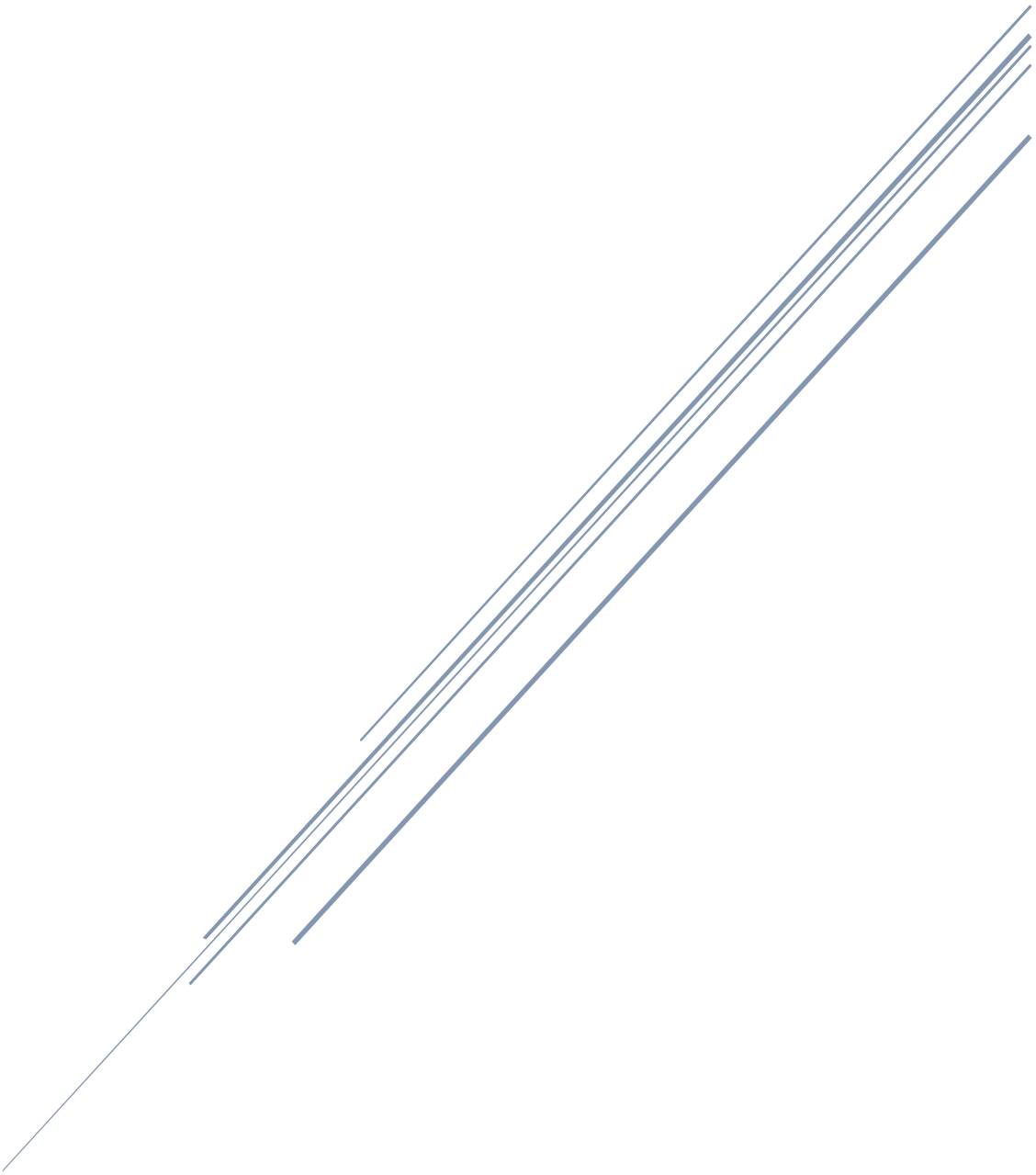
PRACTICA 2

LPM - Laboratorio de programación multiparadigma



MITSS
Sergi Sanz Carreres

Índice:

