II. APRENDIZAJE INDUCTIVO

1. INTRODUCCIÓN Y CONCEPTOS GENERALES.

- **Inferencia, inducción y deducción.**

Se entiende por ***inferencia*** todo proceso lógico que a partir de un conocimiento de un determinado nivel de genera conocimiento de mayor nivel. La mayor parte de los tipos de aprendizaje conllevan una inferencia en mayor o menor grado. En aprendizaje dos procesos de inferencia usuales son la ***inducción*** y la ***deducción*** que caracterizan a dos de los principales tipos de aprendizaje. En el primero se basa el aprendizaje que se va a estudiar en esta ocasión: ***el aprendizaje inductivo***, mientras que el segundo originará otro tipo de aprendizaje que estudiaremos más adelante: el *aprendizaje deductivo*. La combinación de los dos también origina un tipo de sistemas que serán tema de estudio: *sistemas integrados* o de *revisión general de teorías*.

La inducción generaliza propiedades obtenidas de un subconjunto de elementos un dominio. Desde un punto de vista lógico la inducción *no preserva la verdad* del sistema; es decir, lo aprendido por inducción no obligatoriamente tiene que ser verdad en todo el dominio al que se asigna el conocimiento aprendido, siempre hay un riesgo de una generalización excesiva o para la que existan excepciones. Es ese un aspecto distintivo de la inducción frente a la deducción, proceso de inferencia este último que sí *preserva la verdad*.

La inducción en un sentido amplio consiste en encontrar propiedades comunes a un subconjunto finito de elementos de un cierto dominio y considerar esas propiedades extensibles a cualquier elemento del dominio. Una buena inducción tratará de preservar la verdad, obviará aquellos ejemplos que contengan errores produciendo ***ruido*** en la información que proporcionan y hasta puede contemplar ***excepciones*** al conocimiento extraído. Es un concepto estudiado profusamente desde la Lógica aunque no está cerrado, como suele ser habitual en todos los problemas en los que interviene la comprensión de los procesos mentales que originan el conocimiento. Precisamente la posibilidad de inducción en el dominio de los números naturales es su principal característica, recogida con el nombre de principio de inducción en la introducción axiomática de estos por Peano.

- **El dominio de la inducción, los ejemplos y el objeto del aprendizaje.**

Los dominios que se manejan en el aprendizaje inductivo son tales que sus elementos vienen dados por un conjunto de propiedades o, más específicamente, por pares de la forma ***atributo:valor***. Por ejemplo:

(estatura: alta, color_ojos: azules, color_pelo: moreno) **[II.1.1]**

es un elemento.

El ***objeto del aprendizaje*** inductivo puede ser ***un concepto***, ejemplificado positiva o negativamente por cada uno de los elementos del dominio o ***una estructuración del dominio*** en subconjuntos que guarden una relación jerárquica también denominado ***clustering***. Los

conceptos se identifican con subconjuntos del dominio; es decir un concepto está constituido por todos aquellos elementos del dominio que poseen una o unas determinadas propiedades: un paso más allá, un concepto se puede identificar con la función característica del subconjunto de ejemplos que los constituyen. Sea **C** un concepto sobre un dominio **D**, entonces se puede identificar **C** con la *función objetivo*:

$$\chi_{\mathbf{C}}: \mathbf{Domino} \longrightarrow \{0,1\}.$$

Sin embargo, en el caso de que **C** fuese el concepto que se trata de aprender por un sistema que siga un proceso inductivo, por la propia naturaleza de la inducción y por alguna otra restricción añadida que trataremos también más adelante, no está garantizado que se alcance como resultado del aprendizaje esta función. El conocimiento aprendido se puede identificar con una función:

$$\mathbf{h}: \mathbf{Domino} \longrightarrow \{0,1\}$$

que aproxime a la anterior; es decir que:

para la *mayoría* de los d$\in$ Domino se verifique que: $\mathbf{h(d) = \chi_C(d)}$.

- **El espacio de hipótesis**

Como vimos, el conocimiento aprendido aproximará *a su manera* el concepto objetivo del aprendizaje. Precisamente la *manera* o lenguaje en el que la función aprendida venga expresada limitará, la posibilidad de determinar exactamente la función objetivo. Es decir, si denominamos *espacio de hipótesis* a todas las posibles funciones que se puedan representar con el lenguaje disponible, y si ese lenguaje tiene sus limitaciones para hacer el proceso inductivo operativo -como suele ser habitual por necesidad- se tiene que el espacio de las hipótesis consideradas por el sistema no coincidirá con el de todas las hipótesis posibles:

Espacio de hipótesis $\subset$ Espacio de todas las hipótesis posibles

- **Las clases**

Si el elemento anterior **[II.1.1]** perteneciese al conjunto de los elementos del dominio que *verificasen* o *definiesen* un cierto concepto que interesa aprender, para indicar tal hecho necesitaría de un atributo añadido que guardase la información de tal clasificación. Cuando no hay un nombre mejor para ese atributo se le suele denominar ***atributo clase*** o simplemente *clase* y el elemento anterior, con la información que ahora nos interesa, quedaría así:

(estatura: alta, color_ojos: azules, color_pelo: moreno, **clase: +**).

Ello en el supuesto de que el elemento representase al concepto; es decir, disfrutase de la propiedad o propiedades que caracterizan al concepto. Cada elemento del domino en tal circunstancia se denomina ***ejemplo positivo***; en otro caso, el elemento sería un ***ejemplo negativo*** del concepto y su forma sería:

(estatura: alta, color_ojos: azules, color_pelo: moreno, **clase: -**).

Cuando existe este atributo clase en los ejemplos se dice que el aprendizaje es ***supervisado*** ya que al sistema se le indica que el ejemplo es un ejemplo positivo o negativo del concepto que se trata de aprender. En otro caso el aprendizaje es ***no supervisado*** y el objeto del aprendizaje es un ***clustering***.

En realidad el número de dos clases se puede extender a un número finito de ellas, tomando así el problema un cariz más acusado de *aprendizaje de una clasificación* de los elementos, mejor que un problema de aprendizaje de un concepto, tal y como se introdujo para el caso de un atributo clase que fuera binario. Sin embargo, académicamente suele trabajarse con clases binarias para evitar procesos operacionales complejos que no enriquecen conceptualmente el estudio del aprendizaje inductivo. Por otra parte, los elementos suelen denominarse ejemplos en

tanto ejemplifican -positiva o negativamente-el concepto que se desea aprender incluso, y por extensión, también cuando el aprendizaje es no supervisado.

- **La hipótesis del aprendizaje inductivo.**

> ***Cualquier función que aproxime bien a la función objetivo sobre un conjunto de entrenamiento suficientemente grande también la aproximará bien sobre el resto de los ejemplos del dominio.***
>
> [MITCHELL, 97]

Es la hipótesis sobre la que se apoya todo aprendizaje inductivo; quizás su punto más problemático sea la estimación de *lo bien* que se aproxima la función aprendida a la función objetivo, toda vez que no conocemos ésta y que aproximarse muy bien a los ejemplos de entrenamiento, en el sentido pleno de la palabra, generalmente no es la mejor manera de que cubrir bien al resto de los ejemplos.

