A.F. No Determinista

AUTÓMATA FINITO NO DETERMINISTA

- Definimos el **autómata finito determinista** como un modelo formal: $\mathbf{AFD = (\Sigma, Q, q_0, A, \mathit{f}\,)}$

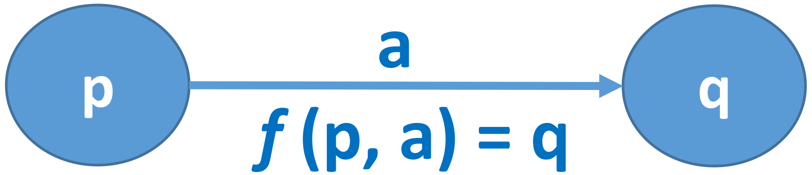
  que se utiliza para reconocer lenguajes:

  $$\mathbf{L(AFD) = \{\ \alpha \in \Sigma^* \ /\ (q_0, \alpha) \vdash^* (q_f, \lambda)\ con\ q_f \in A\}}$$

- El término **DETERMINISTA** se refiere a que el modelo tiene **sólo una forma** de realizar su tarea utilizando la función:

  $$\mathbf{\mathit{f}: Q \times \Sigma \rightarrow Q}$$

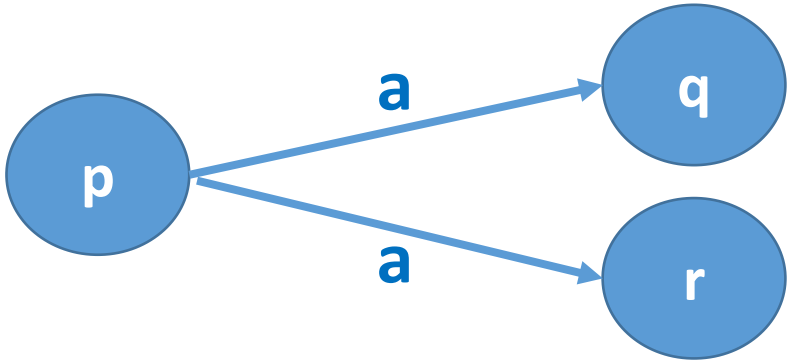
- Para un estado actual y un símbolo de entrada, la función da como resultado siempre un **único próximo** estado:



A.F. No Determinista

AUTÓMATA FINITO NO DETERMINISTA

- Exploraremos ahora la forma de dar al autómata **más de una forma de hacer las cosas**. En particular queremos permitir:

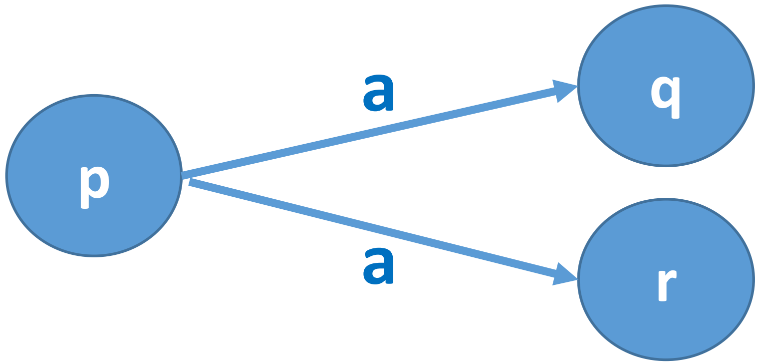


Que el autómata pueda transitar a **más de un estado** al leer **el mismo** símbolo de entrada.

Que el autómata pueda transitar a **otro estado** sin leer símbolos en su entrada (***transición λ*** o ***transición espontánea***)

AUTÓMATA FINITO NO DETERMINISTA

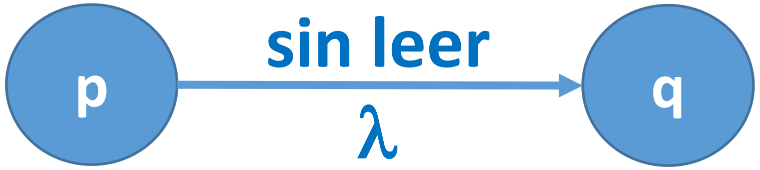
- Si tiene **más de una forma de hacer las cosas** se dice que el autómata se comporta en forma **NO DETERMINISTA** y se lo denomina **autómata finito no determinista**.



Para dar esta característica, se debe cambiar la función de transición:

$$f\colon Q \times \Sigma \to \mathcal{P}(Q)$$

donde $\mathcal{P}(Q)$ es el conjunto potencia de **Q**, permitiendo transitar a **todo un subconjunto de estados**.



