Métodos Formales Industriales
Tarea Individual - El lenguaje Maude

Tomando como partida el módulo `BATTERY-MAUDE` definido en la Práctica 1 de la asignatura, se puede extender con nuevos operadores. Por ejemplo, el módulo `BATTERY-EXT` extiende el módulo funcional `BATTERY-MAUDE` con una nueva función `charge` que devuelve el tanto por cierto de la carga de la bateria:

```
fmod BATTERY-EXT is
  protecting BATTERY-MAUDE .
  protecting BOOL .
  protecting NAT .

  op charge : Battery -> Nat .

  var EBt : EBattery .
  var Bt : Battery .

  eq charge(EBt) = 0 .
  eq charge(EBt ^ o ^ Bt) = 100 + charge(Bt) .
  eq charge(EBt ^ + ^ Bt) = 50 + charge(Bt) .
endfm
```

Los módulos funcionales `BOOL` y `NAT`, que vienen predefinidos en el preludio de Maude, nos permiten importar y utilizar el tipo `Bool` y `Nat` y sus operaciones básicas.

En esta tarea individual se pide entregar un archivo `.maude` que contenga los módulos funcionales `BATTERY-MAUDE` y `BATTERY-EXT` anteriores y se añada un nuevo módulo `BATTERY-ALL` de sistema (en vez de funcional) que contenga la definición de tres nuevos operadores:

1. `op size : Battery -> Nat .`

   Un operador que indica cuántos elementos tiene la batería. Una ejecución de ejemplo sería la siguiente:

   ```
   Maude> reduce in BATTERY-ALL : size(- ^ o ^ o ^ o) .
   rewrites: 9 in 0ms cpu (0ms real) (~ rewrites/second)
   result NzNat: 4
   ```

2. `op half-charge? : Battery -> Bool .`

Un operador que comprueba si una batería tiene al menos un 50 % de carga. Para definir esta función es conveniente usar la función `_quo_` que devuelve el cociente de la división entre dos números naturales. Una ejecución de ejemplo sería la siguiente:

```
reduce in BATTERY-ALL : half-charge?(- ^ o ^ o ^ o) .
rewrites: 19 in 0ms cpu (0ms real) (1900000 rewrites/second)
result Bool: true
```

3. `op consume-all : Battery -> Battery .`

   Un operador que consume gradualmente y en orden aleatorio todas las celdas de la batería. Una ejecución de ejemplo sería la siguiente:

```
search in BATTERY-ALL : consume-all(- ^ o ^ o) =>* EBt .

Solution 1 (state 8)
states: 9  rewrites: 12 in 0ms cpu (0ms real) (141176 rewrites/second)
EBt --> - ^ - ^ -

No more solutions.
states: 9  rewrites: 14 in 0ms cpu (0ms real) (125000 rewrites/second)
```

Notas:

1. El archivo `.maude` entregado no debe tener errores de sintaxis.
2. El archivo entregado debe incluir el módulo `BATTERY-MAUDE` original.
3. En la entrega de la tarea se deben adjuntar capturas como las incluidas arriba, incluyendo una nueva con la salida del siguiente comando:

```
show search graph .
```