ST0270-031
Clase 9

J.F. Cardona

Universidad EAFIT

25 de febrero de 2020

Agenda

1. Capítulo 2. Sintaxis
   - Transformaciones de gramáticas y formas normales

Forma Chomsky o normal binaria

- Hay dos tipos de reglas:
  1. *binaria homogénea*: $A \rightarrow BC$, donde $B, C \in V$
  2. *terminal con una única parte derecha*: $A \rightarrow \alpha$, donde $\alpha \in \Sigma$
- Más aun, si cadena vacía está en el lenguaje, allí hay una regla $S \rightarrow \epsilon$ pero el axioma no es permitido en el lado derecho.
- Con tales restricciones cualquier nodo interno de un árbol síntactico pueden tanto dos hermanos no-terminales o un solo terminal hijo.

Forma Chomsky o normal binaria

- Dada una gramática, suponiendo como hipótesis sin no-terminales que deriven a nulo.
- Cada regla $A_0 \to A_1 A_2 \ldots A_n$ de longitud $n > 2$ es convertida a longitud 2 escogiendo escogiendo el primer símbolo A_1 y el sufijo remanente $A_2 \ldots A_n$.
- Entonces un nuevo no-terminal auxiliar es creado, y nombrado $\langle A_2 \ldots A_n \rangle$ y la nueva regla
$$\langle A_2 \ldots A_n \rangle \to A_2 \ldots A_n$$
- Ahora la regla original es reemplazada por
$$A_0 \to \langle A_1 \rangle \langle A_2 \ldots A_n \rangle \quad \langle A_1 \rangle \to A_1$$
donde $\langle A_1 \rangle$ es un nuevo no-terminal auxiliar.
- Continue aplicando la misma serie de transformaciones a la gramática así obtenida, hasta que todas las reglas están en la forma requerida.

Ejemplo. Conversion a la forma normal Chomsky

La gramática

$$S \to dA \mid cB \quad A \to dAA \mid cS \mid c \quad B \to cBB \mid dS \mid d$$

se transforma en

$$S \to \langle d\rangle A \mid \langle c\rangle B \quad A \to \langle d\rangle\langle AA\rangle \mid \langle c\rangle S \mid c \quad B \to \langle c\rangle\langle BB\rangle \mid \langle d\rangle S \mid d$$
$$\langle d\rangle \to d \qquad \langle c\rangle \to c \qquad \langle AA\rangle \to AA$$
$$\langle BB\rangle \to BB$$

Conversión de gramáticas recursivas por la izquierda a recursivas por la derecha

- Otra forma normal denominada no-recursiva por la izquierda, es caracterizada por la ausencia de reglas recursivas por la izquierda o derivaciones (*l-recursions*);
- esta es indispensable para analizadores sintácticos descendentes.

Transformación de *l-recursions* inmediatas

- El caso más común y más fácil es cuando las *l-recursion* a ser eliminada es inmedidata.
- Considere las *l-recursion* alternativas de un no-terminal A:

$$A \rightarrow A\beta_1 \mid A\beta_2 \mid \ldots \mid A\beta_h, h \geq 1$$

donde ningún β_i es vacío, y tenemos

$$A \rightarrow \gamma_1 \mid \gamma_2 \mid \ldots \mid_k, \gamma_k \geq 1$$

sean las alternativas remanentes.

- Cree un nuevo no-terminal auxiliar A' y reemplace las reglas anteriores con las siguientes:

$$A \rightarrow \gamma_1 A' \mid \gamma_2 A' \mid \ldots \mid \gamma_k A' \mid \gamma_1 \mid \gamma_2 \mid \ldots \mid \gamma_k$$
$$A' \rightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_h A' \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_h$$

Transformación de recursiones por la izquierda inmediata

Ahora cada derivación original recursiva por la izquiera, como por ejemplo

$$A \Rightarrow A\beta_2 \Rightarrow A\beta_3\beta_2 \Rightarrow \gamma_1\beta_3\beta_2$$

es reemplazada con la equivalente derivación recursiva por la derecha

$$A \Rightarrow \gamma_1 A' \Rightarrow \gamma_1\beta_3 A' \Rightarrow \gamma_1\beta_3\beta_2$$

Ejemplo

La gramática

$$E \to E + T \mid T \quad T \to T \times F \mid F \quad F \to (E) \mid i$$

Se transforma

$$E \to TE' \mid T \quad E' \to +TE' \mid +T$$
$$T \to FT' \mid F \quad T' \to \times FT' \mid \times F$$
$$F \to (E) \mid i$$

Transformación de recursiones por la izquierda no inmediata

- *Hipótesis:* la gramática G que se transformará es homogénea, no contiene derivaciones nulas, con reglas terminales únicas;
- en otras palabras, las reglas son como las forma normal Chomsky, pero entonces dos no-terminales son permitidos en la parte derecha.
- Sea $V = \{A_1, A_2, \ldots, A_m\}$ sea el alfabeto de los no-terminales y A_1 el axioma.
- Para un examen ordenado, veremos los no-terminales como un conjunto ordenado (arbitrario), de 1 a m.

