Ciencias de la Computación II

Simplificación de Gramáticas

Eduardo Contrera Schneider

Universidad de la Frontera

13 de octubre de 2016

Introducción

Hasta ahora hemos visto gramáticas en su forma general, por tanto, las producciones involucradas no cuentan con ninguna regla en específica para su creación. Éste puede ser un problema ya que los árboles a generar pueden ser muy tupidos o muy delgados. En cualquier caso pueden haber producciones que resulten inútiles a la hora de generar cadenas.

Motivación

Ejemplo

Consideremos la siguiente gramática:

- $S \rightarrow Aa|B|D$
- $B \rightarrow b$
- $A \rightarrow aA|bA|B$
- $S \rightarrow abd$

Para remover todo este tipo de producciones inútiles, enunciaremos algoritmos que se encargarán al respecto.

Algoritmo 1

Sea $G = (N, \Sigma, S, P)$ una gramática independiente del contexto. Transformaremos G en $G' = (N', \Sigma, S, P')$ de forma que $L(G) = L(G')$ y, para todo $A \in N'$, se obtenga que $A \Rightarrow w$ para algún $w \in \Sigma^*$. Se procede de la siguiente manera:

1. Inicializar N' con todos los no terminales A para los que $A \to w$, es una producción de G, con $w \in \Sigma^*$.
2. Inicializar P' con todas las producciones $A \to w$ para los cuales $A \in N'$ y $w \in \Sigma^*$.
3. Repetir: Añadir a N' todos los no terminales A para los cuales $A \to w$, para algún $w \in (N' \cup \Sigma)^*$ que sea una producción de P y añadirla a P', hasta que no se puedan agregar más no terminales a N'.

Ejemplo

1. Aplique el algoritmo para transformar la gramática anterior.
2. Transforme la gramática:
   - $S \rightarrow aA|\epsilon$
   - $A \rightarrow aA|bB|\epsilon$
   - $B \rightarrow bB$

Se puede ver que en el primer ejemplo de los anteriores, aún quedan producciones no deseadas que derivan en una cada terminal que nunca se llega a formar en el proceso de sustitución. Por tal motivo, el siguiente algoritmo busca eliminar tales producciones a partir de los no terminales y terminales que sólo pueden ser obtenidos a partir de S.

Algoritmo 2

Sea $G = (N, \Sigma, S, P)$ una gramática independiente del contexto. Transformaremos G en $G' = (N', \Sigma, S, P')$ de forma que $L(G) = L(G')$ y, para todo $X \in N' \cap \Sigma'$, se obtenga que $S \Rightarrow \alpha X \beta$ para $\alpha, \beta \in (N' \cap \Sigma')^*$. Se procede de la siguiente manera:

1. Inicializar N' de forma que contenga el símbolo inicial S, e inicializar P' y Σ como $\emptyset$.
2. Repetir: Para $A \in N'$, si $A \to w$ es una producción de P, entonces:
    1. Introducir $A \to w$ en P'.
    2. Para todo no terminal B de w, introducir B en N'.
    3. Para todo terminal σ de w, introducir σ en Σ'.

    hasta que no se puedan agregar nuevas producciones.

Ejemplo

- Aplique el algoritmo 2 al primer ejemplo revisado anteriormente.
- Considere la gramática:
  - $S \rightarrow AB|a$
  - $A \rightarrow a$

  ¿Importa el orden en que apliquemos los algoritmos?

Producciones ϵ

Las producciones del tipo $A \to \epsilon$ se llaman producciones ϵ. A veces son necesarias, pero otras veces no. Sin duda que si $\epsilon \in L(G)$ entonces no tiene sentido eliminarla.

No Terminales Anulables

Se dice que un no terminal A es anulable si $A \Rightarrow \epsilon$.

Si queremos eliminar todas las producciones ϵ entonces se deben identificar todos los no terminales anulables.

Algoritmo 3

1. Inicializar K con todos los no terminales A para los cuales existe una producción ϵ, es decir $A \to \epsilon$.
2. Repetir: Si $B \to w$ para algún $w \in (N \cup \Sigma)^*$ y todos los símbolos de w están en K, agregar B a K.

hasta que no se puedan añadir más no terminales a K. Por último, se procede a crear P' de la siguiente manera: Si $B \to X_1X_2...X_n$ es una producción de P, entonces P' tiene todas las producciones de la forma $B = Y_1Y_2...Y_n$ que satisfagan:

- $Y_i = X_i$ si X_i es no anulable.
- $Y_i = X_i$ o ϵ si X_i es anulable.
- Y_i no es ϵ para todo i.