Para permitir transitar sin leer, el **dominio de *f*** debe ser: $Q \times (\Sigma \cup \{\lambda\})$

AUTÓMATA FINITO NO DETERMINISTA

MODELO FORMAL AFND

Definición: Un ***autómata finito no determinista*** es un modelo matemático compuesto por una quíntupla:

$$\mathbf{AFND = (\Sigma, Q, q_0, A, \mathit{f}\,)}$$

cuyos componentes son:

- **Σ: alfabeto de símbolos de entrada**
- **Q: conjunto finito y no vacío de estados posibles**
- **$q_0 \in Q$: estado inicial de operaciones del autómata**
- **$A \subseteq Q$: conjunto de estados de aceptación**
- **$f : Q \times \Sigma \rightarrow \mathcal{P}(Q)$: función de transición**

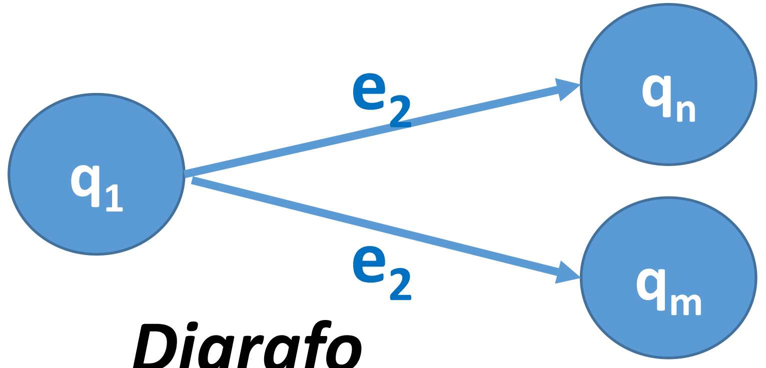
AUTÓMATA – NO DETERMINISTA – RECONOCEDOR

AUTÓMATA FINITO NO DETERMINISTA

MODELO FORMAL AFND

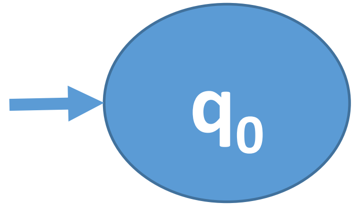
$f\!:Q\times\Sigma\rightarrow\mathscr{P}(Q)$ $\qquad$ $f(q_1,e_2) = \{q_n, q_m\}$

f		Σ			
		e_1	e_2	…	e_k
Q	$\rightarrow q_0$	…	…	…	…
	q_1	…	$\{q_n, q_m\}$	…	…
	…	…	…	$\{q_m\}$	…
	$*\ q_n$	…	…	…	…



Digrafo

Estado Inicial



Estados de Aceptación

$\alpha\in\Sigma^*$ $\longrightarrow$ **AFND** $\xrightarrow{\text{Acepta / Rechaza}}$

AUTÓMATA FINITO NO DETERMINISTA

$$\mathbf{AFND = (\Sigma, Q, q_0, A, \mathit{f}\,)}$$

Funcionamiento: Siempre iniciando su funcionamiento desde el estado inicial q_0, el autómata:

1. Lee un símbolo de entrada e_{leido} y, en el modelo mecánico mueve el cabezal de lectura a la derecha una posición.
2. Transita a los estados $\{q_1, q_2 \ldots, q_k\} = f(q_{actual}, e_{leido})$.
3. Repite los pasos 1-2 "***no determinísticamente***" y al terminar de leer la cadena de entrada:
   - Si algún $q_f \in A$, se dice que **ACEPTA** la cadena leída.
   - Si ningún $q_f \in A$, se dice que **RECHAZA** la cadena.

AUTÓMATA – NO DETERMINISTA –RECONOCEDOR

AUTÓMATA FINITO NO DETERMINISTA

Funcionamiento: Merece una discusión el **funcionamiento no determinístico** de esta máquina abstracta.

Al comienzo, el AFND parte del estado inicial q_0. Al trabajar guiado por su función de transición puede ser que en algún instante se encuentre en **más de un estado posible**.

Entonces sigue desde cada uno de esos estados leyendo su entrada y transitando a nuevos estados por **múltiples caminos de ejecución**. ¿Cómo lo hace? Es **implementación**.

Al finalizar de leer la cadena de entrada, si por algún camino llegó a un estado de aceptación $q_f \in A$, entonces la cadena es **aceptada**. Si por ningún camino llega a aceptar, **la rechaza**.

Aceptación de palabras por un AFND

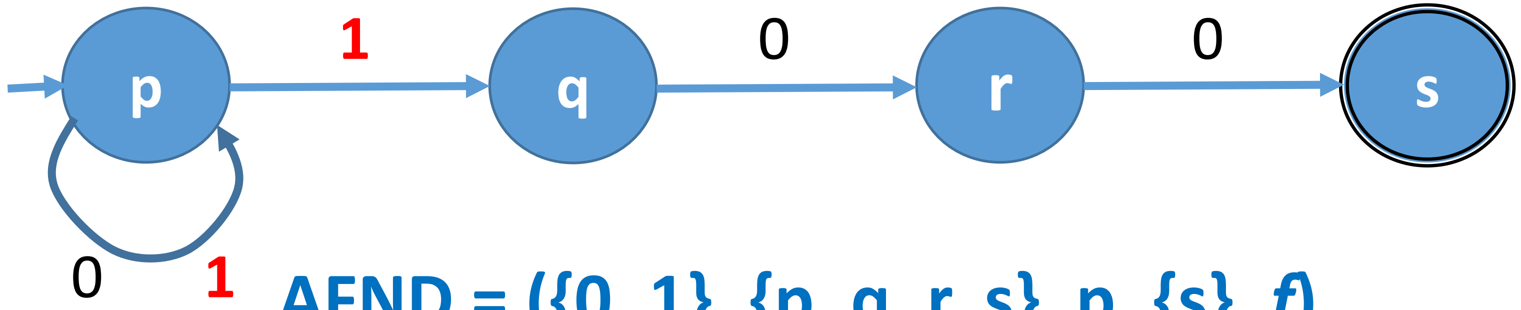
Un **AFND = (Σ, Q, q_0, A, f)** **acepta** o **reconoce** una cadena de símbolos de entrada $\alpha \in \Sigma^*$ si **es posible** que partiendo de una configuración inicial pueda moverse a una configuración de aceptación. En símbolos:

$$(q_0, \alpha) \vdash^* (q_f, \lambda) \text{ con } q_f \in A$$

El "**es posible**", se refiere a que el AFND debe seguir todos los caminos posibles **no determinísticamente** y si por alguno de ellos se llega a un **estado de aceptación** al terminar de leer la palabra completamente, entonces la misma es **aceptada**.