- **Las entradas y la salida.**

Los ***datos de entrada*** de un sistema basado en el aprendizaje inductivo serán, pues, ejemplos constituidos como vectores de pares *atributo: valor* donde uno de ellos se denomina atributo clase o *atributo decisión*. El conjunto de ejemplos de entrada se suele denominar conjunto de ***ejemplos de entrenamiento***. La estructura del conocimiento que se obtiene como ***datos de salida*** del sistema puede ser diversa: reglas o árboles de decisión, grafos, fórmulas lógicas, funciones, *frames*, etc.

- **El régimen de entrenamiento.**

Según el *régimen del entrenamiento* o aprendizaje; esto es, el tratamiento que haga de los datos de entrada, un sistema puede ser ***no incremental***, cuando aprende de un conjunto de ejemplos de entrenamiento y la posibilidad de reconsiderar el conocimiento aprendido ante la disponibilidad de nuevos ejemplos de entrenamiento consiste solamente en hacer *borrón y cuenta nueva*, es decir, realizar un nuevo aprendizaje con el conjunto de ejemplos inicial más los nuevos ejemplos despreciando el conocimiento de la fase de aprendizaje anterior. Por el contrario un sistema se denominan ***incremental*** si ante la disponibilidad de nuevos ejemplos de entrenamiento el sistema realiza un nuevo proceso de aprendizaje a partir del conocimiento extraído en el primer proceso y lo acomoda a la información nueva que proporcionan los ejemplos añadidos. Dentro de los incrementales se subdividen en sistemas de ***memoria plena***, cuando en cada proceso de aprendizaje el sistema considera todos los ejemplos introducidos hasta el momento más el último conocimiento generado, y los que mantienen en memoria solamente el último conocimiento aprendido y trabajan con él y con los nuevos ejemplos. Estos últimos son los más genuinos como sistemas incrementales.

- **La evaluación del aprendizaje.**

Supóngase ahora que un sistema realizó un proceso de aprendizaje. La siguiente cuestión es la ***evaluación del conocimiento aprendido***. Cuando se dispone de muchos ejemplos una posibilidad muy utilizada es dividir esos ejemplos en dos conjuntos; uno de ellos se utilizará de entrenamiento y sobre el otro, que se suele denominar *conjunto de test,* se comprobará el rendimiento del conocimiento aprendido. Sin embargo, el procedimiento solamente es aceptable cuando se dispone d euna gran cantidad de ejemplos y por eso con el tiempo se ha ido imponiendo una técnica conocida como ***cross validation*** que goza de rigor estadístico como método de evaluación. Consiste este método en dado un número ***n***, menor o igual al de ejemplos disponibles, distribuir estos en ***n*** subconjuntos de igual tamaño y con las clases distribuidas uniformemente en ellos. Se realizarán ***n*** procesos de aprendizaje de forma que en cada uno se utilizarán como conjunto de entrenamiento todos los ejemplos menos los de uno de los n conjuntos. Con el conocimiento aprendido en cada una de las ***n*** veces, se obtienen los errores cometidos en la clasificación de los ejemplos no utilizados en el aprendizaje en esa ocasión y, por último, al finalizar los ***n*** procesos de aprendizaje se obtiene la media de los errores de clasificación cometidos en cada vez. En un intento de apurar más el proceso de evaluación, y

que sea más fiable, se puede repetir el *cross validation* ***m*** veces y calcular la media de las medias de errores cometidos obtenidas durante esas ***m*** veces.

- **El *bias***

Se ha dejado intencionadamente el concepto del ***bias***[1]. para último lugar ya que será más fácil de comprender en su sentido general. Cuando tratemos con sistemas específicos se incidirá siempre este concepto que adquiere connotaciones muy particulares en cada sistema. Es éste un concepto muy importante sin el que los sistemas no podrían aprender. Mitchell lo define así:

> ***Consiste el bias en los requisitos añadidos mínimos y suficientes para justificar como deductivas las inferencias inductivas de un sistema***
>
> [MITCHELL 97]

En realidad podemos considerar el *bias* integrado por una serie de requisitos o *bias* más elementales. Por ejemplo,

2. INDUCCIÓN GRAMATICAL.

Si apelamos a la tesis de Church, se puede decir que todo problema computacional es traducible al reconocimiento de un cierto lenguaje y por tanto clasificable según la jerarquía de Chomsky. Entonces, la Jerarquía de Chomsky se puede utilizar como un modelo de categorización de los problemas computacionales según su complejidad. Con esta observación presente resulta adecuado comenzar el estudio del aprendizaje inductivo introduciendo los sistemas que inducen un lenguaje regular -o un autómata finito- desde una muestra de cadenas del lenguaje que se pretende aprender. Este tipo de inducción forma parte de la teoría conocida como ***inducción gramatical***, cuyo objetivo es la inducción de lenguajes de cualquier categoría a partir de muestras. Sin embargo, el problema de la inducción de lenguajes libres de contexto es un problema sobre el que hasta la fecha hubo avances muy modestos por lo que no lo consideramos materia de estudio en un curso como éste.

- **La inducción gramatical y al aprendizaje inductivo supervisado**

Podría considerarse el estudio de este tipo de aprendizaje dentro del aprendizaje inductivo supervisado, pero se ha preferido introducirlo antes, y por separado, debido a la gran diferencia que se establece entre el enfoque de la inducción autómatas finitos, o lo que es equivalente, de lenguajes regulares, y el del aprendizaje inductivo supervisado que no hace referencia explícita a lenguajes.

Aquí se presentarán sencillos algoritmos que sin estar del todo superados no son realmente representativos de la potencia alcanzada por los más actuales sistemas que utilizan técnicas neuronales o genéticas, que, a pesar de las consabidas críticas sobre el consumo de recursos de los algoritmos basados en estos enfoques, producen buenos resultados. Somos de la opinión de que los consumos son relativos al grado de desarrollo de la tecnología y a las necesidades del usuario, por lo que si las exigencias de esos algoritmos entran dentro de lo aceptable para el usuario, que se supone tiene una estimación de los medios disponibles, son una herramienta más para tener en cuenta.

- **La minimización de autómatas como concepto subyacente.**

Estudiaremos el algoritmo IM1 con diferentes heurísticos que le hacen proporcionar resultados diferentes; en su estudio queda clara la forma de actuar típica de muchos de estos algoritmos de inferencia. Se suelen basar en una minimización suavizada de un autómata en forma de árbol que reconoce a la muestra de partida; es decir, las exigencias a la hora de *juntar* estados para obtener un autómata cociente son menores que en una minimización normal, por lo que el

[1] Se mantiene sin traducir porque su traducción, *sesgo*, tiene un significado que podría confundir en la comprensión del concepto.

lenguaje inducido tiende a resultar más general que el constituido por la propia muestra, que es el lenguaje reconocido por el autómata-árbol de partida. Los heurísticos no son más que la función característica que determina la unión de dos estados.

Se utilizará la definición de lenguaje regular basada en los autómatas finitos, equivalente, como se sabe, a las definiciones fundamentadas en el concepto de gramática regular y en el concepto de expresión regular respectivamente.

El objetivo es el estudio de técnicas algorítmicas que permitan construir autómatas finitos partiendo de la información que proporciona la muestra y en algunos casos de información añadida: valores de parámetros, ejemplos negativos, etc. Estas técnicas se basan en lo que se conoce como ***inducción gramatical***. Una utilidad final puede ser la creación de estructuras de control para sistemas basados en reglas.

