Implementación de métodos computacionales

TC2037.2

Actividad Integradora 6.1:

Análisis de herramientas usadas

Profesor(a):

Alejandro de Gante

Abraham Mendoza Pérez	A01274857
Luis Alonso Martínez García	A01636255
Aldo Alejandro Degollado Padilla	A01638391

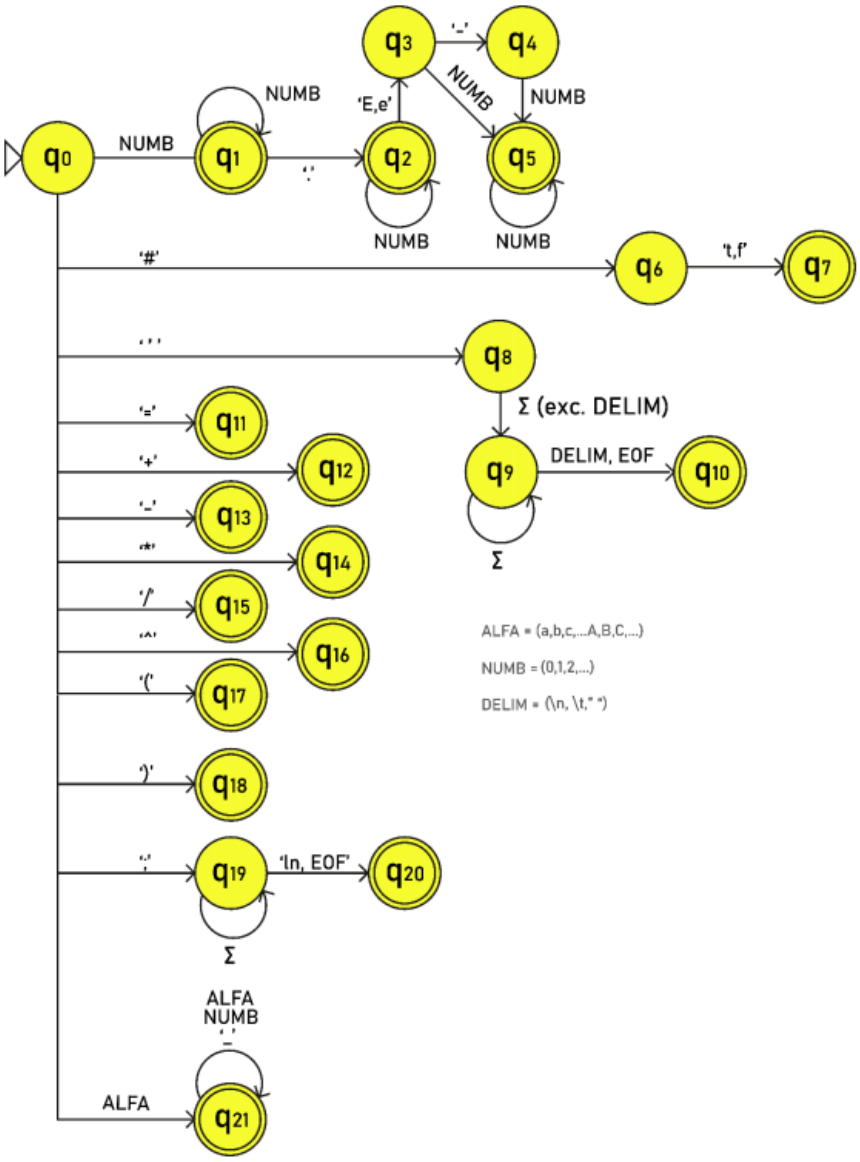
Viernes 28 de mayo de 2021

Conforme se iba estudiando los distintos temas de la materia de Implementación de métodos computacionales (TC2037), se nos presentó la primera parte de lo que sería la situación problema. En la cual, por medio de una las aplicaciones que tienen los autómatas finitos determinísticos, nuestro objetivo como equipo era el realizar un código capaz de identificar y regresar el tipo de tokens dado mediante un archivo de texto. Posteriormente, este proyecto tendría modificaciones para que ahora se pudieran detectar las categorías léxicas, al igual que se realizaría la implementación de HTML+CSS para poder resaltar el léxico detectado por el programa.

