Gramáticas Libres de Contexto

Alan Reyes-Figueroa
Teoría de la Computación (Aula 12) 28.agosto.2024

Formalismo, Derivaciones, Forma de Backus-Naur, Derivaciones izquierdas y derechas

Comentarios informales

- Una *gramática libre del contexto* (context-free grammar) es una notación para describir lenguajes.
- Es más general que los autómatas finitos y que las regexp, pero no describe a todos los posibles lenguajes.
- Útil para describir estructuras anidadas, e.g., paréntesis.

Comentarios informales

- La idea básica consiste en usar “variables” para describir conjuntos de cadenas.
- Estas variables se definen en forma recursiva, en términos de otras.
- Las reglas de recursión (“producciones”) envuelven sólo a la concatenación.
- Reglas alternativas, puede incluir unión.

Gramáticas nivel 2

- Sintaxis = estudia las reglas y principios que gobiernan la combinatoria de constituyentes sintácticos y la formación de unidades superiores a estos.
- En sistemas matemáticos y en el contexto de compiladores, la sintaxis es expresada comúnmente usando la notación BNF (Backus-Naur Form).

Gramáticas nivel 2

- Introducida por John Backus de IBM para describir la sintaxis de ALGOL 58 (1959), y, más adelante, simplificada por Peter Naur (1963).
- Esta notación describe conjuntos de reglas gramaticales, denominados gramáticas. Nos concentramos en un tipo específico de gramáticas, llamadas **gramáticas de tipo 2 o gramáticas libres de contexto**.

Gramáticas nivel 2

Una gramática libre de contexto posee:

- Un conjunto de símbolos *terminales*, definidos como elementos inmediatamente reconocibles e indivisibles (e.g., símbolos,
  palabras reservadas,
  la cadena vacía ε).
- Un conjunto de símbolos *no-terminales* o *variables*, que sirven para crear símbolos compuestos que representan estructuras sintácticas. Por ejemplo:
  frases,
  oraciones,
  estructuras

Gramáticas nivel 2

- Su composición se especifica con *producciones*, esto es, reglas conformadas por (de izquierda a derecha) un símbolo **no-terminal**, una flecha (→) y una secuencia de símbolos **terminales y/o no-terminales**.

  NT → T/NT | T/NT |... | T/NT

- El término del lado izquierdo de la flecha es la *cabeza de la producción*, mientras que lo que está en el lado derecho es el *cuerpo*.
- Un no-terminal designado como el *símbolo de inicio*, regularmente identificado por estar a la cabeza de la primera producción en la gramática.

Gramáticas nivel 2

- Se pueden definir varias producciones con una misma cabeza, es normal agrupar éstas en una única producción con diferentes cuerpos separados por el símbolo “|”, que se lee como “or” (similar a como hicimos con las definiciones regulares).

Ejemplo:

- Símbolos terminales: {+, –, 0, 1, …, 9}
- Símbolos no-terminales: {list, digit}
- Símbolo inicial = list
- Producciones:

