3 SINTAXIS Y BNF

Un Lenguaje de Programación (LP) es una notación utilizada para describir algoritmos y estructuras de datos que resuelven problemas computacionales.

En este capítulo nos ocuparemos de la SINTAXIS de los Lenguajes de Programación y cómo se describe la misma. Desde el punto de vista sintáctico, hay una relación muy importante entre los LPs y los Lenguajes Formales ya que un LP está formado, básicamente, por un conjunto de LRs y un conjunto de LICs, como veremos en este capítulo.

La descripción precisa de esta sintaxis se realiza, normalmente, en base a notaciones que tienen ciertas propiedades similares a las de las Gramáticas Formales que estamos utilizando. A estas notaciones las llamaremos, genéricamente, BNF.

3.1 INTRODUCCIÓN A LA SINTAXIS

Como ya se ha dicho, sintácticamente un LP está compuesto por un conjunto de LRs y otro conjunto de LICs. Las CATEGORÍAS LÉXICAS o TOKENS (palabra en inglés muy utilizada) −como son los identificadores, las palabras reservadas, los números enteros, los números reales, los caracteres, las cadenas constantes, los operadores, y los caracteres de puntuación− constituyen diferentes LRs. Algunos de estos lenguajes son finitos, como los operadores, y otros son infinitos, como sucede con los identificadores y los números (tanto enteros como reales).

Por otro lado, las expresiones y las sentencias de un LP son, en general, LICs. Los llamaremos CATEGORÍAS SINTÁCTICAS. Como tales, estos lenguajes no pueden ser generados por GRs ni por GQRs –como sí ocurre con los LRs– sino que requieren ser generados por GICs.

Es importante recordar que todos estos tipos de gramáticas se caracterizan porque sus producciones tienen un único noterminal en el lado izquierdo.

➔ En esta área de estudio no tenemos en cuenta las restricciones físicas que imponen las computadoras (cantidad de bytes para almacenar un número real, cantidad máxima de caracteres en una línea del monitor, etc.). Por eso podemos hablar de lenguajes infinitos cuyos elementos tienen longitudes indeterminadas aunque finitas, como ocurre con los Lenguajes Formales.

➔ Una Gramática Formal no solo GENERA un Lenguaje Formal, sino que también se la puede utilizar para DESCRIBIR la sintaxis del lenguaje que genera. Este concepto está muy relacionado con el desarrollo de este capítulo.

La sintaxis de un LP debe describirse con precisión, utilizando una notación sin ambigüedades. La gramática cumple con este requisito, pero tiene muy pocos metasímbolos; esto dificulta, muchas veces, la escritura y posterior comprensión de la sintaxis definida.

Por ello, la notación que se utiliza para describir la sintaxis de un LP es la BNF, que en sus comienzos era muy similar a "producciones de una GIC o de una GQR" y que luego se fue extendiendo con nuevos metasímbolos.

> Hopcroft y Ullman [1979] dicen: “El origen de la formalización de las GICs se encuentra en Chomsky [*1956, "Three models for the description of language"*]. La notación relacionada llamada BNF fue usada para describir el lenguaje ALGOL en Backus [*1959, "The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference"*] y Naur [*1960, "Report on the algorithmic language ALGOL 60"*]. La relación entre GIC y BNF fue percibida en Ginsburg y Rice [*1962, "Two families of languages related to ALGOL"*].”

Comenzaremos con dos secciones introductorias en las que trabajaremos con Gramáticas Formales y luego pasaremos a ver la utilización de BNF en la descripción de la sintaxis de un Lenguaje de Programación. De esta manera, percibiremos claramente la evolución de la notación BNF.

3.2 IDENTIFICADORES Y SU SINTAXIS

El elemento más utilizado en todo LP es el IDENTIFICADOR: una secuencia de uno o más caracteres que nombra diferentes entidades de un LP (variables, funciones, procedimientos, constantes, tipos, etc.). Los “identificadores” constituyen un LR infinito. Habitualmente, las PALABRAS RESERVADAS, que también están compuestas por una secuencia de caracteres, forman un LR finito, independiente de los identificadores.

