Breve introducción al funcionamiento de un interprete del lenguaje Lisp

Christian Chinchilla Brizuela

Escuela Ciencias de la Computación e Informática
Universidad de Costa Rica
San José, Costa Rica
christian.chinchilla@ecci.ucr.ac.cr

Abstract. This article gives a little description about the process to interpret Lisp code. The process to interpreted code Lisp is called list process and with a simple example demonstrated the process. Actuality the systems Lisp incorporated several additional functions, but in this case we use a basic interpreted Lisp with essentials functions.

Palabras clave: interprete, átomo, imperativo, inteligencia artificial, prefija.

1 Introducción

En este artículo se brinda una breve descripción del lenguaje Lisp y de él funcionamiento de un interpretador para lenguaje Lisp, además de las utilidades de trabajar con expresiones simbólicas.

El lenguaje Lisp se diseño para la manipulación de fórmulas simbólicas, por lo tanto esta orientado al que hacer y no a como hacer las cosas. Es un lenguaje imperativo. Después este lenguaje fue aplicado a la inteligencia artificial gracias a su habilidad para expresar algoritmos recursivos que manipulen estructuras de datos dinámicas. Para resolver un problema en Lisp se implementa la solución escribiendo lo que se quiere hacer, pero sin indicar paso a paso la secuencia de acciones que la computadora debe de realizar, a esta área se le conoce como computación simbólica.

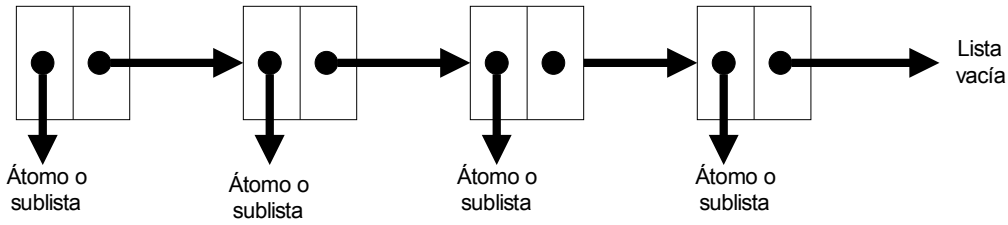
Lisp fue diseñado a finales de los años 50 por el pionero de la Inteligencia Artificial John McCarthy[1] y en la actualidad es ampliamente usado ya que no necesita un amplio conocimiento en programación, es muy fácil de usar. Las implementaciones de Lisp recientes involucran nuevas funcionalidades como por ejemplo un recolector de basura que hace que el lenguaje tenga un mejor desempeño, un depurador que ayuda al programador a corregir y optimizar el código. Sin embargo, vamos a analizar una versión básica de Lisp que contempla las características esenciales de un interpretador de Lisp.

2 Tipos de datos en Lisp

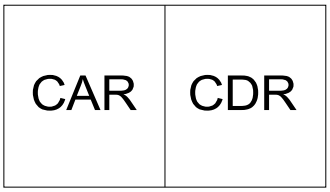
En Lisp existen dos tipos básicos de datos los átomos y las listas [3]. Todas las estructuras definidas posteriormente son basadas en estos tipos de datos. Los átomos pueden ser combinaciones alfanuméricas de las 26 letras del alfabeto en inglés en conjunto con los 10 dígitos del sistema numérico en base 10.

El segundo tipo de datos con los que trabaja Lisp son las listas y es gracias a este tipo de datos que deriva su nombre *List processor* (procesador de listas). Una lista puede ser una secuencia de átomos separados por un espacio y encerrados por paréntesis redondos, las listas pueden tener sublistas o pueden no tener elementos en ese caso seria una lista vacía. Las listas son el tipo de datos más importante de Lisp.

La estructura interna de una lista para Lisp es como se muestra en el siguiente diagrama:



Las celdas de una lista constan de dos partes CAR y CDR.



La parte CAR apunta al elemento de la lista, devuelve el primer elemento de la lista cuando se invoca. La parte CDR apunta al siguiente elemento de la lista, cuando se invoca a la parte CDR para una entrada, devuelve el siguiente elemento de esa entrada.

El corazón de cualquier sistema Lisp es la función EVAL, su trabajo consiste en evaluar expresiones Lisp, una expresión es una función y un conjunto de entradas. Las funciones en Lisp se representan como listas donde el primer elemento de la lista es el nombre de la lista y el resto de los elementos de la lista son los argumentos de la función.

