# ALGORITMO PARA REDUCIR LA COMPLEJIDAD COMPUTACIONAL EN LA CONVERSIÓN DE AFNDs. A AFDs.

## Algorithm to reduce the computational Complexity Converting an NFA to a DFA


## RESUMEN

Al convertir un Autómata Finito No Determinístico (AFND) a un Autómata Finito Determinístico (AFD) los algoritmos descritos en la mayoría de la documentación presentan una complejidad computacional del tipo exponencial ($O(2^n)$), lo cual no es deseable. Esto se debe a las múltiples combinaciones que se dan al hallar los posibles estados equivalentes entre autómatas. El presente trabajo propone un algoritmo que reduce dicha complejidad.


**PALABRAS CLAVES:** AFD, AFND, Autómata, Complejidad Computacional, Estados.


## ABSTRACT

When converting a non deterministic automata to a deterministic automata the algorithms described in most papers have an exponential computational complexity ($O(2^n)$), which is no desired. This is due to the multiple combinations that are possible to find equivalent states between automatas. This paper proposes an algorithm that reduces this complexity.


**KEYWORDS:** Automata, DFA, NFA, Computational Complexity, States.


**JORGE IVAN RIOS P**
Ingeniero Industrial,
M. Sc. Ingeniería de Sistemas,
M. Sc Ingeniería del Conocimiento.
Profesor Asociado.
Programa de Ingeniería de Sistemas y Computación.
Universidad Tecnológica de Pereira
jirios@utp.edu.co

**HUGO HUMBERTO MORALES PEÑA**
Ingeniero de Sistemas.
Profesor Auxiliar
Programa de Ingeniería de Sistemas y Computación.
Universidad Tecnológica de Pereira
huhumor@utp.edu.co

**AUGUSTO ANGEL AGUDELO ZAPATA**
Ingeniero Electricista, Esp.
Profesor Auxiliar
Programa de Ingeniería de Sistemas y Computación.
Universidad Tecnológica de Pereira
a3udeloz@utp.edu.co


## 1. INTRODUCCIÓN

Un problema que se presenta, cuando se realizan transformaciones de Autómatas Finitos No Deterministas (AFNDs.), a Autómatas Finitos Deterministas (AFDs.), concretamente en su equivalencia computacional, es que en dicha transformación, se presenta una alta complejidad de tipo computacional: $O(2^n)$, debido a que el algoritmo que se propone en toda la bibliografía revisada y referenciada [1], [2], [3], [4], [5], [6], [7], [8] y [9], tiene en cuenta todas las posibles combinaciones de los estados del AFND, al realizar dicha transformación.

Cuando se trata de pequeños AFNDs, es decir cuando el número de estados es pequeño (no mayor que de 4), la complejidad computacional de la transformación no suele ser impactante y se puede recurrir al algoritmo propuesto en la antes mencionada bibliografía.

Lo anterior se debe, a que el algoritmos que establecen la equivalencia computacional entre AFNDs. y AFDs., basan su fundamento, en que un conjunto de estados en un AFND es equivalente a uno en el AFD:

$$\delta(\tau,\sigma) = \cup_{q\in\tau}\ \Delta(q,\sigma), \quad \text{donde } \tau \subseteq Q,\ q\in Q \text{ y } \sigma\in\Sigma \quad (1)$$

En este artículo, proponemos un algoritmo, que reduce la complejidad en el caso promedio, en la conversión de un AFND a AFD.

## 2. ALGORITMO PARA DETERMINAR LA EQUIVALENCIA COMPUTACIONAL ENTRE AFNDs. Y AFDs.

### 2.1 Definiciones Básicas

**Definición**: Sea un AFD, M = (Σ, Q, s, F, δ) donde Σ es el alfabeto, Q el conjunto de estados, $s=q_0$ el estado inicial, F el conjunto de estados de aceptación, y δ es la función de transición $\delta$: $Q \times \Sigma \rightarrow Q$, y dicha función

