Hacia las Gramáticas Propias II

Gramáticas sin Ciclos

Universidad de Cantabria

Outline

Definición

Definición

Diremos que una gramática libre de contexto $G := (V, \Sigma, Q_0, P)$ *es acíclica (o libre de ciclos) si no existe ningún símbolo no terminal* $A \in V$ *tal que existe una computación no trivial (en el sistema de transición asociado):*

$$A \longrightarrow B_1 \longrightarrow \cdots \longrightarrow B_k = A.$$

Gramáticas Propias

Definición (Gramáticas Propias)

Diremos que una gramática libre de contexto $G := G := (V, \Sigma, Q_0, P)$ *es propia si verifica las siguientes propiedades:*

- G *es acíclica,*
- G *es* $\lambda-$*libre,*
- G *es libre de símbolos inútiles.*

Observaciones

Notar que son condiciones independientes, aunque los algoritmos que se han visto los realizan en varias etapas que necesitan gramáticas con diferentes requerimientos.

Resultado

Teorema (Interacción entre los algoritmos expuestos)

Se dan las siguientes propiedades:

1. *Si G es una gramática libre de contexto que es $\lambda-$libre y está libre de producciones simples, entonces G es acíclica.*
2. *Sea G una gramática libre de contexto, $\lambda-$libre. Sea $\bar{G}$ la gramática obtenida después de aplicar a G el algoritmo de eliminación de producciones simples. Entonces, $\bar{G}$ sigue siendo $\lambda-$libre.*
3. *Sea G una gramática libre de contexto, libre de producciones simples y $\lambda-$libre. Sea $\bar{G}$ la gramática obtenida después de aplicar a G el algoritmo de eliminación de símbolos inútiles. Entonces, $\bar{G}$ sigue siendo libre de producciones simples y $\lambda-$libre.*

Ideas de la demostración

Supongamos que la gramática fuera $\lambda-$libre y libre de producciones simples, pero hubiera un estado que generara un ciclo. Es decir, supongamos que existe:

$$A \to B_1 \to \cdots \to B_k = A,$$

con $k \geq 1$. Entonces, puedo suponer que

$$\omega_k := \alpha_0 X_1 \alpha_1 \cdots \alpha_{n-1} X_n \alpha_n,$$

donde $\alpha_i \in \Sigma^*$, $X_i \in V$.

Ideas de la demostración

Primero, tenemos que todos los símbolos terminales tienen que ser λ de otra manera, tendrían que aparecen en la parte derecha y no podría haber un ciclo. Por lo tanto:

$$B_k = X_1 \cdots X_n \to A.$$

De aquí tenemos que $n \leq 2$, ya que solo podemos sustituir en cada paso una variable.

Ideas de la demostración

Tenemos dos casos:

- $B_k = X_1$, pero entonces la gramática tiene producciones simples.
- $B_k = AX_2$, y en este caso tenemos que $X_2 \mapsto \lambda$.

Ideas de la demostración

Tenemos dos casos:

- $B_k = X_1$, pero entonces la gramática tiene producciones simples.
- $B_k = AX_2$, y en este caso tenemos que $X_2 \mapsto \lambda$.

Ideas de la demostración

Si $B_k = AX_2$ y la gramática es $\lambda-$libre esto solo pasa en un caso.

Tenemos que tener X_2 es la variable inicial y llegamos a una contradicción.

Ideas de la demostración

Si $B_k = AX_2$ y la gramática es $\lambda-$libre esto solo pasa en un caso.
Tenemos que tener X_2 es la variable inicial y llegamos a una contradicción.

Ideas de la Demostración: II Resultado

La ideas para demostrar que el algoritmo de eliminación de producciones simples mantiene la propiedad de ser $\lambda-$libre es notar dos cosas:

- No se añaden nuevas producciones $A \mapsto \lambda$.
- Si la variable Q_0 no aparece en la parte derecha de ninguna producción entonces no se añaden producciones nuevas $A \mapsto Q_0$.

Ideas de la Demostración: III Resultado

Eliminar símbolos inútiles no añade ninguna nueva producción, es más, quita producciones por lo tanto se mantienen las buenas propiedades de la gramática.

Resultado Final

Teorema

Toda gramática libre de contexto es equivalente a una gramática propia. Dicha equivalencia es calculable algorítmicamente.

Resultado Final

Para hallar esta gramática, hay que realizar los siguientes pasos:

- Transformarla en $\lambda-$libre.
- Eliminar las producciones simples.
- Eliminar los símbolos inútiles.

Resultado Final

Para hallar esta gramática, hay que realizar los siguientes pasos:

- Transformarla en $\lambda-$libre.
- Eliminar las producciones simples.
- Eliminar los símbolos inútiles.

Resultado Final

Para hallar esta gramática, hay que realizar los siguientes pasos:

- Transformarla en $\lambda-$libre.
- Eliminar las producciones simples.
- Eliminar los símbolos inútiles.

Problema de la Palabra

Dada una gramática libre de contexto $G = (V, \Sigma, Q_0, P)$ y dada una palabra $\omega \in \Sigma^*$, decidir si $\omega \in L(G)$.

Resultado

Teorema

Sea $G = (V, \Sigma, Q_0, P)$ *una gramática libre de contexto y* $\lambda-$*libre. Sea* $\omega \in L(G)$ *una palabra aceptada por la gramática y sea:*

$$Q_0 \rightarrow \alpha_1 \rightarrow \alpha_2 \rightarrow \cdots \rightarrow \alpha_n = \omega,$$

una computación aceptadora de ω*, donde* $\alpha_i \in (V \cup \Sigma)^*$ *son formas sentenciales de la gramática. Entonces, la longitud de cada una de estas formas sentenciales verifica:*

$$|\alpha_i| \leq |\omega|, \ \forall i \in \{1, \ldots, n\}.$$

Ideas de la demostración

De hecho, basta con observar que si tenemos dos configuraciones (i.e., dos formas sentenciales) $\alpha_i \rightarrow_G \alpha_{i+1}$ y si la gramática es $\lambda-$libre, entonces o bien se aplica una $\lambda-$producción o bien $|\alpha_i| \leq |\alpha_{i+1}|$. Como la gramática es propia, entonces todas las formas sentenciales son distintas.

Consecuencias

Este resultado nos da una cota de cuantas producciones tenemos que aplicar, tantas como diferentes combinaciones haya, sabemos que si aplicamos el algoritmo un tiempo suficiente, acabara dando la respuesta.

A menos, que el lenguaje sea un lenguaje regular, el proceso no es posible de realizar por un autómata finito.

Consecuencias

Este resultado nos da una cota de cuantas producciones tenemos que aplicar, tantas como diferentes combinaciones haya, sabemos que si aplicamos el algoritmo un tiempo suficiente, acabara dando la respuesta.
A menos, que el lenguaje sea un lenguaje regular, el proceso no es posible de realizar por un autómata finito.

Eficiencia

Decidibilidad no significa eficiencia. Es decir, el hecho de la existencia de un algoritmo para el problema de palabra no significa de modo inmediata que se pueda usar ese algoritmo en la práctica, este algoritmo en la práctica es exponencial en el peor caso.