Algoritmo para la eliminación de la recursividad por la izquierda

for $i = 1$ to m **do**
  for $j = 1$ to $i - 1$ **do**
    reemplace cada regla de tipo $A_i \rightarrow A_j\alpha$, donde $i > j$, con las reglas:
    $A_i \rightarrow \gamma_1\alpha \mid \gamma_2\alpha \mid \ldots \mid \gamma_k\alpha$
    (– posiblemente creando recursiones a la izquierda inmediatas –)
    donde $A_j \rightarrow \gamma_1 \mid \gamma_2 \mid \ldots \mid \gamma_k$ son las alternativas del no-terminal A_j
  end for
  eliminando, por medio del algoritmo anterior, cualquier recursión por la izquierda inmediata que puede surgir como una alternativa de A_i, creando un no-terminal auxiliar A_i'
end for

Ejemplo

Para la gramática G

$$A_1 \to A_2 a \mid b \quad A_2 \to A_2 c \mid A_1 d \mid e$$

i	j		**Grammar**
1		Elimina la recursiones por la izquierda inmediatas de A_1	$A_1 \to A_2 a \mid b$
2	1	Reemplaza $A_2 \to A_1 d$	$A_2 \to A_2 c \mid A_1 d \mid e$ $A_1 \to A_2 A \mid b$ $A_2 \to A_2 c \mid A_2 ad \mid bd \mid e$ $A_1 \to A_2 a \mid b$
		Elimina la recursions por la izquierda inmediatas obteniendo G_3':	$A_2 \to bdA_2' \mid eA_2' \mid bd \mid e$ $A_2' \to cA_2' \mid adA_2' \mid c \mid ad$

Formas normales Greibach y de tiempo-real

- En la forma normal de *tiempo-real* cada regla inicia con un terminal:

$$A \to a\alpha \quad \text{donde } a \in \Sigma, \quad \alpha \in \{\Sigma \cup V\}^*$$

- Un caso especial de la anterior es la forma normal *Greibach*:

$$A \to a\alpha \quad \text{donde } a \in \Sigma, \quad \alpha \in V^*$$

- Cada regla excluyen la cadena vacía del lenguaje.

Formas normales Greibach y de tiempo-real

- Para la forma de tiempo-real:
  - Eliminar primero toda la recursividad por la izquierda;
  - entonces, por transformaciones elementales, expanda cualquier no-terminal que ocurre en la primera posición en un parte derecha, hasta que un prefijo de terminales es producido.
- Entonces continué para la forma Greibach:
  - Si en cualquier otra posición diferente a la primera, un terminal ocurre, reemplacelo por un auxiliar no-terminal y adicione la regla que lo deriva a este.

Ejemplo

La gramática

$$A_1 \to A_2 a \quad A_2 \to A_1 c \mid bA_1 \mid d$$

1. Elimina las recursividades por la izquierda

$$A_1 \to A_2 a \quad A_2 \to bA_1 A_2' \mid dA_2' \mid d \mid bA_1 \quad A_2' \to acA_2' \mid ac$$

2. Expande los no-terminales a la primera posición hasta que un prefijo terminal es producido

$$A_1 \to bA_1 A_2' a \mid dA_2' a \mid da \mid bA_1 a \quad A_2 \to bA_1 A_2' \mid dA_2' \mid d \mid bA_1$$
$$A_2' \to acA_2' \mid ac$$

3. Substituye cualquier terminal en una posición diferente a la primera por no-terminales auxiliares:

$$A_1 \to bA_1 A_2' \langle a \rangle \mid dA_2' \langle a \rangle \mid bA_1 \langle a \rangle \quad A_2 \to bA_1 A_2' \mid dA_2' \mid d \mid bA_1$$
$$A_2' \to a \langle c \rangle A_2' \mid a \langle c \rangle$$
$$\langle a \rangle \to a \qquad \langle c \rangle \to c$$

ST0270-031
Clase 10

J.F. Cardona

Universidad EAFIT

28 de febrero de 2020

Agenda

1. Capítulo 2. Sintaxis
   - Gramáticas de lenguajes regulares
     - De expresiones regulares a gramáticas independientes de contexto
     - Gramáticas de lineales
     - Ecuaciones de gramáticas lineales

De expresiones regulares a gramáticas independientes de contexto

- Dada una expresión regular, es sencillo escribir una gramática para el lenguaje.
- Se analiza la expresión y se mapea sus subexpresiones dentro de reglas gramaticales.
- El núcleo de la construcción, los operadores interactivos (la estrella y la cruz) son reemplazados por reglas recursivas unilaterales.

