Gramáticas Libre del Contexto

Alan Reyes-Figueroa

Teoría de la Computación **(Aula 15) 07.septiembre.2022**

Eliminar ambigüedad

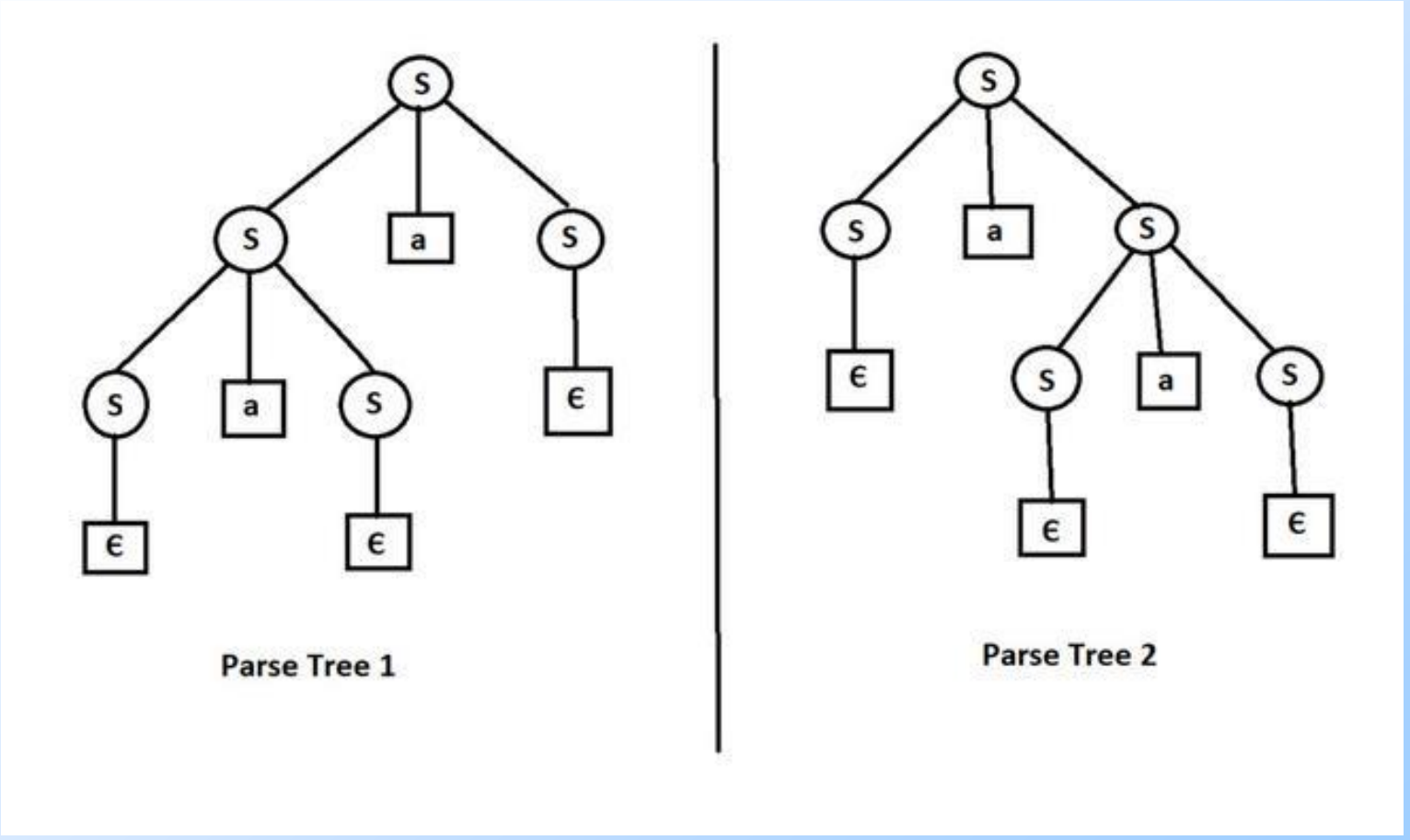
Eliminación de ambigüedad

- Consideremos la siguiente gramática:

