III.2 PROGRAMACIÓN DIRIGIDA POR SINTAXIS

Un Método de Desarrollo de Sistemas Basado en el Lenguaje del Usuario

*Fernado Galindo Soria**

Resumen

La Programación Dirigida por Sintaxis es un método de desarrollo basado en la Lingüística Matemática y orientado a facilitar la comunicación hombre-máquina.

Este método tiene la característica de permitir desarrollar el sistema de información a partir del lenguaje del usuario, por lo que, el manejo de requerimientos se realiza en un "lenguaje natural restringido" parecido al que utiliza el usuario en su comunicación dentro del área problema.

La idea original partió de los métodos de construcción de compiladores con los cuales a partir de la gramática de un lenguaje se puede construir un traductor capaz de reconocer instrucciones en ese lenguaje, por lo que, originalmente se construyeron reconocedores que tenían como entrada la gramática de un lenguaje muy restringido (tipo FORTRAN, BASIC ó PASCAL) y que interpretaba instrucciones de éste lenguaje. Más adelante se amplio el concepto de lenguaje y se construyeron traductores de lenguajes orientado a base de datos, sistemas operativos y validación de datos entre otros, para lo cual únicamente se cambió la gramática, ya que el reconocedor siguió siendo el mismo, por lo que, se vio que, ahora el problema principal era encontrar la gramática representativa de un lenguaje dado, ya que contando con la gramática el proceso de reconocimiento se tenía resuelto (y en realidad existen muchisimos métodos de reconocimiento de un lenguaje a partir de su gramática en los libros de construcción de compiladores), por lo que, se buscaron y desarrollaron métodos de inferencia gramatical, con los cuales se puede encontrar la gramática de un lenguaje, a partir de ejemplos de éste (ejemplos de estos métodos aparecen en trabajos sobre reconocimiento sintáctico de patrones), más adelante se reordenaron éstos métodos para que quedaran en una forma lógica, en la cual, primero se estudia el área y se encuentran ejemplos significativos del lenguaje (Análisis), más adelante se obtiene una gramática con atributos que representa el lenguaje (Diseño) y finalmente se construye el reconocedor del lenguaje (Implantación).

---

* Fernando Galindo Soria, enero de 1988

INTRODUCCIÓN

En la actualidad se considera que el proceso de razonamiento se desarrolla principalmente mediante métodos deductivos e inductivos; en los **métodos deductivos** se aplica un conjunto de **reglas preestablecidas** para encontrar la solución de un problema dado y por su parte los **métodos inductivos** buscan **obtener precisamente reglas generales o patrones** a partir de casos particulares, de donde se ve que son métodos complementarios.

Sin embargo, en la actualidad, prácticamente todas las formas de programación que se utilizan se orientan al manejo de **sistemas de tipo deductivo** (o sea que aplican un conjunto de reglas preestablecidas para resolver un problema) y esto ocurre independientemente de la aplicación (nomina, manejador de base de datos, sistema experto, simulador de juegos, etc.) y del lenguaje o herramienta automatizada que se utilice (COBOL, FORTRAN, Pascal, LISP, PROLOG, Lotus, etc.), por lo que el informático o programador tiene que desarrollar la parte inductiva del problema a mano (lo cual ha propiciado el surgimiento de métodos para encontrar reglas generales para resolver problemas, como son: el Desarrollo de Sistemas, Algorítmica, Ingeniería de Software, Ingeniería de Conocimientos, etc.).

Por esto surge la idea de desarrollar y en su caso automatizar métodos de **Programación de tipo Inductivo,** mediante los cuales, sea relativamente fácil encontrar el conjunto de reglas generales de un problema a partir de casos particulares.

En éste documento se presenta un método de programación de tipo inductivo basado en la Teoría de Lenguajes y en los métodos de Reconocimiento de Patrones (o formas) y orientados a facilitar la comunicación entre el usuario y la computadora, en un lenguaje lo más natural posible y al manejo por parte de la computadora de formas ó patrones más que de datos o conocimientos específicos.

Este método tiene la característica de permitir desarrollar el sistema de información a partir del lenguaje del usuario. Por lo que, el manejo de requerimientos se realiza en un "lenguaje natural

restringido" parecido al que utiliza el usuario en su comunicación dentro del área problema. A pesar de que tradicionalmente, en la literatura se presenta el problema de manejo del lenguaje natural, como un problema de alta complejidad y que por tal motivo. sólo puede ser atacado por personas de muy alto nivel y no es factible su manejo por la generalidad de los informáticos, se ha detectado que éste planteamiento es erróneo y que por el contrario, atacar los problemas informáticos a base del "lenguaje natural restringido", no sólo es más sencillo que los enfoque tradicionales sino que, además, permite resolver los problemas de una forma más completa, ya que, al encadenarse con los métodos de reconocimiento sintáctico patrones, se tiene un doble mecanismo en el cual a partir de múltiples ejemplos del lenguaje del usuario, se puede encontrar una gramática, en la cual se encuentra precisamente representada la forma del lenguaje, y a partir de esta gramática, es relativamente fácil construir por ejemplo, las instrucciones de un sistema de información o el conjunto de reglas de inferencia de un sistema experto, de donde, el sistema experto o el sistema de información se transforma en la parte deductiva de un sistema mucho más poderoso.