Se denotará por $\mathbf{S^+}$ a la ***muestra positiva*** o conjunto de ejemplos positivos**;** es decir, cadenas que deben pertenecer al lenguaje objetivo de la inducción..
Se denotará por $\mathbf{S^-}$ a la ***muestra negativa*** o conjunto de ejemplos negativos **;** es decir, cadenas que no deben pertenecer al lenguaje objetivo de la inducción.

Ejemplo II.2.2. Si se parte de una muestra positiva $\mathbf{S^+ = \{aabb, ab, aab, abbb\}}$, el problema que nos ocupa consiste en inducir la gramática o autómata que genere un lenguaje regular englobando la muestra y para el que la muestra resulte *representativa*.
En este caso el lenguaje podría ser $\mathbf{a^+\ b^+}$, o bien, $\mathbf{aa^*\ bb^*}$[2].. La solución no tiene por qué ser única.

Se expondrán diferentes algoritmos que infieren un lenguaje regular a partir de una muestra.

2.1 COMPENDIO DE CONCEPTOS Y DEFINICIONES

- **Definiciones generales**

Sea Σ un alfabeto, el conjunto de cadenas de cualquier longitud que pueden formarse por concatenación de elementos del alfabeto Σ, incluida λ, la cadena vacía, se representa por Σ^*.

Un **lenguaje formal**, o simplemente lenguaje, es un subconjunto de Σ^*

Conjunto de **prefijos de un lenguaje L**: $P_r(L) = \{ u : \exists v \in \Sigma^* \wedge uv \in L \}$

Si $u \in Pr(L)$ el conjunto de **sufijos de la cadena u** es: $T_L(u) = \{ v \in \Sigma^* : uv \in L \}$

Conjunto de **sufijos de un lenguaje L**: $\cup(T_L(u): u \in Pr(L))$

Conjunto de **sufijos de longitud k de una cadena u**: $T_L^k(u) = \{ v : uv \in L \wedge |v| \leq k \}$

S^+ es una **muestra positiva** de un lenguaje L si $S^+ \subset L$
S^- es una **muestra negativa** de un lenguaje L si $S^- \cap L = \varnothing$

Una muestra positiva S^+ es una **muestra característica** es de un cierto lenguaje L de una clase de lenguajes ζ, si éste es el menor lenguaje de ζ que contiene a S^+.

(2) El símbolo '*' indica que ese carácter puede aparecer de 0 a n veces. El símbolo '+' indica que ese carácter debe aparecer al menos una vez.

Dado un conjunto S, una **partición de S**, que denotaremos por Π_S, es una colección de conjuntos $\Pi_S = \{ S_i \}_{i \in I}$, tal que:

$$\text{i)}\ \cup(S_i: i \in I) = S \qquad \text{y} \qquad \text{ii)}\ S_r \cap S_s = \varnothing\ ,\ \forall\ r,s \in I,\ r \neq s$$

Para cada $s \in S$, el **bloque o la clase de s relativa a la partición Π_S** denotada por es B(s, Π_S), consiste en el S_i, $i \in I$, tal que $s \in S_i$.

Recuérdese que toda partición Π_s sobre S induce una relación de equivalencia R sobre S, y viceversa, en donde los bloques o clases de la partición coinciden con las clases de equivalencia de la relación inducida.

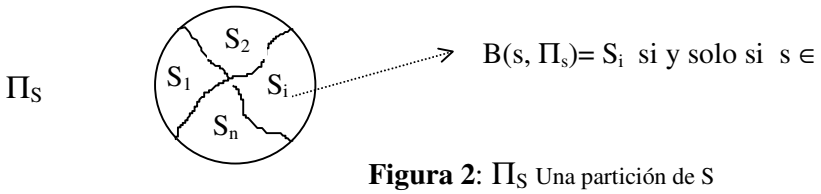


Figura 2: Π_S Una partición de S

La partición $\Pi_S = \{S_i\}_{i \in I}$ de S es un **refinamiento** de la partición $\Pi_S' = \{S_j'\}_{j \in J}$, también de S, si y sólo si, $\forall\ S' \in \Pi_S'$, $S' = \cup(S_h: h \in H)$ y $H \subset I$; es decir, cada bloque de Π_S' coincide con una partición de bloques de Π_S.

Figura 3::Una partición y un refinamiento suyo.

Un autómata finito determinístico/no determinístico que denotaremos por A, consiste en una quíntupla A= (Q, Σ, δ, I, F), donde,

Q representa el ***conjunto de estados*** y es finito
Σ representa el *alfabeto*.
δ representa la ***función de transición***

Si el autómata finito es determinístico (AFD):

$$\delta : Q \times \Sigma \longrightarrow Q$$
$$(q\ ,\ x) \rightsquigarrow \delta\ (q,\ x) \in Q$$

Si el autómata finito es no determinístico (AFND):

$$\delta : Q \times \Sigma \longrightarrow P(Q)^{(3)}$$
$$(q\ ,\ x) \rightsquigarrow \delta\ (q,\ x) = \{...,q_i,...\} \in P(Q)$$

I representa al **e*stado inicial*,** en ocasiones es un conjunto de estados iniciales
F representa el conjunto de ***estados finales o aceptores***

(3) Partes de Q.

Función de transición generalizada, δ^*:

Autómata Finito Determinístico (AFD): $\delta^* : Q \times \Sigma^* \longrightarrow Q$

i) $\delta^*(q, \lambda) = q$

ii) $\delta^*(q, wx) = \delta (\delta^*(q, w), x)$.

Es decir, dada una entrada en un cierto estado q, se pasa a un único estado.

Autómata Finito No Determinístico (AFND): $\delta^* : P(Q) \times \Sigma^* \rightarrow P(Q)$

i) $\delta^*(\{q\}, \lambda) = \{q\}$

ii) $\delta^*(\{q\}, wx) = \cup \delta (q', x) \; q' \in \delta^* (\{q\},w)$

Transiciones características de un AFD y de un AFND

Es decir, dada una entrada en un cierto estado q, se puede pasar a distintos estados.

Para todo AFND se puede construir un AFD, de tal forma que el lenguaje reconocido por ambos autómatas coincida.

Una cadena ***w es reconocida o aceptada por un autómata finito*** si $\delta^*(q,w) \in F$ en un AFD ó $\delta^*(q,w) \cap F \neq \varnothing$ en un AFND.

A los autómatas finitos se les suele denominar también ***aceptores*** o ***reconocedores***.

Representaremos como ***L(A) al lenguaje aceptado o reconocido por el autómata A***.

Definición II.2.3. Un ***lenguaje es regular*** si existe un autómata finito que lo reconoce.

En términos informales la equivalencia de definiciones mencionada podría enunciarse así:

Para todo autómata finito existe una gramática regular y una expresión regular tales que el lenguaje generado por la gramática y el representado por la expresión regular coincide con el lenguaje reconocido por el autómata finito.

Autómata aceptor en forma de árbol de prefijos, o simplemente ***árbol de prefijos***, aceptor de una muestra positiva S^+, $PT(S^+)$, es un AFD definido por la quíntupla $(Q, \Sigma, \delta, I, F)$, donde:

$Q \rightarrow$ Conjunto de Estados = $P_r(S^+)$

$\Sigma \rightarrow$ Alfabeto.

