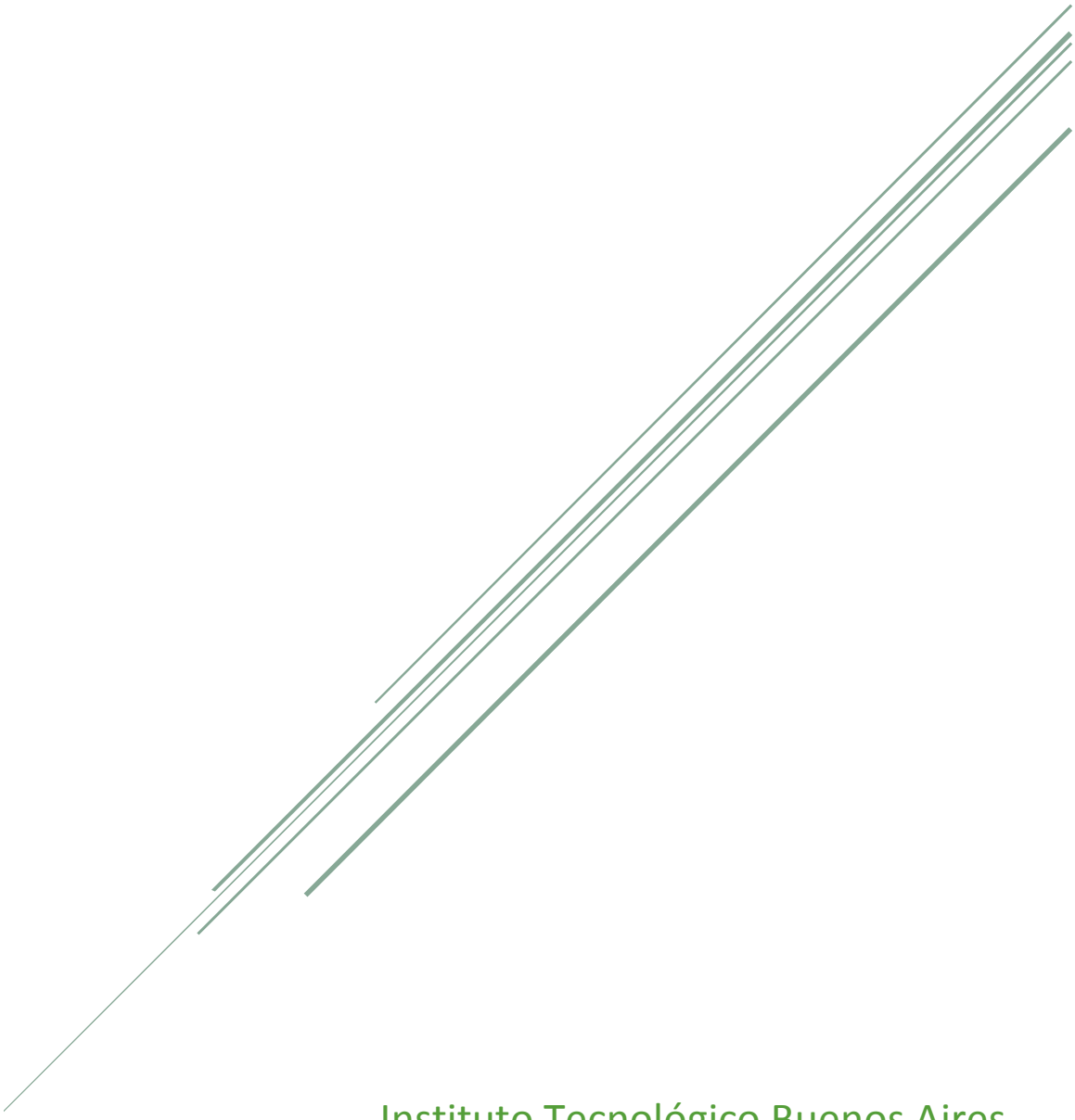
Grupo: María Florencia Besteiro
Maximiliano Valverde
Leandro Matías Rivas

TRABAJO PRÁCTICO

Analizador Léxico Genómico y Compilador Secuenciador de Genomas



Instituto Tecnológico Buenos Aires
Autómatas, Teoría de Lenguajes y Compiladores

Índice

Consideraciones

Para el desarrollo de este trabajo fue necesario tomar decisiones sobre temas que no estaban especificados en la consiga provista por la cátedra. A continuación, se detallarán las más relevantes.

- **Validaciones de tokens de inicio/fin**
  En caso que alguna entrada no respete estrictamente el uso de los tokens de inicio y fin, se descarta completamente, por lo que no produce ninguna salida y se muestra un mensaje de error. Sin embargo, si hay más archivos en cola, serán procesados luego del mensaje de error.
- **Procesamiento de toma de decisión {A, B, C}**
  Se decidió elegir una de las tres cadenas posibles y descartar las otras. El criterio de elección se basa en elegir siempre el primer carácter que se encuentra dentro de las llaves.

