REPASO TEORÍA FORMAL DE LENGUAJES, GRAMÁTICAS Y AUTÓMATAS

Lenguajes y Gramáticas

- Definiciones
  - Alfabeto Σ
    - Conjunto no vacío finito de letras o símbolos
  - Palabra w
    - Secuencia finita de letras del alfabeto
  - Palabra vacía λ, ε
    - Palabra de longitud 0
  - Universo $W(\Sigma)$
    - Todas las palabras que pueden formarse con letras de Σ
  - Lenguaje $L(\Sigma)$
    - Todo subconjunto de $W(\Sigma)$

Lenguajes y Gramáticas

- Operaciones con palabras
  - Concatenación
    - $x \bullet y$
  - Potencia
    - $x^i = \underbrace{x \bullet x \bullet \ldots \bullet x}_{i}$
  - Reflexión
    - $x = A_1 A_2 \ldots A_n \rightarrow x^{-1} = A_n \ldots A_2 A_1$

Lenguajes y Gramáticas

- Operaciones con lenguajes
  - Unión
    - $L_1 \cup L_2 = \{x + y \mid x \in L_1 \vee y \in L_2\}$
  - Concatenación
    - $L_1 \bullet L_2 = \{x \bullet y \mid x \in L_1 \wedge y \in L_2\}$
  - Potencia
    - $L^i = \underbrace{L \bullet L \bullet \ldots \bullet L}_{i}$
  - Clausura $L^* = \bigcup_{i=0}^{\infty} L^i ; \Sigma^* = W(\Sigma)$
    -
  - Reflexión
    - $L^{-1} = \{x^{-1} \mid x \in L\}$

Gramáticas Formales

- Definición: $G=\{\Sigma_T, \Sigma_N, S, P\}$
  - $\boldsymbol{\Sigma_T}$ es el alfabeto de símbolos terminales
  - $\boldsymbol{\Sigma_N}$ es el alfabeto de símbolos no terminales
    - $\Sigma = \Sigma_T \cup \Sigma_N$
    - $\Sigma_T \cap \Sigma_N = \varnothing$
  - $\boldsymbol{S}$ es el axioma ($S \in \Sigma_N$)
  - $\boldsymbol{P}$ es el conjunto finito de reglas de producción
    - $P=\{u::=v \mid u\in\Sigma^+,\ v\in\Sigma^*,\ u=xAy,\ x,y\in\Sigma^*,\ A\in\Sigma_N\}$

Gramáticas Formales

- Definiciones
  - Forma sentencial
    - x es forma sentencial si $S \rightarrow_* x$
  - Sentencia
    - x es sentencia si es forma sentencial y $x\in\Sigma_T^*$
  - Lenguaje asociado a una gramática
    - $L(G)=\{x \mid S\rightarrow_* x,\ x\in\Sigma_T^*\}$
  - Recursividad
    - G es recursiva si $\exists\ A\in\Sigma_N \mid (A\rightarrow xAy) \in P$
      - Recursiva a izquierdas: $x=\lambda$
      - Recursiva a derechas: $y=\lambda$
  - Regla compresora
    - $x \rightarrow \lambda$

Tipos de Gramáticas

- Tipo 0 (sin restricciones)
  - $P=\{u::=v|u\in\Sigma^{+}\wedge v\in\Sigma^{*}\wedge u=xAy\wedge x,y\in\Sigma^{*}\wedge A\in\Sigma_{N}\}$
- Tipo 1 (de contexto libre)
  - $P=\{(S::=\lambda)\cup(xAy=xvy)|x,y\in\Sigma^{*}\wedge A\in\Sigma_{N}\wedge v\in\Sigma^{+}\}$
- Tipo 2 (independientes del contexto)
  - $P=\{(S::=\lambda)\cup(A=v)|\,A\in\Sigma_{N}\wedge v\in\Sigma^{+}\}$
- Tipo 3 (regulares o lineales)
  - Lineales por la izquierda
    - $P=\{(S::=\lambda)\cup(A=Ba)\cup(A=a)|\,A,B\in\Sigma_{N}\wedge a\in\Sigma_{T}\}$
  - Lineales por la derecha
    - $P=\{(S::=\lambda)\cup(A=aB)\cup(A=a)|\,A,B\in\Sigma_{N}\wedge a\in\Sigma_{T}\}$