$\delta \rightarrow$ Aplicación de la forma $\delta : Q \times \Sigma \longrightarrow Q$

$$u, x \rightsquigarrow \delta(u, x) = \begin{cases} ux & \text{si } ux \in P_r(S^+) \\ \perp \text{ (indefinida)} & \text{en otro caso} \end{cases}$$

$I \rightarrow$ Estado inicial = λ (la cadena vacía siempre es un prefijo)

$F \rightarrow$ Conjunto de estados finales = S^+

Ejemplo II.2.4. Sea una muestra positiva $S^+ = \{ab, aab, abbb\}$. El árbol de prefijos $PT(S^+) = (Q, \Sigma, \delta, I, F)$, está determinado por:$\Sigma = \{a,b\}$, para cada cadena de S^+ se señalan, mediante subrayados, sus prefijos:

ab: $\underline{\lambda}ab$, $\underline{a}b$, $\underline{ab}\lambda$, aab: $\underline{\lambda}aab$, $\underline{a}ab$, $\underline{aa}b$, $\underline{aab}\lambda$, abbb: $\underline{\lambda}abbb$, $\underline{a}bbb$, $\underline{ab}bb$, $\underline{abb}b$, $\underline{abbb}\lambda$.

Luego:

$Q = P_r(S^+) = \{ \lambda, a, ab, aa, aab, abb, abbb \}$(7 estados)

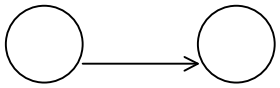
$\delta : Q \times \Sigma \longrightarrow Q$

$(u , x) \rightsquigarrow \delta(u, x) = ux$ si $ux \in P_r(S^+)$

$\delta(\lambda, a) = a$ pues $a \in P_r(S^+)$

$\delta(\lambda, b) = \perp$ pues $b \notin P_r(S^+)$

Es decir, como 'a'$\in P_r(S^+)$ entonces está definido un arco en el autómata de la forma:



Como 'b'$\notin P_r(S^+)$ entonces no está definido ese arco en el autómata.

$I = \lambda$

$F = S^+ = \{ab, aab, abbb\}$.

El grafo del aceptor resultantes se puede ver en la **figura 4.**

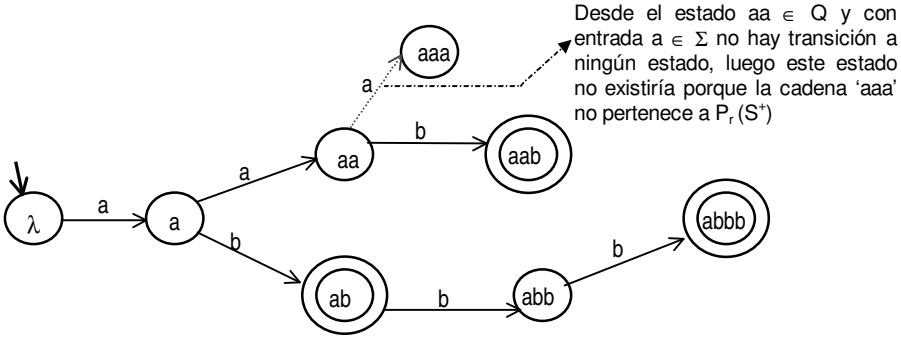


Figura 4:Árbol de prefijos para la muestra S^+.

El ***autómata cociente*** determinado por la partición Π del conjunto de estados Q del AFD A consiste en el siguiente autómata finito que puede resultar no determinístico, es denotado por A/Π y se obtiene a partir de A:

$$A/\Pi = (Q/\Pi, \Sigma, \delta_\Pi, I_\Pi, F_\Pi)$$

Alfabeto: Σ

Conjunto de estados: Q/Π

Aplicación transición:: $\delta_\Pi : Q/\Pi \times \Sigma \longrightarrow Q/\Pi$

Si $B1 \in Q/\Pi$, es decir, un bloque de la partición Π en Q,

entonces $\delta'(B1, x) = B2$ si y solo si $\exists q \in B1$, tal que, $\delta(q, x) = q' \in B2$

Estado inicial: I_Π, clase correspondiente al bloque $B(I,\Pi)$; es decir, bloque en el que está el antiguo estado inicial.

Conjunto de estados finales: $F_\Pi = \{B \in Q/\Pi: B \cap F \neq \varnothing\}$; es decir, los estados finales serán aquellos bloques que contengan algunos de los antiguos estados finales, pudiendo contener otros estados que antes no eran finales.

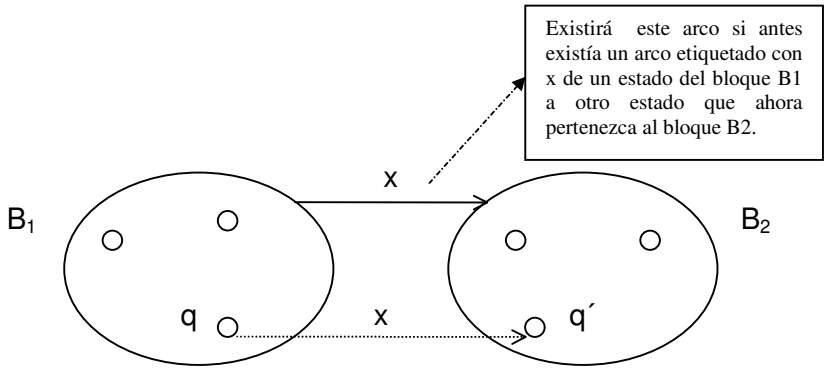


Figura 5: Transición entre bloques.

El autómata mínimo obtenido de un AFD, es un caso particular de autómata cociente.

Sea M = (Q, ∑, δ, I, F) un autómata finito determinístico, el ***autómata mínimo*** que reconoce el mismo lenguaje que M será el autómata cociente: M´ = M/Π = (Q/Π, ∑, δ_Π, I_Π, F_Π) donde Π está definida como sigue:

$\forall\, q_i, q_j \in Q$: $B(q_i, \Pi) = B(q_j, \Pi)$ si y solo si $(\delta^*(q_i, w) \in F \Leftrightarrow \delta^*(q_j, w) \in F)$, $\forall w \in \Sigma^*$

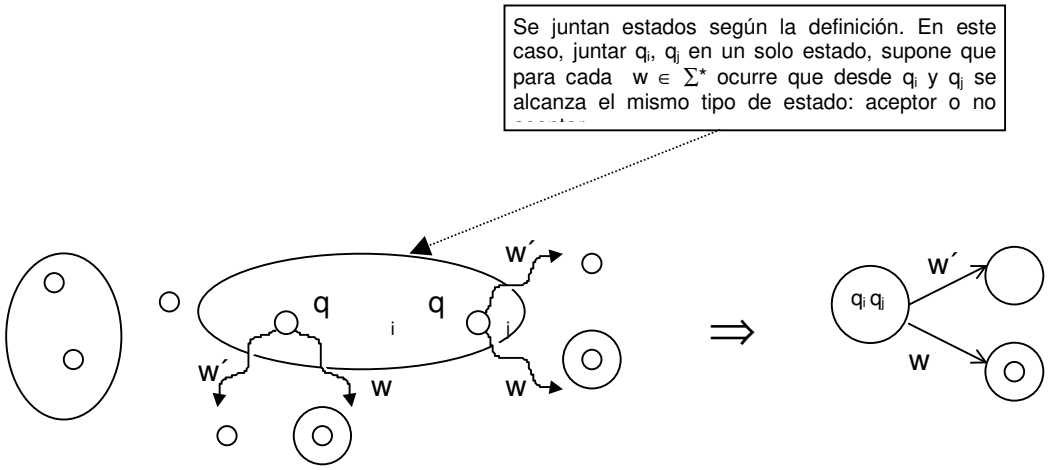


Figura 6: Unión de estados en la minimización.

