Lógica Computacional, 2017-2

Práctica 4: Relación de α-equivalencia y Algoritmo de Martelli Montanari

Manuel Soto Romero   Víctor Zamora Gutiérrez

Fecha de inicio: 15 de marzo de 2017

Fecha de término: 29 de marzo de 2017

Instrucciones generales

- Completar de manera clara y ordenada las funciones del archivo `form.hs`.
- Para tener derecho a calificación, la práctica debe ejecutarse sin errores ni advertencias. No está permitido utilizar primitivas de Haskell que resuelvan directamente los ejercicios, ni modificar la firma de ninguna función.
- La entrega es por **equipos de 3 a 4 integrantes**. Seguir los lineamientos especificados en: `http://sites.ciencias.unam.mx/logica-computacional-2017-2/laboratorio/lineamientos`.

Parte I: Relaciones de α-equivalencia

Definición *Decimos que dos fórmulas φ_1, φ_2 son α-equivalentes y escribimos $\varphi_1 \sim_\alpha \varphi_2$ si y sólo si φ_1 y φ_2 difieren a lo más en los nombres de sus variables ligadas.*

Por ejemplo, las siguientes expresiones son α-equivalentes:

$$\forall x P(x,y) \rightarrow \exists y R(x,y,z) \sim_\alpha \forall w P(w,y) \rightarrow \exists v R(x,v,z) \sim_\alpha \forall z P(z,y) \rightarrow \exists u R(x,y,z)$$

Usando la α-equivalencia, la operación de sustitución en fórmulas se vuelve una función total por lo que siempre está definida. Por ejemplo, se tiene que:

$$\forall x (Q(x) \rightarrow R(z,x))[z := f(x)] = \forall y (Q(y) \rightarrow R(z,y))[z := f(x)] = \forall y (Q(y) \rightarrow R(f(x), y))$$

Ejercicios:

Usando como base el archivo `form.hs`, completar el cuerpo de las siguientes funciones:

1. Función `vAlfaEq :: FORM -> FORM -> Bool` que verifica si dos fórmulas son α-equivalentes. Ejemplo:

```
> vAlfaEq (PT "x" (EX "y" (Impl (Pr "P" [V "x"]) (Pr "P" [V "y"])))) (PT "w" (EX
"z" (Impl (Pr "P" [V "w"]) (Pr "P" [V "z"]))))
True
```

2. Función `renVL :: FORM -> FORM` que renombra las variables ligadas de un fórmula de manera que las listas de variables libres y ligadas sean ajenas. Eso es un caso particular para la función del ejercicio 3. Ejemplo:

```
> renVL $ Conj (PT "x" (Pr "P" [V "x"])) (Pr "P" [V "x"])
Conj (PT "x0" (Pr "P" [V "x0"])) (Pr "P" [V "x"]) (o equivalente)
```

3. Función `renVLconj :: FORM -> [Nombre] -> FORM` que renombra las variables ligadas de una fórmula de forma que los nombres sean ajenos a los de una lista dada. Ejemplo:

```
> renVLconj (EX "y" (Pr "P" [(F "f" [V "x", V "y", V "z"])])) ["y"]
(EX "y0" (Pr "P" [(F "F" [V "x", V "y0", V "z"])])) (o equivalente)
```

4. Función `apsubF2 :: FORM -> Sust -> FORM` que implementan la sustitución en fórmulas usando la α-equivalencia. Ejemplo:

```
> apsubF2 (Impl (PT "x" (PT "z" (Conj (Pr "F" [V "x", V "y"]) (Pr "G" [V
"z"])))) (Pr "G" [V "z"])) [("z", F "f" [V "x"])]
(Impl (PT "x0" (PT "z0" (Conj (Pr "F" [V "x0", V "y"]) (Pr "G" [V "z0"])))) (Pr
"G" [F "f" [V "x"]]))
```

Parte II: Algoritmo de unificación de Martelli Montanari

A continuación se describe el algoritmo de unificación de Materlli-Montanari:

- Entrada: un conjunto de ecuaciones $\{s_1 = r_1, ..., s_k = r_k\}$ tales que se desea unificar simultáneamente s_i con r_i, $1 \leq i \leq k$.
- Salida: un unificador más general μ tal que $s_i\mu = r_i\mu$ para toda $1 \leq i \leq k$.

