Gramáticas libres de contexto

Conceptos básicos

- El siguientes es un ejemplo de una gramática libre de contexto, a la cual llamaremos G_1.

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

- Una gramática consiste de una colección de ***reglas de sustitución***, también llamadas ***producciones***.
- Cada regla aparece como una línea en la gramática, comprendiendo un símbolo y una palabra separada por una flecha, en donde el símbolo es llamado ***variable***.
- La palabra consta de variables y otros símbolos llamados ***terminales***. Las variables a menudo son representadas con letras mayúsculas. Las terminales son análogas al alfabeto de entrada y a menudo son representadas por letras minúsculas, números o símbolos especiales.
- Una variable es designada como la ***variable de inicio***. Esto usualmente ocurre en el lado izquierdo en lo más alto de la regla.
- Por ejemplo, la gramática G_1 contiene tres reglas, siendo las variables A y B, donde A es la variable de inicio y sus terminales son 0, 1, y #.
- Se usan las gramáticas para describir un lenguaje mediante la generación de cada palabra del lenguaje de la siguiente manera:
  1. Se escribe la variable de entrada.
  2. Se encuentra una variable para sustituirse y una regla que empiece con esa variable. Se remplaza la variable con el lado derecho de esa regla.
  3. Se repite el paso 2 hasta que no queden variables.
- Por ejemplo en la G_1 genera la palabra 000#111. La secuencia de sustituciones para obtener una palabra es llamada ***derivación***. Una derivación de la palabra 000#111 en la gramática G_1 es: $A \rightarrow 0A1 \rightarrow 00A11 \rightarrow 000A111 \rightarrow 000B111 \rightarrow 000\#111$
- Se puede representar la misma información gráficamente mediante un ***árbol parser*** de la siguiente manera:

DIBUJO DE PARSER

- Todas las palabras generadas de esta forma constituyen el ***lenguaje de la gramática***.
- Escribimos $L(G_1)$ para el lenguaje de la gramática G_1. Experimentando con la gramática G_1 se muestra que $L(G_1)$ es $\{0^n\#1^n \mid n \geq 0\}$. Cualquier lenguaje que pueda ser generado por alguna gramática libre de contexto es llamado un ***lenguaje***

libre de contexto (CFL por sus siglas en ingles ***context-free language***).

- Por conveniencia se muestra una gramática libre de contexto, abreviamos varias reglas con la misma variable del lado izquierdo, tal como A → A1 y A → B, en un sola línea A → 0A1 | B, usando el símbolo "|" como un "or".
- El siguiente es un ejemplo de una gramática libre de contexto, llamada G_2, que describe un fragmento del lenguaje Ingles.

[SENTENCE] → [NOUN-PHRASE] [VERB-PHRASE]
[NOUN-PHRASE] → [CMPLX-NOUN] | [CMPLX-NOUN] | [PREP- PHRASE]
[VERB-PHRASE] → [CMPLX-VERB] | [CMPLX-VERB] | [PREP-PHRASE]
[PREP-PHRASE] → [PREP] | [CMPLX-NOUN]
[CMPLX-NOUN] → [ARTICLE] [MOUN]
[CMPLX-VERB] → [VERB] [VERB] [NOUN-PHRASE]
[ARTICLE] → a | the
[NOUN] → boy | girl | flower
[VERB] → touches | likes | sees
[PREP] → with

- La gramática G_2 tiene 10 variables (los términos gramaticales en mayúsculas escritos entre corchetes); 27 terminales (el alfabeto Ingles estándar mas un carácter de espacio); y 18 reglas. Las palabras en L(G_2) incluyen
  - A boy sees
  - The boy sees a flower
  - A girl with aflower likes theboy
- Cada una de estas oraciones tiene una derivación en la gramática G_2. La siguiente es una derivación de la primera oración en la lista.

[SENTENCE] → [NOUN-PHRASE] [VERB-PHRASE]
→ [CMPLX-NOUN] [VERB-PHRASE]
→ [ARTICLE] [MOUN] [VERB-PHRASE]
→ a [MOUN] [VERB-PHRASE]
→ a boy [VERB-PHRASE]
→ a boy [CMPLX-VERB]
→ a boy [VERB]
→ a boy sees

Definición formal de una gramática libre de contexto

- Una gramática libre de contexto es una 4-tupla (V, Σ, R, S), donde:

1. V es un conjunto finito llamado ***variables***,
2. Σ es un conjunto finito disjunto de V, llamado ***terminales***,
3. R es un conjunto finito de ***reglas***, con cada regla siendo una variable y una palabra de variables y terminales, y
4. $S \in V$ es la variable de inicio.