Un subaceptor de un autómata finito A es otro autómata A' tal que L(A´) ⊂ L(A); es decir, el lenguaje reconocido por el también autómata finito A´, L(A´), está contenido en el lenguaje reconocido por A, L(A).

Sea A = (Q, ∑, δ, I, F) un AFD**,** ***A^r, el autómata invertido de A*****,** es el AFND definido como sigue: A^r = (Q, ∑, δ^r, I´, F´) donde:

$\delta^r \rightarrow$ Aplicación de la forma $\delta^r: Q \longrightarrow P(Q)$

$(q,x) \qquad \delta^r(q,x) = \{q' \in Q \,/\, \delta(q',x) = q\}$

Conjunto de estados iniciales: I´= F, (aunque se demuestra que para un AFND con varios estados iniciales siempre existe un autómata equivalente con un solo estado inicial)

Conjunto de estados finales: F´= { I }

Figura 7: Relación entre las transiciones en A y A^r.

- **Definiciones más específicas de la inducción gramatical**

La ***partición*** Π_L sobre el conjunto de prefijos del lenguaje L se define como sigue:

$\forall\ u,v \in P_r(L)\quad B(u, \Pi_L) = B(v, \Pi_L) \Leftrightarrow T_L(u) = T_L(v)$

Dado un lenguaje L, se define la ***partición*** $\Pi_{Pr(S^+)}$ ***originada por*** Π_L sobre el conjunto de prefijos de una muestra positiva S^+ como la restricción de Π_L al conjunto de prefijos de la muestra positiva S^+; es decir:

$\forall\ u,v \in P_r(S^+)\quad B(u, \prod_{Pr(S^+)}) = B(v, \prod_{Pr(S^+)}) \Leftrightarrow T_L(u) = T_L(v)$

Todos los algoritmos de inducción que parten del árbol de prefijos aceptor se basan indirectamente en la siguiente proposición.

Proposición II.2.5. Sea L un lenguaje, S^+ una muestra positiva de L, A_0 el árbol de prefijos aceptor de S^+, $A_0 = PT(S^+)$, y $\prod_{Pr(S^+)}$ la restricción de Π_L a $P_r(S^+)$ entonces $A_0 / \prod_{Pr(S^+)}$ es isomorfo a un subaceptor de L; es decir, el lenguaje aceptado por ese autómata está contenido en L: $L(A_0 / \prod_{Pr(S^+)}) \subset L$.

Por la anterior proposición se tiene garantizado que si se conoce $\prod_{Pr(s+)}$ se podrá conocer un lenguaje contenido en el lenguaje L objetivo de la inducción. Gold demostró que cualquier partición supuesta en los estados de A_0 se podrá identificar con la $\prod_{Pr(S^+)}$generada por un cierto lenguaje L. Así, se tiene que cualquier partición de A_0 serviría como medio para inducir un subconjunto de un cierto lenguaje L que, obviamente, dependerá de la partición elegida. Este lenguaje L será el objetivo de la inducción y que no está garantizado que se alcance. Esto también aclara que las soluciones o resultados de la inducción puedan ser múltiples y dispares. Obsérvese que, generalmente, no se induce L sino un lenguaje contenido en L y que por tanto lo aproxima.

2.2. ALGORITMOS

2.2.1 El algoritmo IM1

Dada una muestra positiva S^+, este algoritmo calcula un autómata cociente $A_0 / \prod_{Pr(S^+)}$ del árbol de prefijos de la muestra, A_0, relativo a una partición de los estados, $Pr(S^+)$,determinada por un heurístico χ de forma que:

$B (u, \prod_{Pr(S^+)}) = B (v, \prod_{Pr(S^+)})$, si y sólo si, $\chi(u,v) = 1$.

En los siguientes subapartados se verán varios heurísticos para definir distintas funciones características y por tanto diferentes particiones sobre $PT(S^+)$.

- **Pseudocódigo**

Entradas: una muestra positiva no vacía, S^+, y un parámetro k para el heurístico.
Salida: $A_0 / \prod_{Pr(s+),}$ autómata finito reconocedor de un cierto lenguaje que contiene a S^+.

Procedimiento **INICIALIZAR**
Comienzo
A_0: = (Q_0, Σ, δ_0, I_0, F_0) = $PT(S^+)$ /*árbol de prefijos de S^+*/
$\prod_0$ /*Partición trivial en Q_0. Cada elemento de Q_0 es un elemento de la partición, cada conjunto contiene un solo elemento*/
i:= 0
Fin

Procedimiento **JUNTAR**
Comienzo
Para Cada (u,v) $\in$ Q_0 x Q_0 **hacer**
Si χ(u,v) =1 **entonces**
B1 = B (u, $\prod_i$)
B2 = B (v, $\prod_i$)
$\prod_i' = \prod_i$ - B1 - B2
$\prod_{i+1} = \prod_i'$ + (B1 $\equiv$ B2)
i = i + 1
fin_si
fin_para
f = i
retorna $\prod_f$ (= Q_i / $\prod_i$)
Fin.

- **Heurísticos**

De los K-sufijos o de Biermann y Feldman

χ(u,v) =1, si y sólo si, $T^k_{S^+}(u) = T^k_{S^+}(v)$ k = 0, 1, 2, ...

Una cota superior de k podría ser la longitud de la palabra más larga.

De Levine

χ(u,v) =1, si y sólo si, *stren*(u,v) $\geq$ **s**,

donde **s** es un valor de la imagen de la función *stren* y

$$stren(u,v) = \max_{i\ 0 \leq i \leq k} \left[\frac{2\left|T^i_{S^+}(u) \cap T^i_{S^+}(v)\right|}{\left|T^i_{S^+}(u)\right| + \left|T^i_{S^+}(v)\right|} \right]$$

De Miclet

Heurístico general que, con la particularización que emplea Miclet, equivale al heurístico de Levine cuando se elige como **s** el menor valor no nulo de la tabla de la función *stren*.

- **Algoritmos que induce sobre una subclase de los lenguajes regulares.**

Se presentan seguidamente dos algoritmos que inducen un lenguaje regular dentro de una subclase más especifica; en el primer caso dentro de la clase de los lenguajes k-reversibles y en el segundo de los k-contextuales.

2.2.2. Algoritmo KR

El algoritmo KR induce el menor autómata K-reversible que reconoce a la muestra; o también, el menor lenguaje K-reversible que contiene a la muestra.

- **Lenguajes k-reversibles**

Definición II.2.6. Un autómata A es ***K-reversible*** si A es determinístico y A^r es determinístico en exploración K-ulterior.