  list $\rightarrow$ list + digit

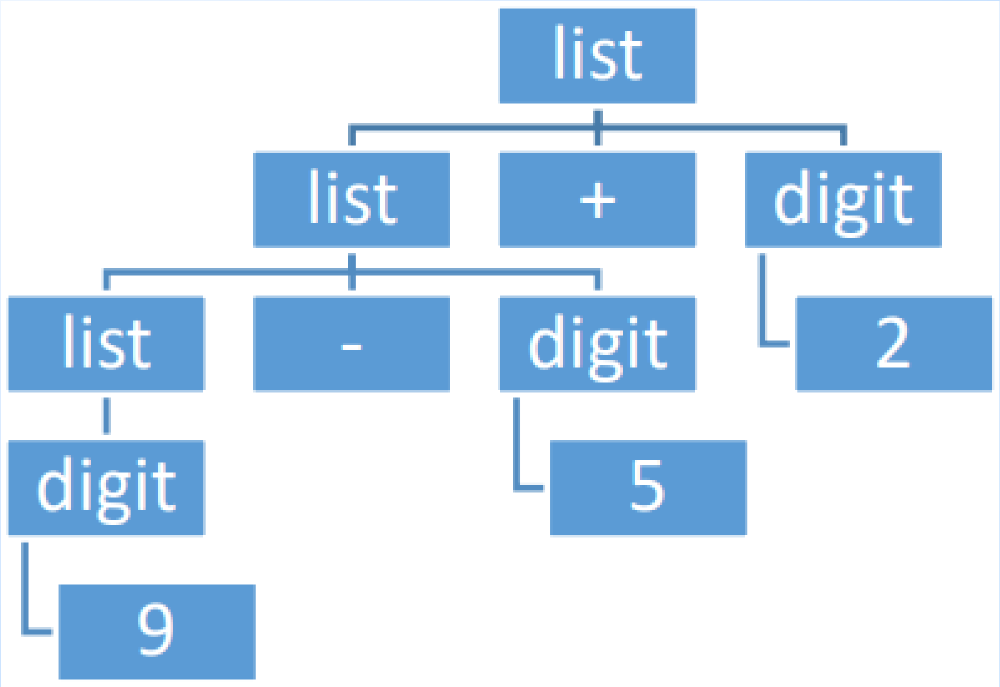
  list $\rightarrow$ list – digit

  list $\rightarrow$ digit

  digit $\rightarrow$ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
- Lenguaje: cadenas de dígitos separados por sumas y restas.

Ejemplo: árbol sintáctico

- Por ejemplo, la cadena **9 – 5 + 2**, se puede construir a partir de las producciones como:



Ejemplo: $\{0^n1^n: n \geq 1\}$

Construir una gramática CFG para el lenguaje $\{0^n1^n: n \geq 1\}$.

- Terminales: {0,1}
- No-terminales: {S}
- Símbolo inicial = S
- Producciones:

$$S \rightarrow 01$$
$$S \rightarrow 0S1$$

Producciones

- Variable → cadena de variables y terminales.
- Convención:
  - A, B, C,… denotan variables (no-terminales).
  - a, b, c,… denotan terminales.
  - …, X, Y, Z denotan terminales o variables.
  - …, w, x, y, z son cadenas de terminales solo.
  - α, β, γ,… son cadenas de terminales o variables.

Derivaciones

- Una cadena en el lenguaje de una CFG se *deriva* comenzando con el símbolo inicial, y reemplazando, repetidamente, alguna variable A por el lado derecho de alguna de sus producciones.
  - Las "producciones for A" son aquellas que tienen A en el lado izquierdo de →.

Derivaciones

- Escribimos

$$\alpha \; A \; \beta => \alpha \; \gamma \; \beta$$

si $A \rightarrow \gamma$ es una producción.

- Ejemplo: $S \rightarrow 01$
  $S \rightarrow 0S1.$
- $S => 0S1 => 00S11 => 000111.$
- Podemos escribir: $S =>^* 000111$

Derivación Iterada

- =>* significa "cero o más pasos de derivación."

- Base: α =>* α para toda cadena α.
- Inducción: si α =>* β y β => γ, entonces α =>* γ.

Ejemplo: Derivación Iterada

- $S \rightarrow 01$
  $S \rightarrow 0S1.$
- S => 0S1 => 00S11 => 000111.
- Luego
  S =>* S
  S =>* 0S1
  S =>* 00S11
  S =>* 000111.

Formas oracionales

- Cualquier cadena de variables o terminales derivada a partir del símbolo inicial se llama una *forma oracional (sentential form)*.

- Formalmente, α es una forma oracional si, y sólo si, $S =>^* \alpha$.

Lenguaje de una Gramática

- Si G es una CFG, entonces L(G), el *lenguaje de G*, es {w: S =>* w}.
  - Nota: w debe ser una cadena terminal, S es el símbolo inicial.
- Ejemplo: G tiene producciones

$$S \rightarrow \epsilon$$
$$S \rightarrow 0S1.$$

Note: ϵ es un lado derecho legítimo.

- $L(G) = \{0^n1^n \mid n \geq 0\}$.

Lenguajes Libres de Contexto

- Un lenguaje que es definido por una CFG se llama un *lenguaje libre de contexto (CFL)*.
- Todo lenguaje regular es un CFL.
- Existen CFLs que no son regulares (ej.).
- No todo lenguaje es CFL.
- Intuitivamente: CFLs pueden contar dos cosas, pero no tres.

Notación BNF

- Las gramáticas para los lenguajes de programación usualmente se escriben en notación BNF (*Backus-Naur Form*).
- Las variables son cadenas en <…>;
  Ejemplo: <statement>.
- Los terminales son usualmente cadenas multicaracteres indicadas por negritas o subrayadas
  Ejemplo: **while** or <u>WHILE</u>.