La idea de éste método surge de los métodos de construcción de compiladores, con los cuales a partir de la gramática de un lenguaje, se puede construir un traductor capaz de reconocer instrucciones de ese lenguaje, por lo que, originalmente se construyeron reconocedores que tenían como entrada la gramática de un lenguaje muy restringido (tipo FORTRAN, BASIC ó Pascal) y que interpretan instrucciones de éste lenguaje. Más adelante se amplió el concepto de lenguaje y se construyeron traductores de lenguajes orientados a base de datos, sistemas operativos y validación de datos, entre otros. Para lo cual únicamente se cambió la gramática, ya que el reconocedor siguió siendo el mismo, por lo que se vió que ahora, el problema principal era el de encontrar la gramática representativa de un lenguaje dado, ya que, contando con la gramática el proceso de reconocimiento se tenía resuelto (y en realidad existen muchisimos métodos de reconocimiento de un lenguaje a partir de su gramática, en los libros de construcción de compiladores) por lo que se buscaron y

desarrollaron métodos de inferencia gramatical, con los cuales se puede encontrar la gramática de un lenguaje a partir de ejemplos de éste. (ejemplos de éstos métodos aparecen en trabajos sobre reconocimiento sintáctico de patrones), más adelante se ordenaron estos métodos para que quedaran en una forma lógica y al método resultante se le llamó Programación Dirigida por Sintaxis.

En la actualidad, éste método se ha utilizado para desarrollar sistemas en áreas tan disímbolas como: construcción de compiladores, sistemas operativos, editores, bases de datos, comunicación de datos, seguridad, etc.; lenguajes para paquetes estadísticos, de validación de datos, de sistemas de nominas, de control de paquetes de graficación y simuladores de robots; y aplicando el concepto de lenguaje en áreas no tradicionales se han utilizado para enseñar a la computadora a reconocer reglas ortográfica, análisis de contenido, reconocimiento de imágenes, aprendizaje de juegos (viendo las imágenes y las jugadas como ejemplos de oraciones y para desarrollar modelos neurales y secuencias lógicas de actividades), por lo que, como se observa, el potencial de ésta herramienta es muy grande, ya que en contra de lo que se maneja tradicionalmente, es relativamente sencillo construir sistemas capaces de reconocer un "lenguaje natural restringido", por lo que ésta herramienta ha trascendido el área académica y se utiliza actualmente tanto en empresas privadas como en diversas áreas del sector público, con el fin de desarrollar sus sistemas con una orientación mayor hacia el usuario, por lo que, uno de los objetivos de éste trabajo, es el de difundir éste tipo de métodos y lograr que un grupo mayor de la comunidad de I.A. se involucre en su utilización y en su estudio, ya que, es un área nueva y existen múltiples temas abiertos de investigación, como por ejemplo:

1) *Ampliación del concepto de lenguaje.*

2) *Inferencia Gramatical*

3) *Automatización de las diferentes etapas del método.*

4) *Construcción de Sistemas Evolutivos.*

Con este documento se pretende entre otras cosas:

1) *Mostrar un método de programación no tradicional.*

2) *Lograr que los sistemas de información tengan comunicación más orientada al usuario.*

3) *Mostrar que los métodos de construcción de compiladores no son exclusivos de los especialistas en esa área*, sino que lo mismo se puede aplicar para construir un compilador, un sistema manejador de base de datos, un sistema operativo, una nómina ó sistema de inventarios, un reconocedor de imágenes, con lo cual se ve en un momento dado que tener diferentes aplicaciones no implica que no puedan usar la ,misma herramienta.

4) *Presentar un proceso integrado para resolver problemas* y que actualmente no se maneja de ésta forma (por ejemplo, en los cursos de compiladores. no es común que se indique como obtener el lenguaje ó a partir de éste cómo obtener la gramática).

5) *Mostrar la gran importancia que tiene la teoría de gramáticas,* como herramientas para el desarrollo de sistemas.

6) *Mostrar la gran importancia que tiene la Teoría de Reconocimiento de Formas* (en particular el Reconocimiento Sintáctico de Patrones) como herramientas para el desarrollo de sistemas y mostrar como su integración con la Teoría de Gramáticas, proporciona una herramienta aplicable en diversas aplicaciones.

I CONCEPTOS GENERALES

Como se ha comentado anteriormente, éste método se basan principalmente en herramientas de Lingüística Matemática y de Reconocimiento Sintáctico de Patrones y en particular en las Gramáticas Libres de Contexto (o tipo 2 de la Jerarquía de Chomsky), por lo que, como primer punto se presentará en forma general éste concepto y más adelante se mostrará su ámbito de aplicación.

1 DEFINICIÓN DE GRAMÁTICAS LIBRE DE CONTEXTO

Una Gramática Libre de Contexto es una herramienta, con la cual se puede describir la estructura de las oraciones de un lenguaje, ésta herramienta consiste básicamente en cuatro componentes:

G = <Vn , Vt, S, P > donde:

1) Vn es un conjunto de elementos o variables no terminales, que sirven para representar los cambios de un estado a otro, normalmente las variables no terminales se denotan con letras mayúsculas y equivalen a los nombres de los procesos de un sistema.

2) Vt es un conjunto de elementos o variables terminales, que equivalen a las señales u órdenes sobre el sistema, normalmente se denotan con letra minúscula.

3) Se tiene un elemento no terminal que indica el estado inicial o axioma del sistema, se denota como S y equivale al programa principal de un sistema.

4) P es un conjunto de reglas con las cuales se describe la estructura del problema y son de la forma:

$$A \rightarrow \alpha$$

donde A indica que esa regla sólo se aplica si se encuentra el problema en el estado A (o se llama al proceso A) y α es una cadena de estados y señales (equivale al código del proceso).