De expresiones regulares a gramáticas independientes de contexto

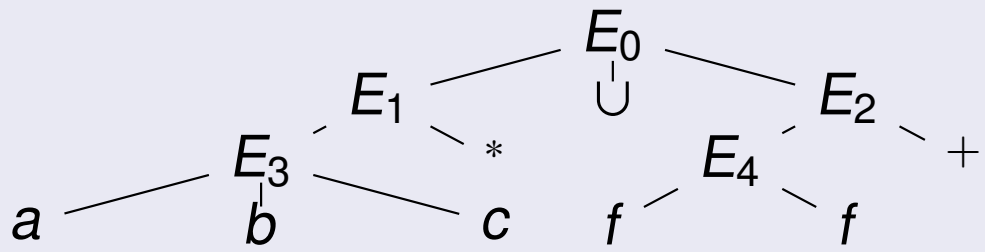
- Primero, identificamos y numeramos las subexpresiones contenidas dentro de la expresión regular dada.
- De cada definición de una expresión regular, los posibles casos y las correspondientes reglas (con no-terminales en mayúsculas) están en la tabla siguiente.

	subexpresión	*regla de la gramática*
1	$r = r_1 \cdot r_2 \ldots r_k$	$E \to E_1 E_2 \ldots E_k$
2	$r = r_1 \cup r_2 \cup \ldots \cup \ldots \cup r_k$	$E \to E_1 \cup E_2 \cup \ldots \cup E_k$
3	$r = (r_1)^*$	$E \to EE_1 \mid \epsilon$ ó $E \to E \to E_1 E \mid \epsilon$
4	$r = (r_1)^+$	$E \to EE_1 \mid E_1$ ó $E \to E_1 E \mid E_1$
5	$r = b \in \Sigma$	$E \to b$
6	$r = \epsilon$	$E \to \epsilon$

Ejemplo. De expresión regular a gramática

La expresión regularw

$$E = (abc)^* \cup (ff)^+$$



Mapeo	Subexpresión	Regla gramática
2	$E_1 \cup E_2$	$E_0 \rightarrow E_1 \mid E_2$
3	E_3^*	$E_1 \rightarrow E_1 E_3 \mid \epsilon$
4	E_4^+	$E_2 \rightarrow E_2 E_4 \mid E_4$
1	abc	$E_3 \rightarrow abc$
1	ff	$E_4 \rightarrow ff$

Propiedad

La familia de los lenguajes regulares *REG* está estrictamente incluida en la familia de los lenguajes independientes de contexto *CF*, de forma que, $REG \subset CF$.

Gramáticas lineales

- Para un lenguaje regular es posible restringir una gramática a una forma muy simple, llamada lineal o de tipo 3.
- Tal forma produce evidencia de algunas propiedades fundamentales y conduce a la construcción sencilla del autómata, el cual reconoce las cadenas de un lenguaje regular.
- Se dice que una gramática es *lineal*, si para cada regla esta tiene la forma:

$$A \rightarrow uBv \quad \text{donde } u, v \in \Sigma^*, B \in (V \cup \epsilon)$$

  donde al menos un no-terminal esta su parte derecha.
- Visualizando su correspondiente árbol sintáctico, se observa que este nunca se subdivide en dos subárboles, pero este una estructura lineal hecha de un tronco con hojas directamente unidas a este.
- Las gramáticas lineales no son tan poderosas para generar los lenguajes regulares.

Lenguaje líneal regular

El lenguaje

$$L_1 = \{b^n e^n \mid n\} = \{be, bbee, \ldots\}$$

La gramática líneal:

$$S \rightarrow bSe \mid be$$

Gramáticas lineales

- Una regla de la siguiente forma es llamada *lineal por la derecha*:

$$A \rightarrow uB \text{ donde } u \in \Sigma^*, B \in (V \cup \epsilon)$$

- Simétricamente, una regla *lineal por la izquierda* tiene la forma

$$A \rightarrow Bu \text{ donde } u \in \Sigma^*, B \in (V \cup \epsilon)$$

- Una gramática tal que todas las reglas son lineales por la derecha o todas lineales por la izquierda es llamada *uni-lineal* o de tipo 3.

Gramáticas de lenguajes regulares

Ejemplo

La expresión regular

$$(a \mid b)^* aa(a \mid b)^* b$$

1. La gramática líneal G_r:

$$S \to aS \mid bS \mid aaA \quad A \to aA \mid bA \mid b$$

2. La gramática líneal G_l:

$$S \to Ab \quad A \to Aa \mid Ab \mid Baa \quad B \to Ba \mid Bb \mid \epsilon$$

Una gramática equivalente no-unilineal para esta expresión regular

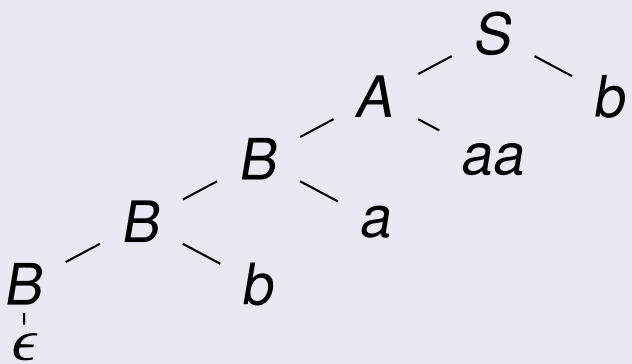
$$E_1 \to E_2 aaE_2 b \quad E_2 \to E_2 a \mid E_2 b \mid \epsilon$$

