

Lenguajes de Programación 2020-1

Facultad de Ciencias UNAM

Ejercicio Semanal 11

Sandra del Mar Soto Corderi
Edgar Quiroz Castañeda

7 de noviembre de 2019

1. Considera el siguiente programa

```
exception Excpt of int ;  
fun twice (f , x ) = handle f ( f ( x ) ) with Excpt ( x ) = > x ;  
fun pred ( x ) = if iszero x then raise Excpt ( x ) else x - 1;  
fun smart ( x ) = handle 1 + pred ( x ) with Excpt ( x ) = > 1;
```

Evalúa las siguientes expresiones utilizando la semántica operacional de la máquina K extendida para manejar excepciones con valor. (pueden obviarse algunos pasos pero no los que involucren manejo de excepciones. En cada caso describe qué excepción fue lanzada y en dónde.)

- `twice(smart, 0)`

Cuando se tenga una aplicación con valores bloqueados, se saltarán los pasos hasta el cálculo que los concierna, ignorando los pasos intermedios para verificar que son valores.

```
□ > app(twice, (smart, 0))  
→k □ > twice[f := twice, x := 0]  
→β □ > handle(app(smart, app(smart, 0)), x.x)  
→k handle(_, x.x) > app(smart, app(smart, 0))  
→k app(smart, _), handle(_, x.x) > app(smart, 0)  
→k app(smart, _), handle(_, x.x) > smart[x := 0]  
→β app(smart, _), handle(_, x.x) > handle(sum(1, app(pred,0)), x.1)  
→k handle(_, x.1), app(smart, _), handle(_, x.x) > sum(1, app(pred,0))  
→k sum(1, _), handle(_, x.1), app(smart, _), handle(_, x.x) > app(pred,0)  
→k sum(1, _), handle(_, x.1), app(smart, _), handle(_, x.x) > pred[x:=0]  
→β sum(1, _), handle(_, x.1), app(smart, _), handle(_, x.x) > if(iszero(0), raise(0), dif(0, 1))  
→k if(_, raise(0), dif(0, 1)), sum(1, _), handle(_, x.1), app(smart, _), handle(_, x.x) > iszero(0)  
→k if(_, raise(0), dif(0, 1)), sum(1, _), handle(_, x.1), app(smart, _), handle(_, x.x) < true  
→k sum(1, _), handle(_, x.1), app(smart, _), handle(_, x.x) << raise(0)
```

En este punto se lanza una excepción de tipo `Excpt` con un valor de 0.

```

→k handle(_, x.1), app(smart, _), handle(_, x.x) ≤ raise(0)
→k app(smart, _), handle(_, x.x) > 1[x:=0]
→β app(smart, _), handle(_, x.x) > 1
→k handle(_, x.x) > smart[x:=1]
→β handle(_, x.x) > handle(sum(1, app(pred, 1)), x.1)
→k handle(_, x.1), handle(_, x.x) > sum(1, app(pred, 1))
→k sum(1, _), handle(_, x.1), handle(_, x.x) > app(pred, 1)
→k sum(1, _), handle(_, x.1), handle(_, x.x) > pred[x:=1]
→β sum(1, _), handle(_, x.1), handle(_, x.x) > if(iszero(1), raise(1), dif(1, 1))
→k if(_, raise(1), dif(1, 1)), sum(1, _), handle(_, x.1), handle(_, x.x) > iszero(1)
→k if(_, raise(1), dif(1, 1)), sum(1, _), handle(_, x.1), handle(_, x.x) < false
→k sum(1, _), handle(_, x.1), handle(_, x.x) > dif(1, 1)
→k sum(1, _), handle(_, x.1), handle(_, x.x) < 0
→k handle(_, x.1), handle(_, x.x) > sum(1, 0)
→k handle(_, x.1), handle(_, x.x) < 1
→k handle(_, x.x) < 1
→k □ < 1

```

- twice(pred, 1)

Cuando se tenga una aplicación con valores bloqueados, se saltarán los pasos hasta el cálculo que los concierna, ignorando los pasos intermedios para verificar que son valores.

```

□ > app(twice, (pred, 1))
→k □ > twice[f := pred, n := 1]
→β □ > handle(app(pred, app(pred, 1)), x.x)
→k handle(_, x.x) > app(pred, app(pred, 1))
→k app(pred, _), handle(_, x.x) > app(pred, 1)
→k app(pred, _), handle(_, x.x) > pred[x:=1]
→β app(pred, _), handle(_, x.x) > if(iszero(1), raise(1), dif(1, 1))
→k if(_, raise(1), dif(1, 1)), app(pred, _), handle(_, x.x) > iszero(1)
→k if(_, raise(1), dif(1, 1)), app(pred, _), handle(_, x.x) < false
→k if(_, raise(1), dif(1, 1)), app(pred, _), handle(_, x.x) > iszero(1)
→k app(pred, _), handle(_, x.x) > dif(1, 1)
→k app(pred, _), handle(_, x.x) < 0
→k handle(_, x.x) > app(pred, 0)
→k handle(_, x.x) > pred[x:=0]
→β handle(_, x.x) > if(iszero(0), raise(0), dif(0, 1))
→k if(_, raise(0), dif(0, 1)), handle(_, x.x) > iszero(0)
→k if(_, raise(0), dif(0, 1)), handle(_, x.x) < true
→k handle(_, x.x) ≤ raise(0)

```

En este punto se lanza una excepción de tipo `Excpt` con un valor 0.

Se continua la evaluación del programa.

$$\rightarrow_k \Box \gamma \mathbf{x}[\mathbf{x}:=0]$$

$$\rightarrow_\beta \Box \gamma 0$$

$$\rightarrow_k \Box \gamma 0$$