AUTÓMATA FINITO NO DETERMINISTA

Ejemplo: Diseñar un AFND que acepte cadenas binarias que terminen en **100**; esto es, el formato de las cadenas queda descripto por la expresión regular **(0+1)*100**.



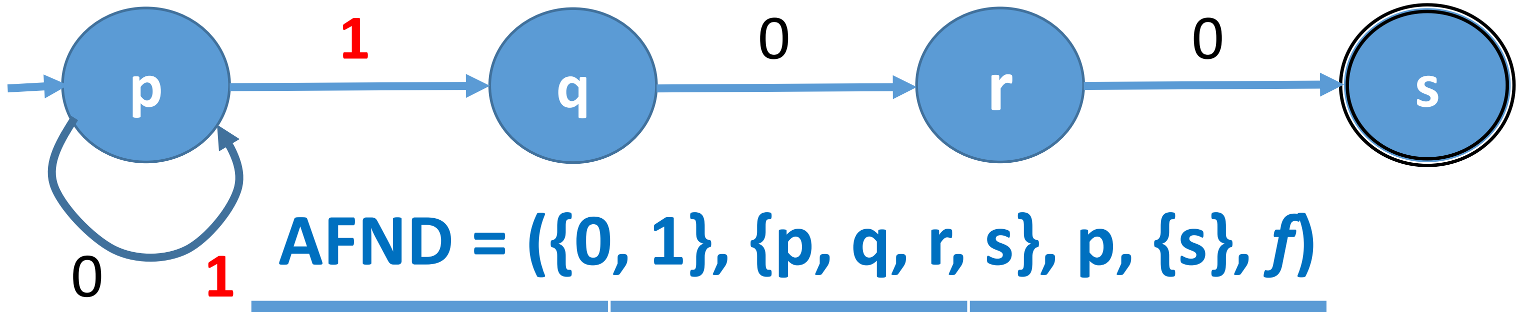
AFND = ({0, 1}, {p, q, r, s}, p, {s}, *f*)

Notar el no determinismo en ***f*(p, 1) = {p, q}**.

Un AFD también puede hacer la tarea, **pero es mucho más sencillo pensar** el autómata usando el no determinismo.

AUTÓMATA FINITO NO DETERMINISTA

Ejemplo: Diseñar un AFND que acepte cadenas **(0+1)*100**.

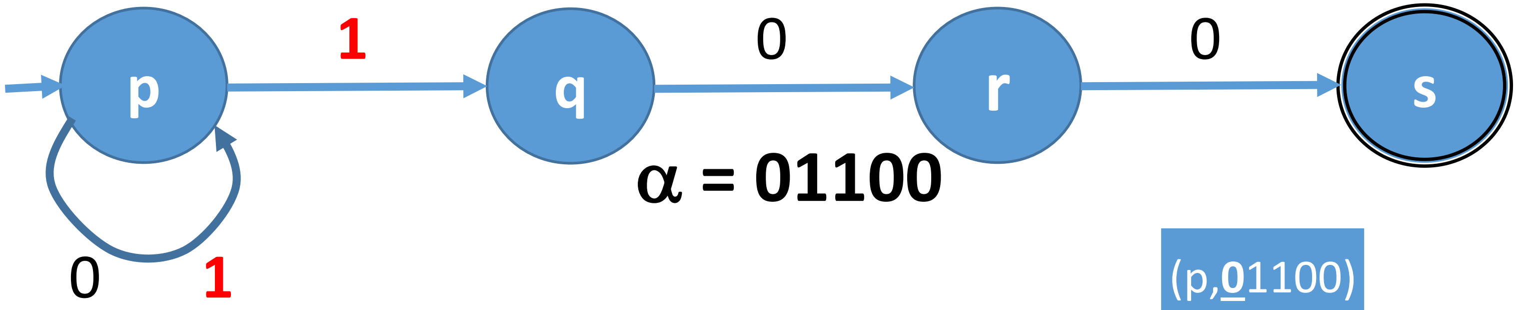


AFND = ({0, 1}, {p, q, r, s}, p, {s}, *f*)

f	0	1
→p	{p}	{p, q}
q	{r}	
r	{s}	
*s		

¿Cómo procesa este autómata la cadena α = **01100**?