Ejemplo 1
Como escribe David Watt en su libro *Programming Language Syntax and Semantics* [1991]: "La siguiente es la especificación informal de la sintaxis de identificadores, parafraseada de un manual de un viejo lenguaje de programación: Un identificador es una secuencia de letras mayúsculas, posiblemente con la inclusión de guiones bajos en medio." A partir de esta especificación en lenguaje natural, es evidente que PI, CANTIDAD y C_TOTAL son identificadores correctos. También es muy claro que PI34, 78AA y CANTIDAD___ no son identificadores válidos.

* Ejercicio 1 *
Verifique que la afirmación escrita en el último párrafo del Ejemplo 1 es correcta para cada una de las cadenas descriptas.

Pero, si nos atenemos a la definición dada en el Ejemplo 1, pueden surgir preguntas como las siguientes:

(1) ¿Es una única letra, como X, un identificador válido? (la especificación dice "una secuencia de letras");

(2) ¿La frase "guiones bajos en medio" significa que puede haber varios consecutivos (la frase está en plural) o que deben estar separados?

* Ejercicio 2 *
En base a la especificación dada en el Ejemplo 1, ¿puede responder con seguridad a las preguntas formuladas en el párrafo anterior?

* Ejercicio 3 *
¿Se le ocurre alguna otra ambigüedad en la especificación informal que estamos analizando?

Estas imprecisiones o ambigüedades son casi inevitables en una especificación informal de una sintaxis, como la que se realiza, habitualmente, utilizando un lenguaje natural.

Ahora bien; supongamos que, para que resulte más claro, usamos una GIC para describir estos identificadores, pero con una pequeña modificación respecto de lo visto en el Capítulo 2: los noterminales serán representados por nombres significativos (por ejemplo, Nombre), en lugar de una sola letra mayúscula. Como siempre, y por convención, el axioma, que es lo que estamos definiendo, aparece en la primera producción.

Entonces, una definición precisa de los identificadores del LP mencionado en el Ejemplo 1 podría ser la siguiente:

Ejemplo 2

```
Identificador -> Letra |
                 Identificador Letra |
                 Identificador GuiónBajo Letra
GuiónBajo -> _
Letra -> A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
          P | Q | R | S | T | U | V | W | X | Y | Z
```

Análisis: Recordemos que toda producción tiene un lado izquierdo, el metasímbolo "operador de producción" y un lado derecho. Recordemos, también, que en la escritura de las producciones solo existen noterminales, terminales y metasímbolos. En este caso, los noterminales son palabras o frases significativas (sin espacios), los únicos metasímbolos son los caracteres `->` y `|`, y los terminales son los restantes caracteres.

> ➔ Es muy importante recordar que todo noterminal debe aparecer en el lado izquierdo de, al menos, una producción; por otro lado, un terminal jamás puede aparecer en el lado izquierdo de una producción. Esta ubicación en el contexto de las producciones también facilita saber cuál es un noterminal y cuál es un terminal, aunque ambos tengan el mismo nombre.

Se nota fácilmente que Identificador debe ser el axioma porque es lo que estamos definiendo. Observe que la primera producción nos dice que un identificador puede ser solo una letra (por ejemplo, X), mientras que las otras dos producciones son recursivas (porque Identificador aparece en ambos lados de la producción) y, por lo tanto, permiten generar identificadores de cualquier longitud. Por supuesto, la primera producción debe ser utilizada para terminar el proceso recursivo. Además, se observa que la última producción no permite que haya dos guiones bajos consecutivos.

En otras palabras: el lenguaje de los "identificadores" descripto en el Ejemplo 1 está perfectamente definido mediante esta GIC. Esto es: si una cadena es un identificador de este LP, la podremos derivar a partir del axioma Identificador; caso contrario, no la podremos derivar.