  $S \rightarrow aSbS, \quad S \rightarrow bSaS, \quad S \rightarrow \varepsilon$

- La gramática resultante es ambigua. Por ejemplo, para la cadena **abab**, tenemos
- **S -> aSbS ->aSbaSbS ->* abab**
- **S -> aSbS ->abSaSbS ->* abab**

Eliminación de ambigüedad



Eliminación de ambigüedad

- Si se utilizan diferentes operadores, consideraremos la precedencia de los operadores.
  - El nivel al que está presente la producción denota la prioridad del operador.
  - La producción a niveles más altos tendrá operadores con menor prioridad.
  - La producción en los niveles inferiores tendrá operadores con mayor prioridad.

Eliminación de ambigüedad

Asociatividad:

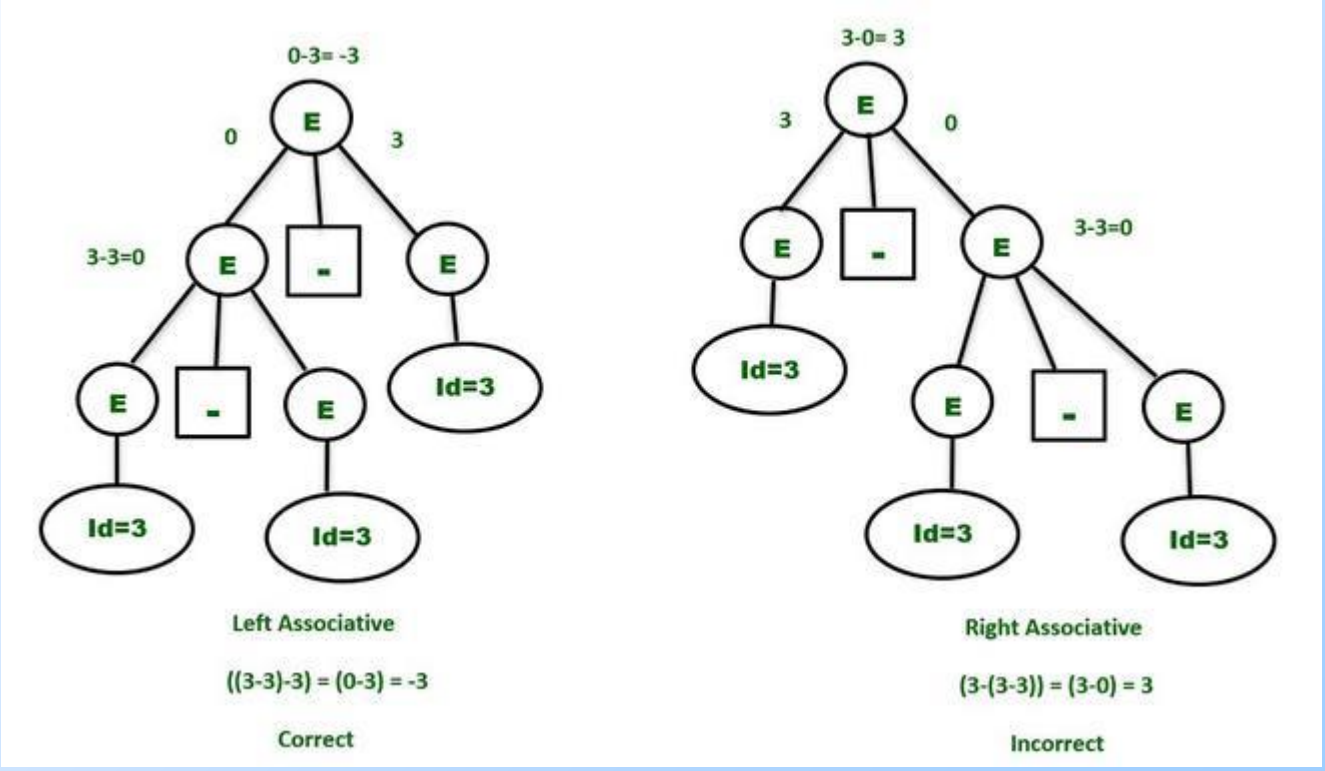
- Si los mismos operadores de precedencia están en producción, entonces tendremos que considerar la asociatividad.
- Si la asociatividad es de izquierda a derecha, entonces tenemos que provocar una recursión a la izquierda en la producción. Si la asociatividad es de derecha a izquierda, entonces tenemos que provocar la recursión a la derecha en las producciones.