Fecha de Recepción: 25 de Enero de 2011
Fecha de Aceptación: 24 de Abril de 2011

extendida, $\delta'$ : Q × $\Sigma^*$ → Q, la cual queda definida recursivamente así:

$$\delta'(q,\omega) = \begin{cases} q & si\ \omega = \lambda \\ \delta(\delta'(q,x),\sigma) & si\ \omega = x\cdot\sigma,\ x\in\Sigma^*, \sigma\in\Sigma \end{cases} \quad (2)$$

Por lo tanto un lenguaje aceptado por un AFD cualquiera, está definido, por:

$$L(M) = \{\ \omega \mid \omega \in \Sigma^*\ \wedge \delta'(q_0, \omega) \in F\} \quad (3)$$

De forma análoga a lo anterior, se tiene para los AFNDs., la siguiente definición:

**Definición**: Sea un AFND, definido como M' = (Σ, Q, s, F, Δ) donde Δ es la relación de transición Δ: Q × Σ → Q, y dicha relación extendida, $\Delta'$: Q × $\Sigma^*$ → Q, la cual queda definida recursivamente así:

$$\Delta'(q,\omega) = \begin{cases} \{q\} & si\ \omega = \lambda \\ \cup_{p\in\Delta'(q,x)}\ \Delta(p,\sigma) & si\ \omega = x\cdot\sigma,\ x\in\Sigma^*, \sigma\in\Sigma \end{cases} \quad (4)$$

Por lo tanto, y de manera similar que en el AFD, el lenguaje de aceptación, de un AFND, es:

$$L(M') = \{\ \omega \mid \omega \in \Sigma^*\ \wedge\ \Delta'(q_0, \omega) \cap F \neq \varnothing\} \quad (5)$$

De lo anterior, es fácil determinar que si los dos lenguajes (4) (7) son iguales, los dos autómatas son equivalentes: AFD ≡ AFND.

### 2.2 Equivalencia Computacional entre AFDs. y AFNDs.

La equivalencia y por lo tanto, el algoritmo para determinarlo, entre dos autómatas uno AFND y otro AFD, pasa por la demostración del anterior teorema.

Según [5] [6] y [8], se debe definir el AFD en función del AFND así: M'' = (Σ, Q', s', F', δ), dónde:

Σ : Alfabeto
Q' : Colección de los subconjuntos de Q: P(Q)
s' : $\{q_0\}$
F' : Colección de subconjuntos de Q, que contienen estados de F, es decir F' = {P ∈ Q' | P ∩ F ≠ ∅}
δ : Función de transición[1] Q'× Σ → Q', donde:

$$\delta(P,\sigma) = \begin{cases} \varnothing & si\ P = \varnothing \\ \cup_{q\in P}\ \Delta(q,\sigma) & si\ P \neq \varnothing \end{cases} \quad (6)$$

Es fácil colegir, que el nuevo conjunto de estados (Q'), será la nueva colección de posibles subconjuntos de los estados del AFND, es decir $2^{n+1}$ estados.

Por lo tanto, si Q = $\{q_o, q_1, \ldots, q_n\}$, entonces:

$$Q' = \{\{\},\{q_o\},\{q_1\},\ldots,\{q_n\},\ldots,\{q_o,q_1\},\ldots\{q_o,q_n\},\ \ldots\{q_1,q_2\},\ \ldots,\{\ q_o, q_1, \ldots, q_n\}\}^{2} \quad (7)$$

Al realizar las transiciones del nuevo AFD equivalente, el orden de complejidad seria:

$$m\cdot 2^{n+1} = O(2^n),\ \text{donde } |Q| = n+1,\ |\Sigma| = m \quad (8)$$

### 2.3 Ejemplo 1:

Con el fin ilustrar cómo funciona el algoritmo, y determinar su complejidad, proponemos el siguiente ejemplo.