Nota 1

OBSERVACIÓN IMPORTANTE. Esta GIC produce un Identificador de derecha a izquierda por el formato que tienen las producciones recursivas. Este formato se denomina recursiva a izquierda porque el noterminal del lado izquierdo aparece, también, como primer símbolo del lado derecho.

* Ejercicio 4 *

Construya una tabla de derivación para generar el identificador R_X_A.

* Ejercicio 5 *

Verifique, mediante derivación, que A__B (hay dos guiones bajos consecutivos) no es un identificador válido para el LP que estamos analizando.

* Ejercicio 6 *
Convierta la GIC del Ejemplo 2 en una GQR y compare ésta con la GIC utilizada..

El Identificador que hemos definido en el Ejemplo 2 mediante una GIC recursiva a izquierda también puede ser definido mediante una GIC recursiva a derecha, aunque sus producciones resultarán más complejas:

Ejemplo 3

```
Identificador -> Letra |
                 Letra Resto
Resto -> Letra Resto |
         GuiónBajo Letra Resto |
         ε
GuiónBajo -> _
Letra -> A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
         P | Q | R | S | T | U | V | W | X | Y | Z
```

* Ejercicio 7 *
Construya una tabla de derivación para generar el identificador R_X_A a partir de la gramática del Ejemplo 3.

* Ejercicio 8 *
Verifique mediante derivación, utilizando la GIC del Ejemplo 3, que A__B (hay dos guiones bajos consecutivos) no es un identificador válido para el LP que estamos analizando.

3.3 LAS EXPRESIONES Y LA SINTAXIS

Uno de los puntos más importantes en la sintaxis de un LP está representado por las EXPRESIONES que se pueden escribir en ese LP. Una GIC para expresiones no solo debe producir todas las expresiones válidas, sino que también debe preocuparse por la precedencia (o prioridad) y la asociatividad (evaluación de izquierda a derecha o de derecha a izquierda para operadores con igual precedencia) de cada uno de los operadores existentes en ese LP.

Para comenzar, desarrollaremos una GIC que genera las expresiones aritméticas con multiplicación y suma como únicos operadores. Esta GIC también reconoce el uso de paréntesis, empleado especialmente para modificar las prioridades, como, por ejemplo, (2+34)*5.

Los operandos, para simplificar la situación, son solo números enteros sin signo; además, asumiremos que ya están definidas las producciones para el noterminal Número.

Entonces, en esta GIC los terminales son los caracteres + (suma), * (multiplicación) y los paréntesis; el noterminal Número representará cualquier número entero no negativo que actuará como operando.

Por lo tanto, una GIC para las expresiones aritméticas con estas restricciones se describe en el siguiente ejemplo:

Ejemplo 4

```
Expresión -> Término |
             Expresión + Término
Término -> Factor |
           Término * Factor
Factor -> Número |
          ( Expresión )
```

* Ejercicio 9 *

Complete la GIC del Ejemplo 4, desarrollando las producciones para el noterminal Número.

➔ Representación de las Prioridades de los Operadores en una GIC: Cuánto más cerca del axioma, menor es la prioridad de un operador. Vea lo que sucede con la suma, la multiplicación y los paréntesis (si éstos fueran un operador) en la GIC diseñada en el Ejemplo 4. Esta GIC respeta la prioridad y la asociatividad de los operadores que intervienen.

Veamos un ejemplo de derivación con esta GIC para analizar esta situación:

Ejemplo 5

Sea la expresión 1+2*(3+4)+5. Leyéndola de izquierda a derecha, observamos que primero debemos resolver la subexpresión que está entre paréntesis; luego podremos realizar la multiplicación y, a continuación, seguir adelante con la evaluación. Apliquemos el proceso de derivación a izquierda.