Definición II.2.7. Un autómata A es ***determinístico en exploración k-ulterior*** si para dos estados q_1, q_2 que sean o bien iniciales, o bien que exista un q_3 y una $x \in \Sigma$ con $\delta(q_3,x)= q_1$ y $\delta(q_3,x)= q_2$, entonces no existe una cadena $u \in \Sigma^*$, de longitud k, saliente de ellos. Es decir, que no se den ninguna de las situaciones que se muestran en la **figura 8.**

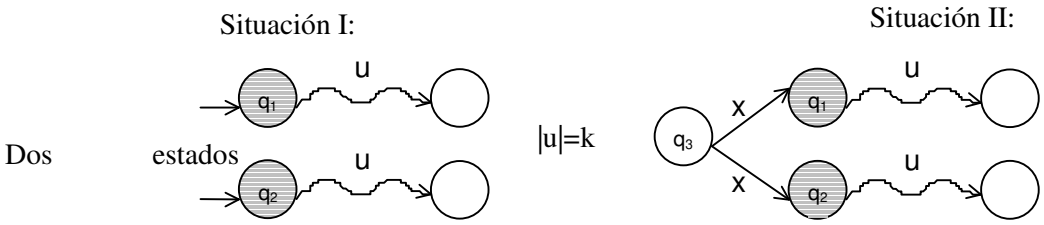


Figura 8: Esquemas de estados que transgreden la k-ulterioridad

Las exigencias de la k-reversibilidad se traducen sobre un cierto autómata en que no se den ninguna de las situaciones reflejadas en los tres esquemas de estados de la **figura 9,** que reflejan el determinísmo del autómata tratado y la k-ulterioridad de su autómata invertido.

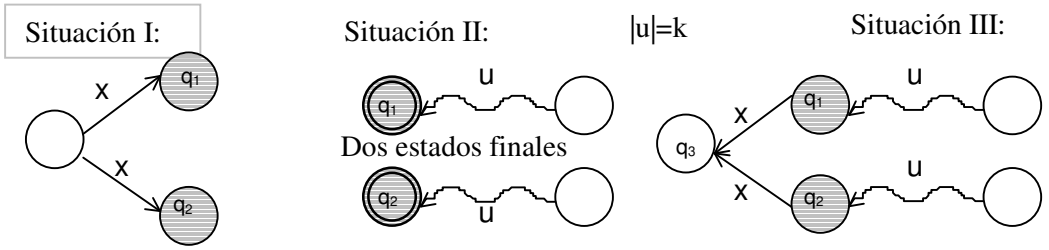


Figura 9: Esquemas de estados que transgreden la k-reversibilidad

Angluin propuso un algoritmo de complejidad $\theta(n^3)$ basado en la eliminación de las condiciones que transgreden la k-reversibilidad, a base de unir estados, para inducir el menor autómata que reconoce a la muestra. Muggleton mediante un nuevo concepto de lenguaje consiguió inducir el mismo lenguaje rebajando la complejidad a $\theta(n^2)$. Seguidamente introducimos dicho algoritmo denominado KR pero previamente definimos algunos conceptos necesarios.

- **Lenguajes KU-reversibles**

Definición II.2.8. Un autómata finito se denomina ***aceptor τ-terminal*** si existe un símbolo $\tau \in \Sigma$ tal que: $\delta(q,x) \in F$, si y sólo si, $x = \tau$. Es decir que todos los arcos incidentes en estados terminales están etiquetados con τ.

Definición II.2.9. Un autómata finito se denomina ***aceptor de estado objetivo*** si posee un único estado final y además de él no parte ninguna transición ni bucle.

Definición II.2.10. Un autómata finito se dice que es un ***aceptor destacadamente terminal*** si es τ-terminal y aceptor de estado objetivo. Es decir, que sólo tiene un estado final (sin bucles) y todas las flechas hacia él están etiquetadas con el mismo símbolo τ.

Existe una biyección h_τ natural que asocia a un aceptor A un aceptor destacadamente terminal $h_\tau(A)$, tal que si L es el lenguaje reconocido por A entonces L $(h_\tau(A)) = L \bullet \{ \tau \}$. Se supone que τ no pertenece a L.

Definición II.2.11. Un autómata finito A se dice que es un ***aceptor KU-reversible*** si es destacadamente terminal y $h_\tau^{-1}(A)$ es k-reversible. Un ***lenguaje es KU-reversible*** si existe un autómata finito KU-reversible que lo reconoce.

El proceder del algortimo KR es ir eliminando sucesivamente las condiciones que transgreden la k-reversibilidad sobre el autómata destacadamente terminal que le corresponde en la biyección natural citada, al árbol de prefijos que reconoce a la muestra de partida. Puesto que solamente hay un estado final las situaciones que se deben detectar y eliminar son las del tipo I y III de la **figura 9**. La eliminación se realiza uniendo los estados q_1 y q_2 de los esquemas.. El resultado final es normalizado para conseguir un autómata K-reversible usando h_τ^{-1}.

- **Pasos del algoritmo**

  a) Partiendo de S^+ creación de S_0^+. Consiste en concatenar τ a todos los ejemplos de la muestra como sufijo: $S_0^+ = S^+ \cdot \{\tau\}$ y por tanto $\Sigma_0 = \Sigma \cup \{\tau\}$, será el nuevo alfabeto.
  b) Construcción del árbol de prefijos aceptor $PT(S_0^+)$
  c) Aunar estados finales. $h_\tau (PT (S_0^+))$

  Los tres primeros pasos corresponden a la conversión del autómata en destacadamente terminal.

  d) Cálculo iterativo del autómata cociente KU-reversible según el heurístico de unión de estados χ: $\chi(u,v) = 1$, sii, u, v transgreden las condiciones de KU-reversibilidad, en las sucesivas iteraciones.

- **Pseudocódigo**

Entradas: S^+, muestra positiva no vacía; el valor del parámetro k.
Salida: A, el menor autómata finito K-reversible que reconoce la muestra.

Procedimiento **INICIALIZAR**

Comenzar

$S_0^+ := S^+ \cdot \{\tau\}$
$A_0 := (Q_0, \Sigma_0, \delta_0, I_0, F_0) = PT(S_0^+)$ → árbol de prefijos de S_0^+
$\prod_{0:}$:= Partición trivial en Q_0. Cada elemento de Q_0 es un elemento de la partición, cada conjunto contiene un solo elemento.
LISTA: = vacio
Para cada ($b \in \Sigma_0$) y ($q \in Q_0$) **hacer**
s ({q}, b) = δ_0 (q, b)
p ({q}, b) = δ_0 (q, b)
fin para

Elegir algún $q' \in F_0$
Para cada $q \in F_0 - \{q'\}$ **hacer**
InsertarPar(LISTA, (q´, q))
fin para
i = 0
Fin.

Procedimiento **JUNTAR**
Comenzar
Mientras LISTA ≠ vacio **hacer**
borrar elemento (q_1, q_2) de LISTA
$B1 = B(q_1, \prod_i)$
$B2 = B(q_2, \prod_i)$
Si $B1 \neq B2$ **entonces**
$B3 = B1 \equiv B2$
$\prod_{i+1} = \prod_i - B1 - B2 + B3$
Para cada $b \in \sum_0$ **hacer**
ActualizarFuncion_S (B1,B2, B3, b)
ActualizarFuncion_P (B1,B2, B3, b, k)
fin para
i = i + 1
fin si
fin mientras

f = i
retorna $h_\tau^{-1}(A_0 / \prod_f)$
Fin.