3 Reglas de evaluación de expresiones [3]

Existen reglas de evaluación para un sistema Lisp que se deben de respetar para su correcto funcionamiento, estas reglas son las siguientes:

1. La evaluación de un número siempre es el mismo número.
2. La evaluación de los símbolos especiales T (true) y NIL (lista vacía, nulo o falso) siempre es el mismo símbolo.
3. La regla para evaluar listas consiste en evaluar los argumentos de la función antes de que sean usados por la función,
4. Para evitar que una expresión se evalúe podemos colocar una comilla simple antes de la expresión, esto le indica al intérprete de Lisp que la expresión que viene a continuación no se debe evaluar, es decir se evalúa como un dato simple y los datos simples se evalúan a si mismos.

El interpretador de Lisp utiliza una notación prefija [2], es decir los operadores van primero que los operandos. Por ejemplo, para sumar 2 + 5 en Lisp sería + 2 5. Esta forma aunque al inicio parece un poco complicada es muy útil porque con un solo operador + podemos sumar una cantidad variable de argumentos.

4 Proceso de interpretación del código Lisp

Las implementaciones de Lisp poseen gran cantidad de funciones primitivas, estas funciones ya están declaradas e implementadas en el sistema, algunas de las funciones básicas que deben existir en todo sistema Lisp son las funciones aritméticas, funciones para la creación y manipulación de listas, entre las encontramos las funciones CAR CDR para obtener el primer elemento de una lista y los siguientes elementos de una lista a partir de una entrada, respectivamente. Funciones para crear nuestras propias funciones

Una función que le indica al sistema Lisp que el argumento de esta función no debe ser evaluado (*quote*). Además de estas funciones todo sistema Lisp posee una función que es la encargada de evaluar las expresiones, esta función generalmente es llamada *eval* y es aquí por donde inicia la interpretación del código Lisp [4].

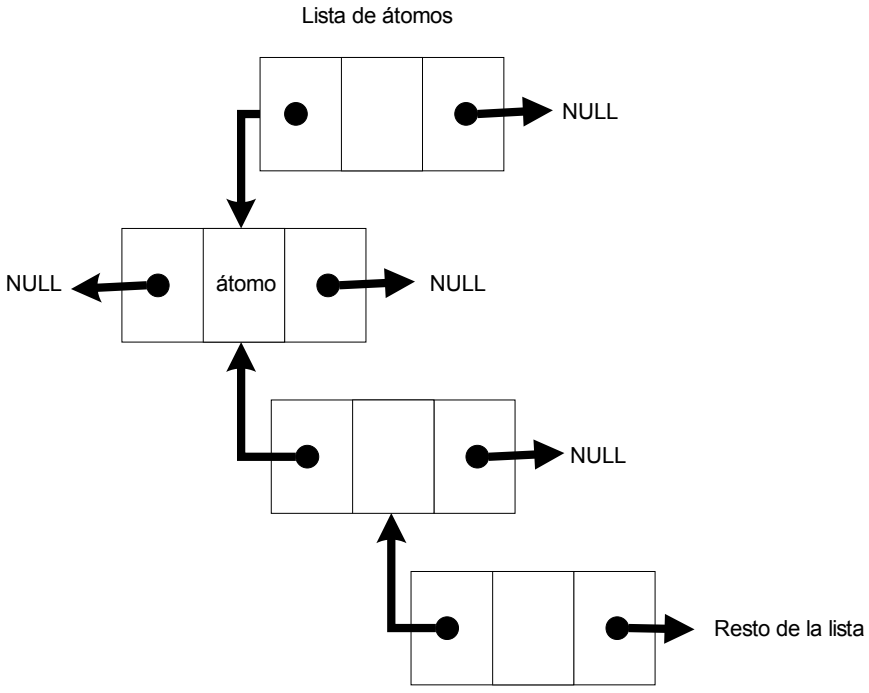
El código Lisp puede estar escrito en un archivo o digitarlo directamente en el interprete interactivo del sistema, sea cual sea la entrada el sistema Lisp inicia el análisis reconociendo carácter por carácter de la entrada. El sistema descarta de la entrada los espacios en blanco, los cambios de línea y las tabulaciones, si encuentra un carácter ";" descarta hasta el final de línea porque sería un comentario en un programa Lisp.