Sea un AFND M' = (Σ, Q, s, F, Δ) donde:

- Σ = {a,b}
- Q = $\{q_o, q_1, q_2\}$
- s = $q_0$
- F = $\{q_2\}$
- Relación Δ (definida en la Tabla 1)

| Q \ Σ | *a* | *b* |
|---|---|---|
| $q_0$ | $\{q_1\}$ | ∅ |
| $q_1$ | $\{q_1\}$ | $\{q_1, q_2\}$ |
| $q_2$ | ∅ | $\{q_2\}$ |

Tabla 1. Relación Δ de transiciones del AFND

La representación gráfica del AFND (Figura 1) es:

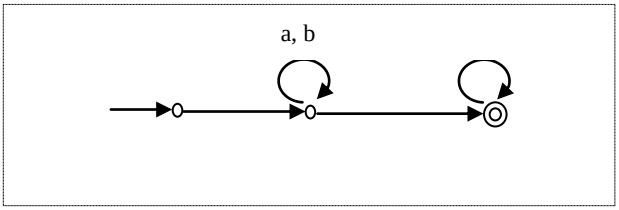


Figura 1. Diagrama de transición del AFND propuesto

[1] δ ahora es una función que operará sobre colecciones de estados

[2] En el resto del artículo de forma indiferente se utilizará el símbolo ∅ o el par de llaves sin elementos {} para representar el conjunto vacío.

Aplicando (6) y (7), se obtiene el siguiente AFD M'' = (Σ, Q', s', F', δ), equivalente:

- Σ = {a,b}
- s' = {$q_0$}
- Q'=P(Q)={{},{$q_o$},{$q_1$},{$q_2$},{$q_o$,$q_1$},{$q_o$,$q_2$}, {$q_1$,$q_2$},{$q_o$, $q_1$, $q_2$}}
- F' = {{$q_2$},{$q_o$,$q_2$}, {$q_1$,$q_2$},{$q_o$, $q_1$, $q_2$}}
- Función δ (definida en la Tabla 2)

| Σ / Q' | *a* | *b* |
|---|---|---|
| {} | {} | {} |
| {$q_0$} | {$q_1$} | {} |
| {$q_1$} | {$q_1$} | {$q_1$,$q_2$} |
| {$q_2$} | {} | {$q_2$} |
| {$q_o$,$q_1$} | {$q_1$} | {$q_1$,$q_2$} |
| {$q_o$,$q_2$} | {$q_1$} | {$q_2$} |
| {$q_1$,$q_2$} | {$q_1$} | {$q_1$,$q_2$} |
| {$q_o$, $q_1$, $q_2$} | {$q_1$} | {$q_1$,$q_2$} |

Tabla 2. Función δ de transiciones del AFD equivalente

La representación gráfica del AFD equivalente (Figura 2) es:

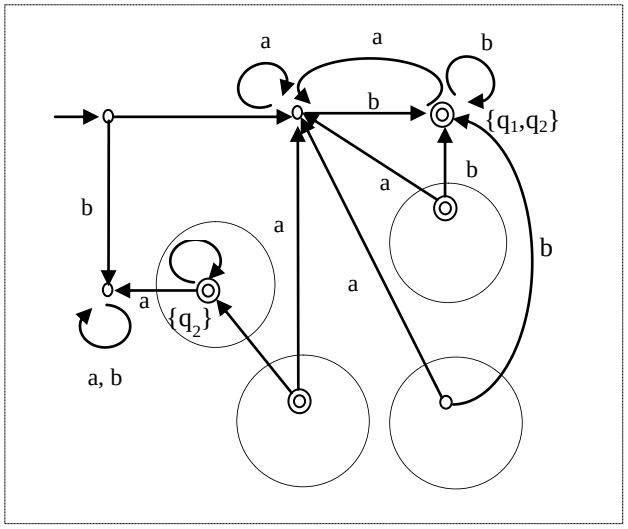


Figura 2. Diagrama de transición del AFD equivalente

Eliminando los estados inalcanzables[3] {$q_2$}, {$q_o$,$q_1$} , {$q_o$,$q_2$} y {$q_o$,$q_1$, $q_2$} (en la figura 2 aparecen con círculos punteados), y renombrando los estados del mismo, obtenemos el siguiente AFD M = (Σ, P, s, F, δ), donde:

- Σ = {a,b}
- P = {$p_0$, $p_1$, $p_2$, $p_3$}, donde P ⊆ Q' y $p_0$ = {$q_0$}, $p_1$ = {$q_1$}, $p_2$ = {$q_1$, $q_2$} y $p_3$ = {}
- s = $p_0$
- F = {$p_2$}
- Función δ: P × Σ → P, (definida en la tabla 3)

| Σ / P | *a* | *b* |
|---|---|---|
| $p_0$ | $p_1$ | $p_3$ |
| $p_1$ | $p_1$ | $p_2$ |
| $p_2$ | $p_1$ | $p_2$ |
| $p_3$ | $p_3$ | $p_3$ |