P es un conjunto de relaciones o reglas de producción (conjunto de procesos del sistema) con domino y contradominio en V* (V = Vn $\cup$ Vt). V* es el conjunto de todas las posibles cadenas formadas por la concatenación de elementos de V. Si $\alpha \in$ V* se dice que α es una cadena de elementos.

Por ejemplo:

El siguiente conjunto de reglas de una gramática, espera señales que indiquen sumas de variables:

$S \rightarrow v\ A$
$A \rightarrow +\ v\ A$
$A \rightarrow ;$

Las reglas indican que,

si se está en el estado S entonces se debe verificar que llegue la señal **v**, si llega **v** entonces se pasa al estado A.

En el caso del estado A se tienen dos posibles caminos: si llega la señal **+** se toma uno y si llega la señal **;** se toma el otro. Por simplicidad cuando existen varios caminos se puede denotar como:

$A \rightarrow +\ v\ A \mid ;$

donde el símbolo **|** indica camino alterno.

Si no llega alguna de las señales esperadas, entonces se detecta que ese no era un camino válido y se regresa al estado anterior a buscar otro camino, si ya no existe ninguno camino posible entonces la señal es errónea.

En éste ejemplo en particular las cadenas de señales válidas es de la forma:

v ;
v + v ;
v + v + v ;
…
v + v + . . . + v ;

2 EQUIVALENCIA ESTRUCTURAL ENTRE GRAMÁTICAS Y SISTEMAS DE INFORMACIÓN

Si partimos de que, un lenguaje es un mecanismo, con el que se puede representar y transmitir el conocimiento y observamos el lenguaje que utiliza algún área problema, podemos encontrar que

en éste, se encuentran en forma natural los componentes de un sistema de información, **ya que podemos encontrar un conjunto de acciones que se refieren a ciertos objetos en un orden dado**, donde las acciones del lenguaje se pueden representar como acciones en el sistema; a partir de los objetos se pueden obtener las entidades, atributos y valores de la estructura de datos y finalmente a partir del orden en que aparecen las acciones y objetos en la oración se puede encontrar el orden o estructura de control del Sistema de Información.

En particular, mediante la gramática que representa el lenguaje se puede representar la estructura del Sistema de Información. Esta posibilidad, de representar un sistema como una gramática permite por ejemplo analizar y mejor los sistemas, para lo cual, se obtiene la gramática del Sistema de Información y se estudia la estructura del sistema, y en su momento se puede encontra una nueva gramática que realice lo mismo (gramática equivalente) y a partir de ésta proponer un mejor sistema.

Por lo que a continuación se verá que cualquier sistema, programa o estructura de datos se puede representar mediante una gramática, y las gramáticas se pueden ver como un mecanismo para representar sistemas (como los diagramas de Warnier, de Flujo, de Flujo de Datos, etc.)

2.1 EQUIVALENCIA DE GRAMÁTICAS Y PROGRAMAS

Como primer punto, se verá que la estructura de un programa se puede representar con una gramática y viceversa.

La equivalencia entre una gramática y un programa se puede observar si se toma un programa típico.

```
Programa típico
Rutina A
    Ejecuta B
    Si C entonces D sino E
    Mientras se cumple F ejecuta G
    Llama a D
Fin rutina
```

ya que en éste se pueden encontrar cuatro estructuras de control básicas:

1) *Sentencias*

2) *Interacciones*

3) *Decisiones*

4) *Recursión* (Nota: no aparece en el ejemplo)

Y cada una de estas se puede representarlas mediante reglas de una gramática o producciones:

1) *Sentencias*: $A \rightarrow B$

2) *Iteraciones*: $A \rightarrow E^*$

3) *Decisiones:* $A \rightarrow C \mid D$

4) *Recursiónes:* $A \rightarrow d\ A$

Por lo que una rutina de la forma:

```
Rutina A
    B
    D ó E
    (G)*
Fin rutina
```

se puede representar mediante la siguiente producción

$A \rightarrow B \{C \mid D\} G^*$

o de otra forma

$A \rightarrow B\ X\ Y$
$X \rightarrow D \mid E$
$Y \rightarrow G^*$

Por otro lado, a partir de la gramática se puede encontrar el programa, ya que:

1) $S \rightarrow D$

equivale a

```
Rutina S
    llama a D
Fin rutina
```

2) $S \rightarrow A\ B\ C$

equivale a

```
Rutina S
    llama a A
    llama a B
    llama a C
Fin rutina
```

3) $S \rightarrow a\ A$

equivale a

```
Rutina S
    Si a entonces llama a A
Fin rutina
```

4) $S \rightarrow a\ A \mid b\ B$

equivale a

```
Rutina S
    Si a entonces llama a A
    sino
      Si b entonces llama a B
    Fin si
Fin rutina
```

Cualquier programa es equivalente estructuralmente a una gramática por lo que se puede tomar un programas, encontrar su gramática y detectar:

1) El tipo de lenguaje que tiene.

2) Problemas estructurales.

3) Oraciones que no reconoce.

4) etc., etc.

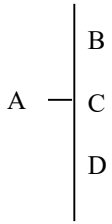
Se puede tomar una gramática y construir un programa que la representa, con lo que se tendría el Sistema de Información.