Ejemplo de gramática tipo 2

- Notación BNF

```
Sentencias::= Sentencia ; Sentencias | Sentencia
Sentencia::= Asignación | Condición | Iteración
Asignación::= Variable := Expresión
Condición::= if Lógica then Sentencias else Sentencias |
  if Lógica then Sentencias
Iteración::= while Lógica do Sentencias
Expresión::= Var_NUM Operación Expresión | Var_NUM
Var_NUM::= Variable | Número
```

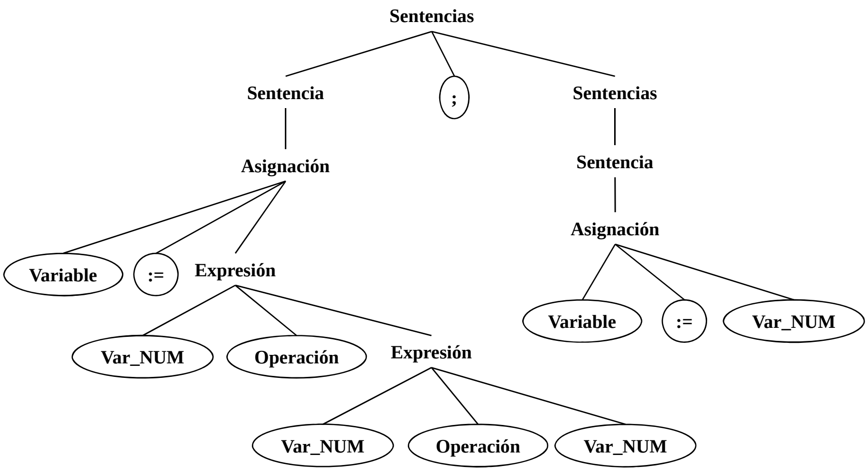
Derivación

- Derivación
  - Aplicación de una producción a una forma sentencial
    - **Sentencias**→**Sentencia** ; **Sentencias** → **Asignación** ; **Sentencias** → …
  - Derivación más a la izquierda
    - Se sustituye primero el símbolo no terminal más a la izquierda
  - Derivación más a la derecha
    - Se sustituye primero el símbolo no terminal más a la derecha

Ambigüedad

- Se define a distintos niveles
  - Sentencia:
    - Si puede obtenerse por dos derivaciones diferentes
  - Gramática:
    - Si puede obtener una sentencia con dos derivaciones más a la izquierda
  - Lenguaje:
    - Si existe una gramática que lo genera
    - Si todas las gramáticas que lo generan son ambiguas entonces es inherentemente ambiguo

Árbol de Derivación



Recursividad a Izquierdas I

- Algoritmo de Eliminación RI

  $\forall P \subseteq \wp \mid P=(A \to A \cdot \alpha \mid \beta)$

  1. $\Sigma_N = \Sigma_N \cup \{A'\}$
  2. $\wp = (\wp - P)$
  3. $\wp\,' = \wp \cup \{A \to \beta \cdot A', A' \to \alpha \cdot A' \mid \lambda\}$

- Ejemplo:
  - $\wp : E \to E * T \mid T$
  - Solución: $\wp\,' : E \to T E', E' \to * T E' \mid \lambda$

Recursividad a Izquierdas II

- Algoritmo de Eliminación RI (General)

$\forall P \subseteq \wp \mid P=(A \rightarrow A \cdot \alpha_1 \mid A \cdot \alpha_2 \mid \ldots \mid A \cdot \alpha_n \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m \mid)$

1. $\Sigma_N = \Sigma_N \cup \{A'\}$
2. $\wp = (\wp - P)$
3. $\wp' = \wp \cup \{A \rightarrow \beta_1 \cdot A' \mid \beta_2 \cdot A' \mid \ldots \mid \beta_m \cdot A' \mid ,$ $A' \rightarrow \alpha_1 \cdot A' \mid \alpha_2 \cdot A' \mid \alpha_n \cdot A' \mid \lambda\}$

Recursividad a Izquierdas II

- Ejemplo:
  - $\wp : E \rightarrow E * T \mid E + T \mid T$
- Solución:
  - $\wp' :$ $E \rightarrow T E'$

    $E' \rightarrow * T E' \mid + T E' \mid \lambda$

Recursividad a Izquierdas III

15

- Recursividad en varios pasos:
  - $\wp$: $A \rightarrow B\,a$
    $B \rightarrow A\,b \mid \lambda$
- Algoritmo

```
1. Ordenar (cualquiera) los símbolos no terminales:
```
$A_1, A_2,.., A_n$