- **Procesamiento de Junk-DNA**
  Cualquier cadena o carácter que sea considerada Junk-DNA, es decir, que no representa ningún nucleótido, es salteada por el programa.

Desarrollo

Manejo de Entradas

Durante el desarrollo se contempló la posibilidad de usar varios modos de entrada para facilitar tanto la ejecución como las pruebas.

Para realizar pruebas rápidas se usa el modo **manual**, el cual se accede con el parámetro –**m**. Este modo permite escribir la entrada en el momento y recibir un resultado al presionar la tecla *enter*.

Para facilitar la ejecución, se usa el modo **file,** usando el parámetro **–f.** En este modo se pueden especificar la cantidad de archivos a procesar para luego agregarlos a la cola de ejecución. De esta manera, se pueden obtener múltiples salidas complejas en una sola ejecución.

Analizador léxico genómico

Para la primera parte de este trabajo, debimos implementar un analizador léxico genómico donde dada una entrada (cadena de codones) debíamos generar una salida (cadena de aminoácidos). Para esto implementamos un programa en Lex. Este analiza las expresiones regulares dadas (entrada) y las traduce a una cadena de aminoácidos. Dicha traducción la obtuvimos a partir de una base de datos, que relaciona una cadena de 3 codones de DNA a su aminoácido correspondiente.

Compilador secuenciador de genomas

En la segunda entrega de este trabajo, debimos implementar un compilador secuenciador de genomas que a partir de una entrada (cadena de aminoácidos) lance una respuesta, indicando a qué especie corresponde dicha entrada. En la base de datos utilizadas, se muestra que cadena de aminoácidos corresponde a una proteína, qué cadenas de proteínas corresponden a una condición genética y qué condiciones genéticas, definen a una especie.

Para esto extendimos la implementación realizada para la primera entrega, agregando reglas al programa. Para la traducción final se realizó una implementación en YACC .

Gramática

La gramática[1] que utilizamos para generar las cadenas de genomas es la siguiente:

G = < {S,B,M,L,N,P,Q,R,V}, {A,U,G,'{','}','[',']','-',1,2,3,4,5,6,7,8,9,0,STOP,T,C,*}, S, P >

Donde P = {

S -> AUGBV

B -> {M} * P | {N M} | {-} | {-} * P | L

M -> AL | TL | CL | GL

L -> ʌ| M

N -> M, N | L

[1] *Adjuntado en la entrega se encuentra la gramática en la forma **Backus-Naur**.*

P -> [Q] B

Q -> 1R | 2R | 3R | 4R | 5R | 6R | 7R | 8R | 9R

R-> ʌ| 0 | 0Q | Q

V -> STOP | TAA | TAG | TGA

}

Problemas encontrados

En la primer entrega nos costó entender los requerimientos funcionales. Una vez comprendidos tuvimos inconvenientes con el Lex, ya que era un lenguaje que desconocíamos. Más allá de esto, pudimos lograr el funcionamiento correcto del ***analizador léxico***.

En la segunda entrega, tuvimos inconvenientes para agregar todas las reglas que necesitamos, ya que lex no permitía una gran cantidad de las mismas. Para esto nos vimos en la necesidad que compilar nuestro programa con la opción **–Ca** que resolvió nuestro primer problema. Luego, tuvimos que analizar el funcionamiento del YACC , una vez entendido tuvimos inconvenientes con los ejemplos provistos por la cátedra. Creamos ejemplos propios, como el enunciado indicaba. 6 ejemplos de especímenes que responden a cierta cadena de proteínas. Con nuestros ejemplos, en un principio tuvimos inconvenientes ya que al ser 165 reglas a tener en cuenta, por accidente obviamos algunas. Al resolver esto el programa funcionó correctamente.

Observaciones

En el comienzo del desarrollo, se habían subestimado las capacidades del lenguaje FLEX, por lo que se pensó una manera de realizar los reemplazos usando únicamente el lenguaje C. Para esto se creó una matriz tridimensional usando estructuras. De esta manera se podían realizar búsquedas rápidamente carácter a carácter hasta formar una cadena valida.

Luego de investigar un poco más el lenguaje, descartamos la estructura, ya que era mucho más sencillo de implementar usando el analizador sintáctico y realizando escapadas ocasionales a funciones de C para un procesamiento más específico.

Extensiones

Extender nuestro código es simple, pero extenso ya que cada cadena de aminoácidos puede ser muy extensa y nuevas cadenas de características genéticas mas aún. Pero agregarlas para reconocer nuevas especies, es una tarea simple.