2.2 EQUIVALENCIA ENTRE GRMÁTICAS Y DIAGRAMAS DE WARNIER

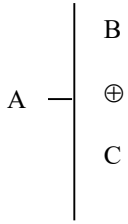
Los diagramas de Warnier son herramientas para representar sistemas, ya que absorben las características de los diagramas Top/Down y de la programación estructurada, con la ventaja de ser sencillos de manejar y con ellos se puede representar fácilmente, por ejemplo, Sistemas de Información, Estructuras de Datos y programas de tipo administrativo.

En estos diagramas se permite:

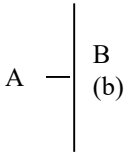
1) La secuencia. La rutina A se compone de B, C y D



2) La decisión. En la rutina A se ejecuta B ó C



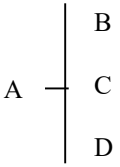
3) La iteración. En la rutina A se repite el proceso B, b veces



Dado que es muy fácil obtener los programas y el sistema de información del diagrama de Warnier, esté es de aplicación general, por lo que veremos que es equivalente a una gramática.

1) $A \rightarrow B\ C\ D$

Equivale al diagrama de Warnier

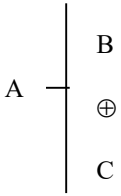


Y este equivale a la regla gramatical:

$A \rightarrow B\ C\ D$

2) $A \rightarrow B \mid C$

Equivale al diagrama de Warnier

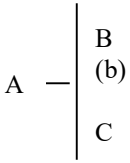


Y este equivale a la regla gramatical:

$A \rightarrow B \mid C$

3) $A \rightarrow \{B\}^{b}\ C$

Equivale al diagrama de Warnier



Y este equivale a la regla gramatical:

$A \rightarrow \{B\}^{b}\ C$

como no es de uso común el exponente se puede sustituir por una gramática recursiva:

$A \rightarrow X\ C$
$X \rightarrow B\ X$

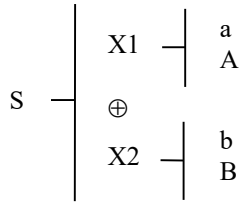
4) $S \rightarrow a\ A \mid b\ B$

En este caso cada parte de la expresión se ve como una rutina independiente, o sea que la regla de producción
$S \rightarrow a\ A \mid b\ B$

equivale a las siguientes reglas

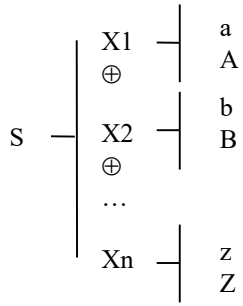
$S \rightarrow X1 \mid X2$
$X1 \rightarrow a\ A$
$X2 \rightarrow b\ B$

Que equivale al siguiente diagrama de Warnier



4) $S \rightarrow a\ A \mid b\ B \mid c\ C \mid \ldots \mid z\ Z$

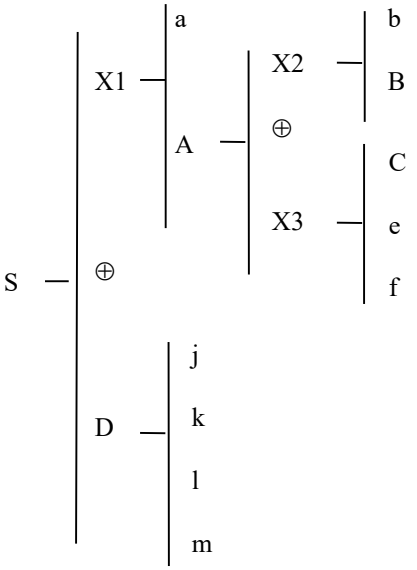
equivale a



5) De la Gramatica

$S \rightarrow a\ A \mid D$
$A \rightarrow b\ B \mid C\ e\ f$
$X \rightarrow j\ k\ l\ m$

Se obtiene el siguiente sistema



2 MÉTODO DE PROGRAMACIÓN DIRIGIDA POR SINTAXIS

El hecho de que sea relativamente fácil representar un sistema de información como una gramática, y que por el otro lado, la gramática represente los patrones o reglas generales de algún lenguaje, plantea la posibilidad de contar con un mecanismo que nos permita obtener a partir del lenguaje, la estructura de un sistema de información y de ahí obtener el sistema.

A continuación, describiremos el método de Programación Dirigida por Sintaxis y en los siguientes puntos se presentará a detalle cada paso con el fin de mostrar su uso y algunas de sus posibilidades.

El método en general consta de una fase de Análisis, en la cual se estudia el área problema y se encuentran ejemplos significativos del lenguaje, otra de Diseño en la cual se encuentra una gramática con atributos, que representa los patrones del comportamiento del lenguaje y la estructura del sistema y finalmente una fase de Implantación, donde se desarrollan las herramientas para reconocer el lenguaje.

Detallando un poco, el método consta de las siguientes parte:

1) Análisis:

i) Detección de un área de aplicación o campo problema.

ii) Estudio del lenguaje del área y detección de ejemplos significativos del lenguaje.

2) Diseño:

i) Detección de las unidades léxicas del enunciado.

ii) Obtención de una Gramática Generativa Canónica, mediante la sustitución de cada una de las unidades léxicas. por un símbolo que indica su tipo de unidad léxica, en las oraciones, e introduciendo un axioma que genere las oraciones.

iii) Obtención de una Gramática Generalizada mediante Técnicas de Inferencia Gramatical.

iv) Obtención de la Gramática con Atributos mediante la inclusión de rutinas semánticas.

v) Detección de problemas estructurales y obtención de una gramática determinística (si existe) o de una gramática con pocos problemas.

