EIF400 – Paradigmas de Programación
Proyecto de programación #1

Profesor: Mag. Georges Alfaro S.

PLANTEAMIENTO DEL PROBLEMA

El problema consiste en construir un programa que pueda aplicar un conjunto de reglas de producción o sustitución a una hilera de entrada para producir un resultado determinado.

El conjunto de reglas a aplicar forma un sistema de reescritura o sustitución llamado **algoritmo de Markov**, concebido por el matemático ruso Andrey Andreevich Markov (1903-1979), quien hiciera contribuciones importantes a la teoría de algoritmos y funciones recursivas. Un algoritmo de Markov es un sistema completo según el modelo de Turing, lo cual quiere decir que permite, en teoría, evaluar cualquier función calculable. De esta manera, un algoritmo de Markov puede servir como modelo computacional. Un lenguaje de programación basado en este modelo debería definir la comparación y empate de patrones (*pattern matching*) como una operación fundamental.

Existe un lenguaje de programación llamado **Refal**, basado en este modelo (http://www.refal.net/). Refal es un lenguaje de programación desarrollado hace unos 50 años, que tiene una estructura muy simple en comparación a otros lenguajes, y ha probado ser útil para la interpretación de documentos XML.

Otros lenguajes de programación se han desarrollado para el manejo de hileras usando el mismo modelo, tales como SNOBOL.

OBJETIVOS DEL PROYECTO

El proyecto tiene como objetivo estudiar algunas de las técnicas fundamentales de análisis de lenguajes formales, en este caso el uso de gramáticas para definir un sistema capaz de efectuar un proceso simple de traducción.

También se busca estudiar técnicas de comparación, reconocimiento, empate y sustitución de patrones.

DESCRIPCIÓN DEL PROYECTO

Descripción general.

El programa a desarrollar debe mostrar una única ventana principal dividida en dos partes: en una sección, el usuario puede examinar y editar las reglas de producción de un algoritmo. Aquí también se puede observar la traza de ejecución del algoritmo . En otra sección, que corresponde con una consola de salida, el programa despliega los resultados obtenidos.

El funcionamiento del programa es muy similar al de la aplicación DrRacket (http://racket-lang.org/), que divide la interfaz principal en un panel de definiciones y otro de interacción.

Descripción general de los algoritmos de Markov.

Un algoritmo de Markov es un conjunto de reglas de producción o sustitución, similares a las de una gramática, que se aplican de manera sistemática sobre una hilera de entrada.

Las reglas son una secuencia de pares de hileras, con la forma:

```
<patrón> → <hilera de reemplazo>
```

Cada regla puede ser una regla simple o terminal.

Para ejecutar o aplicar un algoritmo, dada una cadena de entrada:

- Se comprueban las reglas en orden desde arriba a abajo (es decir, en el orden en que se encuentran definidas las reglas) para ver si alguno de los patrones se puede encontrar o coincide con la cadena de entrada. Todos los patrones se comparan desde la izquierda, y sólo se toma en cuenta la primera aparición del patrón.
- Si uno (o más) se encuentra, usar la primera de ellas para reemplazar la ocurrencia más a la izquierda del texto coincidente en la cadena de entrada con su reemplazo.
- El procedimiento se vuelve a ejecutar desde el inicio y se detiene hasta que ocurra una de dos condiciones:
    - No se puede encontrar ningún patrón que coincida, es decir, no se puede aplicar ninguna regla
    - La regla aplicada es una regla terminal (que finaliza con un punto '.')

Tenga en cuenta que después de cada aplicación de una regla la búsqueda comienza de nuevo desde la primera, excepto en el caso de que se hayan definido etiquetas.

Las reglas pueden etiquetarse opcionalmente. Una etiqueta puede ser cualquier nombre, pero en nuestro caso, serán identificadores formados por letras (mayúsculas o minúsculas, números y el carácter de subrayado '_' (*underscore*). Las reglas para nombrar una etiqueta son las mismas que las que se usan en general en cualquier lenguaje de programación para nombrar variables: el nombre debe comenzar con una letra o el símbolo de subrayado, y debe contener al menos una letra.

Cuando una regla especifica una etiqueta entre paréntesis al lado derecho, esto indica que, una vez aplicada la regla, el control debe dirigirse a la regla indicada en lugar de comenzar desde la primera regla. Esto permite la definición de algoritmos más eficientes que además no son tan dependientes del orden de declaración de las reglas.

El alfabeto de la hilera de entrada debe poder especificarse en el programa, pero normalmente se utilizarán todos los caracteres alfabéticos normales, en mayúscula y minúscula, números y algunos caracteres especiales. El punto('.') no se permite ya que se utilizará como indicador para la producción terminal[1]. El programa debe permitir el uso de algoritmos etiquetados y no etiquetados.

Los símbolos que van a ser utilizados como variables deben ser previamente declarados en el algoritmo, usando una sintaxis adecuada.

[1] Para permitir el uso del punto, se podría incluir el patrón de reemplazo entre comillas o usar una secuencia de escape ('\').