Entradas: B1,B2, B3 bloques o clases de estados; $b \in \sum_0$ un símbolo del alfabeto ampiado.
Procedimiento **Actualizar_Funcion_S**
Comienzo
Si (s (B1, b) AND s (B2, b)) ≠ nulo **entonces**
InsertarPar (LISTA, (s (B1, b), s (B2, b)))
finsi
Si (B1, b) ≠ nulo **entonces** s (B3, b) = s (B1, b)
sino s (B3, b) = s (B2, b)
finsi
Fin

Entradas: B1,B2, B3 bloques o clases de estados; b∈ $\sum_0$ un símbolo del alfabeto ampliado; k parámetro.
Procedimiento **Actualizar_Funcion_P**
Comenzar
Para cada $q_1 \in p(B1,b)$ y $q_2 \in p(B2,b)$ **hacer**
Si PrefijoComun $(q_1, q_2, \prod_i, k)$ **entonces**
InsertarPar(LISTA , (q_1, q_2))
fin_si
fin_para
$p(B3, b) = p(B1, b) \cup p(B2, b)$
Fin

Entradas: q_1 , q_2 estados; $\prod_i$ partición actual.
Salida: valor booleano.
Función **Prefijo_Comun**
Comenzar
$q_1 \rightarrow B(u_1v_1, \prod_i)$
$q_2 \rightarrow B(u_2v_2, \prod_i)$ con $u_1v_1 , u_2v_2 \in Pr(S^+)$
Si $(|v_1| = k = |v_2|)$ AND $(v_1 = v_2)$ **entonces** retorna TRUE **si no** retorna FALSE
fin_si
Fin

- **Versión incremental**

Sea S^+ una muestra positiva y A el autómata obtenido como salida del algoritmo KR para esa muestra. Para la obtención del autómata inducido por KR para $S^+ \cup \{w\}$, es decir, con una nueva cadena añadida, se realizan los siguientes pasos.

- **Pasos del algoritmo:**

a) Comprobar si w es reconocida por A, en caso afirmativo, parar. En otro caso seguir.
b) Construir un camino desde el estado inicial hasta un estado final nuevo reconociendo w y aprovechando en el primer tramo el camino que hubiese en A.
c) Ejecutar KR sobre el autómata resultante del paso b.

- **Pseudocódigo**

Entrada: w Cadena.
Salida → A Autómata finito KU-reversible.

Procedimiento **INICIALIZAR**
Comenzar
$w_0 = w \,.\, \{\tau\}$
$S_0^+ = S_0^+ \cup \{w_0\}$
Fin

Si_ ReconocidaCadena(w_0 **) entonces** FIN
si no
Para cada b∈ w_0 y partiendo de la primera transición indefinida
añadir nuevos estados y transiciones
fin_para
Marcar el último estado al que se llegó como final
InsertarPar (LISTA, (estado, estadofinal de A))
Aplicar KR
fin_si

Entradas: w∈ $\sum_0^*$ cadena sobre el alfabeto expandido.
Salida: valor booleano.
Función **Reconocida_Cadena**
Comenzar

Reconocida = FALSE
E = I de A
b = PrimerSimbolo de w
Mientras (b ≠ FinCadena) AND (δ(E,b) ≠ nulo) **hacer**
b = SiguienteSimbolo de w
Si b ≠ FinCadena entonces E = δ(E,b) **fin_si**
fin_mientras
Si (b = FinCadena) AND (δ(E,b) = EstadoFinal)
entonces Reconocida = TRUE
fin_si
retorna Reconocida
Fin.

- **Versión con ejemplos negativos**

Partiendo de una muestra de ejemplos positivos S^+ y de otra muestra de ejemplos negativos S^-los pasos a seguir son los siguientes.

- **Forma de operar del algoritmo**

a) $0 \le K < \max_i \{\text{longitud de cadena}_i \in S^+\}$
b) Aplicar el algoritmo KR infiriendo lenguajes para K = 0 ... n.
c) Parar cuando se encuentre un lenguaje (L_k) que contenga todas las cadenas de la muestra positiva y ninguna de las cadenas de la muestra negativa. En otro caso devolver: "No hay solución"

- **Pseudocódigo**

Entradas: S^-, S^+ muestras de ejemplos.

Procedimiento **INICIALIZAR**
Comenzar
Calcular longitudes de las cadenas de S^+
k:= 0
maxK:= max { longitud de las cadenas de S^+}
Continuar = TRUE
Fin.

??????????

Mientras (k ≤ maxK) AND (Continuar = TRUE) **hacer**
Aplicar KR usando S^+
NoReconocidas = TRUE
Para cada w ∈ S^- **hacer**
NoReconocidas = NoReconocidas AND ReconocidaCadena(w)
fin_para
Si NoReconocidas = TRUE **entonces** Continuar = FALSE **fin_si**
k = k + 1
fin_mientras

Si Continuar = FALSE **entonces** Encontrada la solución para k = k -1
sino FALLO No se encontró solución para ese rango de k
fin_si

2.2.3. Algoritmo KC

- **Lenguajes k-contextuales**

Definición II.2.12. Sea L un lenguaje regular, L será ***k-contextual*** si para cualquiera cadenas u_1vw_1 , $u_2vw_2 \in L$, con $|v| = k$, se verifica que $T_L(u_1v) = T_L(u_2v)$.

Un AF, A, se denominará k-contextual si el lenguaje que acepta es k-contextual.

El algoritmo KC induce el menor lenguaje k-contextual que reconoce a la muestra. Su principio de funcionamiento es similar al del KR. Realiza sucesivas uniones de los estados que transgreden la k-contextualidad. La eliminación de una de esas situaciones puede conllevar la aparición de otras. Finaliza cuando no existan estados en esas condiciones.

- **Función que determina la unión de estados**

Se define χ , la función que determina la unión de estados a partir del árbol de prefijos, como sigue:

$$\chi(u,v) = \begin{cases} 1, \text{ si y sólo si, } \exists\, w, \ |w| = k, \text{ tal que} \begin{cases} u = u_1\, w \\ v = v_1\, w \end{cases} \\ 0, \text{ en otro caso} \end{cases}$$

Obsérvese que $\chi(u, _) = 0$ siempre que $|u| < k$. Con esta definición de la función que determinan la unión de estados se eliminan las situaciones indeseables si las hubiera, y en otro caso se minimiza el autómata en lo referente a los estados que se unen sin realizar inducción alguna.

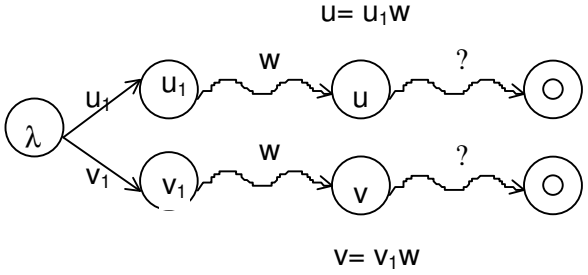


Figura 10: Situaciones en A_0 que transgreden la k-contextualidad

- **Pseudocódigo**

Entradas: S^+, muestra positiva no vacía y k el parámetro que fija el valor k de la k-contextualidad.

Salida: Autómata aceptor del menor lenguaje k-contextual que reconoce a la muestra.

Procedimiento **INICIALIZAR**
Comienzo
$A_0 = (Q_0, \Sigma, \delta_0, I_0, F_0)$ = PT(S^+ /*árbol de prefijos de $S^{+*/}$
$\prod_0 = \{ \{u\}: u \in Q_0 , |u| < k \}$
$Q'_0 = Q_0 - \cup (q: q \in \prod_0)$
i= 0
Fin.