3) Implantación:

En este caso se tienen 2 posibles caminos, el primero consiste en reescribir la gramática como un sistema (usando las técnicas presentadas anteriormente y de ahí generar el sistema de computo, si no se tiene mucha experiencia, este es el camino que recomiendo, ahora si tienen algo de experiencia

recomiendo el segundo camino que se describe a continuación.

i) Construcción de una Tabla Gramatical en la que se representa la estructura del sistema a partir de la gramática.

ii) Construcción o uso de algún Reconocedor de Lenguaje (tipo compilador o interprete) que recibe por un lado, la gramática de lenguaje (que representa los patrones de comportamiento del sistema) y por otro los requerimientos del usuario en un lenguaje natural, orientado a la aplicación y genera respuestas a los requerimientos.

III ESTUDIO DEL LENGUAJE Y DETECCIÓN DE EJEMPLOS SIGNIFICATIVOS

Dado que se tiene un área de aplicación o algún problema a resolver, el primer punto importante consiste en la detección del lenguaje involucrado.

Tradicionalmente el concepto de lenguaje está muy restringido y por lo común o se piensa en instrucciones muy restringidas (tipo FORTRAN, Pascal, PROLOG, JCL, QUERY, etc.) o si bien va en términos de "Lenguaje natural restringido", con lo que se quiere indicar algún subconjunto de algún lenguaje natural escrito, como el Español o Ingles, sin embargo el concepto de lenguaje es mucho más amplio, ya que cualquier mecanismo en el que se encuentre un conjunto de elementos (alfabeto), sobre los que se puede aplicar un conjunto de reglas para relacionarlas (sintaxis) y asociarle un significado (semántica), se puede decir que cuenta con un lenguaje, en su momento puede existir lenguajes de múltiples tipos: natural escrito, simbólico, hablado, visual, etc.; por lo que cuando se desarrolla un sistema mediante éste método, el universo de posibles mecanismos de comunicación se amplia enormemente y en su momento es un factor a tomar en cuenta, ya que sería poco adecuado restringirse a lenguajes tradicionales (menú,

ensamblador, pseudocódigo, natural restringido, etc.) si existe algún otro lenguaje más adecuado.

1) En diagnóstico médico, la conversación del paciente con el médico.

2) En la nómina, la descripción escrita de los procesos a realizar y de los cambios o modificaciones requeridas.

3) En visión, la imagen a reconocer. La cual por ejemplo, se puede transformar en una oración tradicional mediante la aplicación de las técnicas de los primeras vecinos y número de forma.

4) Juegos de Tablero, la trayectoria de la pieza. En éste caso la idea es que la trayectoria de una pieza en el tablero, se puede ver como una imagen visual y por tanto aplicarle las herramientas de 3)

5) Características Especiales, en éste caso el concepto de lenguaje pasa a otro nivel, ya que lo que se pretende es representar conceptos no tangibles como: arriba, abajo, al lado, etc. En éste caso se requiere la confluencia de varios mecanismos de captación de información, que la integran en una sola oración, por ejemplo (mecanismo visual, mecanismo de posición y mecanismo escrito).

6) Atributos Generales, partiendo de la idea anterior del uso de varios mecanismos de captación que integran en una sola oración se pueden captar características como: más claro, más obscuro, verde, azul, curvo, etc.

7) Características Temporales, se puede integrar en una sola opción, eventos ocurridos en diferentes tiempos, (introduciendo un operador temporal ↑) y captando por ejemplo cambios lógicos en un sistema (por ejemplo el crecimiento de una célula, se rige por esos cambios lógicos, ya que no se transforma en forma brusca).

8) En pruebas de personalidad, éste es un ejemplo en el cual el concepto de lenguaje es de una simplicidad extrema y su fuerza es muy grande, ya que en éste caso la oración está

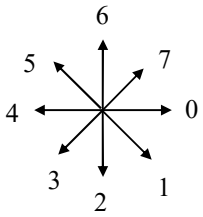
formada únicamente por una cadena de letras, donde cada letra representa la respuesta a una pregunta de la prueba (normalmente tienen un máximo de 5 posibles respuestas) y la cadena representa todas las posibles respuestas dadas por una persona, de donde una oración típica se parece a una cadena binaria.

De los ejemplos anteriores se observa que existen muchos posibles lenguajes y es importante saber elegir el que se adecue más a nuestro problema. Ya que se eligió un lenguaje X el siguiente paso consiste en captar muchos ejemplos de oraciones (grabando, observándolo, sientiéndolo, etc.), como ejemplos distintivos o que muestran alguna característica gramatical que los distinga.