La sintaxis general de la definición de un algoritmo se resume en las siguientes líneas:

```
#symbols {<lista-simbolos>};
#vars {<lista-variables>};
#markers {<lista-marcadores>};
[<etiqueta>:] <patrón-búsqueda> → <patrón-sustitución> [(<etiqueta>)];
% [<comentario>]
```

Las líneas que inician con el símbolo # son directivas para definir los parámetros del algoritmo. Las líneas que inician con el símbolo % son comentarios ignorados por el programa.

En una regla, los elementos indicados por '[' y ']' son opcionales. La flecha que separa el patrón de búsqueda y el patrón de reemplazo es el carácter UNICODE (carácter \u2192). Podría escribirse también como una combinación de caracteres "->". Los patrones de búsqueda y reemplazo no contienen espacios en blanco. Si se desea incluir un espacio en blanco, habrá que encerrar el patrón correspondiente usando comillas dobles o usando una secuencia de escape. Se puede definir una secuencia de escape de la manera tradicional utilizada en varios lenguajes de programación, por medio de un *backslash* '\'.

Los símbolos '<' y '>' no forman parte de la hilera.

Las etiquetas al lado derecho de una regla de producción irán encerradas entre paréntesis.

Un patrón puede encerrarse con comillas dobles, que no pertenecen a la hilera. Se puede usar el símbolo '\' para indicar secuencias de escape, es decir, para utilizar símbolos sin su interpretación usual según la sintaxis.

Las reglas pueden contener los mismos patrones, pero no se pueden repetir las etiquetas. Una variable puede ser un símbolo del alfabeto, pero se interpreta siempre como una variable en el patrón de búsqueda. Cada variable en el patrón de reemplazo debe existir en el patrón de búsqueda para poder tener un valor asignado. Los marcadores NO pueden ser símbolos del alfabeto.

Si no se incluyen las directivas para definir el alfabeto (*symbols*), las variables (*vars*) o los marcadores (*markers*), se asumen las siguientes declaraciones por defecto:

```
#symbols abcdefghijklmnopqrstuvwxyz0123456789
#vars wxyz
#markers αβγδ
```

Las declaraciones pueden repetirse, mientras sean consistentes.

La hilera de entrada puede contener marcadores, pero los marcadores no pueden ser asignados a variables en ningún patrón.

Un algoritmo de ejemplo podría ser:

```
% Algoritmo 1 (Ejemplo canónico).
% Este es un algoritmo de ejemplo que se encuentra en muchos textos
% que describen los algoritmos de Markov y algunas implementaciones.

#symbols abcdefghijklmnopqrstuvwxyz0123456789
#markers ABST

"A" -> "apple"
"B" -> "bag"
"S" -> "shop"
"T" -> "the"
"the shop" -> "my brother"
```

Otro algoritmo de ejemplo:

```
% Algoritmo 2.
% Este algoritmo elimina el último carácter de la hilera de entrada.
% La hilera de entrada debe contener al menos un carácter.

#symbols abcdefghijklmnopqrstuvwxyz0123456789
#vars x
#markers β

P1:    βx → xβ       (P1)
P2:    xβ → Λ.
P3:    x → βx        (P1)
```

En este segundo algoritmo se declaran los símbolos del alfabeto, una variable x y un marcador β. La variable representa en este caso un carácter cualquiera, que puede reemplazar a cualquier símbolo del alfabeto. Una variable representa el mismo valor en una regla particular, pero no tiene alcance más allá de la regla. Los marcadores son símbolos que NO se encuentran en el alfabeto y se utilizan como auxiliares para marcar posiciones en la hilera. El símbolo lambda mayúscula 'Λ' representa una hilera vacía. Cuando la hilera vacía aparece en el patrón a buscar, en el lado izquierdo de la regla, cualquier hilera empata correctamente, coincidiendo desde la izquierda.

Si la hilera de entrada es "abcd", la ejecución del algoritmo produciría, después de aplicar cada regla:

```
abcd  →  βabcd      (aplicando P3. P1 y P2 no se pueden aplicar,
                     al no encontrar el símbolo β en la hilera)
      →  aβbcd      (aplicando P1)
      →  abβcd      (aplicando P1)
      →  abcβd      (aplicando P1)
      →  abcdβ      (aplicando P1)
      →  abc        (aplicando P2. P2 es una regla terminal)
```

Funcionamiento del programa.

El programa debe permitir abrir, editar y guardar algoritmos desde un archivo simple de texto o en formato XML. La idea es permitir que los algoritmos escritos puedan también ser abiertos y modificados usando cualquier editor de texto.

Una vez cargado el algoritmo, el programa permitirá editarlo y ejecutarlo, para lo cual solicitará del usuario una hilera de entrada. La ejecución del algoritmo debe poder mostrarse paso a paso, tal como se hace en el modo de depuración de cualquier IDE; aunque también el algoritmo puede evaluarse completamente de una sola vez. Cuando se ejecute el algoritmo paso a paso, el programa debe mostrar el contenido de la hilera de entrada al aplicar cada producción.

Deberá incluir una opción para ejecutar el algoritmo sobre una lista de hileras de entrada tomada desde un archivo plano de texto.