Tabla 3. Función δ de transiciones del AFD depurado

La representación gráfica del AFD depurado (Figura 3) es:

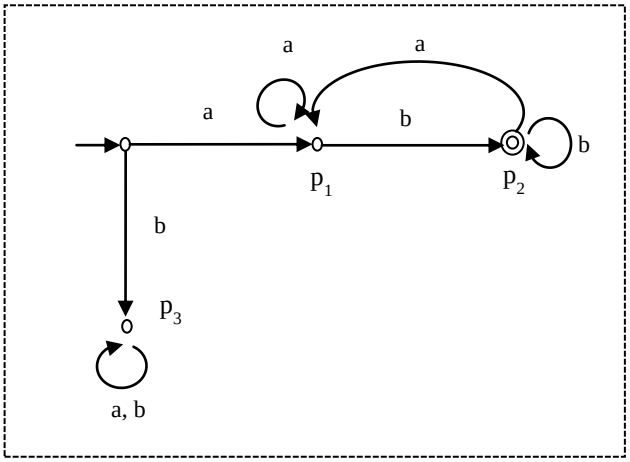


Figura 3. Diagrama de transición del AFD depurado

### 2.4 Propuesta de un nuevo Algoritmo para reducir la complejidad

Como se puede observar, el nuevo AFD (con diagrama de transición Figura 2), cuando se realiza la ejecución de (6) y (7), como parte del mismo del mismo algoritmo, aparecen ocho estados ($2^{|Q|} = 2^3$), de los cuales, cuatro son inalcanzables, desde el estado inicial. Esto es parte inherente de la transformación, donde se tiene en cuenta todos los posibles subconjuntos del conjunto de estados del AFND. La magnitud, de este nuevo conjunto, es de orden $2^n$.

La generación de la función δ, para el AFD equivalente, cuando el número de estados es alto en el AFND, la complejidad computacional de esta generación se vuelve del orden exponencial. Por ejemplo un AFND con cuarenta estados, generaría, en primer lugar, un conjunto de $2^{40}$ (1'099.511'627.776) subconjuntos de estados, que multiplicados por el número de símbolos del alfabeto, nos daría el total de transiciones del nuevo AFD equivalente.

[3] Un estado se define como inalcanzable, cuando a partir del estado inicial ($q_0$), de manera directa o a través de otros estados accesibles desde $q_0$, dicho estado no es accesible, para ningún símbolo del alfabeto.

Obsérvese también, que en la generación de los nuevos estados del AFD, aparecen estados inalcanzables, lo cual agrega un cálculo computacional innecesario, tal como se puede observar en el diagrama de transición del AFD del ejemplo propuesto (Figura 2).

Para resolver la anterior problemática, partimos del concepto de estado alcanzable, es decir a partir del estado inicial s = {$q_0$}, en la aplicación del algoritmo solo tendremos en cuenta aquellos estados que se alcancen con cada uno de los símbolos del alfabeto.

**Algoritmo para transformar AFND en AFD:**

1. Inicializar $\Gamma = \{\{q_0\}\}$
2. Para todo $\tau \in \Gamma$
3. Para todo $\sigma \in \Sigma$
4. Si $\tau = \emptyset$
5. $\delta(\tau, \sigma) = \emptyset$
6. Si $\tau \neq \emptyset$
7. $T = \cup_{q \in \tau} \Delta(q, \sigma)$
8. $\Gamma = \Gamma \cup T$
9. $\delta(\tau, \sigma) = T$

Aplicando nuestro algoritmo al ejemplo anteriormente propuesto, se obtiene lo siguiente:

- $\tau = \{q_0\}$

| $\Gamma$ \ $\Sigma$ | $a$ | $b$ |
|---|---|---|
| $\{q_0\}$ | $\{q_1\}$ | $\{\}$ |