Recordemos las producciones de la GIC:

```
1 Expresión -> Término |
2              Expresión + Término
3 Término -> Factor |
4            Término * Factor
5 Factor -> Número |
6           ( Expresión )
7 Número -> 1 | 2 | 3 | 4 | 5
```

(agregado para completar el proceso de derivación)

Entonces, el proceso de derivación será:

```
Expresión
Expresión + Término
Expresión + Término + Término
Término + Término + Término
Factor + Término + Término
Número + Término + Término
1 + Término + Término
1 + Término * Factor + Término
1 + Factor * Factor + Término
1 + Número * Factor + Término
1 + 2 * Factor + Término
1 + 2 * ( Expresión ) + Término
1 + 2 * ( Expresión + Término ) + Término
1 + 2 * ( Término + Término ) + Término
1 + 2 * ( Factor + Término ) + Término
1 + 2 * ( Número + Término ) + Término
1 + 2 * ( 3 + Término ) + Término
1 + 2 * ( 3 + Factor ) + Término
1 + 2 * ( 3 + Número ) + Término
```

```
1 + 2 * ( 3 + 4 ) + Término
1 + 2 * ( 3 + 4 ) + Factor
1 + 2 * ( 3 + 4 ) + Número
1 + 2 * ( 3 + 4 ) + 5
```

➔ Recordatorio: la secuencia de terminales y noterminales que se obtiene en cada paso de una derivación se denomina CADENA DE DERIVACIÓN.

Ejemplo 6
En el proceso de derivación anterior, `1+2*(Número+Término)+Término` es una de las cadenas de derivación.

* Ejercicio 10 *
¿Cuál es la menor expresión que se puede derivar desde la GIC definida en el Ejemplo 4 y re-escrita en el Ejemplo 5? Escríbala y justifique su respuesta.

* Ejercicio 11 *
¿Es ((2)) una expresión válida? Demuestre que sí o que no por derivación. Para cada cadena de derivación obtenida, indique la producción que fue aplicada.

* Ejercicio 12 *
Intente derivar 1+2++3.

3.3.1 LA EVALUACIÓN DE UNA EXPRESIÓN, PRECEDENCIA Y ASOCIATIVIDAD

La EVALUACIÓN de una expresión se realiza como un proceso inverso al de la derivación, hasta llegar al axioma. En otras palabras: debemos hacer una REDUCCIÓN de la tabla generada por la derivación. Es aquí donde se ve si la precedencia o prioridad de los operadores está correctamente determinada por la gramática diseñada.

Esta reducción debe hacerse en el orden inverso a la derivación; es decir: el último paso de la derivación será el primero de la reducción, y así sucesivamente. Para ello, debe tenerse en cuenta la producción aplicada en cada paso de la derivación. En el próximo ejemplo re-escribiremos la derivación realizada en el Ejemplo 5, pero indicando la producción aplicada en cada paso. Luego, haremos la evaluación de la expresión mediante el proceso de reducción.

Ejemplo 7
Repetimos la derivación para la expresión 1+2*(3+4)+5. Aplicamos el proceso visto en el Ejemplo 5 a partir de la GIC con producciones

```
1 Expresión -> Término |
2              Expresión + Término
3 Término -> Factor |
4            Término * Factor
5 Factor -> Número |
6           ( Expresión )
7 Número -> 1 | 2 | 3 | 4 | 5
```

y construimos la siguiente TABLA DE DERIVACIÓN:

PRODUCCIÓN APLICADA	CADENA DE DERIVACIÓN OBTENIDA
(axioma)	Expresión
2	Expresión + Término
2	Expresión + Término + Término
1	Término + Término + Término
3	Factor + Término + Término
5	Número + Término + Término
7	1 + Término + Término
4	1 + Término * Factor + Término
3	1 + Factor * Factor + Término
5	1 + Número * Factor + Término
7	1 + 2 * Factor + Término
6	1 + 2 * (Expresión) + Término
2	1 + 2 * (Expresión + Término) + Término
1	1 + 2 * (Término + Término) + Término
3	1 + 2 * (Factor + Término) + Término
5	1 + 2 * (Número + Término) + Término
7	1 + 2 * (3 + Término) + Término
3	1 + 2 * (3 + Factor) + Término
5	1 + 2 * (3 + Número) + Término
7	1 + 2 * (3 + 4) + Término
3	1 + 2 * (3 + 4) + Factor
5	1 + 2 * (3 + 4) + Número
7	1 + 2 * (3 + 4) + 5

Ahora, evaluaremos esta expresión. Para ello, como ya adelantamos, aplicaremos el proceso de reducción, que es el proceso inverso al de derivación: partimos de la secuencia de terminales que forman la expresión y finalizamos cuando llegamos al axioma de la GIC.

Construiremos una tabla que llamaremos TABLA DE EVALUACIÓN. Como verán en la tabla, junto a cada noterminal se agrega el número (terminal) del cual proviene, para tenerlo presente al realizar la correspondiente evaluación. La Tabla de Evaluación es la siguiente:

CADENA DE DERIVACIÓN A REDUCIR	PRODUCCIÓN A APLICAR	OPERACIÓN
1 + 2 * (3 + 4) + 5	7	
1 + 2 * (3 + 4) + Número5	5	
1 + 2 * (3 + 4) + Factor5	3	
1 + 2 * (3 + 4) + Término5	7	
1 + 2 * (3 + Número4) + Término5	5	
1 + 2 * (3 + Factor4) + Término5	3	
1 + 2 * (3 + Término4) + Término5	7	
1 + 2 * (Número3 + Término4) + Término5	5	
1 + 2 * (Factor3 + Término4) + Término5	3	
1 + 2 * (Término3 + Término4) + Término5	1	
1 + 2 * (**Expresión3 + Término4**) + Término5	2	3 + 4 = 7
1 + 2 * (Expresión7) + Término5	6	
1 + 2 * Factor7 + Término5	7	
1 + Número2 * Factor7 + Término5	5	
1 + Factor2 * Factor7 + Término5	3	
1 + **Término2 * Factor7** + Término5	4	2 * 7 = 14
1 + Término14 + Término5	7	
Número1 + Término14 + Término5	5	

Factor1 + Término14 + Término5	3	
Término1 + Término14 + Término5	1	
Expresión1 + Término14 + Término5	2	1 + 14 = 15 **(*)**
Expresión15 + Término5	2	15 + 5 = 20
Expresión20	(axioma)	Resultado Final

(*) Aquí se ve un ejemplo muy claro de "qué es la ASOCIATIVIDAD". El operando 14 se puede sumar al operando 1 (a su izquierda) o al operando 5 (a su derecha). Como la suma es asociativa a izquierda, el 14 se suma al 1 y no al 5.

En general: cuando hay tres operandos vinculados con operadores de igual precedencia (como en este caso), el operando central operará primero con el operando de la izquierda si la operación es asociativa a izquierda (como sucede aquí); pero si la operación es asociativa a derecha, el operando central operará primero con el operando de la derecha. Un ejemplo de esta última situación lo veremos más adelante, cuando analicemos el BNF del lenguaje ANSI C.

Además, en la Tabla de Evaluación se ve claramente cómo la GIC determina la PRECEDENCIA de los operadores: la expresión entre paréntesis es evaluada primero, luego evalúa la multiplicación y, finalmente, las sumas.

Nota 2
Para construir una Tabla de Evaluación es indispensable partir de la correspondiente Tabla de Derivación y proceder en orden inverso (reducción).

