Limpiar y bien formar siempre

Tema 3

AFD → AFD mínimo

1. Buscar Q/E_0, que divide los estados en finales y no finales.
2. Hacer Q/E_1, Q/E_2… pueden separarse, pero nunca juntarse de nuevo.
3. Cuando se repitan las particiones hemos terminado. Como máximo tendremos que hacer $Q/E_{(n-2)}$ iteraciones.

AFND → AFD

1. Calcular T*.
2. Quitar la columna λ y añadir la columna λ*.
3. Sustituir la columna a, b… por λ* a λ*, λ* b λ*…
4. El nuevo estado inicial será p λ*.
5. Transformar los caminos múltiples en estados combinados (finales si alguna de las letras es final) y las transiciones no definidas en transiciones al sumidero.

Tema 4

G3 LD → G3 LI

1. Quitar el axioma inducido (A → aS) introduciendo un nuevo símbolo (que haga lo mismo que el axioma pero no se copia lambda).
2. Construir un grafo dirigido en el que los nodos son los Σ_{NT} y las flechas son los Σ_T.
3. Intercambiar las etiquetas de λ y S, y dar la vuelta a las flechas.
4. Interpretar el grafo
   Nota: las que iban de S van a λ, y de λ no puede salir nada.

Lenguaje vacío (G2)

1. Generar el árbol de derivación hasta llegar a n (número de estados). Si no genera sentencias y se repiten los Σ_{NT}, es un lenguaje vacío.

Lenguaje infinito (G2)

1. Construir un grafo cuyos nodos están etiquetados con los Σ_{NT}. Si existen ciclos accesibles desde el axioma, entonces es un lenguaje infinito.

Limpieza y bien-formación de gramáticas

1. Eliminar reglas innecesarias (A → A)
2. Eliminar símbolos inaccesibles (para ello se construye un vector con los simbolos T y NT).
   Ir marcando desde el axioma los que vaya produciendo.
3. Eliminar reglas superfluas con el algoritmo de marcado:
   a. Se marcan los $\Sigma_{NT} \rightarrow \Sigma_T$ y los $\Sigma_{NT} \rightarrow \lambda$.
   b. Se marcan los que contengan un Σ_{NT} marcado en la derecha.
   c. Se repite hasta que no se puedan marcar más.
   d. Se eliminan todas las reglas no marcadas.

4. Eliminar los símbolos no generativos, es decir, aquellos que solo aparecen en reglas superfluas.
5. Eliminar las reglas no generativas, las de tipo A → λ. Cada vez que aparezca A en la λ. Se admite parte derecha, se añade la posibilidad de que sea Axioma→ λ, OJO si cuando eliminamos solo quedaba un simbolo (C->M y eliminamos M), ponemos lambda (C->λ) y repetimos el proceso de eliminacion (para C).
6. Eliminar las reglas de redenominación, las de tipo A → B. Por cada regla de la forma B → x, se añade A → x

G2 → FNC

1. Se separa el primer simbolo de la derecha del resto, por ejemplo, de A → aBb sacamos A → DE y D → a, E → Bb→ BC, C→ b (previo paso debe estar limpia y bien formada).

G2 → FNG

1. Limpiar y bien formar la gramática.
2. Quitar recursividad a izquierdas si la hay. λ no se toca en este paso.
3. Ordenar el αbeto Σ_{NT} (A, B) y clasificar las reglas en G2 (AB) o G3(BA).
4. Pasar las de G3 a G2. Se hace por sustitución, no sustituir con las reglas que dan lambda.
    a. Quitar recursividad a izquierdas si apareciese.
5. Pasar de G2 a G1, empezando por la que me deje meter un Σ_T en la cabeza.
6. Si hay un Σ_T que no esté en la cabeza, sustituirlo por un Σ_{NT} que dé ese Σ_T.

Quitar recursividad a izquierdas

Dada una regla de tipo A → Aα | β (donde α y β son cualquier cosa…)

Se transforma en:

- A → β | βX
- X → αX | α

Si tuviéramos varias (A → Aα | β_1 | β_2), entonces se transforma en:

- A → β_1 | β_2 | β_1X | β_2X
- X → αX | α

Paso de G3LD (FNG) → AF y viceversa

Por cada regla A → aB:

- A y B son estados del autómata y se realiza la transición de A a B con ‘a’.

Si tenemos una regla del tipo A → a:

- Se realiza la transición de A a un estado final con ‘a’.

Para pasar de AF a G3LD se hace exactamente igual.

Tema 5

Teorema de síntesis (Nota: $D_{ab}(\alpha) = D_b(D_a(\alpha))$)

1. Derivar la expresión respecto de todos los símbolos y todas las que me vayan saliendo.
   Las voy llamando R_x, donde x es un número.
2. Si R_x puede ser λ, se añade una regla, $R_x \to \lambda$.
3. Si $D_y(R_{x1}) = R_{x2}$, se añade una regla $R_{x1} \to yR_{x2}$
4. Luego se aplica el paso de G3LD (FNG) a AF para obtener el AF correspondiente.