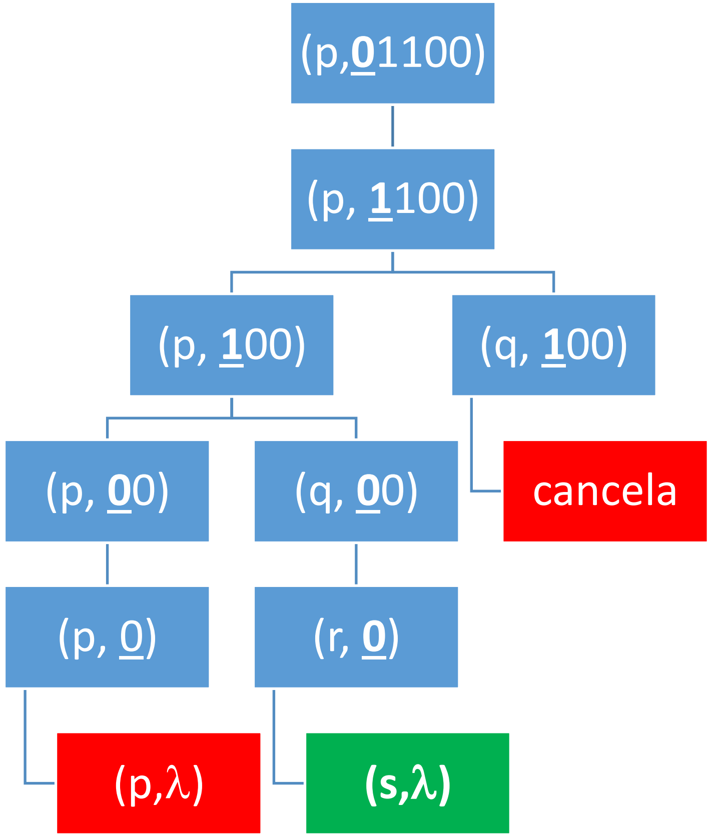
AUTÓMATA FINITO NO DETERMINISTA



α = **01100**

El árbol de configuraciones se ramifica en el lugar donde existe no determinismo.
El AFND sigue todos los caminos del árbol, y si por alguno de ellos, al terminar de leer la cadena llega a "**s**", ACEPTA.

Factor de ramificación: |f(q,a)|



AUTÓMATA FINITO NO DETERMINISTA

MODELO FORMAL AFND-λ

Definición: Un ***autómata finito no determinista con transiciones espontáneas*** es un modelo matemático compuesto por:

$$\mathbf{AFND\text{–}\lambda = (\Sigma, Q, q_0, A, \mathit{f}\,)}$$

cuyos componentes son:

- **Σ: alfabeto de símbolos de entrada**
- **Q: conjunto finito y no vacío de estados posibles**
- **$q_0 \in Q$: estado inicial de operaciones del autómata**
- **$A \subseteq Q$: conjunto de estados de aceptación**
- **$f : Q \times (\Sigma \cup \{\lambda\}) \rightarrow \mathcal{P}(Q)$: función de transición**

AUTÓMATA – NO DETERMINISTA CON λ – RECONOCEDOR

A.F. No Determinista

- Concepto
- Definición AFND
- Funcionamiento
- Aceptación de pal.
- Ejemplo AFND
- **Definición AFND-λ**
- Funcionamiento
- Aceptación de pal.
- Ejemplo AFND-λ
- Transición lambda
- Equivalencias
  - AFND-λ y AFND
  - AFND y AFD
  - AFND-λ y AFD
- Isomorfismos
  - AFD y GR
  - GR y AFND
  - ER y AFND-λ
  - Alg. Thompson
- Análisis Léxico en un compilador