IV DETECCIÓN DE LAS UNIDADES LÉXICAS

En ésta etapa es necesario analizar las oraciones con el fin de detectar las unidades léxicas (Unidades mínimas con significado en el lenguaje) y clasificarlas en tipo de unidades léxicas, por ejemplo:

1) En la información visual se puede tener la siguiente convención:



Donde la dirección de las flechas indican la dirección que se sigue al recorrer una figura. Y las unidades léxicas se obtienen a partir del recorrido de la figura digitalizada.

de donde la figura

se puede representar como: 0 0 0 6 6 6 4 4 3 2 2

2) En un lenguaje se consulta de una base de datos se pueden tener instrucciones como:

Dame	los	nombres	de	las	personas
acción	artículo	atributo	preposición		entidad
a	**b**	**c**	**d**	**b**	**e**

mayores	de	edad	30 años	
operador			valor	
f	**d**	**c**	**g**	

Dime	el	total	de	canciones	en
a	**b**	**a**	**d**	**g**	**d**

S.L.Potosi

g

Escribe	el	CP	de	la	colonia
a	**b**	**g**	**d**	**b**	**c**

Portales

g

Unidades Léxicas

a	b	c	d	…	
Dame Dime Escribe	los las el la	nombres edad	de en		

V. Obtención de la Gramática Generalizada Mediante Inferencia Gramatical

Ya que se ha detectado los componentes del lenguaje (Unidades Léxicas) el siguiente paso consiste en encontrar una Gramática con la cual se obtiene la estructura sintáctica del lenguaje, para lo cual se aplicará el Método de Inferencia Gramatical.

Esta etapa es precisamente la más fuerte dentro del método, ya que permite encontrar los patrones o formas generales de un lenguaje, a partir de ejemplos particulares, con lo que se encadenan las áreas de Lingüística Matemática y Reconocimiento de Formas y se obtiene un mecanismo de programación de tipo inductivo.

V.1 Inferencia Gramatical

La Inferencia Gramatical consiste en encontrar una Gramática a partir de un conjunto de ejemplos del lenguaje que se quiere representar.

Dados 2 conjuntos R+ y R- donde:

$R+ \subseteq L(G)$ y $R- \subseteq L'(G)$

es decir R+ es un subconjunto del lenguaje generado por una Gramática G y R- es un subconjunto de las cadenas que no se deben generar a partir de G, entonces la idea es encontrar la gramática G a partir de la información contenida en R+ y R-.

En particular se verán algunos métodos con los cuales se pueden obtener una gramática a partir de R+ (conjunto de ejemplos de oraciones válida en el lenguaje).

$\{\alpha\} \rightarrow$ ALGORITMO $\rightarrow$ G

donde L(G) = $\{\alpha\}$

Como primer paso de este método se obtiene una gramática canónica, sustituyendo cada unidad léxica en las oraciones ejemplo, por su tipo de unidad léxica e integrando todas las oraciones resultantes en una gramática generada por un axioma S (que se introduce explícitamente).

Dado que se tiene una Gramática Canónica, prácticamente todos los métodos de Inferencia Gramatical se basan en una combinación de operaciones de **Factorización** e introducción de **Recursividad**.

La factorización es una operación sobre las gramáticas parecida a la factorización algebraica, por ejemplo si se tiene:

S $\rightarrow$ a b c d | a b m l (se puede factorizar y queda)
S $\rightarrow$ a b {c d | m l} de donde se obtiene
S $\rightarrow$ a b X
X $\rightarrow$ c d | m l

La factorización tiene la característica de transformar la gramática, por otra gramática débilmente equivalente (o sea que la nueva gramática tiene otra estructura, pero maneja exactamente las mismas oraciones que la vieja) con la cual se absorbe en una sola estructura varios ejemplos particulares.

El proceso de introducción de recursividad nos permite generalizar y obtener a partir de ejemplos concretos estructuras generales (se considera que este mecanismo es el que permite al ser humano manejar una lengua sin necesidad de tener **todas** las oraciones en el cerebro) sin embargo, por esta misma característica se corre el peligro de generalizar tanto que se llegue a aceptar oraciones no válidas en el lenguaje, como ejemplo de recursividad se tiene:

A $\rightarrow$ a B | a a B | a a a B

En este caso se propone que el lenguaje admite cadenas indefinidas de a's y se propone:

$A \rightarrow a\ A \mid B$

como gramática recursiva.

Ejemplos:

1) Factorizar cadenas

$S \rightarrow a\ b\ a\ a\ b\ c \mid a\ b\ a\ b\ b\ a \mid a\ b\ b\ b\ b\ a$

es equivalente a:

$S \rightarrow a\ b\ \{a\ a\ b\ c \mid a\ b\ b\ a \mid b\ b\ b\ a\}$

equivalente a :

$S \rightarrow a\ b\ X$

$X \rightarrow a\ a\ b\ c \mid a\ b\ b\ a \mid b\ b\ b\ a$

equivalente a :

$S \rightarrow a\ b\ X$

$X \rightarrow a\ a\ b\ c \mid \{\ a \mid b\ \}\ b\ b\ a$

equivalente a:

$S \rightarrow a\ b\ X$

$X \rightarrow a\ a\ b\ c \mid Y\ b\ b\ a$

$Y \rightarrow a \mid b$

2) Factorización empotrada (método de Solimonoff modificado)

$S \rightarrow a\ \alpha 1\ B \mid a\ \alpha 2\ B \mid \ldots \mid a\ \alpha n\ B$

produce

$S \rightarrow a\ X\ B$

$X \rightarrow \alpha 1 \mid \alpha 2 \mid \ldots \mid \alpha n$

3) Introducir recursividad (izquierda o derecha) a partir de:

$S \rightarrow a\ a\ b\ b\ c \mid a\ a\ b\ b\ b\ c \mid a\ a\ b\ b\ b\ b\ c$

se obtiene mediante recursividad

$S \rightarrow a\ a\ X\ c$

$X \rightarrow b\ X$

4) Introducir recursividad empotrada a partir de :

