Joel Isaí Ramos Hernández

A01245083

Implementación de métodos computacionales Gpo 14

Prof. Román Martínez Martínez

Instituto Tecnológico y de Estudios
Superiores de Monterrey

2 de mayo de 2021

Análisis de herramientas para la solución de la situación problema (Segundo Período)

Al igual que en el entregable pasado, unas de las herramientas/tecnologías más comunes que pueden ser usadas para la resolución de esta situación problema son:

- Flex/Lex: la cual es muy usada por su rapidez para crear scanners (lexers) usando el lenguaje C. Su mayor desventaja es que al estar basado en un AFD no tiene la capacidad de retroceder para encontrar la mayor cantidad de lexemas posibles.
- Coco/R: la cual es muy usada por que permite ser usada en más lenguajes como C++, C# y Java.

En la siguiente tabla de comparación, analizo estas herramientas/tecnologías (junto con la que yo use en mi solución propuesta) en 3 categorías, el conocimiento necesario para su uso, la rapidez y qué tanto es usado para tanto este tipo de soluciones como de otras.

Tabla de comparación

Herramientas	Conocimientos para de uso	Rapidez	Que tanto y en donde es utilizado
AFD en Scheme (Paradigma funcional)	Requiere todo un proceso previo de definir el AFD paso por paso y además requiere conocer y comprender el paradigma funcional en un nivel básico.	La rapidez va a depender directamente de como sea manejado el autómata por el programador, por lo que una gran implementación puede hacerlo muy rápido al igual que una mala implementación puede hacerlo exageradamente lento. Una gran ventaja es que al ser un AFD, esto ya	No es común que sea usado para este tipo de casos específicos. Como es necesario crear básicamente toda la programación, es más común que se usen otras herramientas. La única ventaja es que, al ser programador desde cero, si el programador es bueno, tiene la posibilidad de hacerlo más eficiente que con

		asegura una mayor rapidez contra uno AFN, siendo que la implementación del programador es la misma.	herramientas preprogramadas.
Flex/Lex	Además de entender las expresiones regulares, esta herramienta tiene su propia sintaxis necesaria de comprender.	Es más limitado al estar basado en un Autómata Finito Determinístico, pero esto mismo lo hace mucho más rápido.	Es ampliamente usado para la generación de scanners o lexers por su rapidez.
Coco/R	A diferencia de los otros, su sintaxis requiere que se sepa usar el Extended Backus–Naur Form (EBNF).	De igual manera está basado en un AFD, por lo que es rápido	De igual manera es muy usado tanto en la generación de scanners, lexers e incluso permite también hacer parsers.

A pesar de haber podido generar un resaltador de sintaxis con una gran rapidez, estas otras herramientas especializadas para la creación de lexers que mencioné son de enorme importancia, dado que no sólo simplifican el proceso, sino también evitan que uno mismo caiga errores de rapidez y/o eficiencia.

Todas estas herramientas formales son esenciales en las ciencias computacionales, dado que promueven la creación de software sin tener que exactamente entender qué es lo que está pasado por detrás. Y como acabo de mencionar, estas herramientas ayudan mucho puesto que alguien más ya se encargó de encontrar esta forma rápida y eficaz, de manera que los demás podemos aprovechar este avance evitando así errores en el método base.

Por otro lado, también está en este caso el paradigma funcional con el que trabajé para la solución de la situación problema, y del cual he podido entender su gran importancia en la computación. A pesar de ser un paradigma que consume mucha más memoria, esta permite crear código complejo de una manera muy sencilla gracias a que es un paradigma declarativo. Hasta cierto punto se parece un poco al paradigma de objetos en el aspecto que lo complejo esta (se programa) por detrás y el programa en sí se convierte en un lenguaje muy parecido al natural. En lo personal los dos temas de este paradigma que me ayudaron mucho y me hicieron entender su importancia fueron las listas, dado que básicamente todo es una lista y las funciones de primera clase que aunque apenas soy novato en ese tema, puedo imaginarme las infinitas posibilidades que esto aporta a las ciencias computacionales.

Referencias

Fuertes, J. L. (2011). Herramientas de Compiladores. Compiladores. http://www-lt.ls.fi.upm.es/compiladores/Herramientas.html.

JKU. (2018). The Compiler Generator Coco/R. JKU Computer Science System Software. https://ssw.jku.at/Research/Projects/Coco/.

JKU. (s.f.). Compiler Generator Coco/R. JKU Computer Science System Software. https://ssw.jku.at/Research/Projects/Compiler.html.

Mittal, S. (2019). Flex (fast lexical Analyzer generator). https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/.

Mössenböck, H. (2018). The Compiler Generator Coco/R. Johannes Kepler University Linz. https://ssw.jku.at/Research/Projects/Coco/Doc/UserManual.pdf.