2. Para i=1 hasta n
    3. Para j=1 hasta n
        3.1 Reemplazar cada producción: $A_i \rightarrow A_j \cdot \gamma$
        por: $A_i \rightarrow \delta_1 \cdot \gamma \mid \delta_2 \cdot \gamma \mid \ldots \mid \delta_k \cdot \gamma$
        donde: $A_j \rightarrow \delta_1 \mid \delta_2 \mid \ldots \mid \delta_k$ son todas las producciones de A_j
        3.2 Eliminar la RI de A_i

Factorización a Izquierdas

16

- Para las producciones:
  - $\wp$: $A \rightarrow \beta \cdot \alpha_1 \mid \beta \cdot \alpha_2$
- Algoritmo

1. $\Sigma_N = \Sigma_N \cup \{A'\}$
2. $\wp = (\wp - P)$
3. $\wp' = \wp \cup \{A \rightarrow \beta \cdot A',$
   $A' \rightarrow \alpha_1 \mid \alpha_2\}$

Factorización a Izquierdas

- Ejemplo:
  - $\wp$:
    $S \rightarrow$ ***if*** C ***then*** S ***else*** S
    $S \rightarrow$ ***if*** C ***then*** S
    $S \rightarrow$ ***repeat*** S ***until*** C
    $S \rightarrow$ ***repeat*** S ***forever***
- Solución:
  - $\wp$:
    $S \rightarrow$ ***if*** C ***then*** $S\,A'$
    $S \rightarrow$ ***repeat*** $S\,A''$
    $A' \rightarrow$ ***else*** $S \mid \lambda$
    $A'' \rightarrow$ ***until*** $C \mid$ ***forever***

Conceptos de Gramáticas, I

- Equivalencia de gramáticas:
  - G_1 y G_2 son equivalentes si $L(G_1)=L(G_2)$
- Forma Normal de Chomsky
  - $\wp = \{ (A \rightarrow BC) \cup (S \rightarrow \lambda) \cup (A \rightarrow a) \mid A,B,C \in \Sigma_N,\ a \in \Sigma_T$
- Forma Normal de Greibach
  - $\wp = \{ (A \rightarrow aX) \cup (S \rightarrow \lambda) \cup (A \rightarrow a) \mid A \in \Sigma_N,\ X \in {\Sigma^*}_N,\ a \in \Sigma_T$

Conceptos de Gramáticas, II

- Gramáticas bien formadas
  - Limpias:
    - Sin reglas innecesarias:
      - $A \to A$
    - Sin símbolos inaccesibles:
      - $A \in \Sigma_N \mid \sim\exists(B \to xAy) \in \wp$
    - Sin símbolos superfluos:
      - $\sim\exists\, A \to_* x \mid x \in \Sigma^*_T$
  - Sin reglas no generativas:
    - $A \to \lambda \wedge A \neq S$
  - Sin reglas de redenominación:
    - $A \to B \mid A, B \in \Sigma_N$

Conceptos de Gramáticas, III

- Algoritmos de limpieza de GIC
  - Eliminación de Reglas innecesarias:
    - Sea una GIC y $\wp$ contiene producciones $\wp_i$ de la forma $A \to A$, entonces $\wp' = \wp - \wp_i$
  - Eliminación de Símbolos inaccesibles y superfluos:
    - Sea la GIC $G = (\Sigma_T, \Sigma_N, S, \wp)$ y la G transformada $G' = (\Sigma'_T, \Sigma'_N, S, \wp')$

    1. $\Sigma'_N = \{S\}$, $\Sigma'_T = \varnothing$, $\wp' = \varnothing$
    2. Repetir

       $\forall A \in \Sigma'_N$ y $\forall A \to w \in \wp$

       2.1. $\wp' = \wp' \cup \{A \to w\}$

       2.2. $\forall B \in w$, $\Sigma'_N = \Sigma'_N \cup \{B\}$

       2.3. $\forall a \in w$, $\Sigma'_T = \Sigma'_T \cup \{a\}$

       Hasta que no se puedan añadir nuevas reglas a $\wp'$

Conceptos de Gramáticas, IV

- Algoritmos de limpieza de GIC
  - Eliminación de Reglas no generativas:
    - Sea la GIC $G = (\Sigma_T, \Sigma_N, S, \wp)$ y la transformada $G' = (\Sigma_T, \Sigma_N, S, \wp')$