Puede mostrar una paleta que sirva para seleccionar caracteres especiales que no puedan generarse con el teclado[2]. En especial, deben incluirse todas las letras del alfabeto griego (en minúscula) para poder ser usadas como marcadores. Incluya también símbolos especiales, como el carácter para indicar la hilera nula (Λ[3]).

[2] En algunos sistemas operativos ya existe un mecanismo para incluir símbolos y caracteres especiales.

[3] El símbolo para la hilera nula corresponde con la letra griega lambda (λ) en mayúscula.

Algoritmos de prueba.

Para comprobar el funcionamiento del programa, deberá incluir los siguientes algoritmos para resolver problemas en particular. En este caso, deberá probar algoritmos para:

1. Invertir una hilera.
2. Duplicar una hilera (es decir, concatenar una copia de la misma hilera)
3. Averiguar si una hilera es o no un palíndromo (utilice las hileras “verdadero” o “falso” para indicar el resultado)
4. Convertir un número binario en su equivalente unario (se reemplaza un número binario por una secuencia de símbolos cuya longitud es igual al número)
5. Realizar una suma de números binarios
6. Triplicar números binarios (multiplicarlos por 3)
7. Realizar una multiplicación de números binarios arbitrarios
8. Realizar una suma de números decimales
9. Eliminar los caracteres repetidos en una hilera de manera consecutiva
10. Verificar la estructura de una hilera que contenga únicamente paréntesis, que se encuentren correctamente balanceados.

Los algoritmos deberían poder probarse con cualquiera de los programas elaborados por todos los grupos de trabajo.

CONSIDERACIONES DE IMPLEMENTACIÓN

Escriba el programa utilizando el lenguaje de programación Python. Debe utilizar una interfaz gráfica adecuada. Los programas serán probados de preferencia en la plataforma Linux, pero éstos deberán poderse ejecutar correctamente también en Windows o Mac OS X.

Incluya en el programa las opciones necesarias para poder guardar y recuperar desde un archivo tanto los algoritmos escritos como las hileras de entrada. Incluya además una opción para guardar la traza de ejecución de cualquier algoritmo en un archivo de texto (los archivos estarán codificados en UNICODE).

ENTREGA Y EVALUACIÓN

El proyecto debe entregarse **por medio del aula virtual, en el espacio asignado para ello**. La entrega se hará al finalizar la semana 12 del curso. (**domingo 14 de octubre de 2018**). No se aceptará ningún proyecto después de esa fecha, ni se admitirá la entrega del proyecto por correo electrónico. El proyecto se puede realizar en grupos de **cuatro personas, como máximo**. Deberán enviar un correo al profesor indicado la lista de participantes del grupo antes del martes de la semana 10 (25 de septiembre de 2018)

Se presentará un avance del proyecto, para verificar su desarrollo, antes del domingo 30 de septiembre. Si el avance no es presentado o es presentado de manera incompleta, se rebajará hasta un 15% de la nota final del proyecto. Para el avance, deberá mostrarse la implementación completa de la interfaz del programa, y el diseño de la estructura para el manejo de los algoritmos y su ejecución.

Incluya comentarios en el código de los programas y describa detalladamente cada una de las clases y métodos utilizados.

En caso de que la aplicación no funcione adecuadamente, efectúe un análisis de los resultados obtenidos, indicando las razones por las cuales el programa no trabaja correctamente, y cuáles son las posibles correcciones que se podrían hacer. Durante la revisión del proyecto, es muy importante poder defender adecuadamente la solución propuesta. De existir disponibilidad en el laboratorio, y siempre que no atrase el desarrollo normal de los temas del curso, se revisaría el proyecto en clase, durante la semana siguiente a la entrega.

El proyecto se evaluará de acuerdo con la siguiente ponderación:

Diseño de la solución y documentación		**10%**
	Documentación y análisis de resultados:	10%
Funcionalidad		**90%**
	Manejo de documentos (abrir y guardar programas, abrir archivos de datos):	10%
	Edición de los algoritmos:	10%
	Reconocimiento de patrones:	15%
	Aplicación de reglas:	15%
	Implementación del modo de depuración:	15%
	Implementación correcta de los algoritmos de prueba:	25%

Observaciones generales:

- Los proyectos deben entregarse con toda la documentación, diagramas, código fuente y cualquier otro material solicitado.
- Se debe indicar en cada documento el nombre completo y cédula de cada participante del grupo, indicando el nombre del curso, ciclo lectivo y descripción del trabajo que se entrega. Esto incluye comentarios en cada archivo fuente entregado.
- Los trabajos no se copiarán de ninguna llave USB u otro dispositivo en el momento, sino que se deben entregar en el formato adecuado.
- Si los materiales de entrega no están completos, se penalizará hasta un 15% de la nota correspondiente. Asimismo cualquier trabajo práctico que no sea de elaboración original de los estudiantes (plagio) se calificará con nota 0 (cero) y se procederá como lo indiquen los reglamentos vigentes de la universidad.
- Los trabajos que se reciban después de la fecha señalada para su entrega, **en caso de ser aceptados, serán penalizados con un 30% de la nota por cada día de atraso.**