3.1. Monty Python Example

1.- ¿Qué devuelve la ejecución de la expresión search bruja(X) =>* X:Bool .?

```
Maude> search bruja(X) =>* X:Bool .
search in BRUJA-SAT : bruja(X) =>* X:Bool .

Solution 1 (state 11)
states: 12  rewrites: 14 in 0ms cpu (1ms real) (36082 rewrites/second)
X:Bool --> pato =:= lola and X =:= pato and X =:= lola

Solution 2 (state 12)
states: 13  rewrites: 15 in 0ms cpu (1ms real) (32967 rewrites/second)
X:Bool --> pato =:= pato and X =:= lola and X =:= jamon

No more solutions.
```

El comando search con =>* devuelve todos los posibles estados a los que se llega con cero o más pasos de reescritura desde el estado inicial y que hagan matching con el patrón. Por tanto, se devolverá todos los posibles estados que hagan matching con X.

2.- ¿Qué devuelve la ejecución de la expresión search bruja(lola) =>* X:Bool .?

```
Maude> search bruja(lola) =>* X:Bool .
search in BRUJA-SAT : bruja(lola) =>* X:Bool .

Solution 1 (state 11)
states: 12  rewrites: 14 in 0ms cpu (0ms real) (91503 rewrites/second)
X:Bool --> pato =:= lola and lola =:= pato and lola =:= lola

Solution 2 (state 12)
states: 13  rewrites: 15 in 0ms cpu (0ms real) (76530 rewrites/second)
X:Bool --> pato =:= pato and lola =:= lola and lola =:= jamon
```

El comando search con =>* devuelve todos los posibles estados a los que se llega con cero o más pasos de reescritura desde el estado inicial y que hagan matching con el patrón. Por tanto, se devolverá todos los posibles estados que hagan matching con lola.

3.- ¿Qué devuelve la ejecución de la expresión search bruja(lola) =>* X:Bool . si cambiamos la segunda regla de mismopeso para que compare pato y lola?

La regla actual incluida en el programa es la siguiente:

```
rl mismopeso(X,Y) => X =:= pato and Y =:= jamon .
```

Que debemos sustituir por:

```
rl mismopeso(X,Y) => X =:= pato and Y =:= lola .
```

Dando como resultado:

```
Maude> search bruja(lola) =>* X:Bool .
search in BRUJA-SAT : bruja(lola) =>* X:Bool .

Solution 1 (state 11)
states: 12  rewrites: 14 in 0ms cpu (0ms real) (60606 rewrites/second)
X:Bool --> pato =:= lola and lola =:= pato and lola =:= lola

Solution 2 (state 12)
states: 13  rewrites: 16 in 0ms cpu (0ms real) (53691 rewrites/second)
X:Bool --> pato =:= pato and lola =:= lola

No more solutions.
```

El comando search con =>* devuelve todos los posibles estados a los que se llega con cero o más pasos de reescritura desde el estado inicial y que hagan matching con el patrón. Donde podemos observar dos soluciones, en una solución se obtiene el estado

pato =:= lola and lola =:= pato and lola =:= lola

Mientras que en la otra solución obtenemos el estado

pato =:= pato and lola =:= lola

3.2. Listas

1.- ¿Qué devuelve la ejecución de la expresión rewrite length(nil) .?

```
Maude> rewrite length(nil) .
rewrite in LISTAS : length(nil) .
rewrites: 1 in 0ms cpu (0ms real) (125000 rewrites/second)
result Zero: 0
```

Al aplicar reescritura se realiza un paso donde obtenemos la largaría de nil que es cero.

2. ¿Qué devuelve la ejecución de la expresión search length(1 : 2 : 3 : 4 : nil) =>! N .?

```
Maude> search length(1 : 2 : 3 : 4 : nil) =>! N .
search in LISTAS : length(1 : 2 : 3 : 4 : nil) =>! N .

Solution 1 (state 5)
states: 6  rewrites: 9 in 0ms cpu (0ms real) (34482 rewrites/second)
N --> 4

No more solutions.
states: 6  rewrites: 9 in 0ms cpu (0ms real) (30821 rewrites/second)
```

El comando search con =>! devuelve todos los posibles estados finales a los que se llega reescribiendo el término inicial que le pasamos y que hacen matching con el patrón de la derecha. Por tanto, obtendremos una solución en la que se nos indica que 6 estados hacen matching, obteniendo como resultado N -- > 4.