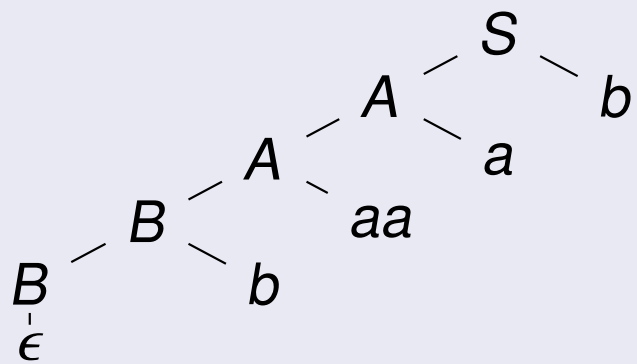
Gramáticas de lenguajes regulares

Ejemplo

Con la gramática G_l los árboles de derivación de la sentencia ambigua *baaab*

$$S \to Ab \quad A \to Aa \mid Ab \mid Baa \quad B \to Ba \mid Bb \mid \epsilon$$





Gramáticas de lenguajes regulares

Ejemplo

El lenguaje

$$L = \{a, a + a, a \times a, a + a \times a, \ldots\}$$

está definido por la gramática lineal derecha G_r:

$$S \rightarrow a \mid a + S \mid a \times S$$

o por la gramática lineal izquierda G_l:

$$S \rightarrow a \mid S + a \mid S \times a$$

Gramáticas de lenguajes regulares

Gramáticas estrictamente uni-lineales

- Las forma de reglas uni-lineales puede ser aun más restringida.
- Una gramática es *estrictamente uni-lineal* si cada regla contiene al menos un carácter terminal, es decir, si tiene la forma

$$A \rightarrow aB \text{ (ó } A \rightarrow Ba), \text{ donde } a \in (\Sigma \cup \epsilon),\ B \in (V \cup \epsilon)$$

- Una simplificación adicional es posible; imponer que las únicas reglas terminales son reglas vacías.
- En este caso se asume que la gramática contiene únicamente los siguientes tipos de reglas:

$$A \rightarrow aB \mid \epsilon \text{ donde } a \in \Sigma, B \in V$$

Ejemplo

La anterior gramática G_r puede ser transformada en una gramática estrictamente lineal por la derecha G'_r:

$$S \to a \mid aA \quad A \to +S \mid \times S$$

o en su equivalente con reglas terminales nulas:

$$S \to aA \quad A \to +S \mid \times S \mid \epsilon$$

Gramáticas de lenguajes regulares

Ecuaciones de lenguajes lineales

- Mostraremos los lenguajes que generan las gramáticas uni-lineales.
- La prueba consiste en la transformación de las reglas a un conjunto de ecuaciones lineales, que tiene lenguajes regulares como sus soluciones.
- Más adelante veremos que cada lenguaje regular puede ser definido como una gramática uni-lineal.
- Por simplicidad tome una gramática $G = (V, \Sigma, P, S)$ en una forma estrictamente lineal por la derecha con reglas terminales nulas.

Ecuaciones de lenguajes lineales

- Cualquiera de esas reglas puede ser transcribidas dentro de una ecuación lineal teniendo los lenguajes desconocidos generados por cada no-terminal, esto es

$$L_A = \{x \in \Sigma^* \mid A \overset{+}{\Rightarrow} x\}$$

- y en particular, $L(G) \equiv L_s$.
- Una cadena $x \in \Sigma^*$ está en el lenguaje L_A si:
  - x es la cadena vacía y P contiene la regla $A \to \epsilon$;
  - x es la cadena vacía, P contiene la regla $A \to B$ y $\epsilon \in L_B$;
  - $x = ay$ inicia con el carácter a, P contiene la regla $A \to aB$ y la cadena $y \in \Sigma^*$ está en el lenguaje L_B.

Gramática de lenguajes regulares

Ecuaciones de lenguajes lineales

- Sea $n = \mid V \mid$ sea un número de no-terminales. Cada no-terminal A_i es definida por un conjunto de alternativas

$$A_i \rightarrow aA_1 \mid bA_2 \mid \ldots \mid \ldots \mid aA_n \mid bA_n \mid \ldots \mid A_1 \mid \ldots \mid A_n \mid \epsilon$$

  algunos posiblemente desaparecidos.
- Escribimos las correspondientes ecuaciones:

$$L_{A_i} = aL_{A_1} \cup bL_{A_2} \cup \ldots \cup aL_{A_n} \cup bL_{A_n} \cup \ldots \cup L_{A_1} \cup \ldots \cup L_{A_n} \cup \epsilon$$

- El último término desaparece si la regla no contiene la alternativa $A_i \rightarrow \epsilon$.
- Este sistema de n ecuaciones simultáneas en n desconocidos (los lenguajes generados por los no-terminales) puede ser resuelta por los métodos bien conocidos de eliminación Gaussiana.