AUTÓMATA FINITO NO DETERMINISTA

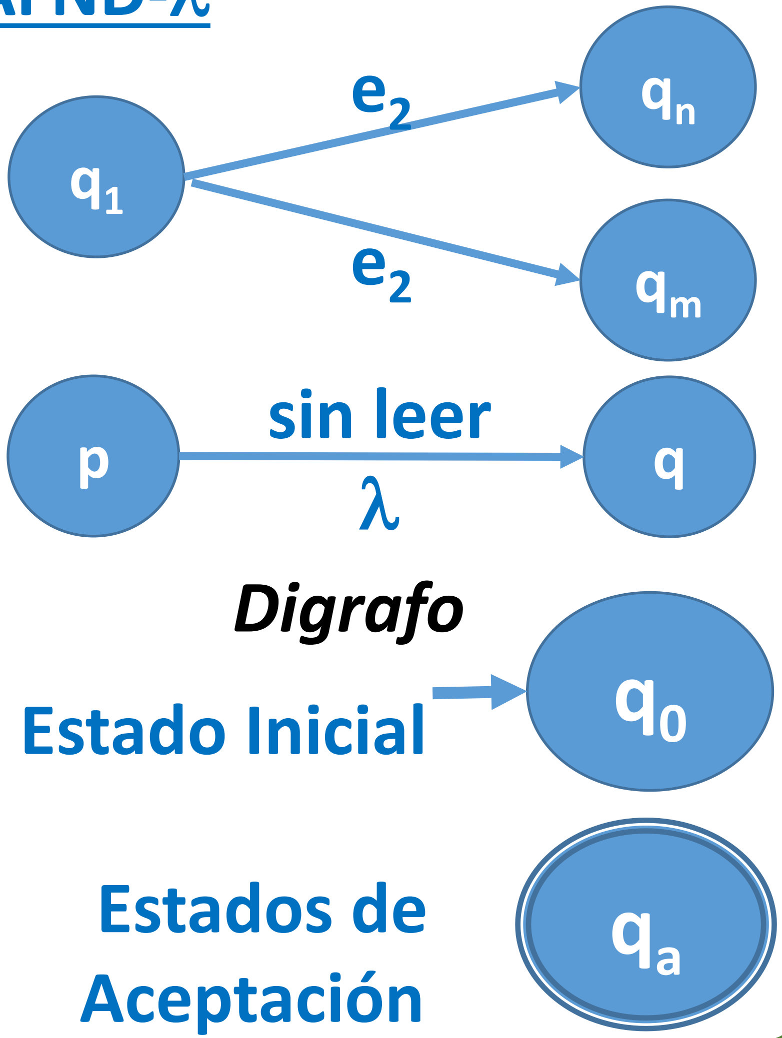
MODELO FORMAL AFND-λ

f:Q×(Σ∪{λ})→𝒫(Q) $f(q_1,e_2) = \{q_n, q_m\}$

f		Σ				λ
		e_1	e_2	…	e_k	
Q	→ q_0	…	…	…	…	…
	q_1	…	$\{q_n, q_m\}$	…	…	$\{q_k\}$
	…	…	…	$\{q_m\}$	…	…
	* q_n	…	…	…	…	…

$f(q_1,\lambda) = \{q_k\}$

$\alpha \in \Sigma^*$ → **AFND** → Acepta / Rechaza



AUTÓMATA FINITO NO DETERMINISTA

$$\text{AFND–}\lambda = (\Sigma, Q, q_0, A, f)$$

Funcionamiento: Siempre iniciando su funcionamiento desde el estado inicial q_0, el autómata:

1. **Lee un símbolo** de entrada $lec = e_{leido}$ y, en el modelo mecánico mueve el cabezal de lectura a la derecha una posición, **o no lee** nada ($lec = \lambda$) y no mueve el cabezal.
2. Transita a los estados $\{q_1, q_2 \ldots, q_k\} = f(q_{actual}, lec)$.
3. Repite los pasos 1-2 "***no determinísticamente***" y al terminar de leer la cadena de entrada:
   - Si algún $q_f \in A$, se dice que **ACEPTA** la cadena leída.
   - Si ningún $q_f \in A$, se dice que **RECHAZA** la cadena.

AUTÓMATA FINITO NO DETERMINISTA

Aceptación de palabras por un AFND-λ

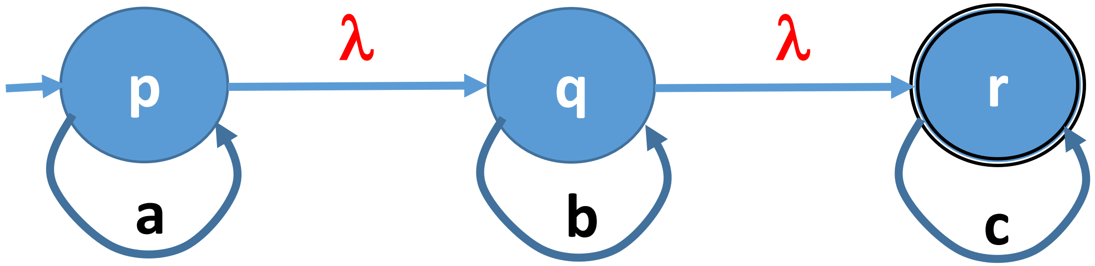
Un **AFND-λ = (Σ, Q, q_0, A, f)** acepta o reconoce una cadena de símbolos de entrada $\alpha \in \Sigma^*$ si **es posible** que partiendo de una **configuración inicial** pueda moverse inclusive con transiciones espontáneas a una **configuración de aceptación**. En símbolos:

$$(q_0 , \alpha) \vdash^* (q_f , \lambda) \text{ con } q_f \in A$$

El "**es posible**", se refiere a que el AFND-λ debe seguir todos los caminos posibles **no determinísticamente** y si por alguno de ellos se llega a un **estado de aceptación al terminar de leer la palabra completamente**, entonces es **aceptada**.

AUTÓMATA FINITO NO DETERMINISTA

Ejemplo: Diseñar un AFND que acepte cadenas que respon-dan al patrón dado por la expresión regular **a*b*c***.



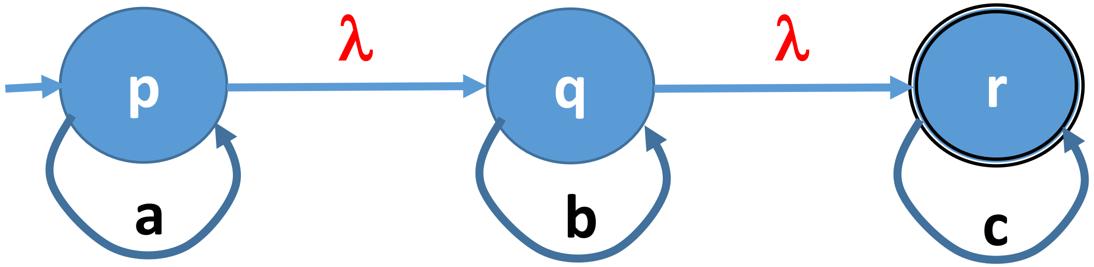
AFND-λ = ({a, b, c}, {p, q, r}, p, {r}, *f*)

Notar el no determinismo en las transiciones *λ*.

Un AFD también puede hacer la tarea, **pero es mucho más sencillo pensar** el autómata usando el no determinismo de las transiciones espontáneas.