Para cada carácter de la entrada el interprete compara si es alguna de las entradas válidas: un carácter alfabético, un dígito, un paréntesis izquierdo "(", un paréntesis

derecho “)” o un apostrofe “ ‘ ”, en caso de no corresponder con alguno de estos, reporta un error de sintaxis.

Cuando el sistema encuentra un carácter alfabético sigue buscando todos los caracteres continuos que sean alfabéticos, numéricos o caracteres especiales permitidos, cuando encuentra el token correspondiente crea un átomo y lo inserta en la lista de átomos del sistema y continua con la búsqueda de tokens. Igual sucede con los dígitos, cuando el sistema encuentra un dígito sigue buscando en la entrada para encontrar todos los dígitos contiguos que formen un solo número, cuando descubre el número correspondiente, crea un nuevo nodo átomo y lo inserta en la lista de átomos del sistema.

Esta línea de código Lisp crea una lista “a” con el valor x, el procedimiento que realiza el sistema para analizarla se muestra en el diagrama:



En este esquema se muestra el procedimiento de un interpretador de Lisp para crear la lista del sistema, primero el interprete inicia extrayendo carácter por carácter de la entrada, cuando descubre el paréntesis izquierdo lo descarta, luego encuentra la letra “s” y continua leyendo mientras los caracteres siguientes sean letras, números o caracteres especiales, cuando llega a un carácter no valido (espacio en blanco) crea un átomo con el token que tiene en ese momento (setq) y lo introduce en la lista de átomos del sistema, en este caso introduce el primer elemento en la lista. En la lista de átomos, la lista de funciones y la lista del sistema no hay duplicidad de datos solo

es necesario representar una vez los nodos de las listas y por medio de apuntadores se accede a los nodos. De esta forma se evidencia como un intérprete de Lisp traduce un programa a una lista de elementos que luego se evalúa para obtener los resultados.

Conclusiones

A pesar de que podemos considerar que el lenguaje es viejo pues surgió hace casi 60 años el lenguaje es ampliamente utilizado en el campo de la inteligencia artificial principalmente.

Un interprete de Lisp realiza un proceso de análisis muy diferente a lenguajes como C, C++, Java C# porque no sigue un método descendente ni ascendente de análisis, podemos decir que sigue un método de procesamiento de listas, porque el sistema traduce todo el programa de entrada en una lista que después evalúa, de acuerdo a reglas establecidas. Gracias a la gramática simple del lenguaje es posible utilizar las listas para representar casi cualquier cosa en Lisp, puesto que las funciones que el usuario puede declarar se manejan como listas, los conjuntos complejos de datos también se manejan como listas. Aunque el lenguaje es simple, no son así los programas que se pueden escribir en él, el lenguaje permite escribir programas complejos con mayor facilidad que en otros lenguajes como C, C++, además al ser un lenguaje interpretado se puede corregir el código sin necesidad de tener que recompilar el programa. Lisp no fue diseñado para tener el mejor desempeño en como realizar las tareas, Lisp fue diseñado para hacer las tareas no pone especial énfasis en como hacerlas, su principal objetivo es hacerlas y este es un principio de la computación simbólica. Actualmente hay muchos sistemas Lisp y son ampliamente usados. Sin embargo, los sistemas Lisp actuales involucran muchas funcionalidades que hacen muy atractivo el lenguaje. Lisp se ha colocado como uno de los lenguajes favoritos en la rama de la inteligencia artificial, una de las razones es que el lenguaje no fue diseñado para obtener resultados de ejecución óptimos en tiempo, ni para realizar cálculos de la manera más adecuada, el lenguaje se diseño para hacer las cosas de una manera fácil y que otorgara la posibilidad de desarrollar y corregir programas sin tener amplio conocimiento en programación de computadores, este es el factor principal que hace que el lenguaje Lisp se coloque como uno de los favoritos, en el área de la inteligencia artificial

Referencias

[1] Michael L. Scott, Programming Language Pragmatic, Morgan Kaufmann Publishers, Inc., SanFrancisco, California. ISBN13:978-0-12-633951-2

[2] Introducción a Lisp, Dr. Alejandro Guerra Hernández, Departamento de Inteligencia Artificial, Universidad Veracruzana, Facultad de Física e Inteligencia Artificial,
Sebastián Camacho No5, Xalapa, Ver., México91000

[3] Touretzky D.S. “Lisp A gentle introduction to symbolic computation”. Ediciones Díaz de Santos S.A., Madrid-Barcelona, 1986.

[4] Wilensky R., Common LISPcraft, WW Norton & Company, primera edición, 1986.