Da(a) = λ
Da(b) = Ø
Da(RS) = Da(R)S + d(R)Da(S)
Da(R+S) = Da(R) + Da(S)
Da(R*) = Da(R)R*
d(a) = Ø
d(a*) = λ   Si puede ser λ, es λ, si no Ø
d(a*+a) = λ

Tema 6

APF → APV

1. Añadir un estado inicial nuevo con una transición y un nuevo "chivato" que nos indique cuándo se vacía la pila.
2. Añadir un estado "final" que desapile todo lo que pudiera quedar y el chivato.

APV → APF

1. Añadir un estado inicial nuevo con su chivato.
2. Añadir un estado final al que se transita desapilando el chivato.

G2 (FNG) → APV

1. Tres tipos de reglas:
   a. A → aBCD: f(q, a, A) = (q, BCD)
   b. A → a: f(q, a, A) = (q, λ)
   c. S → λ: f(q, λ, S) = (q, λ)
2. El autómata tendrá un único estado, q.

APV → G2

1. Se pone una regla del tipo S → (q_0, A_0, pqr…)
2. Se ponen reglas del tipo:
   a. Tipo 1A: f(q, a, B) = (p, DEF)
      El molde será: (qB_) → a(pD_)(_E_)(_F_)

   b. Tipo 1B: f(p, λ, B) = (q, A)
      El molde será: (pB_) → X(qA_)

   c. Tipo 2A: f(p, a, B) = (q, λ) El molde será: (pBq) → a

   d. Tipo 2B: f(p, λ, B) = (q, λ) El molde será: (pBq) → λ

Equivalencia de EERR

1. $(\alpha + \beta) + \sigma = \alpha + (\beta + \sigma)$
2. $\alpha + \beta = \beta + \alpha$
3. $(\alpha \cdot \beta) \cdot \sigma = \alpha \cdot (\beta \cdot \sigma)$
4. $\alpha \cdot (\beta + \sigma) = (\alpha \cdot \beta) + (\alpha \cdot \sigma)$
5. $\alpha \cdot \lambda = \lambda \cdot \alpha = \alpha$
6. $\alpha + \emptyset = \emptyset + \alpha = \alpha$
7. $\lambda^* = \lambda$
8. $\alpha \cdot \emptyset = \emptyset \cdot \alpha = \emptyset$
9. $\emptyset^* = \lambda$
10. $\alpha^* \cdot \alpha^* = \alpha^*$
11. $\alpha \cdot \alpha^* = \alpha^* \cdot \alpha$
12. $(\alpha^*)^* = \alpha^*$
13. $\alpha^* = \lambda + \alpha + \alpha^2 + \ldots + \alpha^n + \alpha^{n+1} \cdot \alpha^*$
14. $\alpha^* = \lambda + \alpha \cdot \alpha^*$
15. $\alpha^* = (\lambda + \alpha)\ n-1 + \alpha n \cdot \alpha^*$
16. $f(a, b, \sigma) + (\alpha + \beta + \sigma)^* = (\alpha + \beta + \sigma)^*$
17. $(f(\alpha^*, \beta^*, \sigma^*))^* = (\alpha+\beta+\sigma)^*$
18. $(\alpha^* + \beta^*)^* = (\alpha^* \cdot \beta^*)^* = (\alpha+\beta)^*$
19. $(\alpha \cdot \beta)^* \cdot \alpha = \alpha \cdot (\beta \cdot \alpha)^*$
20. $(\alpha^* \cdot \beta)^* \cdot \alpha^* = (\alpha+\beta)^*$
21. $(\alpha^* \cdot \beta)^* = \lambda + (\alpha+\beta)^* \cdot \beta$
22. $X = Ax + B \rightarrow X = A^* \cdot B$

Analisis

1. Hacer la ecuaciones del Automata Finito.

   De X0 a X1 con una ‘a’ : X0 = aX1
   De X0 a X2 que es final con una ‘b’: X0 = bX2 + b Si hay varias transiciones se ponen en la misma ecuacion sumandose.
   Si el estado final solo va al sumidero o no tiene ramas se le añade lambda
2. Utilizar las equivalencias de EERR, empezando por las mas lejanas al inicial. Esencialmente se usa la regla de inferencia:

X0 = aX0	X0 = Ø
X1 = bX1 + c	X1 = b*c
X2 = c	X2 = c

Formatos

AFD=(Alfabeto, Q, q0, f, F) F=EstadisFinales f=Funcion transicion
AFND=(Alfabeto, Q, q0, f, F, T) T=Transiciones con lambda
G=(Terminales, NoTerminales, S, P) S=Axioma P=Transiciones
AP=(AlfabetoCinta, AlfabetoPila, Q, A0, q0, f, F) A0=FondoPila
MT=(Alfabetoentrada, AlfabetoCinta, b, Q, q0, f, F) b=SimbolosEspeciales

Jerarquia Chomsky

Todos aceptan axioma para dar λ
G0: Lenguaje sin restricciones, puede ser cualquier cosa, se caracteriza por reglas compresoras (aVs -> d OJO tambien si no es axioma B->λ) y estructura de frases (AS->SA).
G1: Sensible al contexto, puede ser cualquier cosa, sin reglas compresoras y CONTEXTO (aS-> aDC)
G2: De contexto libre, un simbolo a la izquierda pero cualquier cosa a la derecha. Tambien si hay G3LD y G3LI en la misma gramatica.
G3: Gramatica regular, son la de la forma NT->T y un tipo de las siguientes
NT-> NT T G3LI
NT-> T NT G3LD