Propiedad. La identidad Arden

- La ecuación

$$X = KX \cup L \tag{1}$$

- donde K es lenguaje no vacío y L cualquier lenguaje, tiene y solamente una solución

$$X = K^*L \tag{2}$$

- Es simple ver que el lenguaje K^*L es la solución de (1), debido a que si sustituimos esta por la variable a ambos lados obtenemos la identidad

$$K^*L = (KK^*L) \cup L$$

- También es posible demostrar que la ecuación (1) no tiene más soluciones que (2).

Gramáticas de lenguajes regulares

Lenguaje de ecuaciones

La gramática lineal por la derecha

$$S \to sS \mid eA \quad A \to sS \mid \epsilon$$

define un lista de elementos e, divididos por el separador s. Este es definido por el sistema

$$\begin{cases} L_S = sL_S \cup eL_A \\ L_A = sL_s \cup \epsilon \end{cases}$$

Substituyendo la segunda ecuación dentro de la primera:

$$\begin{cases} L_S = sL_S \cup e(sL_S \cup \epsilon) \\ L_A = sL_S \cup \epsilon \end{cases}$$

Lenguaje de ecuaciones

Entonces aplicando la propiedad distribuitiva de la concatenación sobre la unión, para factorizar a la variable L_S como un sufijo común:

$$\begin{cases} L_S = (s \cup es)L_S \cup e \\ L_A = sL_S \cup \ \epsilon \end{cases}$$

Aplicando la identidad de Arden a la primera ecuación, se obtiene:

$$\begin{cases} L_S = (s \cup es)^* e \\ L_A = sL_s \cup \epsilon \end{cases}$$

y entonces $L_A = s(s \ \cup \ es)^* e \cup \epsilon$

ST0270-031
Clase 11

J.F. Cardona

Universidad EAFIT

3 de marzo de 2020

Agenda

1. Capítulo 2. Sintaxis
   - Gramáticas de lenguajes regulares
   - Comparación de lenguajes regulares e independientes de contexto
     - Límites de los lenguajes independientes de contexto
     - Propiedades de clausura de REG y CF
   - Gramáticas con expresiones regulares
   - Jerarquía de Chomsky

Comparación de lenguajes regulares e independientes de contexto

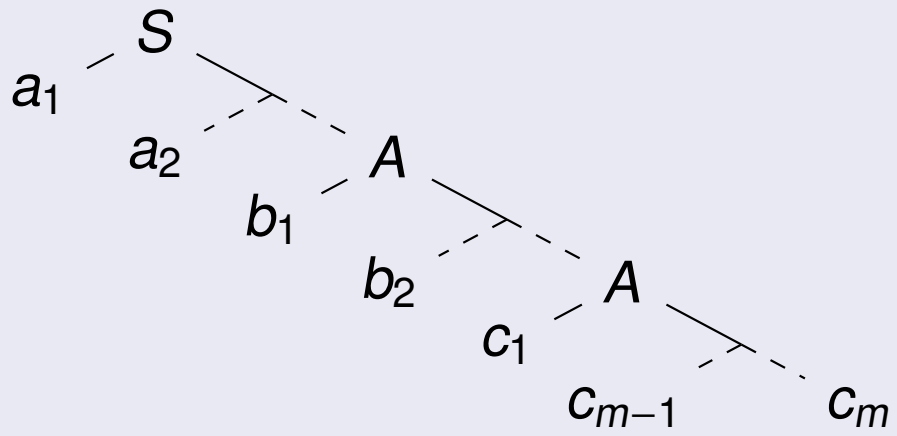
Propiedad. Bombeo de cadenas

- Tome una gramática uni-lineal G.
- Para cualquier sentencia x lo suficientemente larga , con una longitud superior a cualquier constante dependiente de la gramática,
- es posible encontrar una factorización $x = tuv$, donde la cadena u no está vacía,
- de tal forma que, para cada $n \geq 0$, la cadena $tu^n v$ está en el lenguaje.
- Es habitual decir que la sentencia dada puede ser *"bombeada"* al inyectar arbitrariamente muchas veces la subcadena u.

Comparación de lenguajes regulares e independientes de contexto

Prueba de la propiedad de bombeo

- Tome una gramática completamente lineal por la derecha y sea k el número de símbolos no terminales.
- Observe el árbol de sintaxis de cualquier sentencia (frase) x de longitud k ó más;



- claramente dos nodos existen con la misma etiqueta no terminal A.

Comparación de lenguajes regulares e independientes de contexto

Prueba de la propiedad de bombeo

- Considere la factorización de $t = a_1 a_2 \ldots$, $u = b_1 b_2 \ldots$ y $v = c_1 c_2 \ldots c_m$.
- Por lo tanto hay una derivación recursiva:

$$S \overset{+}{\Rightarrow} tA \overset{+}{\Rightarrow} tuA \overset{+}{\Rightarrow} tuv$$

que puede ser repetida para generar las cadenas $tuuv$, $tu \ldots uv$ y tv.