- Si u, v, y w son palabras con variables y terminales, y $A \rightarrow w$ es una regla de la gramática, decimos que uAv produce uwv, escrito como $uAv \rightarrow uwv$. Decimos que u deriva a v, escrito $u \Rightarrow v$, si $u = v$ o si una secuencia $u_1, u_2, \ldots, u_k$ existe para $k \geq 0$ y

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v.$$

- El ***lenguaje de la gramática*** es $\{w \in \Sigma^* \mid S \Rightarrow w\}$.
- En la gramática G_1, $V = \{A, B\}$, $\Sigma = \{0,1,\#\}$, $S = A$, y $R = \{A \rightarrow 0A1, A \rightarrow B, B \rightarrow \#\}$
- En la gramática G_2:
  - V = {[SENTENCE], [NOUN-PHRASE], [VERB-PHRASE], [PREP-PHRASE], [CMPLX-NOUN], [CMPLX-VERB], [ARTICLE], [NOUN], [VERB], [PREP] }, y
  - Σ = {a, b, d,…,z," "}. El símbolo " " es el símbolo blanco, que es invisible después de cada palabra de un lenguaje (a, boy, etc.), por lo que las palabras de un lenguaje no pueden estar juntas.
- A menudo especificamos una gramática escribiendo sólo sus reglas. Podemos identificar las variables como los símbolos que aparecen en el lado izquierdo de las reglas y los terminales como los símbolos restantes. Por convención, la variable de inicio es la variable del lado izquierdo de la primera regla.

Ejemplos de gramáticas liebres de contexto

Ejemplo 3: Considere la gramática $G_3 = (\{S\}, \{a, b\}. R, S)$. El conjunto de reglas, R, es S -> aSb | SS | ε

Esta gramática genera palabras tales como **abab, aaabbb, y aababb**. Se puede ver muy fácilmente que este lenguaje es si se toma a la terminal **a** como el paréntesis izquierdo "(" y a **b** como el paréntesis de la derecha ")". Visto de esta forma, $L(G_3)$ es el lenguaje de todas las palabras o strings con paréntesis anidados apropiadamente.

Ejemplo 4: Considere la gramática $G_4 = (V, \Sigma. R, [EXPR])$.

V es {[EXPR], [TERM], [FACTOR]} y

Σ es {a, +, x, (,)}.

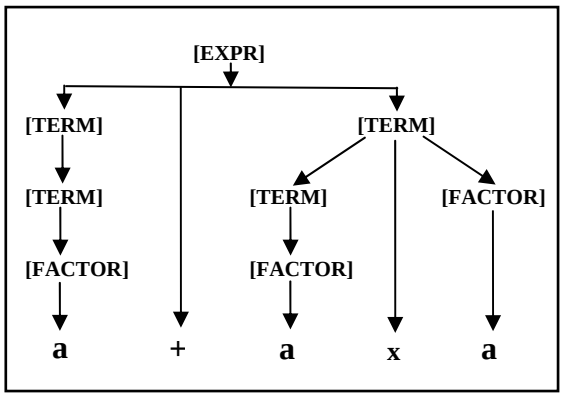
Las reglas son:

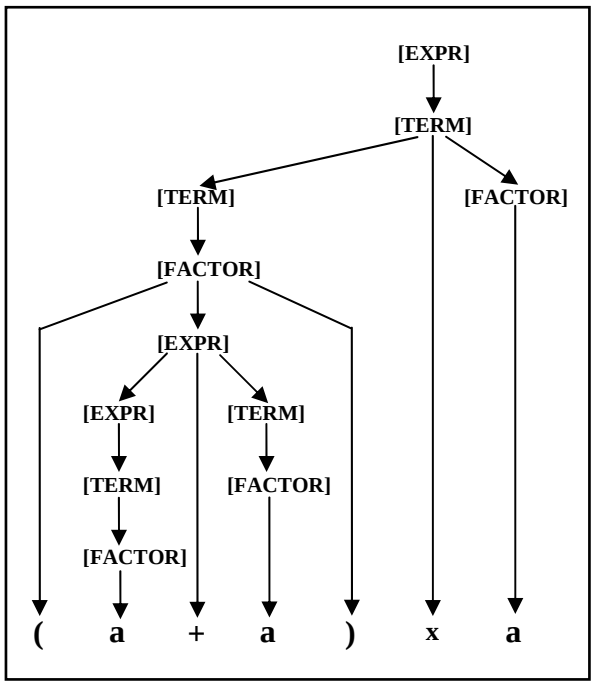
[EXPR] -> [EXPR] + [TERM] | [TERM]