La resolución de toda esta situación problema no sólo conllevó el uso de programación, sino también el uso de distintas herramientas que iban desde diseño, trabajo colaborativo e IDE's. La aplicación de conceptos en alguna de las herramientas mencionadas anteriormente fue de suma importancia para la conclusión exitosa del proyecto. Por ello, en este ensayo, se realizará un análisis a detalle sobre cómo estas se fueron implementando a lo largo del desarrollo de la solución, las alternativas disponibles que pudieran utilizarse y otros criterios de evaluación para la misma.

Desde que se planteó la situación, el equipo se dedicó a analizar de qué manera podríamos resolverla. Fue que, con los primeros temas vistos en clase, uno de los más llamativos y fáciles de aplicar para nosotros, fueron los autómatas finitos deterministas. Desde un inicio, el código que íbamos a realizar principalmente se iba a basar en el uso de un *switch statement*, ya que dependiendo del carácter que se detectará, es que este iba determinando poco a poco la clasificación de token (categoría léxica posteriormente) de acuerdo con las restricciones dadas. Este funcionamiento se asemeja, en su mayoría, al de un autómata. Esta herramienta está principalmente formada por distintos estados o procesos, los cuales se van recorriendo de acuerdo con la entrada de un símbolo donde para poder realizar una transición, es necesario que el símbolo se encuentre presente dentro de los caracteres válidos de ella. De igual manera, estos pueden volver a recorrer el mismo proceso, en caso de ser necesario, y al

cumplirse de manera exitosa un camino de estos procesos, habrá un estado final que nos devolverá qué tipo de token o categoría léxica es el conjunto de caracteres. Así fue como finalmente llegamos a plantear nuestra lógica del funcionamiento del programa en un DFA.



De igual manera, no sólo esta era la única forma de tener un guía base para el desarrollo del código, sino que están presentes las expresiones regulares (también conocidas como regex), otro de los temas revisados durante la materia.

Estos principalmente consisten en una cadena de caracteres utilizadas para encontrar patrones o combinaciones dentro de otras esto conforme a cierto uso de delimitadores y algunas reglas de sintaxis. Los regex cuentan metacaracteres, los cuales se pueden ver como condiciones que crea el mismo usuario en caso de requerirlo.

Como tal estos si pueden representar una opción alternativa bastante llamativa. Debido a que, si planteáramos un nuevo punto de vista de nuestro código, donde ahora nos enfocáramos en realizar la clasificación léxica de acuerdo a que los patrones de estos que coincidieran con la entrada que se está dando en el código, se podría realizar sin ningún inconveniente. Inclusive programas como Java cuentan con un set de funciones tales como exec() y test() de RegExp, con match(), matchAll(), replace(), entre otras, las cuales podrían ser de gran ayuda. Desafortunadamente, debido a que el equipo no tenía una idea muy clara de las expresiones y ninguno había utilizado Java a profundidad anteriormente, decidimos pasar de esta opción. Además de que muy posiblemente nos hubiera tomado mucho más tiempo adaptarnos a este nuevo lenguaje. Lo mejor fue hacer uso del autómata, aunque no se demerita a las expresiones regulares, ya que una vez aprendiendo a utilizar estás, se pueden aplicar en bastante ámbitos de la informática.

En cuanto al trabajo colaborativo, para un mejor manejo de código e igualmente de documentos, nosotros utilizamos GitHub. Esta herramienta, además de ser requisito del profesor para la actividad, es bastante popular, y ya teníamos conocimiento sobre ella, por lo que no dudamos en utilizarla. Sin embargo, existen otras alternativas como Gitlab, que ofrece numerosas y útiles características en su DVCS, como, por ejemplo, un proyecto wiki integrado y una página web de proyecto. Las continuas capacidades de integración de GitLab automatizan el análisis y la entrega del código, lo que permite ahorrar tiempo en la fase de prueba. Con un visor de código, pull requests y un práctico método para solucionar conflictos, GitLab permite acceder a todos los aspectos más importantes de tu proyecto. En documentos, consideramos que una alternativa que sería más

eficiente sería un procesador de texto en línea, como Google Docs, pues facilita la edición, mientras se puede observar, tiempo real, los cambios realizados por los otros colaboradores.