Tabla 4. Paso a paso construcción función δ

$\Gamma = \{\{q_0\}, \{q_1\}, \{\}\}$

- $\tau = \{q_1\}$

| $\Gamma$ \ $\Sigma$ | $a$ | $b$ |
|---|---|---|
| $\{q_0\}$ | $\{q_1\}$ | $\{\}$ |
| $\{q_1\}$ | $\{q_1\}$ | $\{q_1, q_2\}$ |
| $\{\}$ | | |

Tabla 5. Paso a paso construcción función δ

$\Gamma = \{\{q_0\}, \{q_1\}, \{\}, \{q_1, q_2\}\}$

- $\tau = \{\}$

| $\Gamma$ \ $\Sigma$ | $a$ | $b$ |
|---|---|---|
| $\{q_0\}$ | $\{q_1\}$ | $\{\}$ |
| $\{q_1\}$ | $\{q_1\}$ | $\{q_1, q_2\}$ |
| $\{\}$ | $\{\}$ | $\{\}$ |
| $\{q_1, q_2\}$ | | |

Tabla 6. Paso a paso construcción función δ

$\Gamma = \{\{q_0\}, \{q_1\}, \{\}, \{q_1, q_2\}\}$

- $\tau = \{q_1, q_2\}$

| $\Gamma$ \ $\Sigma$ | $a$ | $b$ |
|---|---|---|
| $\{q_0\}$ | $\{q_1\}$ | $\{\}$ |
| $\{q_1\}$ | $\{q_1\}$ | $\{q_1, q_2\}$ |
| $\{\}$ | $\{\}$ | $\{\}$ |
| $\{q_1, q_2\}$ | $\{q_1\}$ | $\{q_1, q_2\}$ |

Tabla 7. Paso a paso construcción función δ

$\Gamma = \{\{q_0\}, \{q_1\}, \{\}, \{q_1, q_2\}\}$

Renombrando los estados como, $p_0$ = {$q_0$}, $p_1$ = {$q_1$}, $p_2$ = {$q_1$, $q_2$} y $p_3$ = {}, el nuevo AFD M = (Σ, P, s, F, δ), donde:

- Σ = {a,b}
- P = {$p_0$, $p_1$, $p_2$, $p_3$}
- s = $p_0$
- F = {$p_2$}
- La función δ: P × Σ → P, representada por la siguiente tabla:

| P \ Σ | $a$ | $b$ |
|---|---|---|
| $p_0$ | $p_1$ | $p_3$ |
| $p_1$ | $p_1$ | $p_2$ |
| $p_2$ | $p_1$ | $p_2$ |
| $p_3$ | $p_3$ | $p_3$ |

Tabla 8. Función δ del ADF equivalente

La tabla anterior (No 8) nos muestra que llegamos al mismo AFD que obtuvimos cuando aplicamos el algoritmo tradicional. En nuestro algoritmo no se tienen en cuenta todos los subconjuntos de estados ($2^{|Q|}$), si no únicamente los que son accesibles desde el nuevo estado inicial {$q_0$}, lo cual descarta los no accesibles y obviamente sus transiciones.

**2.5 Ejemplo 2:**

Con este ejemplo se deja en evidencia que no todas las veces se economiza trabajo con nuestro algoritmo de transformación de AFND a AFD versus el algoritmo tradicional.

Sea el AFND M' = (Σ, Q, s, F, Δ) donde:

- Σ = {a,b}
- Q = {$q_o$, $q_1$, $q_2$}
- s = $q_0$

- F = $\{q_2\}$
- Relación Δ (definida en la Tabla 9)

| Σ / Q | $a$ | $b$ |
|---|---|---|
| $q_0$ | $\{q_1\}$ | $\{q_2\}$ |
| $q_1$ | $\{q_0, q_1\}$ | $\{\}$ |
| $q_2$ | $\{q_1, q_2\}$ | $\{q_0, q_2\}$ |

Tabla 9. Relación de transiciones Δ del AFND

Aplicando nuestro algoritmo al AFND anterior, tenemos:

- $\Gamma = \{\{q_0\}\},\ \tau = \{q_0\}$

$T = \cup_{q\in\tau}\ \Delta(q,\sigma) = \Delta(q_0,a) = \{q_1\}$
$\Gamma = \Gamma \cup T = \{\{q_0\}\} \cup \{q_1\} = \{\{q_0\},\{q_1\}\}$
$\delta(\tau,\sigma) = \delta(\{q_0\},a) = T = \{q_1\}$

$T = \cup_{q\in\tau}\ \Delta(q,\sigma) = \Delta(q_0,b) = \{q_2\}$
$\Gamma = \Gamma \cup T = \{\{q_0\},\{q_1\}\} \cup \{q_2\} = \{\{q_0\},\{q_1\},\{q_2\}\}$
$\delta(\tau,\sigma) = \delta(\{q_0\},b) = T = \{q_1\}$

| Σ / Γ | $a$ | $b$ |
|---|---|---|
| $\{q_0\}$ | $\{q_1\}$ | $\{q_2\}$ |

Tabla 10. Paso a paso construcción función δ

- $\Gamma = \{\{q_0\},\{q_1\},\{q_2\}\},\ \tau = \{q_1\}$

$T = \cup_{q\in\tau}\ \Delta(q,\sigma) = \Delta(q_1,a) = \{q_0,q_1\}$
$\Gamma = \Gamma \cup T = \{\{q_0\},\{q_1\},\{q_2\}\} \cup \{q_0,q_1\}$
$= \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\}\}$
$\delta(\tau,\sigma) = \delta(\{q_1\},a) = T = \{q_0,q_1\}$

$T = \cup_{q\in\tau}\ \Delta(q,\sigma) = \Delta(q_1,b) = \{\}$
$\Gamma = \Gamma \cup T = \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\}\} \cup \{\}$
$= \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\}\}$
$\delta(\tau,\sigma) = \delta(\{q_1\},b) = T = \{\}$

| Σ / Γ | $a$ | $b$ |
|---|---|---|
| $\{q_0\}$ | $\{q_1\}$ | $\{q_2\}$ |
| $\{q_1\}$ | $\{q_0,q_1\}$ | $\{\}$ |

Tabla 11. Paso a paso construcción función δ

- $\Gamma = \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\}\},\ \tau = \{q_2\}$

$T = \cup_{q\in\tau}\ \Delta(q,\sigma) = \Delta(q_2,a) = \{q_1,q_2\}$
$\Gamma = \Gamma \cup T = \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\}\} \cup \{q_1,q_2\}$
$= \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\},\{q_1,q_2\}\}$
$\delta(\tau,\sigma) = \delta(\{q_2\},a) = T = \{q_1,q_2\}$

$T = \cup_{q\in\tau}\ \Delta(q,\sigma) = \Delta(q_2,b) = \{q_0,q_2\}$
$\Gamma = \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\},\{q_1,q_2\}\} \cup \{q_0,q_2\}$
$= \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\},\{q_1,q_2\},\{q_0,q_2\}\}$
$\delta(\tau,\sigma) = \delta(\{q_2\},b) = T = \{q_0,q_2\}$

| Σ / Γ | $a$ | $b$ |
|---|---|---|
| $\{q_0\}$ | $\{q_1\}$ | $\{q_2\}$ |
| $\{q_1\}$ | $\{q_0,q_1\}$ | $\{\}$ |
| $\{q_2\}$ | $\{q_1,q_2\}$ | $\{q_0,q_2\}$ |

Tabla 12. Paso a paso construcción función δ

- $\Gamma = \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\},\{q_1,q_2\},\{q_0,q_2\}\}$, $\tau = \{q_0,q_1\}$

$T = \cup_{q\in\tau}\ \Delta(q,\sigma) = \Delta(q_0,a) \cup \Delta(q_1,a)$
$= \{q_1\} \cup \{q_0,q_1\} = \{q_0,q_1\}$
$\Gamma = \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\},\{q_1,q_2\},\{q_0,q_2\}\}$
$\cup \{q_0,q_1\}$
$= \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\},\{q_1,q_2\},\{q_0,q_2\}\}$
$\delta(\tau,\sigma) = \delta(\{q_0,q_1\},a) = T = \{q_0,q_1\}$