AUTÓMATA FINITO NO DETERMINISTA

Ejemplo: Diseñar un AFND-λ que acepta cadenas **a*b*c***.

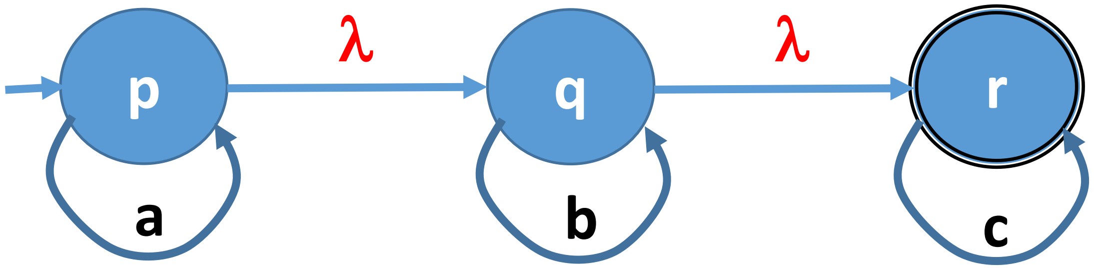


AFND-λ = ({a, b, c}, {p, q, r}, p, {r}, *f*)

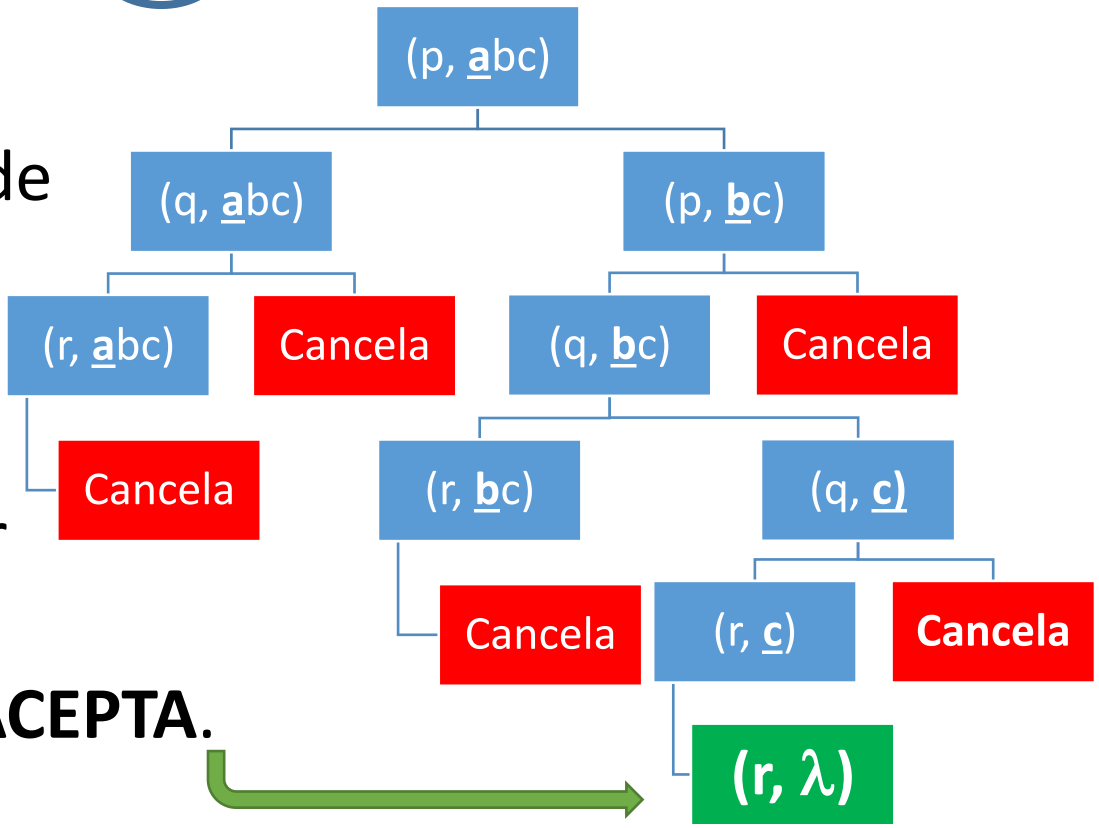
f	a	b	c	λ
→p	{p}			{q}
q		{q}		{r}
*r			{r}	

¿Cómo procesa este autómata la cadena α = **abc**?

AUTÓMATA FINITO NO DETERMINISTA



Procesamiento de la cadena α = **abc**



El árbol de configuraciones se ramifica en el lugar donde existe una transición λ.

El AFND-λ sigue todos los caminos del árbol, y si por alguno de ellos, al terminar de leer la cadena llega al estado de aceptación "**r**", **ACEPTA**.