[TERM] -> [TERM] x [FACTOR] | [FACTOR]

[FACTOR] -> ([EXPR]) | a

Las dos palabras **a+axa** y **(a+a)xa** pueden ser generadas con la gramática G_4.
Los árboles parse son los siguientes:





- Un compilador traduce código escrito en un lenguaje de programación en otra forma, usualmente una mas adecuada para su ejecución.
- Para hacer esto el compilador extrae el significado del código a ser compilado en un proceso llamado parsing.
- Una representación de este significado es el arbol parse para dicho código, en la gramática libre de contexto para el lenguaje de programación.
- La gramática G_4 describe un fragmento de un lenguaje de programación que trata con expresiones aritméticas.

- Los árboles parse de la gramática G_4 agrupan operaciones, el árbol para **a+axa** agrupa el operador **x** y sus operandos (las segundas dos **a**'s) como un operando del operador **+**. En el árbol para **(a+a)xa**, la agrupación es al revés.
- Estos agrupamientos muestran la precedencia estándar de la multiplicación antes que la adición y el uso de los paréntesis para hacer caso omiso de la precedencia estándar. La gramática G_4 es diseñada para capturar estas relaciones de precedencia.

Diseño de gramáticas libres de contexto

- Así como el diseño de un autómata finito, el diseño de una gramática libre de contexto requiere creatividad. Ciertamente las gramáticas libres de contexto son mas difíciles de construir que un autómata finito porque estamos mas acostumbrados a programar maquinas para tareas especificas.
- Cuando tenemos el problemas de construir una gramática libre de contexto, debemos tomar en cuenta:

Primero,

- Que muchos lenguajes libres de contexto son la unión de lenguajes libres de contexto más simples. Si se debe construir una gramática libre de contexto para un lenguaje libre de contexto que se puede partir en partes mas simples, realizamos esa división y entonces construimos gramáticas individuales para cada pieza.
- Esas gramáticas individuales pueden ser fácilmente unidas en una gramática para el lenguaje original mediante la combinación de sus reglas y posteriormente adicionamos una nueva regla $S \rightarrow S_1 \mid S_2 \mid \ldots \mid S_k$, donde las variables S_i son las variables de inicio para las gramáticas individuales.
- Resolver varios problemas simples es a menudo más fácil que resolver uno complicado.
- Por ejemplo, para obtener la gramática para el lenguaje

$$\{0^n 1^n \mid n \geq 0\} \cup \{1^n 0^n \mid n \geq 0\},$$

1. primero construimos la gramática: $S_1 \rightarrow 0S_1 1 \mid \varepsilon$ Para el lenguaje $\{0^n 1^n \mid n \geq 0\}$ y
2. la gramática: $S_2 \rightarrow 1S_2 0 \mid \varepsilon$ Para el lenguaje $\{1^n 0^n \mid n \geq 0\}$
3. y entonces sumamos la regla $S \rightarrow S_1 \mid S_2$ para dar la gramática:

$$S \rightarrow S_1 \mid S_2$$
$$S_1 \rightarrow 0S_1 1 \mid \varepsilon$$
$$S_2 \rightarrow 1S_2 0 \mid \varepsilon$$

Segundo,

- Construir una GLC para un lenguaje que sucede que es regular es fácil si primero se construye un AFD para ese lenguaje.
- Se puede convertir cualquier AFD en su equivalente GLC como sigue:

1. Hacer una variable Ri para cada estado qi del AFD.
2. Adiciona la regla $R_i \rightarrow aR_j$ a la GLC si $\delta(q_i, a) = q_j$ es una transición en el AFD.
3. Adiciona la regla $R_i \rightarrow \varepsilon$ si q_i es un estado de aceptación del AFD.
4. Hacer R_0 la variable de inicio de la gramática, donde q_0 es el estado de inicio de la maquina.
5. Verificar que la GLC resultante genere el mismo lenguaje que el AFD reconoce.