En nuestro caso, por ser un lenguaje en el cuál hemos tenido más práctica, utilizamos C++, dónde se realizó el algoritmo para el resaltador de sintaxis. El programa, al concluir el análisis, generaba un archivo HTML, con los estilos que determinamos en otro archivo CSS. Otra solución que podría mejorar este proceso, sería realizarlo en un lenguaje de programación como Python, donde podemos utilizar algún motor de plantillas como django template, jinja2 o genshi.

Por parte del uso de herramientas UML, app.diagrams.net fue la página utilizada al momento de diseñar el diagrama de nuestro autómata finito para la solución de la situación. Aunque este sitio no está pensado para hacer demostraciones de diagramas de este tipo, nos fue más que suficiente. Esto debido a que app diagrams ofrece una gran variedad de tipos de diagramas que pueden utilizarse para demostrar el funcionamiento más a detalle de un código que esté en desarrollo o incluso ya haya sido finalizado, aspectos más técnicos que sean necesarios mostrarse al equipo de trabajo, o viendo desde un lado más profesional, a un cliente. Además de que se nos ofrece una amplia gama de opciones en caso de necesitar descargar alguno de los trabajos realizados o incluso poder guardar estos en la nube como Google Drive o One Drive en caso de requerirlo. Sin embargo, está no es la única herramienta de diagramas UML disponible en la red, ya que existen otras herramientas alternativas entre las cuales podemos encontrar a LucidChart. Esta al igual que app.diagrams, nos ofrece una amplia variedad de distintos diagramas, aunque en este caso, está herramienta se ve apoyada por distintos plugins de integración de plataformas tales como Java, Slack, Google Drive y Microsoft Teams. Sin embargo, nosotros consideramos que todo el sistema de la interfaz y como funciona en si él sitio, va más dirigido hacia personas que no se dedican al diseño en una manera tan profesional. Por el otro lado, consideramos que diagrams.net, va más orientada a

personas expertas como desarrolladores, administrador de redes, diseñadores, entre otros.

Otra herramienta la cual nos fue de gran utilidad para probar si el autómata finito que habíamos creado sería capaz de cubrir la situación problema planteada fue JFlap. Dicha herramienta nos ofrece opciones como diseñar y usar expresiones regulares, gramáticas, autómatas de pilas y finitos. Esta herramienta funciona con Java por lo cual es necesario contar un JDK para su uso. Al usarla tenemos un menú con muchas opciones y obviamente en nuestro caso, optamos por el autómata finito, donde al usarlo, podemos ir creando los distintos procesos que necesitamos al igual que los símbolos que son necesarios para las transiciones. Lo destacable de esta herramienta es que es funcional, ya que nos permite probar con entradas del usuario y observar cómo dicha entrada recorre el diagrama.

Mientras buscábamos dónde probar el DFA, el equipo encontró una herramienta alternativa conocida como Jsflap. Una herramienta para la construcción de NFA Y DFA por medio de la Web, con la fortuna de que está se encuentra basada en JFlap. Lo cual nos pareció una alternativa excelente en caso de tener algún problema con JFlap. Sin embargo, desde la perspectiva de todos los integrantes, Jsflap es inferior a JFlap en aspectos como opciones de diseño, ya que por parte de Jsflap solo podemos crear un autómata finito o una máquina de Turing. Además de que no cuenta con una versión descargable. Fuera de eso, si se busca crear un diseño rápido y comprobar su funcionalidad, Jsflap cubre esos puntos. De igual manera, acepta entradas del usuario que validen los procesos del autómata, por lo cual es una herramienta alternativa que pudo llegar a cubrir las necesidades que se nos presentarán en el desarrollo del proyecto.