Comparación de lenguajes regulares e independientes de contexto

Propiedad

Cada lenguaje regular es independiente de contexto y allí existen lenguajes independientes de contexto que no son regulares.

$$REG \subseteq CF$$

Comparación de lenguajes regulares e independientes de contexto

Rol de las derivaciones auto-anidadas

- Algunos lenguajes tienen características especiales.
- Tal es el caso de los lenguajes de Dyck o palíndromos y alguno derivados, son claramente no regulares.
- Sus gramáticas tiene algo en común, *todas utilizan derivaciones auto-anidadas*.

$$A \overset{+}{\Rightarrow} uAv \quad u \neq \epsilon \text{ y } v \neq \epsilon$$

- Al contrario, tales derivaciones no pueden ser obtenidas con gramáticas uni-lineales; las cuales permiten únicamente recursiones unilaterales.
- Por ahora, la ausencia de la recursión auto-anidada es la que nos permite resolver las ecuaciones lineales por la identidad de Arden.

Comparación de lenguajes regulares e independientes de contexto

Propiedad

Cualquier gramática independiente de contexto que no produce derivaciones auto-anidadas genera un lenguaje regular.

Comparación de lenguajes regulares e independientes de contexto

Propiedad. El lenguaje con tres potencias iguales

El lenguaje

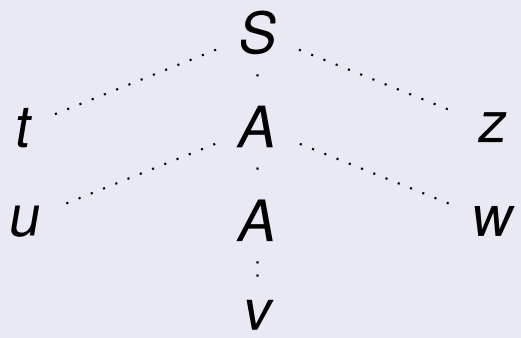
$$L = \{a^n b^n c^n \mid n \geq 1\}$$

no es independiente de contexto.

Comparación de lenguajes regulares e independientes de contexto

Demostración

- Por contradicción, asuma que la gramática G de L existe e imagine el árbol sintáctico de la frase $x = a^n b^n c^n$.
- Enfoquese en los caminos de la raíz (axioma S) a las hojas.
- Al menos un camino debe tener una longitud que se incrementa con la longitud de frase x, y desde que n puede ser arbitrariamente largo, tal camino necesariamente recorre dos nodos con etiquetas no-terminales, digamos A.
- La situación es mostrada en el diagrama.



donde t, u, v, w, z son cadenas terminales.

Comparación de lenguajes regulares e independientes de contexto

Demostración

- Este esquema denota la derivación

$$S \overset{+}{\Rightarrow} tAz \overset{+}{\Rightarrow} t\,uAw\,z \overset{+}{\Rightarrow} t\,u\,v\,w\,z$$

- Esta contiene una subderivación recursiva de A a A, la cual puede ser repetida cualquier número de j veces, produciendo cadenas de tipo

$$y = t\overbrace{u \ldots u}^{j} v \overbrace{w \ldots w}^{j} z$$

- Ahora, examine todos los posibles casos de las cadenas u, w:
  - Ambas cadenas contiene solo uno y el mismo carácter.
  - La cadena u contiene dos o más caracteres, por ejemplo $u = \ldots a \ldots b \ldots$.
  - La cadena u contiene únicamente un carácter, digamos a y la cadena w contiene un carácter diferente, digamos b.

Comparación de lenguajes regulares e independientes de contexto

Lenguaje de copia o replica

- Un extraordinario paradigma es la *replica*, encontrado en mucho contextos técnicos, siempre que dos listas contiene elementos que deben ser idénticos o más generalmente deben coincidir cada uno.
- Un caso concreto es suministrado por la declaración/invocación del procedimiento: la correspondencia entre la lista de parámetros formales y lista actual de parámetros.
- En la forma más abstracta:

$$L_{replica} = \{uu \mid u \in \Sigma^+\}$$

- Para mostrar que el lenguaje de replica no es un lenguaje independiente de contexto, debemos aplicar de nuevo el razonamiento del bombeo.

Comparación de lenguajes regulares e independientes de contexto

Propiedades de clausura de *REG* y *CF*

reflexión	*estrella*	*unión* o *concatenación*	*complemento*	*intersección*
$R^R \in REG$	$R^* \in REG$	$R_1 \oplus R_2 \in REG$	$\neg R \in REG$	$R_1 \cap R_2 \in REG$
$L^R \in CF$	$L^* \in CF$	$L_1 \oplus L_2 \in CF$	$\neg L \notin CF$	$L_1 \cap L_2 \notin CF$ $L \cap R \in CF$

