Proyecto final - Fundamentos de Lenguajes Programación

Robinson Duque, Ph.D
robinson.duque@correounivalle.edu.co

Marzo 25 de 2020

1. Introducción

El presente proyecto tiene por objeto enfrentar a los estudiantes del curso:

- a la comprensión de todos los conceptos vistos en clase
- a la implementación de tres versiones de interpretadores de un lenguaje de programación
- al análisis de estructuras sintácticas, de datos y de control de un lenguaje de programación para la implementación de los interpretadores

Importante: Tenga en cuenta que las descripciones presentadas en este documento son bastante generales y solo pretenden dar un contexto que le permita a cada estudiante entender el proyecto y las características del lenguaje que debe desarrollar. Así mismo, se proponen algunas construcciones sintácticas, es posible que se requiera modificar o adaptar la sintaxis para cumplir con el requerimiento.

2. (50pts) Lenguaje MEZCLA

MEZCLA es un lenguaje de programación (no tipado) con ciertas características de lenguajes de programación declarativos e imperativos. Se propone para este proyecto que usted implemente un lenguaje de programación interpretado para lo cual usted es libre de proponer una sintaxis inspirandose en mínimo 3 lenguajes de programación existentes (e.g., C, C++, Java, Python, R, Scala, Octave, etc). La semantica del lenguaje estará determinada por las especificaciones en este proyecto.

2.1. Sintaxis Gramatical

- **La gramática debe estar documentada con comentarios indicando el lenguaje en el cual se han inspirado para especificar cada regla de producción**.
- La gramática de MEZCLA **debe ser socializada con el profesor** durante las primeras dos semanas después de la asignación de este proyecto. Para esto, deberá utilizar el mecanismo definido por el profesor para enviar un archivo Racket con la gramática definida. Dentro de los siguientes 2 a 4 días hábiles después del envío recibirá una respuesta con observaciones y/o la aprobación de la gramática. Este proceso podrá tomar varios envíos con actualizaciones sobre las recomendaciones realizadas. Así pues, si deja su envío para la segunda semana, quizá sólo tenga 1 retroalimentación sobre su gramática.
- La gramática deberá contener ejemplos de cada producción utilizando llamados a `scan&parse`. Es decir, deberá contener ejemplos de cómo se crean las variables, procedimientos, invocación a procedimientos, etc.
- Todo interpretador cuya gramática esté sin aprobar por el profesor, tendrá una nota de sustentación máxima de 0.2.

Usted debe implementar un interpretador para darle la semántica que se describe en los numerales subsecuentes.

2.2. Valores:

- **Valores denotados:** enteros, flotantes, numeros en base 32, hexadecimales, octales, cadenas de caracteres, booleanos (true, false), procedimientos, listas.
- **Valores expresados:** enteros, flotantes, números en base 32, hexadecimales, octales, cadenas de caracteres, booleanos (true, false), procedimientos, listas.

Aclaración: Los números en una base distinta de 10, deberán representarse así: [x32 0 23 12], [x16 4 1 0 7 14], [x8 2 1 4 0 7], teniendo en cuenta que el primer elemento de la lista indica la base del número y el resto de la lista puede utilizar la representación BIGNUM vista en clase.

Sugerencia: trabaje los valores enteros, flotantes desde la especificación léxica. Implemente los números en base 32, hexadecimales, octales, cadenas de caracteres, booleanos (true, false) y procedimientos desde la especificación gramatical.

2.3. Características del Lenguaje MEZCLA

El lenguaje debe permitir utilizar:

- **Identificadores:** Son secuencias de caracteres alfanuméricos que comienzan con una letra o un simbolo específico dependiendo de las características que desee implementar en su lenguaje
- **Definiciones:** Este lenguaje permitirá crear distintas definiciones:
  - **Constantes:** introduce una colección de identificadores no actualizables y sus valores iniciales. Una constante es de asignación unica y debe ser declarada con un valor inicial, por ejemplo: `Constante y = 9;`. Para este caso `y` no podrá ser modificada durante la ejecución de un programa.
  - **Variables de asignación unica:** introduce una colección de identificadores actualizables una única vez. Una variable de asignación unica puede ser declarada así: `Val z = 9, x = MEZCLAVAL;`. Para este caso, `z` no podrá ser modificada durante la ejecución de un programa puesto que ya ha sido asignada. Sin embargo, `x` podrá ser asignada una única vez a algún valor del lenguaje `MEZCLA`. Por ejemplo: `x ->29;`. En caso que se utilice la variable `x` sin estar ligada, el interpretador deberá lanzar un error.
- **Condicionales:** Son estructuras para controlar el flujo de un programa
- **Expresiones:** las estructuras sintácticas son una expresión. Algunas expresiones producen un valor, otras producen un efecto (ejemplo, las expresiones relacionadas con asignación)
- **Primitivas booleanas:** $<, >, <=, >=, ==, != , ==, and, or, not$. Estas primitivas son binarias (exceptuando el *not*, que es unaria) y permiten evaluar expresiones para generar un valor booleano
- **Primitivas aritméticas para enteros:** $+, -, *, \%, /, add1, sub1$
- **Primitivas aritméticas para hexadecimales, octales y base 32:** $+, -, *, add1, sub1$
- **Primitivas sobre cadenas:** longitud, concatenar
- **Primitivas sobre listas:** se deben crear primitivas que simulen el comportamiento de: `empty?`, `empty`, `cons`, `list?`, `car`, `cdr`, `append`
- **Definición/invocación de procedimientos:** el lenguaje debe permitir la creación/invocación de procedimientos que retornan un valor al ser invocados. El paso de parámetros será por valor y por referencia, el lenguaje deberá permitir identificar de alguna manera la forma como se enviarán los argumentos.
- **Definición/invocación de procedimientos recursivos:** el lenguaje debe permitir la creación/invocación de procedimientos que pueden invocarse recursivamente. . El paso de parámetros será por valor y por referencia, el lenguaje deberá permitir identificar de alguna manera la forma como se enviarán los argumentos.

3. (25pts) MEZCLA+

Extienda el lenguaje MEZCLA y añada:

- **Variables de múltiple asignación (mutables)**: introduce una colección de variables actualizables y sus valores iniciales. El lenguaje deberá distinguir entre constantes, variables mutables y variables asignación única. Una variable mutable puede ser modificada cuantas veces se desee. Una variable mutable puede ser declarada así: `Var a = 5, b = MEZCLAVAL;`. En ambos casos, ambas variables podrán ser modificadas durante la ejecución de un programa. Por ejemplo: `a ->9;` o `b->true;`
- **Secuenciación:** el lenguaje deberá permitir expresiones para la creación de bloques de instrucciones
- **Iteración:** el lenguaje debe permitir la defición de una estrutura de repetición tipo `for`. Por ejemplo: `for x =` a_1 `to` a_2 `do` a_3 `end`. Se sugiere agregar funcionalidad al lenguaje para que permita "imprimir" resultados por salida estándar tipo `print`.
- **Primitivas sobre vectores:** se debe extender el lenguaje y agregar manejo de vectores. Se deben crear primitivas que simulen el comportamiento de: `vector?`, `make-vector`, `vector-ref`, `vector-set!`.
- **Tipos de datos**: el lenguaje debe permitir manejo de tipos de datos. El lenguaje debe incluir solemanete chequeo de tipos (no es necesario implementar inferencia de tipos). Se deben definir reglas para todas las primitivas, condicionales, invocación de procedimientos, iteración y manejo de los distintos tipos de variables.

4. (25pts) MEZCLA++

Extienda el lenguaje MEZCLA+, sin manejo de chequeo de tipos, y añada:

- **Declaración de clases y métodos**
- **Creación de objetos:** simples o planos, cualquier implementación es válida
- **Invocación de métodos y selección de campos:** el lenguaje debe permitir invocar métodos asociados a objetos y obtener los valores asociados a los campos
- **Actualización de campos:** el lenguaje debe permitir actualizar los campos asociados a un objeto
- El lenguaje debe incluir conceptos como: herencia, llamados a métodos de la superclase, polimorfismo

5. Evaluación MEZCLA, MEZCLA+, MEZCLA++

El proyecto podrá ser realizado en grupos de hasta 3 personas utilizando la librería SLLGEN de Dr Racket. Este debe ser sustentado y cada persona del grupo obtendrá una nota entre 0 y 1 (por sustentación), la cual se multiplicará por la nota obtenida en el proyecto.

Sólo la especificación léxica y gramatical del lenguaje base MEZCLA, deberá ser aprobado por el profesor. Los interpretadores MEZCLA+ y MEZCLA++ deben ser variaciones simples del lenguaje base. Dado el caso que un grupo se presente con lenguajes MEZCLA+ y MEZCLA++ ajenos a la gramática definida para el interpretador MEZCLA, obtendrán una nota máxima de 0.2 en la sustentación asociada con dichos interpretadores.