De igual forma, uno de los temas que el equipo empezó a investigar más a cerca de este a pesar de no ser parte de estas dos últimas situaciones problema fue la programación Paralela, ya que para la última versión de este código será necesario adaptar este para que realice el proceso de resaltado de léxico de manera paralela con el fin de aprovechar los múltiples núcleos disponibles en los

CPUs, mejor conocidos como hilos. Aquí la duda es, ¿por qué es necesario aplicar este concepto? Bueno, como se sabe, de manera general, cada vez hay sistemas más complejos, los cuales pueden llegar a representar un problema para el procesamiento de distintos equipos. Es por esto por lo que el tener un buen poder computacional (hardware) se ha vuelto tan importante en distintas áreas de estudio. Y esta necesidad ha provocado que se busquen maneras de aprovechar todo el potencial que un procesador nos puede dar. Es así como nace la programación concurrente paralela. Este tipo de programación es un ejemplo de lo eficaz que pueden llegar a ser los algoritmos de divide y vencerás. La programación concurrente paralela es, en esencia, la división del poder del procesador para realizar pequeños procesamientos a la vez, lo que agiliza y, comúnmente, reduce el tiempo de ejecución. Entre otras ventajas podemos encontrar una disminución de tiempo considerable al correr programas muy demandantes (mayor complejidad), se pueden resolver problemas que no serían posibles con una sola CPU, se pueden abarcar más problemas entre otros beneficios, aunque de igual manera no está libre de desventajas, entre otros. Lamentablemente, no todo son ventajas y entre los puntos negativos se pueden encontrar un mayor consumo de energía, la implementación en código puede llegar a ser algo compleja, la sincronización entre distintas tareas puede ser difícil de lograr, etc. Aunque en esta unidad de formación aprendimos a manejar hilos mediante el lenguaje de Java y el IDE de NetBeans, nuestra poca experiencia tanto en el lenguaje como en el editor, nos hicieron replantearnos en qué entorno lo haríamos. Es por esto que decidimos, que este último trabajo sea realizado en el lenguaje de C++ y con el IDE de Visual Studio Code. Debido a que nos sentimos más cómodos trabajando en este entorno y la implementación de la programación paralela es igualmente posible gracias al uso de hilos.

Como se puede ver, las herramientas son de suma importancia para poder llevar a cabo el desarrollo de proyectos, ya sea desde un editor de texto hasta un lenguaje de programación, y esto posible a que cada vez las herramientas que son de suma relevancia para este estilo de proyecto se van adaptando de acuerdo a las necesidades del usuario como por ejemplo Drive, hace unos diez años nadie

pensaría la importancia que tendría la nube en nuestra actualidad y es que la aparición de nuevas tecnologías va de acuerdo a los problemas que se van presentando ante la sociedad, porque no es hasta el momento que se presenta un obstáculo o se presiente que será un problema en un futuro cercano cuando se empiezan a desarrollar estas nuevas tecnologías, pero por el momento lo que pensamos es que las tecnologías actuales se mantendrá a la vanguardia pero no se descarta que en algún punto llegué una nueva tecnología la cual nadie se espere.

Referencias (formato APA)

Bernal, F et al. (S.F.). Programación Paralela. *Ferestrepoca*. http://ferestrepoca.github.io/paradigmas-de-programacion/paralela/paralela_teoria/

Capterra. (S.F.). *diagrams.net y Lucidchart*. Recuperado de https://www.capterra.es/compare/166985/146136/draw-io/vs/lucidchart

Ionos. (2019). *Alternativas a GitHub: un resumen de las 5 mejores aplicaciones*. Recuperado de https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/alternativas-a-github/

JFLAP. (2005). *JFLAP Version 7.1*. Recuperado de http://www.jflap.org

VictorD3D. (2020). ¿Qué es diagrams.net?. *Conocimiento Libre*. Recuperado de https://conocimientolibre.mx/que-es-diagrams-net/#:~:text=Diagrams.net%20es%20una%20aplicación,para%20Windows%2C%20Linux%20y%20macOS.