Comparación de lenguajes regulares e independientes de contexto

Definición. Trans-literación o homomorfismo alfabético

Considere dos alfabetos: ***fuente*** Σ y el ***objetivo*** Δ.
Una ***transliteración alfabética*** es una función:

$$h : \Sigma \to \Delta \cup \{\epsilon\}$$

La **transliteración o imagen** del carácter $c \in \Sigma$ es $h(c)$, un elemento del conjunto objetivo. Si $h(c) = \epsilon$, el carácter es borrado.
Una transliteración es ***no-borrable*** si, para ningún carácter fuente c, este es $h(c) = \epsilon$.
La **imagen de la cadena fuente** $a_1 a_2 \ldots a_n$, $a_i \in \Sigma$ es la cadena $h(a_1)h(a_2)\ldots h(a_n)$ obtenida de la concatenación de imágenes de los caracteres individuales.
La **imagen de la concatenación de dos cadenas** v y w es la concatenación de las imágenes de las cadenas:

$$H(v.w) = h(v).h(w)$$

Transliteración

Ejemplo

La siguiente transliteración describe el manejo de ciertos carácteres por impresoras ya obsoletas:

$h(c) = c$ Si $c \in \{a, b, \ldots, z, 0, 1, \ldots, 9\}$;
$h(c) = c$ Si c es una marca de puntuación o un espacio;
$h(c) = \square$ Si $c \in \{\alpha, \beta, \ldots, \omega\}$;
$h(\texttt{stxt}) = \epsilon$
$h(\texttt{etxt}) = \epsilon$

Un ejemplo de transliteración es

$$h(\texttt{stxt}\text{ the const. }\pi\text{ has value 3,14 }\texttt{etxt}) = \text{the const. }\square\text{ has value 3,14}$$

Gramáticas con expresiones regulares

Gramáticas independientes de contexto extendidas

- La legibilidad de las expresiones regulares son buenas para listas y estructuras similares.
- Las gramáticas independientes de contexto son confusas para este tipo de casos.
- La idea es combinar expresiones regulares y reglas gramáticales en la notación:
- *Gramáticas independientes de contexto extendidas* o *EBNF*.
- La parte derecha de una producción es una expresión regular.

Gramáticas con expresiones regulares

Ejemplo

- Considere una lista de declaraciones de variables:

```
char text1, text2;
real temp, result;
int alpha, beta2, gamma;
```

- Esta puede ser construida con una expresion regular, cuyo alfabeto $\Sigma = \{c, i, r, v, ', ', ' ; ' \}$ donde c, i, r son `char`, `int`, `real` y v para el nombre de una variable:

$$((c \mid i \mid r)v(, v)^*;)^+$$

- Esta lista puede ser definida también por la gramática

$$D \to DE \mid E \quad E \to AN; \quad A \to c \mid i \mid r \quad N \to v, N \mid v$$

Gramáticas con expresiones regulares

Definición

Una gramática independiente de contexto extendida o gramática *EBNF*; $G = (V, \Sigma, P, S)$ que contiene exactamente $| V |$ reglas cada una de la forma $A \rightarrow \alpha$, donde A es un non-terminal y α es una expresion regular del alfabeto $V \cup \Sigma$.

Gramáticas con expresiones regulares

Derivación

- La parte derecha α de una regla extendida $A \rightarrow \alpha$ es una expresión regular, la cual genera un conjunto infinito de reglas:
- Cada una vista como la parte derecha de una regla no extendida teniendo muchas alternativas sin limite.
- Por ejemplo: $A \rightarrow (aB)^+$ es el conjunto de reglas:

$$A \rightarrow aB \mid aBaB \mid \ldots$$

Gramáticas con expresiones regulares

Derivación

- La derivación puede ser definida para las gramáticas extendidas también, via la noción de derivaciones para las expresiones regulares.
- Para una gramática *EBNF G*, considere una regla $A \rightarrow \alpha$, donde α es una expresión regular.
- Esta expresión regular contiene operadores de elección (*choice*): estrella, cruz, unión y opción.
- Sea α' una cadena derivada de α, que no contiene ningún operador de elección.
- Para cualquiera (posiblemente vacías) cadenas δ y η existe una derivación de un paso:

$$\delta A \eta \Rightarrow \delta \alpha' \eta$$

Gramáticas con expresiones regulares

Derivación

- Entonces podemos definir derivación de múltiples pasos iniciando desde el axioma y produciendo cadenas terminales;
- y finalmente el lenguaje generado por una gramática *EBNF*, en el misma manera como gramáticas básicas.

Gramáticas con expresiones regulares

Ejemplo

La gramática G

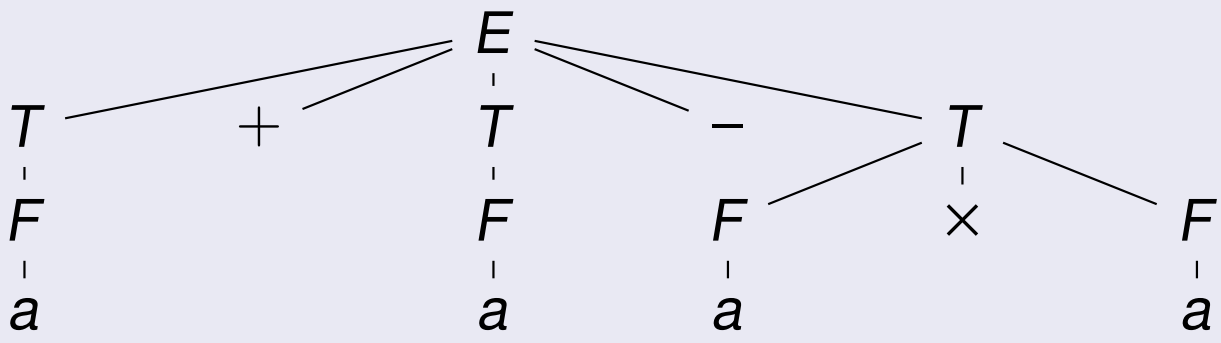
$$E \to [+ \mid -]T((+ \mid -)T)^* \quad T \to F((\times \mid /)F)^* \quad F \to (a \mid' \ ('E')')$$

