Práctica 3 – Conocimiento y Razonamiento Automatizado

Scheme – Aritmética modular

Introducción

El objetivo de esta práctica es implementar una aritmética modular de números enteros mediante lambda cálculo en Scheme, así como un conjunto de operaciones básicas.

Aritmética entera

Codificar enteros tomando como base una codificación de los naturales es relativamente sencillo. Para ello, dado un par (m,n), con m, n $\in \mathbb{N}$, consideraremos que representa al entero m-n.

Al igual que ocurre con otras codificaciones de los enteros, los pares no representan a los enteros de manera única, p.e. (n,m) codifica el mismo entero que (n+k,m+k).

En el fichero *enteros.rkt* de la carpeta Práctica 3 de la asignatura (Blackboard) se puede ver una codificación de los enteros usando λ-cálculo e implementada empleando *Scheme*. Dicha codificación incluye las siguientes operaciones:

- Suma y resta.
- Multiplicación.
- División euclídea.
- Cálculo del máximo común divisor.
- Relaciones de igualdad y de orden.
- Reducción a representante canónico, donde tomaremos (n,0) o (0,n) como representantes canónicos.

Objetivo de la práctica

El conjunto de los números modulo p, $\mathbb{Z}_p$, es el conjunto cociente de $\mathbb{Z}$ mediante la relación de equivalencia

n R m si y sólo si n-m es múltiplo de p

Teniendo en cuenta lo anterior y la codificación, relaciones de orden y operaciones de enteros definidas en el fichero enteros mencionado, se pide codificar en λ-cálculo $\mathbb{Z}_p$. Dicha codificación debe incluir las siguientes operaciones:

(a) Reducción a representante canónico.
(b) Aritmética: suma, producto.
(c) Decisión sobre la inversibilidad y cálculo del inverso en el caso de que exista.

Codificada la aritmética modular, se pide codificar las matrices 2x2 en $\mathbb{Z}_p$ con p primo. Esta codificación debe incluir las siguientes operaciones:

(d) Suma y producto.
(e) Determinante.
(f) Decisión sobre inversibilidad y cálculo de inversa y del rango.
(g) Cálculo de potencias (naturales) de matrices. Este cálculo se tiene que hacer usando el algoritmo binario para el cálculo de potencias, también conocido como exponenciación binaria.

Scheme y λ-cálculo – Definición de términos

Las necesidades de *Scheme* para esta práctica quedan cubiertas, salvo en el caso de la recursión, con la siguiente observación:

El λ-término $\lambda x.M$ se codifica en *Scheme* mediante `(lambda (x) M)`. En el caso de que necesitáramos dar un nombre a un término para su posterior reutilización, la forma de hacerlo sería la siguiente `(define termino (lambda (x) M))`.

Por ejemplo, siguiendo lo visto en clase (página 57 de los apuntes) uno puede definir:

```
(define true (lambda (x y) x)

(define false (lambda (x y) y))

(define if (lambda (p x y) (p x y)))

...
```

Éstos términos se pueden encontrar al principio del fichero *enteros.rkt* y son los únicos λ-términos no currificados que se admitirán en la práctica. El resto de expresiones deben estar currificadas.

Scheme y λ-cálculo – recursividad

El combinador de punto fijo **Y** ha de definirse aplicando una η-expansión (líneas 23 a 29 del fichero *enteros.rkt*).

```
;;;;; Combinador de punto fijo
(define Y
  (lambda (f)
    ((lambda (x) (f (lambda (v) ((x x) v))))
     (lambda (x) (f (lambda (v) ((x x) v)))))))
```

Aun definiendo así **Y**, la recursividad no funciona tal cual se ha visto en clase. En el mismo fichero se pueden encontrar algunos ejemplos sobre cómo se puede simular la recursión. Por ejemplo, en la definición del resto de la división euclídea (líneas 135-153):

```
(define restonataux
    (lambda (n)
        (lambda (m)
            ((Y (lambda (f)
                 (lambda (x)
                    ((((esmayoroigualnat x) m)
                        (lambda (no_use)
                            (f ((restanat x) m))
                        )
                        (lambda (no_use)
                            x
                        )
                    )
                        zero)    ; Pasa zero como argumento de no_use
                )
            ))
                n)  ; Pasa n como el valor inicial de x.
        )
))
```

(La simulación de la recursividad se ha extraído de esta dirección).

Detalles de la entrega

La práctica se podrá realizar en equipos de hasta 3 personas (las mismas que en prácticas anteriores).

Además del **código fuente del programa (.rkt)**, convenientemente estructurado y comentado, se deberá incluir una **breve memoria** (**PDF**) indicando los miembros del equipo, reparto de tareas entre los mismos, grado de cumplimiento de cada uno de los requisitos, errores y/o aspectos no implementados, así como las fuentes consultadas para la resolución de la práctica.

La entrega se realizará **a través de la plataforma** en un **único fichero** .**zip** antes de las **23:59 horas del día 16 de mayo de 2022**. El nombre del fichero será *NombreApellido1Apellido2.zip* de uno de los integrantes del grupo. Una vez recibidos los trabajos y establecido el número de equipos, se publicará una fecha con la hora de defensa para cada equipo.

La entrega de prácticas **copiadas,** total o parcialmente, supondrá el **suspenso del laboratorio para todos los alumnos implicados**.