**Teoría de Lenguajes y Automatas**

**Trabajo Práctico 1**

- ITBA -

Tomas Lori Santiago Ramirez Ayuso Agustín Pagnoni

Índice

1. Consideraciones realizadas . . . . . . . . . . . . . . . . . . . . . . . 3

2. Desarrollo del T.P. 1. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

3. Dificultades encontradas . . . . . . . . . . . . . . . . . . . . . . . . . 5

4. Futuras extensiones . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

5. Referencias . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

1. Consideraciones realizadas

Este trabajo práctico, que consta de un analizador de secuencias de nucleótidos y traductor de dichas secuencias en secuancias de proteinas, se llevó a cabo utilizando el lenguaje de programación C en conjunto al lenguaje Lex, lenguaje que crea un autómata y luego lo traduce al lenguaje C.

Dada la ocurrencia de “{A,B,..,C}”1 , de ahora en adelante “SELECTOR” ,hemos considerado que para el análisis de la secuencia, tomar sólo un camino, por ejemplo el primero, en este caso A. Al no hacerlo, estaríamos frente a la posibilidad de que encontremos más junkDNA2 pero analizar todas las posibles ramificaciones no es el objetivo que tiene este primer T.P.

En cuanto a la aparición de “[N]”abc..,3 tomamos como inválida la secuencia completa si la posicón del nucleótido “a” no es N.

De aparecer “SELECTOR\*” se ignora el simbolo “\*” por la misma razón que se decidió no implementar la búsqueda de la secuencia que de menos junkDNA cuando aparece un SELECTOR.

1 A, B y C alguno de [AUGTC]+

2 Junk DNA llamamos a toda secuencia de nucleótidos que no generan proteínas

3 N es un número natural y “a,b,c” alguno de [AUGTC]

2. Desarrollo del T.P. 1

El desarrollo de este trabajo consiste en el reconocimiento de las tríadas de nucleótidos de la secuencia de entrada, que se logra mediante el autómata interno que genera Lex en base a la gramática que definimos con las estructuras de las proteínas conocidas.

La gramática que definimos es la siguiente:

G : <VT, VN, S, P > siendo VT los símbolos terminales, VN los no terminales, S el símbolo inicial y P el conjunto de producciones .

VT = {

A , U, G, C, T, {, }, ‘,’, [, ] , \*, STOP, AUG, -, 0, 1, 2, 3, 4, 5, 6, 7, 8,9,TAA, TAG TGA

}

S = AUG

El conjunto de producciones surge de la siguiente expresión regular :

AUG ( ( { ( ( ( ( A | C | T | G)+ , )\* ( A | C | T | G )+ ) | - ) } ( "\*" | "\*" ( [ (0-9)+ ] ) | ( [ (0- 9) + ] ) ) ? ) \* | (A | C | T | G)\* ( [ (0-9)+ ] )? )+ ( STOP | TAA | TAG | TGA )

3. Dificultades encontradas

En primer lugar, una dificultad que encontramos es en parte lo que se menciona en la sección 1, en cuanto a la problemática que nace a partir de los SELECTORES y sus multiples opciones a la hora de analizar la secuencia e intentar generar la menor cantidad de junkDNA posible.

Fue motivo de duda ciertas combinaciones del lenguaje, tal como la aparición de “{A}\*” sin un “[N]” que indique la cantidad de iteraciones de “A”. Asimismo, el caso en el que haya “{A}\*BC[N]”, el analisis requeriría de un buffer de tamaño indefinido, con tope máximo la longitud del archivo de entrada

En el tiempo de implementación, las ideas para hacer un procesamiento en paralelo que surgieron en muchos casos creemos que la diferencia en tiempo no se reflejaría y haría la lógica del programa considerablemente más compleja y más permeable a casos de mal funcionamiento.

4. Futuras extensiones

Una extensión posible sería encontrar la forma de generar procesamiento en paralelo.

Una extensión no tan lejana sería calcular el procentaje junkDNA existente.

5. Referencias

Documentación de Flex - <http://flex.sourceforge.net/manual/>