La derivación izquierda

$$\begin{aligned} E &\Rightarrow T + T - T \Rightarrow F + T - T \Rightarrow a + T - T \Rightarrow a + F - T \\ &\Rightarrow a + a - T \Rightarrow a + a - F \times F \\ &\Rightarrow a + a - a \times F \Rightarrow a + a - a \times a \end{aligned}$$

Gramáticas con expresiones regulares

Ejemplo



Gramáticas con expresiones regulares

Ambigüedad en gramáticas extendidas

- Una gramática ambigua permanece ambigua si es escrita como una gramática extendida.
- En las gramáticas EBNF una forma diferente de ambiguedad puede surgir si la expresión regular es ambigua.
- Por ejemplo la siguiente expresion regular es ambigua:

$$a^*b \mid ab^*$$

- Como consecuencia de lo anterior la gramática extendida:

$$S \rightarrow a^*bS \mid ab^*S \mid c$$

También es ambigua.

Jerarquía de Chomsky

- Es una esquema de clasificación para PSG (*Phrase Structure Grammar*) y sus correspondientes PSL (*Phrase Structure Language*) que ellos generan.
- Los PSG pueden ser clasificados en una jerarquía.
- La clasificación de una gramática acuerdo a la jerarquía de Chomsky es basado únicamente en la presencia de ciertos patrones en las producciones.
- Están nombrados con números desde el 0 al 3.
- Donde 0 es la gramática más general.

Gramática tipo 0

Nombre

Sin restricciones

Forma de las reglas

$$\beta \rightarrow \alpha$$

donde $\alpha, \beta \in (\Sigma \cup V)^+$

Familia del lenguaje

Recursivamente enumerable

Tipo de reconocedor

Máquinas de Turing

Gramática tipo 1

Nombre

Sensitiva al contexto

Forma de las reglas

$$\beta \rightarrow \alpha$$

donde $\alpha, \beta \in (\Sigma \cup V)^+$ y $|\beta| \leq |\alpha|$.

Familia del lenguaje

Contextual o dependiente del contexto

Tipo de reconocedor

Máquinas de Turing con complejidad del espacio limitada por la longitud de la cadena de entrada.

Gramática tipo 2

Nombre

Independiente al contexto o BNF

Forma de las reglas

$$A \to \alpha$$

donde A es un non-terminal ($A \in V$) y $\alpha \in (\Sigma \cup V)^*$.

Familia del lenguaje

Independiente del contexto (*CF*) o algebraica.

Tipo de reconocedor

Autómata de pila.

Gramática tipo 3

Nombre

Regular o unilineal (lineal por la derecha o por la izquierda).

Forma de las reglas

Lineal por la derecha

$$A \rightarrow uB$$

Lineal por la izquierda

$$A \rightarrow Bu$$

donde A es un non-terminal ($A \in V$) y $u \in \Sigma^*$ y $B \in (V \cup \epsilon)$

Familia del lenguaje

Regular *REG* o racional o de estado finito.

Tipo de reconocedor

Autómata finito.

Jerarquía de Chomsky

Ejemplo de una gramática tipo 1 de lenguaje de potencia igual de tres elementos

$$L = \{a^n b^n c^n \mid n \geq 1\}$$

Este es generado por la gramática sensitiva al contexto:

$$\begin{array}{lll} 1.S \rightarrow aSBC & 3.CB \rightarrow BC & 5.bC \rightarrow bc \\ 2.S \rightarrow abC & 4.bB \rightarrow bb & 6.cC \rightarrow cc \end{array}$$

Jerarquía de Chomsky

Ejemplo. Gramática tipo 1 de réplica con centro

El lenguaje $L = \{ycy \mid y \in \{a, b\}^+\}$ contiene frases como: *aabcaab*.
La gramática:

$$
\begin{array}{lllll}
S \to X\dashv & XA \to XA' & A'A \to AA' & A' \dashv\!\to a & B'a \to ba \\
X \to aXA & XB \to XB' & A'B \to BA' & B' \dashv\!\to b & B'b \to bb \\
X \to bXB & & B'A \to AB' & A'a \to aa & Xa \to ca \\
 & & B'B \to BB' & A'b \to ab & Xb \to cb
\end{array}
$$

genera frase de dicho lenguaje L.