T R A B A J O P R Á C T I C O I I I

M e l i S e b a s t i á n . R o d r í g u e z J e r e m í a s .

Análisis de Lenguajes de Programación

20 de marzo de 2017

Ejercicio 1

La derivación se encuentra en el archivo derivaciones.pdf .

Ejercicio 2

La función infer retorna un valor del tipo Either String Type porque, dado un término, se dará uno de los siguientes dos casos: Que tenga un tipo válido o no. Si no puede asignarle ningún tipo a un término, devuelve un left string detallando el error; caso contrario (el término tiene un tipo válido), utiliza el constructor right.

Si el tipo de retorno de infer fuera Type, tendríamos que agregar un tipo Error y todas las reglas para este tipo, para devolver un resultado adecuado al inferir tipos de expresiones mal tipadas; o diseñar otra forma de señalar las expresiones mal tipadas y algún detalle del error.

El operador

(>>=)

tiene este tipo:

(>>=) :: Either String Type ->(Type ->Either String Type) ->Either String Type

y esta definición

(>>=) v f = either Left f v

Donde:

either (Left x) f g = f x
either (Right x) f g = g x

Entonces (>>=) toma un valor del tipo Either String Type y analiza su estructura:

- Si es de la forma Left xs, retorna Left xs (lo mismo que recibió).
- Si es de la forma Right r, retorna f r , donde f es una funcion que recive un Type y devuelve un Either String Type.

Esto nos sirve cuando queremos inferir el tipo de algun término, y precisamos inferir recursivamente el tipo de subtérminos. En este caso podemos utilizar (>>=) para manejar las situaciones en que subtérminos están mal tipados (o no).

Ejercicio 3

Para añadir la construcción let, utilizamos un nuevo constructor de datos en el tipo LamTerm:

Let String LamTerm LamTerm

Modificamos el parser para capturar las expresiones let en este constructutor de LamTerm. Luego, decidimos modificar sólamente la función de conversión para interpretar un LamTer let x = e in t como un Term que represente aplición (x : T . t) e; utilizando infer para obtener T. El resto de las funciones no precisan ser modificadas.

Ejercicio 4

Modificamos todas las funciones para extender al operador as.

Ejercicio 5

Ver en derivaciones.pdf

Ejercicio 6

Modificamos todas las funciones para extender al tipo Unit.

Ejercicio 7

Agregamos reglas de evaluación:

$$\frac{t1 \to v1}{(t1\, t\ 2) \to (v1\, t\ 2)}$$

$$\frac{t2 \to v2}{(v1\, t\ 2) \to (v1\, v\ 2)}$$

$$fst\ (v1\, v\ 2) \to v1$$
$$snd\ (v1\, v\ 2) \to v2$$

Ejercicio 8

Análogamente extendimos las funciones.

Ejercicio 9

Ver en derivaciones.pdf

Ejercicio 10

Ejercicio 11