$S \rightarrow a\ b\ c \mid a\ a\ b\ c\ c \mid a\ a\ a\ b\ c\ c\ c \mid a\ a\ a\ a\ b\ c\ c\ c$

se obtiene

$S \rightarrow a\ X\ c$
$X \rightarrow b \mid a\ X\ c$

5) Inferencia Gramatical

$S \rightarrow a\ b\ a\ a\ b\ b\ d \mid a\ b\ c\ a\ b\ b\ d \mid a\ b\ c\ a\ b\ b\ b\ d$

factorizando: a b

$S \rightarrow a\ b\ A$
$A \rightarrow a\ a\ b\ b\ d \mid c\ a\ b\ b\ d \mid c\ a\ b\ b\ b\ d$

factorizando: a a y c a

$S \rightarrow a\ b\ A$
$A \rightarrow a\ a\ B \mid c\ a\ B$
$B \rightarrow b\ b\ d \mid b\ b\ b\ d$

Introducciendo recursividad

$S \rightarrow a\ b\ A$
$A \rightarrow a\ a\ B \mid c\ a\ B$
$B \rightarrow b\ B \mid d$

6) Factorizar

$S \rightarrow a\ b\ a\ d\ a\ d\ a \mid a\ b\ c\ d\ b\ e\ f\ d\ c\ a \mid$
$a\ b\ c\ j\ c\ j\ c\ d\ b\ e\ f\ d\ c\ f\ d\ f\ d\ g\ f\ d\ f\ d\ f \mid a\ c\ e\ f\ c\ g \mid$
$j\ a\ h\ b\ e\ i\ g\ j \mid a\ b\ c\ a\ d\ b\ c\ a \mid$
$a\ b\ g\ d\ b\ c\ g$

Primero se ordena alfabéticamente las oraciones:

a b a d g d g
a b c a d b c g
a b c d b e f d c g
a b c j c j c d b e f d c f d f d g f d f d g
a c e f c g
j a h b e i g j

De ahí se factoriza y queda:

S → a A | j a h b e y g j
A → b B | c e f c g
B → a d g d g | c C | g d b c g
C → a d b c g | d b e f d c g | j c j c d b e f d c f d f g f d f d

7) Aplicando la recursión a las oraciones anteriores:

B1 → a d g d g

resulta

B1 → a B2
B2 → d g B2 | λ

entonces:

B → a B2 | c C | g d b c g
B2 → d g B2 | λ
C1 → j c j c d b e f d c f d f d g f d f d g

al introducir

X → c | g | λ
C1 → C2 d b e C3
C2 → j c C2 | λ
C3 → f d X C3 | λ

entonces

C → a d b c g | d b e f d c g | C2 d b e C3
C2 → j c C2 | λ
C3 → f d X C3 | λ
X → c | g | λ

V.1.2 Método del Pivote

Busca cadena dentro de cadenas.

1) Ordenar los ejemplos por el número de elementos.

a *a
(a * a) *a
a* (a * a)
(a * a) * (a * a)
(a* (a * a) *a)
((a * a) *a) *a
(a* (a*) a*a))

(a* (a * a) * a)

2) Busca subcadenas dentro de las cadenas y se toma el más sencillo como Pivote.

VI MANEJO DE LA SEMÁNTICA

Mediante el léxico y la sintaxis únicamente se presentan los elementos básicos de un lenguaje y las relaciones permitidas entre estos elementos, sin embargo no se presenta nada acerca del significado de las palabras y de las estructuras, esto es estudiado por la Semántica.

El problema del significado se puede contemplar como una cebolla compuesta de múltiples capas, en la primera se encuentran las palabras, en la segunda las estructuras sintácticamente válidas y en las siguientes lo que se podría llamar niveles de significado. En el Lenguaje Natural, una expresión válida sintácticamente puede tener múltiples significados, dependiendo del que habla y el que escucha, de la situación Social, Económica, Psicológica, etc. de los habitantes, del tiempo y lugar donde se realiza la comunicación, etc.

Ahora bien, hasta ahora se ha visto que con una gramática se puede representar la estructura de un problema, sin embargo aparte de la gramática se tiene que encontrar las acciones o rutinas semánticas y los datos sobre los que se opera para tener las tres partes que representa a un problema.

Si se observa una oración típica por ejemplo:

Dame la edad y dirección de Juan Pérez

Se nota que se tiene una acción **Dame** la cual indica una señal para que se ejecute una rutina semántica; por su parte **edad** y **dirección** indican los datos que se requieren sobre el ente Juan Pérez, de donde, a partir de una oración, puedo encontrar las acciones que se necesitan representar como rutinas semánticas, los datos que se

están manejando y el orden de ejecución o estructura del problema.

Existen múltiples herramientas encaminados al manejo de la semántica y en su momento se han desarrollado escuelas completas alrededor de éste punto, en particular en éste documento se contarán sólo dos tipos de herramientas: las Gramáticas Generativo-Tansformacionales y las Gramáticas con atributos.

VI.1 Gramáticas Generativo-Transformacionales

Una Gramática Generativo-Transformacional consta de tres componentes:

1) Componente Léxico

2) Componente Generativa.

3) Componente Transformacional.

La componente Generativa es lo que se ha estudiado en los capítulos anteriores.

Ejemplo:

$S \rightarrow v + S \mid v$

La componente Léxica nos permite caracterizar a los elementos léxicos mediante sus atributos.

Dentro de la lingüística transformacional se considera que las oraciones tienen dos niveles.

