Definiciones currificadas en ISetL

Sea la definición de la función f dada en Discusión 4:
f: N -> N -> Bool
f(n) = f1 donde f1: N -> Bool
f1(m) = (odd °mod1) (n,m).

Para definirla en ISetL, necesitamos definir la composición de funciones y la funcion mod1:

```
co1:=func(f,g);
       if is_func(f) and is_func(g) then
       return func(x);
              return f(g(x));
      end; end; end;


mod1:=func(n,m);
        if is_integer(n) and is_integer(m) then
        return n mod m;
      end;end;
```

Recordemos que mod1 es la versión prefija de mod y la definición de co1 que hemos dado arriba es la versión prefija de °

La expresión (odd °mod1) (56,15) seria en ISetL co1 (odd, mod1) (56,15), cuya evaluación de acuerdo a la definición de co1 seria f(g(x)), es decir, odd (mod1(56, 15)).

Sin embargo, nos encontramos con otra limitación del lenguaje ISetL:
no permite que una variable represente cualquier elemento de cualquier conjunto. Así, la variable x en la definición de co1, no puede representar un par, y la expresión co1 (odd, mod1) (56,15) resulta en el error "!Error: Too many arguments", ya que ISetL no considera al par (56,15) como un argumento sino como dos.

Lo que necesitamos es una versión currificada de mod, para poder aplicarla a un número y no a un par:
En matemática es:

mod2: N -> N -> N
mod2 (n) = f donde f : N -> N

$$f(m) = n \bmod m$$

En ISetL:

```
mod2:=func(n);
        if is_nat(n) then
        return func(m);
                 if is_nat(m) then
                 return n mod m;
               end;end;
end;end;
```

Evalúe en ISetL:

mod2(56);
mod2(56)(12);

Responda:

1. ¿Cual es el dominio de la función mod2(56)?
2. ¿Cual es la expresión de la definición de mod2 arriba que representa dicha función?
3. ¿Existe una definición correspondiente en matemática?

Usando la definición de la función is_nat pedida en el ejercicio 5 de Evaluación 2, la función f puede implementarse así en ISetL:

```
f:=func(m);
      if is_nat(m) then
      return func(n);
                if is_nat (n) then
                return co1(odd,mod2(n))(m);
              end; end;
    end;end;
```

Observar que la expresión

```
func(n);
   if is_nat (n) then
   return co1(odd,mod2(n))(m);
  end; end;
```

corresponde a “donde f1 …" en la definición matemática de f, con la diferencia de que en matematica necesitamos darle un nombre f1 a la función, mientras que en ISetL, las expresiones que comienzan con la palabra reservada *func* y contienen la palabra reservada *return*, representan

funciones y pueden ser usadas *sin asignarles un nombre*, como expresión que devuelve una función,
Observar que la expresión debe terminarse con end; así como también la cláusula if.

1. ¿Cuales son las funciones que se componen en
   co1 (odd, mod2 (56)) (12);?
2. Escriba una definición currificada de la composición, es decir, que pueda aplicarse asi: co (odd) (mod2 (56)) (12); Llámele co y guárdela.

Evalue en ISetL:

f (56);
f (56) (15);
f (18);
f (18) (4);

Observe que f (56) y f (18) son funciones. ¿Cual es su dominio y su co-dominio?

Observe que f (56) es la función f1 y f (18) es la función f2 introducidas en Discusión 4.

Observe que teniendo una definición de una función que soluciona un problema general, es posible definir funciones para casos particulares *sin necesidad de introducir nuevas funciones para eso*, sino que se obtienen aplicando la función general a los casos particulares. Así, teniendo la función f, f(56) es f1 y f(18) es f2.
Si definimos (observar el orden de los argumentos):

```
mod3:=func(n);
        if is_nat(n) then
        return func(m);
                 if is_nat(m) then
                 return m mod n;
               end;end;
end;end;
```

podemos definir mod3(2) como la función que dado cualquier natural m devuelve el resto de la división de m por 2, y podemos también darle un nombre:

resto_2:= mod3(2);

y usarla en expresiones, por ejemplo:

resto_2 (5);

que es equivalente a mod3(2)(5)

Con una definición cuprificada de la función suma, por ejemplo, podemos definir una función que dado un natural, lo incremente en 1, otra que lo incremente en 2, etc, sin necesidad de definir funciones nuevas:

```
mas:=func(n);
        if is_nat (n) then
        return func(m);
                if is_nat (m) then
                return n + m;
       end;end;end;end;
```

Las funciones que incrementan en 1 y en 2,se obtienen aplicando la función mas a 1 y a 2 respectivamente: mas(1) y mas(2).
Podemos definir nuevas funciones:

```
incr_1:=mas(1) ;
co1 (incr_1, resto_2);
```

Los nombres incr_1 y resto_2 para las funciones mas(1) y mod3(2) respectivamente, pueden obviarse y escribir directamente:

co1(mas(1), mod3(2));

Introduzca en ISetL las definiciones anteriores y evalúe todas las expresiones.

Ejercicios

El ejercicio 6 de la Actividad 2 pide:

> Defina matemáticamente una función *mul_con_tope* que tome un par de naturales y devuelva el conjunto de los múltiplos del primero, menores que el segundo. Por ejemplo:
>
> > mul_con_tope(15,312) = {60, 75, 90, 105, 0, 15, 45, 30, 285, 300, 270, 255, 240, 120, 135, 165, 150, 225, 210, 195, 180}

En Evaluación 2 se pide una implementación de la misma.

Dar una definición matemática y una implementación en ISetL de la función *mul_con_tope* con los siguientes dominio y co-dominio:

> mul_con_tope : N -> N -> P(N)

Esta función puede ser aplicada así:

```
mul_con_tope (12) (324);
mul_con_tope (12) (218);
mul_con_tope (17) (218);
```

Qué tipo de objetos son:

mul_con_tope(12)
(mul_con_tope ° abs) (12) (324)

Escriba una función currificada en ISetL que dados dos números enteros n y m, devuelva (mul_con_tope ° abs) (n) (m).