The hulk programming language

Luis Ernesto Amat Cárdenas (C-122)

September 2, 2023

Contents

Lexing

1.1. Consideraciones generales Para el tokenizado se requiere de dos elementos: una serie de tokens, y un sistema que transforme texto ASCII en tokens con los que el Parser puede trabajar.

1.2. Tokens La clase **Tokens** contiene todos los tipos de tokens que **HULK** necesita; se hace un emparejamiento llave-valor usando los atributos de clase. **Tokens** tiene un metodo conveniente (**FromValue**) para obtener el tipo del token en base a su valor (recibe dos valores porque existen tokens compuestos de multiples caracteres). El tipo se le pasa a la clase **Token** junto con el valor que le corresponda para conformar el token que recibe el parser.

1.3. Lexer **Lexer** solo recibe texto ASCII como parámetro. Esta clase también se encarga de tener constancia de la posición del cursor (útil a la hora de lanzar excepciones). También contiene varios atributos, HashSets que agrupan tipos de tokens para facilitar identificarlos a la hora del ejecutar el proceso de parsing. De no ser por el OOP esctricto de Csharp serían variables globales.

El método más complejo que contiene esta clase es **GetResult**. Se usa luego que el tipo del token es identificado para obtener el valor que le corresponde (Ej. Al token tipo STRING la cadena que le corresponde). Recibe una condición que se mapea a una expresión regular que detemina si se deben seguir añadiendo caracteres al string que corresponde al valor del token o detenerse e identificar el siguiente token (o finalizar el lexing).

El proceso de lexing completo (usual) sería el siguiente:

1. Tomar un caracter y mover el cursor
2. Identificar el tipo a través de regex
3. Adquirir el valor que le corresponde usando **GetResult** (o sea, más regex)
4. Devolver el token

Si no se puede identificar el tipo del token mediante regex se comprueba si el valor del token se puede inferir en base al tipo (Ej. "+" implica PLUS). Else, se lanza una excepción.

Parsing

2.1. Consideraciones generales El parser recibe una serie de tokens y se encarga de crear un AST (Abstract Syntax Tree) para el interprete. O sea, pasarle un evaluable al intérprete con el que pueda ejecutar las instrucciones que se programaron

2.2. Context Clase que se utiliza para mantener el contexto para el intérprete. Montada sobre **Dictionary[string, Union[Literal, AST]]**. O sea, un diccionario que le hace corresponder namespaces con el valor que le corresponde. Si es una definición le corresponde un evaluable; caso contrario (una variable), un literal.

2.3. AST Evaluables. El parser identifica que tipo de AST le corresponde un conjunto de tokens (puede ser solo uno), que es básicamente asociar un comportamiento (que se ejecuta al ser evaluado) al conjunto de tokens. El AST se construye con valores que modifican el comportamiento resultante al ser evaluado.

2.3.1 Sobre Function y la recursividad

La propia definición de la función se referencia dentro del contexto local de esta para permitir la recursividad.

2.4. Parser La clase **Parser** se comporta de la misma manera que **Lexer**, solo que trabaja con tokens en lugar de caracteres. Para identificar a que AST le corresponde cada token primero se identifica si es un statement o una expresión. En el caso de este subset HULK solo tenemos declaración de funciones y variables entre los statements. Si es una expresión se utilizan varios métodos para simular el orden operacional de los operadores binarios.

Interpreting

3.1. Consideraciones generales El intérprete recibe un evaluable (con un valor de retorno, con el propósito de mostrarlo en un REPL de ser necesario). Este simplemente ejecuta el método **Eval** de dicho evaluable y retorna el resultado. También mantiene el contexto global y define los Builtins

3.2. Builtins Definiciones que no pueden ser definidas dentro de HULK. ASTs que sirven de puente entre features de Csharp y HULK; esto se implemente para facilitar algunos procesos. Ej. print(), True, False, cos(), ...

Flujo del programa

Se le pasa texto al lexer. Este identifica los tokens que son entregados al parser. El parser entrega un evaluable al intérprete que los evalúa en base al contexto global.