Colocar como ecuaciones las expresiones a unificar, escoger una de ellas de manera no determinista que tenga la forma de alguna de las siguientes opciones y realizar la acción correspondiente:

Nombre de la regla	$t1 = t2$	Acción
Descomposición DESC	$fs_1...s_n = ft_1...t_n$	sustituir $\{s_i = t_i\}$
Des. fallida DFALLA	$fs_1...s_n = gt_1...t_n$ donde $g \neq f$	falla
Eliminación ELIM	$x = x$	eliminar
Intercambio SWAP	$t = x$ donde t no es una variable	sustituir por la ecuación $x = t$
Sustitución SUST	$x = t$ donde x no figura en t	eliminar $x = t$ y aplicar la sustitución $[x := t]$ a las ecuaciones restantes
Sustitución fallida SFALLA	$x = t$ donde x figura en t y $x \neq t$	falla

El algoritmo termina cuando no se puede llevar a cabo ninguna acción o cuando falla. En caso de éxito se obtiene el conjunto vacío de ecuaciones y el unificador más general se obtiene al componer todas las sustituciones usadas por la regla de sustitución en el orden en que se generaron.

Ejemplo Sean $f^{(2)}, g^{(1)}, h^{(2)}$ y $W = \{fgxhxu, fzhfyyz\}$. Mostramos el proceso de ejecución del algoritmo:

1. $\{fgxhxu = fzhfyyz\}$ Entrada.
2. $\{gx = z, hxu = hfyyz\}$ DESC, 1.
3. $\{z = gx, hxu = hfyyz\}$ SWAP, 2.
4. $\{hxu = hfyygx\}$ SUST, 3 $[z := gx]$.
5. $\{x = fyy, u = gx\}$ DESC, 4.
6. $\{u = gfyy\}$ SUST, 5 $[x := fyy]$
7. $\emptyset$ DESC, 6 $[u := gfyy]$

El unificador se obtiene al componer las sustituciones utilizadas desde el inicio.

$$\begin{aligned} \mu &= [z := gz][x := fyy][u := gfyy] \\ &= [z := gfyy, x := fyy][u := gfyy] \\ &= [z := gfyy, x := fyy, u := gfyy] \end{aligned}$$

Ejercicios:

Usando como base el archivo `form.hs`, completar el cuerpo de las siguientes funciones:

1. Función `simpSus :: Sust -> Sust` que dada una sustitución, elimina de ella los pares con componentes iguales correspondientes a sustituciones de la forma $x := x$. Ejemplo:

```
> simpSus [("x", V "x")]
[]
```

2. Función `compSus :: Sust -> Sust -> Sust` que dadas dos sustituciones regresa su composición. Ejemplo:

```
> compSus [(‘‘x’’,V ‘‘y’’),(‘‘y’’,V ‘‘z’’)] [(‘‘z’’, V ‘‘w’’)]
[(‘‘x’’,V ‘‘y’’),(‘‘y’’,V ‘‘z’’),(‘‘z’’, V ‘‘w’’)]
```

3. Función `unifica :: Termino -> Termino -> [Sust]` que dados términos, regresa una lista de sustituciones, de forma que:

   - Si t_1, t_2 no son unificables, la lista es vacía.
   - Si sí lo son, la lista contiene como único elemento al unificador correspondiente.

   Ejemplo:

```
> unifica (V "x") (F "f" [V "y"])
[("x", F "f" [V "y"])]
```

4. Función `unificaConj [Termino] -> [Sust]` que implementa el caso general para unificar un conjunto (lista) $W = \{t_1, ..., t_n\}$. Ejemplo:

```
> let a = F ‘‘f’’ [F ‘‘g’’ [V ‘‘x’’], F ‘‘h’’ [V ‘‘x’’, V ‘‘u’’]]
> let b = F ‘‘f’’ [V ‘‘z’’, F ‘‘h’’ [F ‘‘f’’ [V ‘‘y’’, V ‘‘y’’], V ‘‘z’’]]
> unificaConj [a,b]
[(‘‘z’’, F ‘‘g’’ [F ‘‘f’’ [V ‘‘y’’,V ‘‘y’’]]),
(‘‘x’’, F ‘‘f’’ [V ‘‘y’’, V ‘‘y’’]),
(‘‘u’’, F ‘‘g’’ [F ‘‘f’’ [V ‘‘y’’, V ‘‘y’’]])]
```

Punto extra:

- (1pt.) Definir la función `unificaLit :: FORM -> FORM -> [Sust]` que unifica dos literales.