$T = \cup_{q\in\tau}\ \Delta(q,\sigma) = \Delta(q_0,b) \cup \Delta(q_1,b)$
$= \{q_2\} \cup \{\} = \{q_2\}$
$\Gamma = \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\},\{q_1,q_2\},\{q_0,q_2\}\}$
$\cup \{q_2\}$
$= \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\},\{q_1,q_2\},\{q_0,q_2\}\}$
$\delta(\tau,\sigma) = \delta(\{q_0,q_1\},b) = T = \{q_2\}$

| Σ / Γ | $a$ | $b$ |
|---|---|---|
| $\{q_0\}$ | $\{q_1\}$ | $\{q_2\}$ |
| $\{q_1\}$ | $\{q_0,q_1\}$ | $\{\}$ |
| $\{q_2\}$ | $\{q_1,q_2\}$ | $\{q_0,q_2\}$ |
| $\{q_0,q_1\}$ | $\{q_0,q_1\}$ | $\{q_2\}$ |

Tabla 13. Paso a paso construcción función δ

- $\Gamma = \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\},\{q_1,q_2\},\{q_0,q_2\}\}$, $\tau = \{\}$

$\delta(\tau,\sigma) = \delta(\{\},a) = \{\}$
$\delta(\tau,\sigma) = \delta(\{\},b) = \{\}$

| Σ / Γ | $a$ | $b$ |
|---|---|---|
| $\{q_0\}$ | $\{q_1\}$ | $\{q_2\}$ |
| $\{q_1\}$ | $\{q_0,q_1\}$ | $\{\}$ |
| $\{q_2\}$ | $\{q_1,q_2\}$ | $\{q_0,q_2\}$ |
| $\{q_0,q_1\}$ | $\{q_0,q_1\}$ | $\{q_2\}$ |
| $\{\}$ | $\{\}$ | $\{\}$ |

Tabla 14. Paso a paso construcción función δ

- $\Gamma = \{\{q_0\},\{q_1\},\{q_2\},\{q_0,q_1\},\{\},\{q_1,q_2\},\{q_0,q_2\}\}$, $\tau = \{q_1,q_2\}$

$$T = \cup_{q \in \tau} \Delta(q, \sigma) = \Delta(q_1, a) \cup \Delta(q_2, a)$$
$$= \{q_0, q_1\} \cup \{q_1, q_2\} = \{q_0, q_1, q_2\}$$
$$\Gamma = \{\{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{\}, \{q_1, q_2\}, \{q_0, q_2\}\} \cup \{q_0, q_1, q_2\}$$
$$= \begin{Bmatrix} \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{\}, \{q_1, q_2\}, \{q_0, q_2\}, \\ \{q_0, q_1, q_2\} \end{Bmatrix}$$
$$\delta(\tau, \sigma) = \delta(\{q_1, q_2\}, a) = T = \{q_0, q_1, q_2\}$$

$$T = \cup_{q \in \tau} \Delta(q, \sigma) = \Delta(q_1, b) \cup \Delta(q_2, b)$$
$$= \{\} \cup \{q_0, q_2\} = \{q_0, q_2\}$$
$$\Gamma = \begin{Bmatrix} \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{\}, \{q_1, q_2\}, \{q_0, q_2\}, \\ \{q_0, q_1, q_2\} \end{Bmatrix}$$
$$\delta(\tau, \sigma) = \delta(\{q_1, q_2\}, b) = T = \{q_0, q_2\}$$

| $\Gamma$ \ $\Sigma$ | $a$ | $b$ |
|---|---|---|
| $\{q_0\}$ | $\{q_1\}$ | $\{q_2\}$ |
| $\{q_1\}$ | $\{q_0, q_1\}$ | $\{\}$ |
| $\{q_2\}$ | $\{q_1, q_2\}$ | $\{q_0, q_2\}$ |
| $\{q_0, q_1\}$ | $\{q_0, q_1\}$ | $\{q_2\}$ |
| $\{\}$ | $\{\}$ | $\{\}$ |
| $\{q_1, q_2\}$ | $\{q_0, q_1, q_2\}$ | $\{q_0, q_2\}$ |

Tabla 15. Paso a paso construcción función δ

Ya en estos momentos $\Gamma$ contiene todos los posibles subconjuntos de estados de $Q$, lo que indica que todos los subconjuntos de estados harán parte del AFD equivalente al AFND.