1. Obtención de símbolos anulables Σ_A
   1.1. $\Sigma_A = \{A \mid A \in \Sigma_N$ y $A \to \lambda \in \wp\}$
   1.2. Repetir $\forall B \mid B \to w \in \wp, w \in \Sigma^*_N$
   Si $w \in \Sigma^*_A \Rightarrow \Sigma_A = \Sigma_A \cup \{w\}$
   Hasta que no se añadan más símbolos no terminales a Σ_A
2. Creación de G'
   2.1. $\wp' = \varnothing$
   2.2. $\forall\, B \to x_1x_2\ldots x_n \in \wp, x_1x_2\ldots x_n \in \Sigma^*$ hacer todas las $B \to y_1y_2\ldots y_n$ tal que:
   Si $x_i \notin \Sigma_A \Rightarrow y_i = x_i$
   Si $x_i \in \Sigma_A \Rightarrow y_i = x_i$ o λ
   2.3. Aplicar la propiedad de concatenación para eliminar λ y eliminar las reglas $B \to \lambda$ las reglas resultantes se incluyen en $\wp'$
   2.4. Si $S \in \Sigma_A \Rightarrow \wp' = \wp' \cup \{S \to \lambda\}$

Conceptos de Gramáticas, V

- Algoritmos de limpieza de GIC
  - Eliminación de Reglas no generativas: Ejemplo

Si $B \to x_1x_2x_3$ y $x_2, x_3 \in \Sigma_A$

Entonces hacer las producciones:

$B \to x_1x_2x_3 \mid x_1x_2\lambda \mid x_1\lambda\, x_3 \mid x_1\, \lambda\, \lambda$

Entonces $\wp' = \wp' \cup \{B \to x_1x_2x_3 \mid x_1x_2 \mid x_1x_3 \mid x_1\}$

Conceptos de Gramáticas, VI

- Algoritmos de limpieza de GIC
  - Eliminación de Reglas de Redenominación (no funciona si hay reglas $A \to B \to \ldots \to A$):
    - Sea la GIC $G = (\Sigma_T, \Sigma_N, S, \wp)$ y la transformada $G' = (\Sigma_T, \Sigma_N, S, \wp')$

Repetir:

Para cada regla unitaria $A \to B$

Sean $B \to w_1 | w_1 | \ldots | w_n$

Entonces $\wp' = \wp - \{A \to B\} \cup \{A \to w_1 | w_1 | \ldots | w_n\}$

Hasta que no haya más reglas unitarias

Conceptos de Gramáticas, VII

- Orden de aplicación de los Algoritmos de limpieza y bien formar GIC
  1. Eliminación de Reglas innecesarias
  2. Eliminación de Reglas no generativas
  3. Eliminar Reglas de Redenominación (Menos la regla del axioma $S \to \lambda$)
  4. Eliminación de S. Superfluos
  5. Eliminación de S. Inaccesibles

Autómata Finido Determinista

25

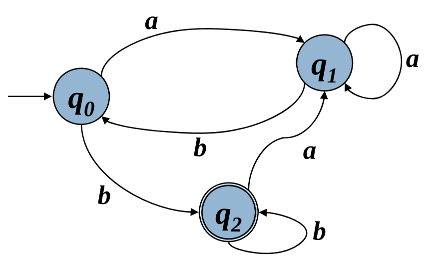
- Definición AFD:
  - $AFD=(\Sigma, Q, f, q_0, F)$
- Donde:
  - Σ es el alfabeto de símbolos de entrada
  - Q es el conjunto de estados
  - $q_0 \in Q$ es el estado inicial
  - $F \subseteq Q$ es el conjunto de estados finales
  - f es la función de transición entre estados $f : Q \times \Sigma \rightarrow Q$

Representación de un AFD

- Tabla de transiciones

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_0
$*q_2$	q_1	q_2

- Diagrama de transiciones



Conceptos de AFD, I

- Extensión a palabras:
  - $f' : Q \times \Sigma^* \rightarrow Q$
    - $f'(q,a\cdot x) = f'(f(q, a), x)$
    - $f'(q,\lambda) = q$
- Lenguaje asociado a un AFD:
  - $L_{AFD}=\{x \mid x\in\Sigma^* \wedge f'(q_0,x) \in F\}$
- Equivalencia de AFD:
  - $\mathrm{AFD}_1 \equiv \mathrm{AFD}_2 \Leftrightarrow L(\mathrm{AFD}_1)= L(\mathrm{AFD}_2)$