Eliminación de ambigüedad

- Para remover la ambigüedad, simplemente hacemos que la gramática sea ***recursiva a la izquierda***.
  Para ello:
- Reemplazamos el símbolo no terminal más a la izquierda en el lado derecho de la producción con otra variable no terminal.

Ejemplo

- E → E - E
  E → 0|1|2|3|4|5|6|7|8|9



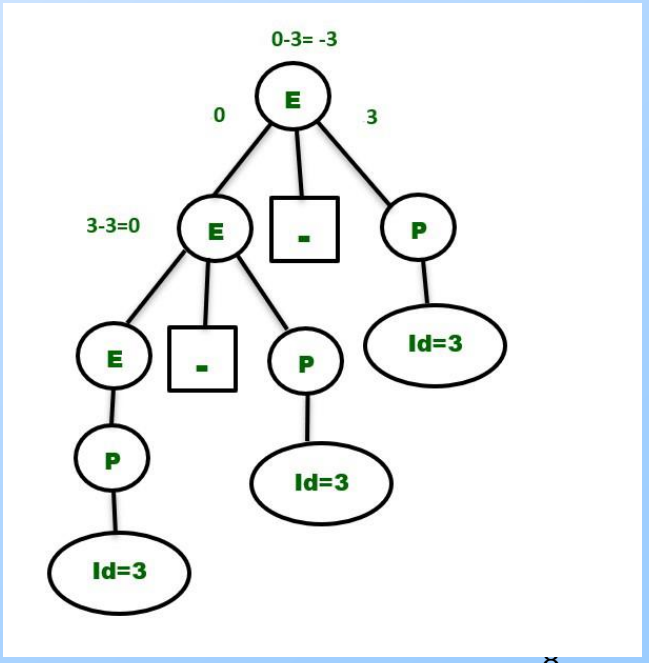
Ejemplo

Removemos la ambigüedad

- $E \rightarrow E - X$

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

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



Ejemplo

- $E \rightarrow E + E$
  $E \rightarrow E * E$
  $E \rightarrow 0|1|2|3|4|5|6|7|8|9$

Removemos la ambigüedad como:

- $E \rightarrow E + F$
  $E \rightarrow F$
  $F \rightarrow F * G$
  $F \rightarrow G$
  $G \rightarrow 0|1|2|3|4|5|6|7|8|9$

Ejemplo

- $S \rightarrow aSbS, \quad S \rightarrow bSaS, \quad S \rightarrow \varepsilon$

Removemos la ambigüedad como:

- $S \rightarrow aSbT$
  $S \rightarrow bSaT$
  $S \rightarrow T$
  $T \rightarrow \varepsilon$

Eliminación de ambigüedad

- Consideremos la siguiente gramática:

  $S \rightarrow aSbS, \quad S \rightarrow bSaS, \quad S \rightarrow \varepsilon$

- La gramática resultante es ambigua. Por ejemplo, para la cadena **abab**, tenemos
- **S -> aSbS ->aSbaSbS ->* abab**
- **S -> aSbS ->abSaSbS ->* abab**

Testando si una variable deriva alguna cadena terminal

- Base: Si existe una producción $A \rightarrow w$, donde w no tiene variables, entonces A deriva una cadena terminal.
- Inducción: Si existe una producción $A \rightarrow \alpha$, donde α consiste sólo de terminales y variables que derivan una cadena terminal, entonces A deriva una cadena terminal.

