

# Practica 4 - Interprete FAE

Profesora: Karla Ramírez Pulido  
Ayudante: Héctor Enrique Gómez Morales

Fecha de inicio: 23 de septiembre de 2015  
**Fecha de entrega: 7 de octubre de 2015**

## 1. Instrucciones

Para esta práctica, requerirás tomar como base el archivo `practica4-base.rkt` e implementar las funciones que se solicitan.

El objetivo de esta práctica es hacer un intérprete del lenguaje FAE (Function, Arithmetic Expression) con *ambientes*. Se tendrán dos sintaxis, la primera **FAES** es una sintaxis que se define explícitamente las expresiones **with**, y la sintaxis **FAE** en las que solo se tienen las operaciones aritméticas, definición de funciones y aplicación de funciones.

Usaremos la definición y aplicación de funciones para implementar la funcionalidad del **with**, dado que:

```
{with {var named-expr} body}
```

lo reemplazamos con

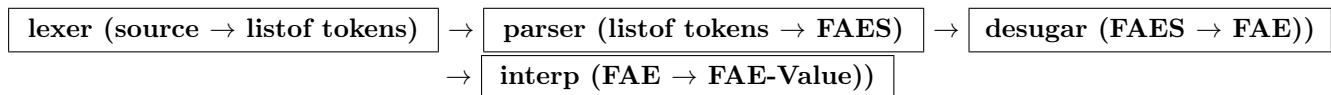
```
{{fun {var} body}  
  named-expr}
```

Es decir las expresiones **with** son *syntactic sugar*. Para realizar esto se definirá una función **desugar** que tomara una expresión en sintaxis **FAES** a una expresión en sintaxis **FAE**. Para luego hacer el interprete del árbol de sintaxis abstracta de **FAE** que debe ser de alcance estático por medio de ambientes.

```
<binop> ::= +  
          | -  
          | *  
          | /
```

```
<FAES> ::= <num>  
        | <id>  
        | {<binop> <FAES> <FAES>}  
        | {with {{<id> <FAES>}+} <FAES>}  
        | {with* {{<id> <FAES>}+} <FAES>}  
        | {fun {<id>*} <FAES>+}  
        | {<FAES> <FAES>*}
```

```
<FAE> ::= <num>  
        | <id>  
        | {<binop> <FAE> <FAE>}  
        | {fun {<id>*} <FAE>+}  
        | {<FAE> <FAE>*}
```



Esta práctica debe ser implementada con la variante `plai`, es decir su archivo con terminación `.rkt` debe tener como primer linea lo siguiente: `#lang plai`.

Todos los ejercicios requieren contar con pruebas mediante el uso de la función `test`:

## 2. Ejercicios

1. (2pts) **desugar** Define una función que toma una expresión en sintaxis FAES y que regresa una expresión en sintaxis FAE.
2. (3pts) **multi-param** Adecua el `interp` de tal manera que las funciones acepten una lista de parámetros y que las aplicaciones de funciones sean de múltiples argumentos.
3. (3pts) **with\*** Adecua a `desugar` de tal manera que se tenga expresiones `with*` que tienen la misma semántica de `let*` de Racket, en donde cada `named-expr` es evaluada una a una y se crea inmediatamente el *binding* con el valor obtenido.
4. (2pts) **interp** En base al código de `p4-base.rkt`, implementar el intérprete, dado un árbol de sintaxis abstracta, evaluar y regresar un valor de tipo `FAE-Value` que considera las variantes `numV` y `closureV`. `closureV` es el constructor de tipo que recibe un parámetro, tiene un cuerpo de tipo `FAE` y un ambiente de tipo `Env`. `Env` considera las variantes de tipo `mtSub` y `aSub`, donde el constructor `aSub` recibe un símbolo, un valor de tipo `FAE-Value` y otro ambiente `Env`.