Conceptos de AFD, II

- Minimización de AFD:
  - $\forall\ \mathrm{AFD}\ \exists\ \mathrm{AFD}_m \mid \mathrm{AFD} \equiv \mathrm{AFD}_m$
- AFD asociado a una G3:
  - $\forall\ \mathrm{G}_3\ \exists\ \mathrm{AF} \mid L(\mathrm{G}_3) = L(\mathrm{AF})$
  - Pero no siempre será AFD
  - Aunque siempre puede transformarse un AF en otro AFD equivalente

Autómata Finido No Determinista

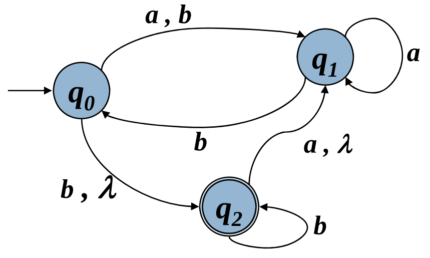
- Definición AFND:
  - ***AFND*=(*Σ*, *Q*, *f*, q_0, *F*)**
- Donde:
  - ***Σ*** es el alfabeto de símbolos de entrada
  - ***Q*** es el conjunto de estados
  - $q_0 \in Q$ es el estado inicial
  - $F \subseteq Q$ es el conjunto de estados finales
  - ***f*** es la función de transición entre estados $f : Q \times (\Sigma \cup \{\lambda\}) \rightarrow P(Q)$

Representación de un AFND

- Tabla de transiciones

	a	b	λ
$\rightarrow q_0$	q_1	$\{q_2, q_1\}$	q_2
q_1	q_1	q_0	
$*q_2$	q_1	q_2	q_1

- Diagrama de transiciones



Conceptos de AFND, I

- Extensión a palabras: $f'': Q \times \Sigma^* \rightarrow P(Q)$
  - $f''(q,x) = \left\{ p \mid x = a_1 \cdot a_2 \cdot \ldots \cdot a_n \wedge q \overset{\lambda^* \cdot a_1 \cdot \lambda^* \cdot a_2 \cdot \ldots \cdot a_n \cdot \lambda^*}{\longrightarrow} p \right\}$
  - $f''(q,\lambda) = \left\{ p \mid q \overset{\lambda^*}{\rightarrow} q \right\}$
- Lenguaje asociado a un AFND:
  - $L_{AFND} = \{x \mid x \in \Sigma^* \wedge f''(q_0,x) \cap F \neq \varnothing\}$
- Equivalencia de AFD y AFND:
  - $\forall \text{AFD} \, \exists \, \text{AFND} \mid \text{AFD} \equiv \text{AFND}$

Conceptos de AFND, II

- AFND asociado a una G3:
  - $\forall \, G_3 \, \exists \, \text{AFD y AFND} \mid \text{AFD} \Leftrightarrow \text{AFND} \Leftrightarrow G_3$

Autómata a Pila

- Definición AP:
  - $AFD=(\Sigma, \Gamma, Q, A_0, q_0, f, F)$
- Donde:
  - Σ es el alfabeto de símbolos de entrada
  - Γ es el alfabeto de símbolos de pila
  - Q es el conjunto de estados
  - $A_0 \in \Gamma$ es el símbolo de pila inicial
  - $q_0 \in Q$ es el estado inicial
  - $F \subseteq Q$ es el conjunto de estados finales
  - f es la función de transición entre estados $f : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow P(Q \times \Gamma^*)$

Conceptos asociados a AP, I

- Autómata a Pila Determinista (APD)
  - $\forall q \in Q, A \in \Gamma$, si $| f(q,\lambda,A)| > 0 \Rightarrow f(q,\lambda,A) = \varnothing \forall a \in \Sigma$
  - $\forall q \in Q, A \in \Gamma, a \in \Sigma^*, | f(q, a, A)| < 2$
- Descripción instantánea
  - $(q, x, A) \mid q \in Q, x \in \Sigma^*, A \in \Gamma^*$
- AP asociado a una G_2:
  - $\forall\ G_2 \ \exists\ AP \mid AP \equiv G_2$
  - Pero el AP puede ser no determinista y no hay algoritmo para pasar de AP a APD

