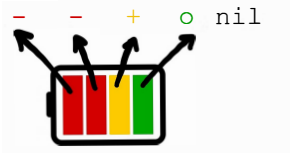
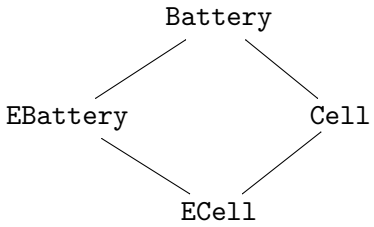
Métodos Formales Industriales
Tarea 1 (Individual) - El lenguaje Maude

Consideremos el siguiente módulo que define una batería como una bolsa o multiconjunto de celdas (es decir, no están ordenadas y se admite repetición), donde `'-'` representa una celda descargada (es decir, vacía), `'+'` representa una celda que está a media carga y `'o'` representa una celda con su carga completa:




```
mod BATTERY-TASK is
  pr NAT .

  sorts ECell Cell EBattery Battery .

  subsorts ECell < Cell < Battery .
  subsorts ECell < EBattery < Battery .

  op o : -> Cell [ctor] .
  op + : -> Cell [ctor] .
  op - : -> ECell [ctor] .

  op nil : -> EBattery [ctor] .
  op __ : EBattery EBattery -> EBattery [ctor assoc comm id: nil] .
  op __ : Battery Battery -> Battery [ctor assoc comm id: nil] .

  var C : Cell .
  var EB : EBattery .
  vars B B1 B2 : Battery .

endm
```

Observa que el módulo introduce una jerarquía de tipos construida usando la relación de subtipado ($<$), que podemos representar mediante el siguiente grafo:



En particular, los tipos de datos `ECell` y `EBattery` se usarán para denotar, respectivamente, el tipo al que pertenece una celda vacía (`-`) y el tipo al que pertenece una batería vacía (es decir, una batería que: (1) no tiene celdas, representado por `nil`; o bien (2) sus celdas están vacías, como por ejemplo la batería `- - -`).

La tarea consiste en:

- Responder a las preguntas del cuestionario que encontrás como examen en PoliformaT (**Test-Tarea1**), eligiendo la respuesta correcta, de entre las distintas opciones propuestas, para cada una de las preguntas planteadas en el ANEXO. Para ello debes primero extender el módulo funcional `BATTERY-TASK` implementando todas las funciones solicitadas.
- A continuación, responde a esta tarea adjuntando como evidencia tu módulo `BATTERY-TASK` completo. La tarea es nula si falta el adjunto o si hay errores de sintaxis en el código.

ANEXO.

Completa la definición de las siguientes funciones seleccionando para cada pregunta la respuesta correcta:

1. Comprobar si una batería está vacía (por no tener celdas o por contener sólo celdas vacías).

```
op is-empty : Battery -> Bool .
eq is-empty(␣) = true .
eq is-empty(B) = false [owise] .
```

Respuestas:

- [A] ECell
- [B] - nil
- [C] EB
- [D] nil

Ejemplos de ejecución:

```
reduce in BATTERY-TASK : is-empty(- - - -) .
result Bool: true

reduce in BATTERY-TASK : is-empty(nil) .
result Bool: true

reduce in BATTERY-TASK : is-empty(o + o) .
result Bool: false
```

---

2. Determinar el número de celdas de una batería.

```
op length : Battery -> Nat .
eq length(␣) = 0 .
eq length(C) = 1 .
eq length(B1 B2) = length(B1) + length(B2) [owise] .
```

Respuestas:

- [A] ECell
- [B] - nil
- [C] EB
- [D] nil

Ejemplos de ejecución:

```
reduce in BATTERY-TASK : length(+ - o -) .
result NzNat: 4

reduce in BATTERY-TASK : length(- -) .
result NzNat: 2

reduce in BATTERY-TASK : length(nil) .
result NzNat: 0
```

---

3. Calcular el tiempo restante de vida de una batería, asumiendo que 'o' representa 30 minutos de carga, '+' representa 15 minutos de carga, y '-' representa una celda descargada.

```
op bat-life : Battery -> Nat .
eq bat-life(EB) = 0 .
eq bat-life(o) = 30 .
eq bat-life(+) = 15 .
eq bat-life(_ _) = bat-life(B1) + bat-life(B2) [owise] .
```

Respuestas:

- [A] -
- [B] B1 ∧ B2
- [C] B1 B2
- [D] B1 and B2

Ejemplos de ejecución:

```
reduce in BATTERY-TASK : bat-life(- - -) .
result Zero: 0

reduce in BATTERY-TASK : bat-life(o + -) .
result NzNat: 45
```

---

4. Testear si existe una posible reordenación capicúa de las celdas de la batería.

```
op capicuizable : Battery -> Bool .
eq capicuizable(B B) = true .
eq capicuizable(_ _) = true .
eq capicuizable(B) = false [owise] .
```

Respuestas:

A EB

B C

C C B C

D B C B

Ejemplos de ejecución:

```
reduce in BATTERY-TASK : capicuizable(- - + +) .
result Bool: true

reduce in BATTERY-TASK : capicuizable(nil) .
result Bool: true

reduce in BATTERY-TASK : capicuizable(- - + o) .
result Bool: false
```

---

5. Consumir de forma aleatoria la mitad de la carga de una celda cualquiera que esté cargada completamente (sólo de una).

```
op degrade-random : Battery -> Battery .
rl degrade-random(      ) => + B .
```

Respuestas:

A B B

B o B

C + B

D - B

Ejemplos de ejecución:

```
Maude> search degrade-random(+ o - + o o) =>! B:Battery .
Solution 1
B --> o o + + + -
```

---

6. Generar todas las formas posibles de consumir incrementalmente las celdas de la batería hasta agotarla:

```
op exhaust-random : Battery -> Battery .
rl exhaust-random(o B) => exhaust-random(+ B) .
rl exhaust-random(+ B) => [          ] .
eq exhaust-random(EB) = EB .
```

Respuestas:

- [A] - B
- [B] exhaust-random(- B)
- [C] exhaust-random(EB)
- [D] nil

Ejemplos de ejecución:

```
Maude> rew exhaust-random(+ o - + o) .
Solution 1
B --> - - - - -
```

---

7. Finalmente, introduce las siguientes reglas en el programa:

```
rl [r1] : o => +  .
rl [r2] : + => -  .
```

e indica el resultado de ejecutar el comando ‘‘Maude> rew (+ o +) .’’

Respuestas:

- [A] - - -
- [B] - o +
- [C] - + -
- [D] nil

ATENCIÓN: Elimina estas reglas cuando ejecutes las funciones definidas en las preguntas 5 y 6 ya que interfieren con su comportamiento.