Práctica 1: Ignacio Ferrer y Alejandro Tauste

Pregunta 1

Con el comando: *red in BATTERY-HASKELL : consume(- ^ - ^ nil)* .

Se obtiene el siguiente resultado:

`result Battery: - ^ - ^ consume(nil)`

Habría que añadir una ecuación para que en el caso de que la pila ya este vacía no ejecute la operación consume añadiendo la siguiente ecuación:

```
eq consume(nil) = nil .
```

Obteniendo:

`result Battery: - ^ - ^ nil`

Pregunta 2

Un ejemplo de tipo EBattery pero no ECell:

`EBattery: nil`

Un ejemplo de tipo Battery pero no EBattery ni Cell:

`Battery: - ^ -`

Pregunta 3

Definimos el subtipo ECell de Cell para diferenciar entre una celda que contiene contiene batería de una que está vacía, con esta implementación evitaremos que al final de la expresión resultante aparezca nil.

Pregunta 4

Para una batería de 2 celdas el conjunto de valores que puede adoptar una variable de tipo EBattery son:

`EBattery : - ^ -`

Para una batería de 2 celdas el conjunto de valores que puede adoptar una variable de tipo Battery son:

```
Battery : o ^ o
Battery : o ^ +
Battery : o ^ -
Battery : + ^ o
Battery : - ^ o
Battery : + ^ +
Battery : + ^ -
Battery : - ^ +
```

Pregunta 5

Obtenemos un tipo EBattery más concretamente:

```
result EBattery: - ^ -
```

Obtenemos este resultado de tipo EBattery ya que al hacer el consume comprueba que todas las celdas estén todas vacías, si es así devuelve el resultado de EBattery.

Pregunta 6

Al ejecutar el comando propuesto se obtendría:

```
Battery: consume-left-right(o ^ o ^ o)
```

El resultado no variará ya que no se con el comando red únicamente se evalúan las ecuaciones, no se evalúan las reglas del módulo.

Pregunta 7

Al ejecutar el comando propuesto se obtendría:

```
Battery: consume-left-right(+ ^ o ^ o)
```

En este caso sí se toman en cuenta las reglas declaradas por lo tanto si variará el resultado.

Pregunta 8

Al ejecutar el siguiente comando obtenemos:

```
Solution 1 (state 1)
Bt --> - ^ + ^ o

Solution 2 (state 2)
Bt --> - ^ o ^ +
```

El resultado de ejecutar el comando search, son todos los posibles resultados que se pueden obtener a partir de la expresión *consume(- ^ o ^ o)*.

Pregunta 9

Al ejecutar el siguiente comando obtenemos:

```
Solution 1 (state 0)
Bt --> consume-left-right(- ^ o ^ o)

Solution 2 (state 1)
Bt --> - ^ + ^ o
```

```
Solution 3 (state 2)
Bt --> - ^ o ^ +
```

Obtenemos un resultado más ya que con este search, obtenemos todos los posibles resultados con tantos pasos de ejecución como sean necesarios.

Pregunta de Respuesta Libre

Una posible solución para la cuestión planteada por el ejercicio es la que se muestra por el módulo siguiente:

```
mod BATTERY-ANY is
    protecting BATTERY-MAUDE .
    op consume-any : Battery -> Battery .
    var EBt : EBattery .
    vars Bt1 Bt2 : Battery .

    rl consume-any(Bt1 ^ o ^ Bt2) => Bt1 ^ + ^ Bt2 .
    rl consume-any(Bt1 ^ + ^ Bt2) => Bt1 ^ - ^ Bt2 .
    rl consume-any(EBt ^ o ^ Bt1) => EBt ^ + ^ Bt1 .
    rl consume-any(EBt ^ + ^ Bt1) => EBt ^ - ^ Bt1 .
    rl consume-any(Bt2 ^ o ^ EBt) => Bt2 ^ + ^ EBt .
    rl consume-any(Bt2 ^ + ^ EBt) => Bt2 ^ - ^ EBt .

endm
```

Este módulo se ha implementado del módulo funcional *"BATTERY-MAUDE"* ya proporcionado por la práctica.