Testando si una variable deriva alguna cadena terminal

- Eventualmente, llegamos a no encontrar más variables.
- Haciendo una inducción sobre el orden en que las variables “aparecen” muestra que cada una deriva una cadena terminal.
- Recíprocamente, cualquier variable que deriva una cadena terminal siempre se puede encontrar mediante este algoritmo.

Algoritmo para eliminar variables que no derivan nada

1. Descubrir todas las variables que derivan cadenas terminales.
2. Para todas las demás variables, remover todas las producciones en donde dichas variables aparecen (ya sea en la izquierda o en la derecha).

Ejemplo: eliminar variables

$S \rightarrow AB \mid C, \quad B \rightarrow bB,$
$A \rightarrow aA \mid a, \quad C \rightarrow c$

- Base: A y C se marcan, ya que $A \rightarrow a$ y $C \rightarrow c$ (ambas derivan terminales).
- Inducción: S se marca, ya que $S \rightarrow C$ (deriva símbolo que deriva terminales).
- Nada más se marca.
- Resultado: $S \rightarrow AB \mid C, \; A \rightarrow aA \mid a, \; C \rightarrow c$

Símbolos inalcanzables

- Otra forma en que un terminal o variable merece ser eliminada es si no puede aparecer en ninguna derivación desde el símbolo de inicio.
- Base: S es alcanzable (S símbolo inicial).
- Inducción: Si podemos alcanzar A desde S, y existe una producción $A \rightarrow \alpha$, entonces α es alcanzable desde S.

Símbolos inalcanzables

- Es simple mostrar (vía inducción) que cuando no podemos descubrir más símbolos alcanzables, tenemos todos y sólo los símbolos que aparecen en las derivaciones de S.
- Algoritmo: Remover de la gramática todos los símbolos no alcanzables desde S y todas las producciones que involucran a dichos símbolos.

Eliminar símbolos sin uso

- Un símbolo es *útil* (*useful*) si éste aparece en alguna derivación de alguna cadena terminal desde el símbolo inicial S. En otro caso, es *sin uso* (*useless*).
- Eliminamos todos los símbolos sin uso:
  1. Eliminar símbolos que derivan cadenas no terminales.
  2. Eliminar símbolos no alcanzables.

Ejemplo: Símbolos sin uso

$$S \rightarrow AB|C,\ \ A \rightarrow C,\ \ C \rightarrow c,\ \ B \rightarrow bB$$

- **El orden es importante!!**
  Si eliminamos símbolos no alcanzables primero, encontraríamos que todo es alcanzable.
- Luego,
  A, C, y c nunca serían eliminados.

¿Por qué funciona?

- Luego del paso (1), todo símbolo remanente deriva alguna cadena terminal.
- Luego del paso (2), los únicos símbolos remanentes son aquellos derivables de S.
- Adicionalmente, estos símbolos aún derivan una cadena terminal, ya que tal derivación sólo envuelve símbolos alcanzables desde S.

Produccciones Épsilon

- Casi podemos evitar usar producciones del tipo $A \rightarrow \varepsilon$ (llamadas producciones-ϵ).
  - El problema es que ϵ no puede pertenecer al lenguaje generado por una gramática que no posee producciones-ϵ.
- Teorema: Si L es una gramática CFL, entonces L-{ϵ} posee una CFG sin producciones-ϵ.

Símbolos Anulables

- Para eliminar producciones-ϵ, primero debemos detectar las *variables anulables* =
  variables A tales que A =>* ϵ.
- Base: Si hay alguna producción A → ϵ, entonces A es anulable.
- Inducción: Si existe una producción A → α, y todos los símbolos de α son anulables, entonces A es anulable.

Ejemplo: Símbolos Anulables

$$S \rightarrow AB,\ \ A \rightarrow aA \mid \epsilon,\ \ B \rightarrow bB \mid A$$

- Base: A es anulable ya que $A \rightarrow \epsilon$.
- Inducción: B es anulable ya que $B \rightarrow A$.

- En nuestro ejemplo:
  Entonces, S es anulable, ya que $S \rightarrow AB$.