Notación BNF

- El símbolo ::= se usa usualmente en lugar de $\rightarrow$.
- El símbolo | se usa para indicar "or."
  - Podemos acortar una lista de producciones con el mismo símbolo izquierdo A.
- Ejemplo: $S \rightarrow 0S1 \mid 01$ es una contracción de

$$S \rightarrow 0S1$$
$$S \rightarrow 01$$

Notación BNF

- El símbolo … se usa para “uno o más”.
- Ejemplo:

  <digit> ::= 0|1|2|3|4|5|6|7|8|9

  <unsigned integer> ::= <digit>…

  - Obs! No funciona igual que la * de Kleene en las RE’s.
- Translation: Reemplazar α… con una nueva variable A y producciones $A \rightarrow A\alpha \mid \alpha$.

Ejemplo: Cerradura de Kleene

- Una gramática para para enteros sin signo pueden describirse por:

$$U \rightarrow UD \mid D$$

$$D \rightarrow 0|1|2|3|4|5|6|7|8|9$$

Ejercicio:

- Construir una CFG para las cadenas de paréntesis balanceados.
- Solución:
  - Terminales: {(,)}
  - No-terminales: {S}
  - Símbolo inicial = S
  - Producciones:

$$S \rightarrow (S)$$

$$S \rightarrow ()$$

$$S \rightarrow SS$$

Derivaciones más a la izquierda y más a la derecha

- Las derivaciones nos sirven para reemplazar cualquiera de las variables en una cadena.
- Esto conduce a diferentes maneras de derivar una misma cadena.
- Al forzar que la derivación más a la izquierda (o más a la derecha) se aplica primero, eliminamos estas distinciones.

Derivaciones más a la izquierda (*leftmost*)

- Denotamos

$$wA\alpha \Rightarrow_{lm} w\beta\alpha$$

si w es una cadena sólo de terminales y $A \rightarrow \beta$ es una producción.

- Denotamos

$$\alpha \Rightarrow^*_{lm} \beta$$

si α se transforma en β mediante una secuencia de 0 ó mas pasos $\Rightarrow_{lm}$.

Ejemplo:

- Gramática para paréntesis balanceados:

  $$S \rightarrow SS \mid (S) \mid ()$$

- $S \Rightarrow_{lm} SS \Rightarrow_{lm} (S)S \Rightarrow_{lm} (())S$
  $\Rightarrow_{lm} (())()$
- Luego, $S \Rightarrow^*_{lm} (())()$
- $S \Rightarrow SS \Rightarrow S() \Rightarrow (S)() \Rightarrow (())()$
  es una derivación, pero no es una **más a la izquierda *(leftmost)***.

Derivaciones más a la derecha (*rightmost*)

- Denotamos

$$\alpha A w \Rightarrow_{rm} \alpha\beta w$$

si w es una cadena sólo de terminales y $A \rightarrow \beta$ es una producción.

- Denotamos

$$\alpha \Rightarrow^*_{rm} \beta$$

si α se transforma β mediante una secuencia de 0 ó más pasos $\Rightarrow_{rm}$.

Ejemplo:

- Gramática para paréntesis balanceados:

  $$S \rightarrow SS \mid (S) \mid ()$$

- $S \Rightarrow_{lm} SS \Rightarrow_{lm} (S)S \Rightarrow_{lm} (())S$
  $\Rightarrow_{lm} (())()$
- Luego, $S \Rightarrow^{*}_{lm} (())()$
- $S \Rightarrow SS \Rightarrow S() \Rightarrow (S)() \Rightarrow (())()$
  es una derivación, pero no es una **más a la izquierda *(leftmost)***.

Ejemplo:

- Gramática para paréntesis balanceados:

  $$S \rightarrow SS \mid (S) \mid ()$$

- $S \Rightarrow_{rm} SS \Rightarrow_{rm} S() \Rightarrow_{rm} (S)()$
  $\Rightarrow_{rm} (())()$
- Luego, $S \Rightarrow^*_{rm} (())()$
- $S \Rightarrow SS \Rightarrow S() \Rightarrow (S)() \Rightarrow (())()$ es una derivación **más a la derecha *(rightmost)***.

Ejemplo:

- Gramática para paréntesis balanceados:

  S → SS | (S) | ()

- La derivación
  S => SS => SSS => S()S => ()()S => ()()()
  no es ni *leftmost* ni *rightmost*.