Tercero,

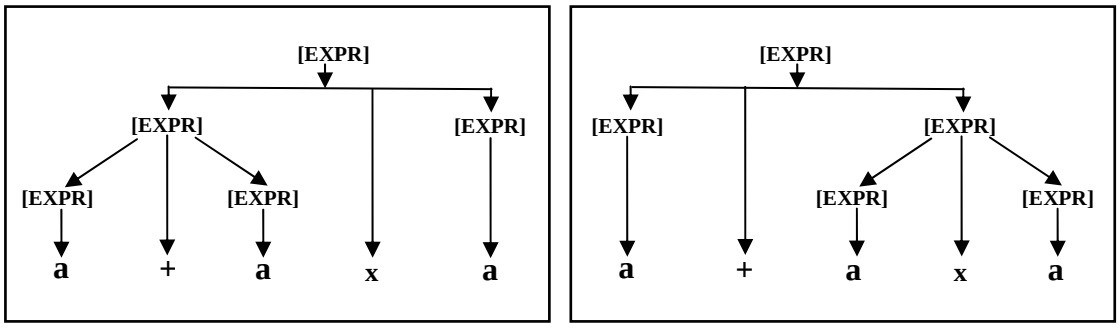
- Ciertos lenguajes libres de contexto contienen strings con dos substrings que son "enlazados" en el sentido de que una maquina para tal lenguaje podría necesitar recordar una ilimitada cantidad de información acerca de uno de los substrings para verificar que corresponda apropiadamente al otro substring.
- Esta situación ocurre en el lenguaje $\{0^n 1^n \mid n \geq 0\}$ porque la maquina podría necesitar recordar el numero de 0's en orden para verificar que es igual al numero de 1's.
- Se puedes construir una GLC para manejar esta situación mediante el uso de la regla de la forma $R \rightarrow uRv$, que genera strings donde la porción que contiene las *u*'s corresponde a la porción que contiene las *v*'s.
- Finalmente, en lenguajes mas complejos, los strings pueden contener ciertas estructuras que aparecen recursivamente como parte de otras (o las mismas) estructuras. Esta situación ocurre en la gramática que genera expresiones aritméticas.

Ambigüedad

- Algunas veces una gramática puede generar el mismo string de diferentes maneras, tal string tendrá diferentes árboles parser y por lo tanto diferentes significados.
- Este resultado puede ser indeseable para ciertas aplicaciones, tal como para los lenguajes de programación, donde dado un programa debe tener una única interpretación.
- Si una gramática genera el mismo string de diferentes maneras, se dice que el string es derivado ambiguamente en la gramática. Si una gramática genera algún string ambiguamente decimos que la gramática es ambigua.
- Por ejemplo, considere la gramática G_5:

  [**EXPR**] → [**EXPR**] + [**EXPR**] | [**EXPR**] x [**EXPR**] | ([**EXPR**]) | **a**

- Esta gramática genera el string a+axa ambiguamente, con dos árboles parser como se muestra:



- Esta gramática no sigue la usual relación de precedencia y puede agrupar el + antes que **x** y viceversa.
- En contraste la gramática G_4 genera exactamente el mismo lenguaje, dado que cada string generado tiene un único árbol parser.
- De aquí que G_4 no es ambigua, mientras que G_5 si lo es.
- La gramática G_2 es otro ejemplo de gramática ambigua, debido a que la sentencia "the girl touches the boy with the flower" tiene dos derivaciones diferentes.
- Formalizando la noción de ambigüedad se puede decir que una gramática genera un string ambiguamente, lo que significa que el string tiene dos árboles parser diferentes (con derivaciones por la izquierda), no dos diferentes derivaciones (puesto que puede haber derivaciones por la derecha también).
- Dos derivaciones pueden diferir meramente en el orden en que son remplazadas las variables y no en toda su estructura.
- Cuando nos referimos a estructura definimos un tipo de derivación que remplaza variables en un orden determinado.
- La derivación de un string *w* en la gramática *G* es una derivación por la izquierda si en cada paso de la derivación se remplaza la variable restante que se encuentra más a la izquierda.
- **Definición:** Un string w es derivado ***ambiguamente*** en una gramática libre de contexto *G*, si *w* tiene dos o más derivaciones por la izquierda que sean diferentes. La gramática *G* es ***ambigua*** si genera algún string ambiguamente.
- Algunas veces cuando tenemos una gramática ambigua se puede encontrar una gramática no ambigua que genere el mismo lenguaje.
- Algunos lenguajes libres de contexto, sin embargo, pueden ser generados solo por gramáticas ambiguas, tales lenguajes son llamados inherentemente ambiguos.

Formal Normal de Chomsky

- Adicionar esta parte para terminar.