Eliminar Producciones-ϵ

- Idea Clave: Convertir cada producción de la forma $A \rightarrow X_1 \ldots X_n$ en una familia de producciones.
- Para cada subconjunto de anulables X's, existe una producción con aquellos eliminados del lado derecho "in advance."
  - Excepto, si todos los X's son anulables, no crear una producción con ϵ en el lado derecho.

Ejemplo: Eliminar producciones-ϵ

$S \rightarrow ABC$, $B \rightarrow bB \mid \epsilon$,

$A \rightarrow aA \mid \epsilon$, $C \rightarrow \epsilon$

- A, B, C, y S son todos anulables.
- New grammar:

$S \rightarrow \cancel{ABC} \mid AB \mid \cancel{AC} \mid \cancel{BC} \mid A \mid B \mid \cancel{C}$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

Note: C es ahora sin uso.
Eliminar sus producciones.

Induction – Continued

- By the IH, if $w_i \neq \epsilon$, then $X_i =>^*_{new} w_i$.
- Also, the new grammar has a production with A on the left, and just those X_i's on the right such that $w_i \neq \epsilon$.
  - Note: they all can't be ϵ, because $w \neq \epsilon$.
- Follow a use of this production by the derivations $X_i =>^*_{new} w_i$ to show that A derives w in the new grammar.

Proof of Converse

- We also need to show part (2) – if w is derived from A in the new grammar, then it is also derived in the old.
- Induction on number of steps in the derivation.
- We'll leave the proof for reading in the text.

Unit Productions

- A *unit production* is one whose right side consists of exactly one variable.
- These productions can be eliminated.
- Key idea: If A =>* B by a series of unit productions, and B -> α is a non-unit-production, then add production A -> α.
- Then, drop all unit productions.

Unit Productions – (2)

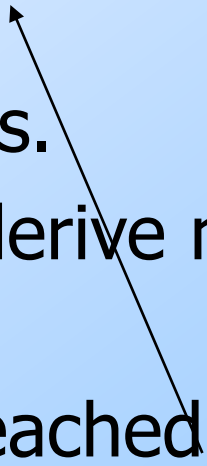
- Find all pairs (A, B) such that A =>* B by a sequence of unit productions only.
- Basis: Surely (A, A).
- Induction: If we have found (A, B), and B -> C is a unit production, then add (A, C).

Cleaning Up a Grammar

- Theorem: if L is a CFL, then there is a CFG for L – {ϵ} that has:
  1. No useless symbols.
  2. No ϵ-productions.
  3. No unit productions.
- I.e., every right side is either a single terminal or has length ≥ 2.

Cleaning Up – (2)

- Proof: Start with a CFG for L.
- Perform the following steps in order:
  1. Eliminate ϵ-productions.
  2. Eliminate unit productions.
  3. Eliminate variables that derive no terminal string.
  4. Eliminate variables not reached from the start symbol.



Must be first. Can create unit productions or useless variables.

Chomsky Normal Form

- A CFG is said to be in *Chomsky Normal Form* if every production is of one of these two forms:
  1. A -> BC (right side is two variables).
  2. A -> a (right side is a single terminal).
- Theorem: If L is a CFL, then L – {ϵ} has a CFG in CNF.

Example: Step 2

- Consider production A -> BcDe.
- We need variables A_c and A_e. with productions A_c -> c and A_e -> e.
  - Note: you create at most one variable for each terminal, and use it everywhere it is needed.
- Replace A -> BcDe by A -> BA_cDA_e.

CNF Proof – Continued

- Step 3: Break right sides longer than 2 into a chain of productions with right sides of two variables.
- Example: A -> BCDE is replaced by A -> BF, F -> CG, and G -> DE.
  - F and G must be used nowhere else.

Example of Step 3 – Continued

- Recall A -> BCDE is replaced by A -> BF, F -> CG, and G -> DE.
- In the new grammar, A => BF => BCG => BCDE.
- More importantly: Once we choose to replace A by BF, we must continue to BCG and BCDE.
  - Because F and G have only one production.