AUTÓMATA FINITO NO DETERMINISTA

Transiciones Lambda: Un ***autómata finito no determinista con transiciones espontáneas*** fue definido como:

$$\textbf{AFND–}\lambda = (\Sigma, Q, q_0, A, f\,) \text{ con } f : Q \times (\Sigma \cup \{\lambda\}) \rightarrow \mathcal{P}(Q)$$

Al trabajar con transiciones-λ, hay transiciones implícitas que no se especifican en la tabla del autómata. Para visualizarlas y tenerlas en cuenta es útil definir:

Relación de transiciones-λ:

$$T = \underbrace{\{ (p, q) \,/\, f(p, \lambda) = q \}}_{\text{explícitas}} \cup \underbrace{\{ (x, x) \,/\, x \in Q \}}_{\text{implícitas reflexivas}}$$

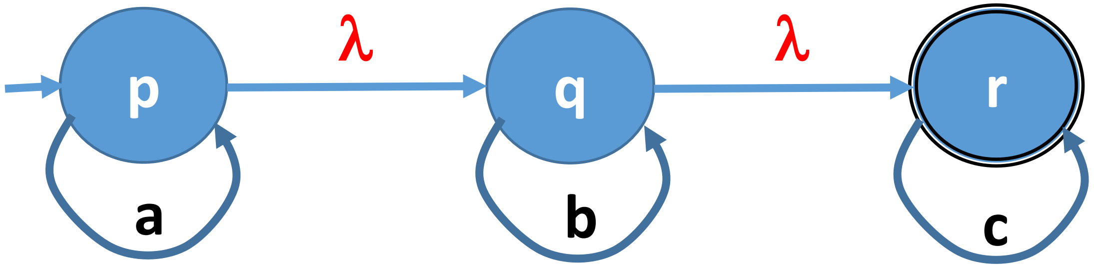
Cierre transitivo de la relación de transiciones-λ:

$$T^* = T \cup \underbrace{\{ (p, r) \,/\, pTq \wedge qTr \}}_{\text{Implícitas transitivas}}$$

Autómata Finito No Determinista

Transiciones Lambda (continuación)

Consideremos el ejemplo:



f	A	B	C	λ
$\rightarrow$p	{p}			{p, q, r}
q		{q}		{q, r}
*r			{r}	{r}

$T = \{ (p, q)/f(p, \lambda)=q \} \cup \{ (x, x)/x \in Q \} = \{ (p, q); (q, r); (p, p); (q, q); (r, r) \}$

$T^* = T \cup \{ (p, r) / pTq \wedge qTr \} = \{ (p, q); (q, r); (p, p); (q, q); (r, r); (p, r) \}$

Equivalencia entre AFND y AFND-λ

Teorema 1: Todo **AFND-λ** tiene un **AFND equivalente**.

Dado un **AFND–λ** = (Σ, Q, q_0, A, f) con $f: Q \times (\Sigma \cup \{\lambda\}) \rightarrow \mathcal{P}(Q)$ siempre se puede construir un **AFND** = (Σ, Q, q_0, A', f') con $f': Q \times \Sigma \rightarrow \mathcal{P}(Q)$ **equivalente**, eliminando las transiciones λ de la siguiente forma:

0. Copiar todas las transiciones de ***f*** a ***f'***, **sin** las transiciones λ. Copiar también el conjunto **A** a **A'**.
1. Si pT^*q y $f(q, e) = r$ entonces **agregar** a ***f'***, $f'(p, e) = r$.
2. Si $f(p, e) = q$ y qT^*r entonces **agregar** a ***f'***, $f'(p, e) = r$.
3. *Si* q_0T^*r y $r \in A$ entonces **agregar** q_0 a **A'**.

Con esto se asegura que **L(AFND-λ) = L(AFND)**.

Equivalencia entre AFND (con o sin λ) y AFD

Teorema 2: Todo **AFND** tiene un **AFD equivalente**.

Caso 1: Dado un **AFND = (Σ, Q, q_0, A, *f*)** con ***f***:**Q** $\times \Sigma \rightarrow \mathcal{P}$**(Q)** siempre se puede construir un **AFD = (Σ, Q', q_0', A', *f'*)** con ***f'***:**Q'** $\times \Sigma \rightarrow$ **Q'** **equivalente** de la siguiente forma:

a) **$c_0 = \{q_0\}$** **nuevo estado inicial q_0' del AFD**.

b) **$c_i = f'(c_k, e)$ $\forall$k<i** y **$\forall e \in \Sigma$** **se va armando *f'* y los nuevos estados** **$c_i = \{ q_j \,/\, f(q_k, e) = q_j$ para cada $q_k \in c_k \} \in Q'$**.

c) **$c_f \in A'$ si tiene algún** **q_f** en **A** como elemento.

d) **Q'** **queda armado con los estados c_i.**

e) ***f'*:Q' x $\Sigma \rightarrow$ Q'** **armada durante la ejecución del punto b).**

Equivalencia entre AFND (con o sin λ) y AFD

Teorema 2: Todo **AFND-λ** tiene un **AFD equivalente**.

Caso 2: Dado un **AFND-λ = (Σ, Q, q_0, A, *f*)** con ***f***:**Q** × **(Σ ∪ {λ})** → **𝒫(Q)** siempre se puede construir un **AFD = (Σ, Q', q_0', A', *f'*)** con ***f'***:**Q'×Σ** → **Q'** **equivalente** de la siguiente forma:

a) $c_0 = f(q_0, \lambda^*)$ **nuevo estado inicial q_0' del AFD**.

b) $c_i = f'(c_k, e)$ $\forall k<i$ y $\forall e \in \Sigma$, **y luego λ. Se va armando *f'***.

$c_i = \{ q_j \,/\, f(f(q_k, e), \lambda^*) = q_j$ **para cada** $q_k \in c_k \} \in Q'$.

c) $c_f \in A'$ **si tiene algún** q_f en **A** como elemento.

d) **Q' queda armado con los estados c_i.**

e) f':Q' x Σ → Q' **armada en b)**. Hay que verificar si es **conexo**.