Conceptos asociados a AP, II

- Lenguaje aceptado por un AP
  - Por estado final
    - $LF_{AP}=\{x \mid (q_0, x, A_0) \vdash_* (p, \lambda, X) \wedge p\in F, x\in \Sigma^*, X\in \Gamma^*\}$
  - Por vaciado de pila
    - $LV_{AP}=\{x \mid (q_0, x, A_0) \vdash_* (p, \lambda, \lambda) \wedge p\in Q, x\in \Sigma^*\}$

Gramática y AP, I

- **Dada una G2, construir un Autómata a Pila:**
  - $G = (\Sigma_T, \Sigma_N, S, P) \quad Ap = (\Sigma, \Gamma, Q, \#, i, f, \{t\})$

1. $\Sigma = \Sigma_T$
2. $\Gamma= \Sigma_T \cup \Sigma_N \cup \{\#\}$
3. $Q=\{i, p, q, t\}$
4. Añadir la transición: $(i, \lambda, \lambda; p, \#)$
5. Añadir la transición: $(p, \lambda, \lambda; q, S)$
6. Añadir las transiciones:
   - $\forall A\rightarrow w \in P, (q, \lambda, A; q, w) \mid A\in\Sigma_N, w\in(\Sigma_N \cup \Sigma_T)^*$
7. Añadir las transiciones:
   - $\forall x \in \Sigma_T, (q, x, x; q, \lambda)$
8. Añadir la transición: $(q, \lambda, \#; t, \lambda)$

Gramática y AP, II

- Ejemplo: Gramática para expresiones con paréntesis balanceados
  - $G = (\Sigma_T=\{(,),o\}, \Sigma_N=\{S,E,E'\}, S, P=\{S \to E\,E', E' \to \lambda, E' \to o\,S, E \to (S), E \to S,\})$
  - $Ap = (\Sigma=\Sigma_T=\{(,),o\}, \Gamma=\Sigma_T \cup \Sigma_N \cup \{\#\} =\{(,),o,S,E,E',\#\}, Q=\{i,p,q,t\}, \#, i,\ f, \{t\})$
  - $f=\{(i, \lambda, \lambda; p, \#), (p, \lambda, \lambda; q, S),(q,\lambda,S;q,EE'), (q,\lambda,E';q,\lambda), (q,\lambda,E';q,oS), (q,\lambda,E;q,(S)), (q,\lambda,E;q,S), (q,(,(;q,\lambda), (q,),);q,\lambda), (q,o,o;q,\lambda), (q, \lambda,\#;t,\lambda) \}$

Expresiones Regulares

- Definición
  - $\varnothing$ es una ER
  - $\boldsymbol{\lambda}$ es una ER
  - $\forall a \in \Sigma$, $\boldsymbol{a}$ es una ER
  - $\forall \alpha\beta$ ER, $\boldsymbol{\alpha+\beta}$ es una ER
  - $\forall \alpha\beta$ ER, $\boldsymbol{\alpha\cdot\beta}$ es una ER
  - $\forall \alpha$ ER, $\boldsymbol{\alpha^*}$ es una ER $\quad \alpha^* = \bigcup_{i=0}^{\infty} \alpha^i$
  - Sólo son ER aquellas que se obtienen por aplicación un número finito de veces las reglas anteriores

Conceptos asociados a ER, I

- ER útiles
  - $\alpha^{+} = \alpha \cdot \alpha^{*} = \alpha^{*} \cdot \alpha$
  - $\alpha? = \alpha \mid \lambda$
  - $[\,abc\,] = a \mid b \mid c$
  - $[\,a\text{-}z\,] = a \mid b \mid \ldots \mid z$
- Dos EERR son equivalentes
  - $\alpha \equiv \beta \Rightarrow L(\alpha) = L(\beta)$

Conceptos asociados a ER, II

- Lenguaje asociado a una ER
  - si $\alpha = \varnothing$, $L(\alpha) = \varnothing$
  - si $\alpha = \lambda$, $L(\alpha) = \lambda$
  - si $\alpha = a \mid a \in \Sigma$, $L(\alpha) = \{a\}$
  - si α y β son EERR $\Rightarrow L(\alpha+\beta) = L(\alpha) \cup L(\beta)$
  - si α y β son EERR $\Rightarrow L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$
  - si α^{*} es una ER $\Rightarrow L(\alpha^{*}) = L(\alpha)^{*}$