* Ejercicio 13 *
Siguiendo el modelo presentado en el Ejemplo 7, construya una Tabla de Evaluación que muestre la evaluación de la expresión: (1 + 2) * (3 + 4)

Ejemplo 8
Para terminar de comprender totalmente la GIC para expresiones utilizada en los ejemplos y ejercicios anteriores, analicemos el siguiente caso. Supongamos que consideramos que no es necesario que haya tantos niveles de noterminales (expresión, término, factor, número) para una GIC que "solo" genera expresiones aritméticas con los operadores de suma y de multiplicación. Diseñamos, entonces, una GIC más simple que solo tiene dos noterminales: E (por expresión) y N (por número). Esta GIC tiene las siguientes producciones:

```
1 E -> E + E |
2      E * E |
3      (E) |
4      N
5 N -> 1 | 2 | 3 | 4 | 5
```

* Ejercicio 14 *
Describa formalmente la GIC del Ejemplo 8.

Ejemplo 9
Replicaremos el Ejemplo 7, pero ahora utilizando la nueva GIC cuyas producciones figuran en el Ejemplo 8.

Sea, nuevamente, la expresión 1+2*(3+4)+5. Aplicamos el proceso de derivación a izquierda y construimos la siguiente Tabla de Derivación:

PRODUCCIÓN APLICADA	CADENA DE DERIVACIÓN OBTENIDA
(axioma)	E
1	E + E
2	E * E + E
1	E + E * E + E
4	N + E * E + E
5	1 + E * E + E
4	1 + N * E + E
5	1 + 2 * E + E
3	1 + 2 * (E) + E
1	1 + 2 * (E + E) + E
4	1 + 2 * (N + E) + E
5	1 + 2 * (3 + E) + E
4	1 + 2 * (3 + N) + E
5	1 + 2 * (3 + 4) + E
4	1 + 2 * (3 + 4) + N
5	1 + 2 * (3 + 4) + 5

Como se ve, no solo hemos podido derivar la expresión dada sino que, además, lo hemos realizado en menor cantidad de pasos que utilizando la GIC del Ejemplo 7. ¡Parecería que esta GIC es mejor que la anterior!

Siguiendo el ejemplo anterior, evaluaremos esta expresión aplicando el proceso de reducción. Nuevamente, junto al noterminal colocaremos el número (terminal) que le corresponde, con esta notación: `noterminal.terminal`.

Ejemplo 10
Obtenemos, entonces, la siguiente Tabla de Evaluación:

CADENA DE DERIVACIÓN A REDUCIR	PRODUCCIÓN A APLICAR	OPERACIÓN
1 + 2 * (3 + 4) + 5	5	
1 + 2 * (3 + 4) + N.5	4	
1 + 2 * (3 + 4) + E.5	5	
1 + 2 * (3 + N.4) + E.5	4	
1 + 2 * (3 + E.4) + E.5	5	
1 + 2 * (N.3 + E.4) + E.5	4	
1 + 2 * (**E.3** + **E.4**) + E.5	1	3 + 4 = 7
1 + 2 * (E.7) + E.5	3	
1 + 2 * E.7 + E.5	5	
1 + N.2 * E.7 + E.5	4	
1 + E.2 * E.7 + E.5	5	
N.1 + E.2 * E.7 + E.5	4	
E.1 + **E.2** * E.7 + E.5	1	1 + 2 = 3
E.3 * **E.7** + E.5	2	3 * 7 = 21
E.21 + **E.5**	1	21 + 5 = 26
E.26	(axioma)	Resultado Final

Como se observa, el resultado final es incorrecto, y esto se debe a que la segunda gramática está mal diseñada porque no resuelve correctamente el problema de precedencia de los operadores.

* Ejercicio 15 *
Utilizando la GIC cuyas producciones están en el Ejemplo 8, obtenga otra Tabla de Derivación para la expresión del Ejemplo 9. Luego, construya la Tabla de Evaluación correspondiente. ¿Cuáles son sus conclusiones?

* Ejercicio 16 *
¿Puede hacer lo mismo que hizo en el ejercicio anterior, pero con la GIC del Ejemplo 7? Justifique su respuesta.