AUTÓMATA FINITO NO DETERMINISTA (CONVERSIONES)

El **teorema 1** demuestra que: **$\forall$AFND-λ, $\exists$ AFND equivalente**

El **teorema 2** demuestra que:

Caso 1: **$\forall$AFND, $\exists$ AFD equivalente**

Caso 2: **$\forall$AFND-λ, $\exists$ AFD equivalente**

Debe notarse que también **$\forall$AFND, $\exists$ AFND-λ equivalente** con sólo agregar la transición reflexiva λ en cada estado, ya que en un **AFND** si se está en un estado **q** y no se lee nada, se permanece en **q**.

O sea, siempre se puede utilizar directamente el caso 2 para convertir cualquier autómata finito no determinista en uno determinista equivalente.

Isomorfismo entre Gramáticas Regulares y AF

AFD→GR: Para cada **AFD = (Σ, Q, q_0, A, *f*)** se puede construir una gramática regular **G** = (Σ_T, Σ_N, **S, P)** tal que **L(AFD) = L(G)**, con el siguiente procedimiento.

a) $\Sigma_T = \Sigma$ **símbolos terminales de G son de entrada del AF**.

b) $\Sigma_N = Q$ **símbolos no terminales de G son estados de AF**.

c) $S = q_0$ **el axioma de G es el estado inicial del AF**.

d) **P** **se arma desde la función de transición *f* del AF**.

- Si ***f*(X, a) = Y** entonces agregar la producción **X := aY**
- Si ***f*(X, a) = Y$\in$A** entonces agregar la producción **X := a**
- Si ***f*(S, λ) = Y$\in$A** entonces agregar la producción **S := λ**

Isomorfismo entre Gramáticas Regulares y AF (continuación)

GR→AFND: Para cada **G = (Σ_T, Σ_N, S, P)** se puede construir un autómata finito no determinista **AFND-λ = (Σ, Q, q_0, A, *f*)** tal que **L(ANFD-λ) = L(G)**, con el siguiente procedimiento.

a) **$\Sigma = \Sigma_T$** **símbolos de entrada del AF son terminales de G**.

b) **$Q = \Sigma_N \cup \{F\}$** **estados del AF son símbolos no terminales de G más un nuevo estado F que será de aceptación**.

c) **q_0 = S** **el estado inicial del AF es el axioma de G**.

d) **A = {F}** **el conjunto de estados de aceptación tiene solo F.**

e) ***f*** **se arma desde el conjunto de producciones de P.**

- Si **X := aY** está en P entonces agregar ***f*(X, a) = Y**
- Si **X := a** está en P entonces agregar ***f*(X, a) = F**
- Si **S := λ** está en P entonces agregar ***f*(S, λ) = F**

AUTÓMATA FINITO NO DETERMINISTA

Isomorfismo entre Expresiones Regulares y AFND-λ

ER→AFND-λ: Para cada **expresión regular** se puede construir un autómata finito no determinista **AFND-λ = (Σ, Q, q_0, A, *f*)** tal que **L(ANFD-λ) = L(ER)**, utilizando el procedimiento denominado **Algoritmo de Thompson**.

La idea de este algoritmo es desarrollar para cada parte de la definición de expresiones regulares, un autómata finito que reconozca las mismas cadenas que genera la expresión regular.

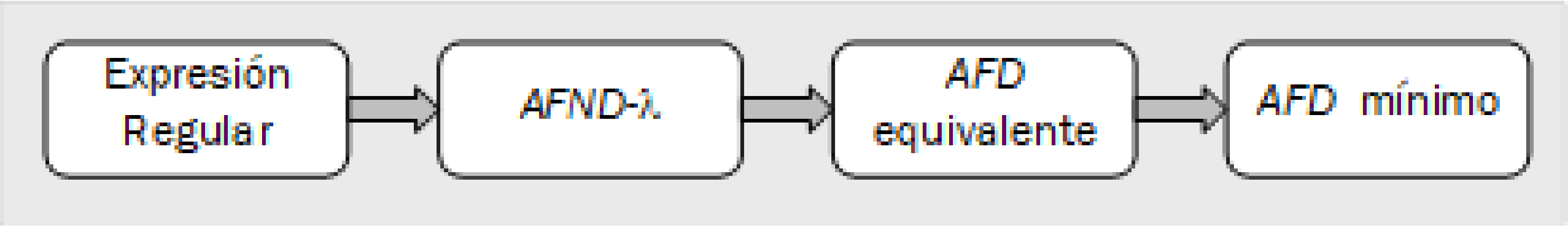
Con esa construcción, indica cómo se puede ir conformando el autómata paso a paso desde la expresión regular.

Algoritmo de Thompson

Ver documento con descripción paso a paso y apunte

Análisis Léxico en un compilador

Los conceptos, operaciones y conversiones que hemos visto sobre gramáticas regulares, expresiones regulares y autómatas finitos deterministas y no deterministas, permiten desarrollar la siguiente secuencia:



Para luego implementar en código el AFD obtenido.

AUTÓMATA FINITO NO DETERMINISTA

Análisis Léxico en un compilador (continuación)

Eso se hace al desarrollar el **análisis léxico** de un compilador de cualquier lenguaje de programación (ver **lex** / **flex** en web) :