Conceptos asociados a ER, III

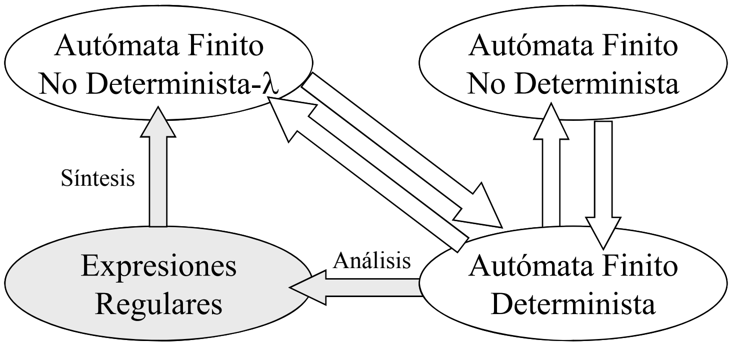
- Propiedades
  1. $(\alpha + \beta) + \sigma = \alpha + (\beta + \sigma)$ (+ es asociativa)
  2. $\alpha + \beta = \beta + \alpha$ (+ es conmutativa)
  3. $(\alpha \bullet \beta) \bullet \sigma = \alpha \bullet (\beta \bullet \sigma)$ (• es asociativa)
  4. $\alpha \bullet (\beta + \sigma) = (\alpha \bullet \beta) + (\alpha \bullet \sigma)$ (+ es distributiva
     $(\beta + \sigma) \bullet \alpha = (\beta \bullet \alpha + (\sigma \bullet \alpha)$ respecto de •)
  5. $\alpha \bullet \lambda = \lambda \bullet \alpha = \alpha$ • tiene elemento neutro)
  6. $\alpha + \Phi = \Phi + \alpha = \alpha$ (+ tiene elemento neutro)
  7. $\lambda^* = \lambda$
  8. $\alpha \bullet \Phi = \Phi \bullet \alpha = \Phi$
  9. $\Phi^* = \lambda$
  10. $\alpha^* \bullet \alpha^* = \alpha^*$
  11. $\alpha \bullet \alpha^* = \alpha^* \bullet \alpha$
  12. $(\alpha^*)^* = \alpha^*$ **(IMPORTANTE)**

Conceptos asociados a ER, IV

- Propiedades
  13. $\alpha^* = \lambda + \alpha + \alpha^2 + .. + \alpha^n + \alpha^{n+1} \bullet \alpha^*$
  14. $\alpha^* = \lambda + \alpha \bullet \alpha^*$ (13 con n=0) **(IMPORTANTE)**
  15. $\alpha^* = (\lambda + \alpha)n\text{-}1 + \alpha n \bullet \alpha^*$ (de 14, sustituyendo)
  16. Sea f una función, $f{:}E_\Sigma^{\,n} \rightarrow E_\Sigma$ se verifica: $f(\alpha, \beta, \ldots, \sigma) + (\alpha + \beta + \ldots + \sigma)^* = (\alpha + \beta + \ldots + \sigma)^*$
  17. Sea f una función, $f{:}E_\Sigma^{\,n} \rightarrow E_\Sigma$ se verifica: $(f(\alpha^*, \beta^*, \ldots, \sigma^*))^* = (\alpha + \beta + \ldots + \sigma)^*$
  18. $(\alpha^* + \beta^*)^* = (\alpha^* \bullet \beta^*)^* = (\alpha + \beta)^*$ **(IMPORTANTE)**
  19. $(\alpha \bullet \beta)^* \bullet \alpha = \alpha \bullet (\beta \bullet \alpha)^*$
  20. $(\alpha^* \bullet \beta)^* \bullet \alpha^* = (\alpha + \beta)^*$
  21. $(\alpha^* \bullet \beta)^* = \lambda + (\alpha + \beta)^* \bullet \beta$
  22. **Regla de Inferencia**:
      sean α, β y δ EERR:
      $$\alpha = \beta^* \bullet \delta \Rightarrow \alpha = \beta \bullet \alpha + \delta$$
      si $\lambda \notin \beta$ se verifica:
      $$\alpha = \beta \bullet \alpha + \delta \Rightarrow \alpha = \beta^* \bullet \delta$$