El nivel superficial que se refiere a la estructura de la oración y el nivel profundo que refiere al significado de la misma, por ejemplo:

1) En A1 “El perro corre en la calle” y A2 “En la calle corre el perro”

Se observa igualdad semántica y diferencia completa sintácticamente , es decir, significado de A1 = significado de A2.

Cuando se agregan reglas transformacionales a la gramática que se tenía entonces se tiene una gramática Generativa-Transformacional.

VI.2 Gramaticas con Atributos

Las gramáticas con atributos son simplemente gramáticas en las que se permite intercalar en las producciones llamadas a rutinas semánticas (una rutina semántica es una rutina que lleva acabo alguna acción) las cuales se representan como Sn donde **n** es el número de la rutina.

En general una gramática con atributos:

$GA = <Vn, Vt, Vs, S, P>$

es una gramática con producciones de la forma:

$A \rightarrow \alpha$

donde

$A \in Vn$

$\alpha \in \{Vn \cup Vt \cup Vs\}^*$

Vn es el conjunto de variables no terminales.

Vt es el conjunto de variables terminales.

Vs es el conjunto de variables semánticas o rutinas semánticas.

$A \rightarrow \alpha$ son las producciones con atributos

VI.3 Normalización de la Estructura

Cuando se construye un sistema de información a partir de la gramática, es posible que el sistema tenga problema de funcionamiento. Lo anterior es porque la gramática representa la estructura del programas y si esta estructura tiene problemas, el programa los tendrá.

Por lo anterior surge la necesidad de Normalizar la gramática con el fin de que la estructura sea la adecuada.

El primer paso de normalización busca que la gramática no tenga islas o variables no usables y por otro lado que siempre genere un lenguaje diferente del vacío.

En el segundo paso de normalización se pretende que, la gramática sea del tipo libre de contexto determinístico, con el fin de que el sistema no tenga indeterminaciones o ambigüedades.

VII IMPLANTACIÓN DEL SISTEMA

Como se comentó anteriormente una gramática es equivalente a un diagrama de Warnier o a un programa, por lo que, realmente la etapa de implantación se puede reducir a tomar la gramática y construir un programa conocido como compilador de compiladores, al cual se le da la gramática y genera el sistema, que reconoce el lenguaje definido por la gramática, sin embargo este método sólo es válido para visualizar el proceso, ya que, en este caso la gramática queda inmersa en el programa, con lo que cualquier modificación a la gramática (o al lenguaje del usuario) requiere modificar el programa, por lo que se han desarrollado otros métodos, en los cuales la gramática se representa mediante alguna tabla, y se construye un programa con el cual se recibe por un lado los requerimientos del usuario en su lenguaje, y por el otro la tabla gramatical y reconoce y ejecuta las oraciones que se le dieron (existen múltiples métodos para realizar este proceso en los libros de compiladores).

Este último enfoque tiene la ventaja de que únicamente se requiere obtener (comprar, desarrollar) una sola vez el programa de reconocimiento y si cambia la gramática únicamente se tiene que cambiar la tabla gramatical con lo que generalizando la idea, prácticamente toda la estructura de control de un sistema puede quedar representada mediante una tabla y ahorrarse gran cantidad de programación.

CONCLUSIÓN

Como puede observarse, este método está orientado al desarrollo de software de cualquier tipo y básicamente integra en un sólo proceso la parte inductiva y deductiva de la programación, ya que,

a partir de un conjunto de ejemplos del lenguaje manejado en un área de conocimiento, se puede encontrar la forma general de este lenguaje (en el documento se manejaron las gramáticas como mecanismo de representación de las formas, pero realmente se puede desarrollar lo mismo con otros sistemas formales, como por ejemplo, con los autómatas) y a partir de esta generar la parte deductiva, que en este caso fue representada mediante las ideas de los compiladores, pero sin embargo se ha generalizado y entre otras muchas aplicaciones se ha encontrado que si:

1) Si en el estudio del lenguaje se detectan objetos (estos se representan en el sistema como entidades, atributos y datos) y acciones (representados como rutinas dentro del sistema) encadenados mediante algún tipo de oración, entonces las rutinas semánticas pueden ir generando una secuencia de llamados a datos y acciones, con lo que se tiene un generador d sistemas como parte deductiva.

2) Si en el estudio del lenguaje se detecta que este consiste en la descripción de un conjunto de síntomas que permiten llegar a un diagnóstico (médico, automotríz, etc.), entonces el método de inferencia gramatical permite obtener precisamente, el conjunto de reglas de inferencia de un sistema experto y la parte deductiva es precisamente el sistema experto.

Por otro lado, prácticamente todo el proceso se puede automatizar, con lo que en su momento las reglas de inferencia del sistema experto se obtienen a partir de ejemplos del usuario en una forma automática.

3) Si en el estudio del lenguaje se detecta que está formada por imágenes o patrones, y nuevamente si el proceso se automatiza, se cuenta con un sistema que "capta" de la realidad la forma de las imágenes y aprende a reconocerlas.

Por otro lado, otro campo de aplicación muy prolífico, se da en el mantenimiento, revisión y reestructuración de sistemas en operación, ya que, a partir del sistema, se puede obtener una gramática con atributos que representa la estructura del sistema y

mediante algoritmos de equivalencia detectar y corregir problemas estructurales.

Si en general el método de Programación Dirigida por Sintaxis se automatiza, tanto en su parte inductiva como deductiva, y se le permite actualizarse continuamente con ejemplos nuevos y con la validez o no validez de sus resultados, entonces se cuenta con un **Sistema Evolutivo**.