_1$
- $\mathbb{W}(U_2) = \Gamma_2 \vdash M_2 : \sigma_2$
- $\mathbb{W}(U_3) = \Gamma_3 \vdash M_3 : \sigma_3$
- $\tau_x = \begin{cases} \alpha_1 \text{ si } x : \alpha_1 \in \Gamma_3, \\ \text{variable fresca si no} \end{cases}$
- $\tau_{rm} = \begin{cases} \alpha_3 \text{ si } rm : \alpha_3 \in \Gamma_3, \\ \text{variable fresca si no} \end{cases}$
- $\tau_{ri} = \begin{cases} \alpha_2 \text{ si } ri : \alpha_2 \in \Gamma_3, \\ \text{variable fresca si no} \end{cases}$
- $\tau_{rd} = \begin{cases} \alpha_4 \text{ si } rd : \alpha_4 \in \Gamma_3, \\ \text{variable fresca si no} \end{cases}$
- $\Gamma_{3'} = \Gamma_3 \ominus \{x, ri, rm, rd\}$
- $S = \mathrm{mgu}\ (\{\sigma_1 \stackrel{?}{=} \mathtt{AT}(\tau_x),\ \sigma_2 \stackrel{?}{=} \sigma_3,\ \tau_{ri} \stackrel{?}{=} \sigma_2,\ \tau_{rm} \stackrel{?}{=} \sigma_2,\ \tau_{rd} \stackrel{?}{=} \sigma_2\} \cup$
  $\{\tau \stackrel{?}{=} \rho \mid x : \tau \in \Gamma_i\ \wedge\ x : \rho \in \Gamma_j\ \wedge\ i, j \in \{1, 2, 3'\}\})$

Además, se considera extendido el algoritmo para la suma de naturales:

$\mathbb{W}(U_1 + U_2) \stackrel{def}{=} S\Gamma_1 \cup S\Gamma_2 \vdash S(M_1 + M_2) : Nat$
donde:

- $\mathbb{W}(U_1) = \Gamma_1 \vdash M_1 : \sigma_1$
- $\mathbb{W}(U_2) = \Gamma_2 \vdash M_2 : \sigma_2$
- $S = \mathrm{mgu}\ (\{\sigma_1 \stackrel{?}{=} Nat,\ \sigma_2 \stackrel{?}{=} Nat\}\ \cup \{\tau \stackrel{?}{=} \rho \mid x : \tau \in \Gamma_1\ \wedge\ x : \rho \in \Gamma_2\}\})$

Inferir el tipo de la siguiente expresión, o demostrar que no es tipable:

$$\mathsf{foldAT}\ \mathtt{Tern}(\underline{1}, \mathtt{TNil}, x, x) \triangleright\ \mathsf{TNil}\ \leadsto rd;\ \mathsf{Tern}(x, ri, rm, rd) \leadsto (ri + rm) + rd$$

b) Demostrar el siguiente teorema usando deducción natural, sin utilizar principios clásicos:

$$(\exists X.P(X)) \Rightarrow (\forall Y.(P(Y) \Rightarrow Q(Y))) \Rightarrow \exists Z.Q(Z)$$