Conceptos asociados a ER, V

- Relación entre ER y AF



Problema de síntesis, I

- Dada una ER, construir un AF, que reconozca el lenguaje que la ER describe
  - Procedimiento :
    - Derivar la ER y obtener una G3LD y de ella un AF
  - Definición de derivada de ER:
    - $D_a(R) = \{ x \mid a.x \in R \}$
  - Sea $\forall\ a, b \in \Sigma$ y R, S EERR
    - $D_a(\varnothing) = \varnothing$
    - $D_a(\lambda) = \varnothing$
    - $D_a(a) = \lambda, \quad a \in \Sigma$
    - $D_a(b) = \varnothing, \quad \forall\ b \neq a,\ b \in \Sigma$
    - $D_a(R + S) = D_a(R) + D_a(S)$
    - $D_a(R \bullet S) = D_a(R) \bullet S + \delta(R) \bullet D_a(S)$
    - $D_a(R^*) = D_a(R) \bullet R^*$

$$\delta(R) = \begin{cases} \lambda \text{ si } \lambda \in L(R) \\ \varnothing \text{ si } \lambda \notin L(R) \end{cases}$$

Problema de síntesis, II

- Regla de la cadena
  - $D_{ab}(R) = D_b(D_a(R))$
- Proceso de obtención de G3 lineal derecha
  - El número de derivadas distintas de una ER es finito
  - Obtenido todas, se puede obtener la G3
- Sea $D_a(R) = S$ con $S \neq \varnothing$
  - $S \neq \lambda \Rightarrow R ::= a\, S \in \wp$
  - $S = \lambda \Rightarrow R ::= a \in \wp$
- Sea $\delta(D_a(R))$
  - $\delta(D_a(R)) = \lambda \Rightarrow R ::= a \in \wp$
  - $\delta(D_a(R)) = \varnothing \Rightarrow$ no añadir regla
- El axioma es R (ER de partida), si $\delta(R) = \lambda \Rightarrow R ::= \lambda \in \wp$
- Σ_T = símbolos que formaban la ER de partida
- Σ_N = letras que distinguen cada una de las distintas derivadas

Problema de síntesis, III

- Ejemplo:
  - Número natural, con o sin signo: $s?d^+$
    - $R_0 = (s+\lambda)dd^*$
    - $D_s(R_0) = D_s(s+\lambda)\cdot dd^* + \delta(s+\lambda)\cdot D_s(dd^*) = dd^* = R_1$
    - $D_d(R_0) = D_d(s+\lambda)\cdot dd^* + \delta(s+\lambda)\cdot D_d(dd^*) = D_d(dd^*) = d^* = R_2$
    - $D_s(R_1) = D_s(d)\cdot d^* + \delta(d)\cdot D_s(d^*) = \varnothing$
    - $D_d(R_1) = D_d(d)\cdot d^* + \delta(d)\cdot D_d(d^*) = d^* = R_2$
    - $D_s(R_2) = D_s(d^*) = \varnothing$
    - $D_d(R_2) = D_d(d^*) = d^* = R_2$
    - Se han obtenido todas las derivadas

Problema de síntesis, IV

- Ejemplo:
  - Derivadas
    - $D_s(R_0)= R_1$
    - $D_d(R_0)= R_2$
    - $D_s(R_1)= \varnothing$
    - $D_d(R_1)= R_2$
    - $D_s(R_2)= \varnothing$
    - $D_d(R_2)=R_2$
  - Deltas de las derivadas
    - $\delta(D_s(R_0)) = \delta(R_1) = \varnothing$
    - $\delta(D_d(R_0)) = \delta(R_2) = \lambda$
    - $\delta(D_s(R_1)) = \varnothing$
    - $\delta(D_d(R_1)) = \delta(R_2) = \lambda$
    - $\delta(D_s(R_2)) = \varnothing$
    - $\delta(D_d(R_2)) = \delta(R_2) = \lambda$
    - $\delta(R_0) = \varnothing$

Producciones

$R_0 \to s\, R_1$
$R_0 \to d\, R_2$

$R_1 \to d\, R_2$

$R_2 \to d\, R_2$

Producciones

$R_0 \to d$

$R_1 \to d$

$R_2 \to d$