3. ¿Qué devuelve la ejecución de la expresión search length(1 : N1 : N2 : nil) =>! N .?

```
Maude> search length(1 : N1 : N2 : nil) =>! N .
search in LISTAS : length(1 : N1 : N2 : nil) =>! N .

Solution 1 (state 4)
states: 5  rewrites: 7 in 0ms cpu (0ms real) (53030 rewrites/second)
N --> 3

No more solutions.
states: 5  rewrites: 7 in 0ms cpu (0ms real) (40697 rewrites/second)
```

El comando search con =>! devuelve todos los posibles estados finales a los que se llega reescribiendo el término inicial que le pasamos y que hacen matching con el patrón de la derecha. Por tanto, obtendremos una solución en la que se nos indica que 5 estados hacen matching, obteniendo como resultado N -- > 3.

3.3. Suma y comparación

1.- ¿Qué devuelve la ejecución de la expresión search leq(sum(0,0),s(Y)) =>* true .?

```
Maude> search leq(sum(0,0),s(Y)) =>* true .
search in SUM-LEQ : leq(sum(0, 0), s Y) =>* true .

Solution 1 (state 2)
states: 3  rewrites: 2 in 0ms cpu (0ms real) (22727 rewrites/second)
empty substitution

No more solutions.
```

El comando search con =>* devuelve todos los posibles estados a los que se llega con cero o más pasos de reescritura desde el estado inicial y que hagan matching con el patrón, devolviendo como resultado final una sustitución vacía.

2.- ¿Qué devuelve la ejecución de la expresión search leq(0,sum(X,Y)) =>* true .?

```
Maude> search leq(0,sum(X,Y)) =>* true .
search in SUM-LEQ : leq(0, sum(X, Y)) =>* true .

Solution 1 (state 1)
states: 2  rewrites: 1 in 0ms cpu (0ms real) (32258 rewrites/second)
empty substitution

No more solutions.
```

El comando search con =>* devuelve todos los posibles estados a los que se llega con cero o más pasos de reescritura desde el estado inicial y que hagan matching con el patrón, devolviendo como resultado final una sustitución vacía al igual que en el ejercicio anterior.

3.4. Resta

El código añadido para posibilitar esta funcionalidad es el siguiente:

```
op minus : Nat Nat -> [Nat] .
rl minus(X,0) => X .
rl minus(s(X),s(Y)) => minus(X,Y) .
```

1. ¿Qué devuelve la ejecución de la expresión search minus(s(0),s(s(0))) =>* X:Nat .?

```
Maude> search minus(s(0),s(s(0))) =>* X:Nat .
search in SUM-LEQ : minus(1, 2) =>* X:Nat .

No solution.
```

Debido a que el s(s(0)) es 2 y el s(0) es 1, nos encontramos con un valor negativo al aplicar la resta, pero ese motivo no se devuelve solución ya que incumple la propiedad de que tienen que ser naturales.

2. ¿Qué devuelve la ejecución de la expresión search minus(s(s(0)),s(0)) =>* X:Nat .?

```
Maude> search minus(s(s(0)),s(0)) =>* X:Nat .
search in SUM-LEQ : minus(2, 1) =>* X:Nat .

Solution 1 (state 2)
states: 3  rewrites: 2 in 0ms cpu (0ms real) (47619 rewrites/second)
X:Nat --> 1

No more solutions.
```

En este caso, se aprecia como s(s(0)) obtiene el valor 2, mientras que s(0) obtiene el valor 1, pudiéndose aplicar la resta satisfaciblemente y obteniendo un valor resultante de Nat → 1

3. ¿Qué devuelve la ejecución de la expresión search minus(0,s(X)) =>* Y:Nat .?

```
Maude> search minus(0,s(X)) =>* Y:Nat .
search in SUM-LEQ : minus(0, s X) =>* Y:Nat .

No solution.
```

Debido a que el s(s(0)) es 1 y 0, nos encontramos con un valor negativo al aplicar la resta, pero ese motivo no se devuelve solución ya que incumple la propiedad de que tienen que ser naturales.