La función δ completa se presenta a continuación (Tabla 16):

| $\Gamma$ \ $\Sigma$ | $a$ | $b$ |
|---|---|---|
| $\{q_0\}$ | $\{q_1\}$ | $\{q_2\}$ |
| $\{q_1\}$ | $\{q_0, q_1\}$ | $\{\}$ |
| $\{q_2\}$ | $\{q_1, q_2\}$ | $\{q_0, q_2\}$ |
| $\{q_0, q_1\}$ | $\{q_0, q_1\}$ | $\{q_2\}$ |
| $\{\}$ | $\{\}$ | $\{\}$ |
| $\{q_1, q_2\}$ | $\{q_0, q_1, q_2\}$ | $\{q_0, q_2\}$ |
| $\{q_0, q_2\}$ | $\{q_1, q_2\}$ | $\{q_0, q_2\}$ |
| $\{q_0, q_1, q_2\}$ | $\{q_0, q_1, q_2\}$ | $\{q_0, q_2\}$ |

Tabla 16. Función δ completa para el AFD equivalente

Con el ejemplo anterior se evidencia que nuestro algoritmo en el peor de los casos genera la misma cantidad exponencial de estados que el algoritmo tradicional, donde podemos garantizar que todos estos son alcanzables desde el estado inicial.

## 3. CONCLUSIONES Y RECOMENDACIONES

En el algoritmo tradicional que consiste de la ecuaciones (6) y (7), las complejidades en el mejor y en el peor de los casos son de orden exponencial ($\Theta(2^n)$). El algoritmo propuesto presenta una mejora sustancial mejorando el mejor de los casos, obteniéndose una complejidad de orden lineal (O(n)) y el caso promedio se aproxima a una cuadrática ($O(n^2)$).

En un próximo artículo se determinará analíticamente la complejidad del caso promedio.

## 4. BIBLIOGRAFÍA


[1] Ding-Zhu Du, Ker-I Ko, *Problem Solving in Automata, Languages, and Complexity*, John Wiley & Sons, INC., Capítulo 2, páginas 23-53, 2001, ISBN 0-471-43960-6

[2] Rodrigo De Castro Korgi, *Teoría de la Computación, Lenguajes Autómatas y Gramáticas*, Universidad Nacional de Colombia, Facultad de Ciencias, UNIBIBLOS, Bogotá D.C, Capítulo 2, páginas 25-42, 2004.

[3] Dean Kelley, *Teoría de Autómatas y Lenguajes Formales*, Editorial Prentice-Hall, ISBN 0-13-497777-7, España, Capítulo 2, páginas 53-69, 1995.

[4] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Second Edition, Addison-Wesley, ISBN 0-201-44124-1, United States of America, Capítulo 2, páginas 45-63, 2001.

[5] Pedro García, Tomás Pérez, José Ruiz, Encarna Segarra, José M. Sempere, y Manuel Vásquez de Parga, *Teoría de Autómatas y Lenguajes Formales*, Alfaomega, ISBN 970-15-0661-8, México, Capítulo 2, páginas 32-44, 2001.

[6] Rafael Cases Muñoz y Lluis Márquez Villodre, *Lenguajes Gramáticas y Autómatas, Curso Básico*, Alfaomega, ISBN 970-15-0775-4, México, Capítulo 4, páginas 74-88, 2002.

[7] Pedro Isasi, Paloma Martínez y Daniel Borrajo, *Lenguajes Gramáticas y Autómatas, Un enfoque Práctico*, Addison-Wesley, ISBN 0-201-65323-0, España, Capítulo 3, páginas 63-81, 1997.

[8] Raul Gómez Marín y Andrés Sicard, *Informática Teórica. Elementos Propedéuticos*, Fondo Editorial Universidad EAFIT, Sección 4.4, páginas 145-150, 2002.

[9] Juraj Hromkovic, *Theoretical Computer Science. Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography*, Spinger Verlag, Berlin, Capítulo 3, páginas 55-87, 2004.