Procedimiento **JUNTAR**
Comienzo
Para cada estado $u_1v \in Q'_0$ **hacer**
Si |v|= k **entonces**
Si existe algún bloque B1 tal que B1= B(u_2v, $\prod_i$)
entonces B2 = B1 $\cup$ {u_1v}
sino B2 = {u_1v}
fin_si
$\prod_{i+1} = \prod_i$ - B1 + B2
i= i + 1
fin_si
fin_para
f= i
retorna $A_0 / \prod_f$
Fin.

- **Versión incremental**

A semejanza de como se realizaba en el algoritmo KR.

- **Versión con ejemplos negativos**

A semejanza de como se realizaba en el algoritmo KR.

2.2.3. Algoritmos SKR y SKC

Ninguno de los algoritmos anteriores puede identificar un lenguaje regulare partiendo solamente de una muestra de ejemplos positivos sin ninguan restricción o información más. Por eso se usan algoritmos que utilizan una muestra positiva y otra información adicional. Normalmente esta información es suficiente para permitir una identificación en el límite. Esta información adicional puede ser el valor k, los ejemplos negativos etc., según se vio en los algoritmos anteriores.

En este apartado veremos un nuevo tipo de información adicional que se denomina *información semántica* y se añade a los ejemplos positivos utilizando pares de la forma *situación/acción*.

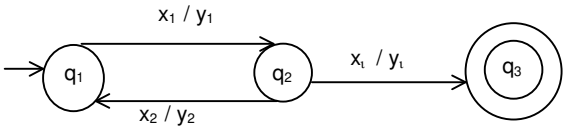
- **Definiciones y resultados básicos**

Definición II.2.13. Una ***máquina de Mealy*** es una sextupla (Q, X, Y, δ, I, F), donde:
Q es un conjunto finito de estados.
X es un conjunto de situaciones.
Y es un conjunto de acciones.
δ es una función de transición definida como sigue:

$$\delta : Q \times X \longrightarrow Y \times Q$$
$$(q , x) \rightsquigarrow \delta (q , x)= (y , q')$$

I es un conjunto de estados iniciales.
F es un conjunto de estados finales.

Su representación gráfica puede verse en la **figura 11**.



$Q = \{q_1 , q_2 , q_3\}$; $X = \{x_1 , x_2 , x_t\}$;
$Y = \{ y_1 , y_2 , y_t\}$
$\delta(q_1 ,x_1) = \{(y_1, q_2)\}$
$\delta(q_2 ,x_2) = \{(y_2, q_1)\}$
$\delta(q_2 ,x_t) = \{(y_t , q_3)\}$
en otro caso $\delta(q , x) = 0$.
$I = \{ q_1 \}$
$F= \{ q_3 \}$

Figura 11: Representación de una máquina de Mealy

Definición II.2.13. Una máquina de Mealy es ***de estado objetivo*** si sólo tiene un estado final.

Definición II.2.14. Una máquina de Mealy es ***τ-terminal*** si sus estados terminales no tienen salidas y todos los arcos que acceden a ellos están etiquetados por la pareja especial x_t / y_t .

Definición II.2.15. Una máquina de Mealy es ***destacadamente terminal*** si es a la vez terminal y τ-terminal.

Definición II.2.16. Una máquina de Mealy es ***determinista s***i verifica que:

1) $\forall q \in Q$ y $\forall x \in X$, si $\delta(q, x)=(y, q')$ entonces no existe otros $q'' \in Q$ e $y' \in Y$ tales que $\delta(q, x)=(y', q'')$
2) $|I| = 1$, es decir, el conjunto de estados iniciales contiene un solo miembro.

,

Definición II.2.17. El ***universo de pares situación/accción***, Σ_{sa}, consiste en le siguiente conjunto*:* $\Sigma_{sa} = (X \times Y)$, siendo X el universo de símbolos de situación e Y el universo de símbolos de acción. Un ***lenguaje situación/acción, L_{sa},*** es un conjunto de sucesiones o cadenas formadas por pares situación/acción: $L_{sa} \subset \Sigma^*_{sa}$.

Definición II.2.18. Una ***muestra positiva***, S_{sa}^+, de un lenguaje situación/acción L_{sa} consiste en un subconjunto de L_{sa}.

Definición II.2.19. Una máquina de Mealy ***terminal*** se dice que genera la sucesión de pares (x_1, y_1) (x_2, y_2) ..... (x_n , y_n), si y sólo si, existe una sucesión de estados, no necesariamente distintos, $q_0, q_1, q_2, ..., qn$, tales que:

$$(y_{i+1}, q_{i+1}) \in \delta(q_i,x_{i+1}) \text{ con } 0 \le i \le (n-1), q_0 \in I \text{ y } q_n \in F$$

Lema II.2.20. Existe una biyección h_b entre pares (situación , acción) $\in \sum_{sa} = (X \times Y)$, y símbolos del alfabeto $\sum = X \bullet Y$[4]

$$h_b : \sum_{sa} \longleftrightarrow \sum$$
$$(s,a) \leftrightsquigarrow s \bullet a$$

Lema II.2.21. Dada la biyección anterior se puede establecer una biyección, h_a, entre máquinas Mealy terminales y autómatas de estados finitos.

Ejemplo II.2.22.

Maquina Mealy:

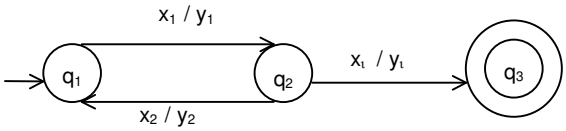


Figura 12:Máquina de Mealy terminal

Autómata Finito:
$\sum = \{ x_1 y_1 , x_2 y_2 , x_t y_t \}$

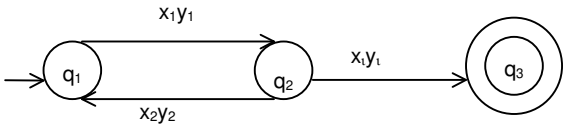


Figura 13:AF de estado objetivo y, además, destacadamente terminal

Lema II.2.23. Dada la biyección h_b existe una biyección h_u que aplica sucesiones de situación /acción en cadenas de $\sum$.

$$h_b : \sum_{sa}^{*} \longleftrightarrow \sum^{*}$$
$$(s_1a_1)(s_2a_2)(s_3a_3) \leftrightsquigarrow s_1a_1s_2a_2s_3a_3$$

Lema II.2.24. Dada la biyección del lema anterior, a partir de ella se define otra biyección h_{s+} que lleva subconjuntos de sucesiones situación/acción en cadenas.

$$h_{s+}: P(\sum_{sa}^{*}) \longleftrightarrow P(\sum^{*})$$
$$\{(s_1a_1)(s_2a_2)(s_3a_3)\} \leftrightsquigarrow \{s_1a_1s_2a_2s_3a_3\}$$

Todos los lemas anteriores permiten obtener un autómata finito a partir de una máquina Mealy y operar sobre él y después hacer el paso inverso para ajustar los resultados a una máquina Mealy.

$$S_{sa}^{+} \xrightarrow{\mathbf{h_{s+}}} S^{+} \longrightarrow L \xrightarrow{\mathbf{h_{s+}^{-1}}} L_{sa}$$

[4] Concatenación de los conjuntos X con Y

2.2.4. Algoritmo SKR

Parte de una muestra S_{sa}^+ de un lenguaje $L_{sa} \subset \Sigma_{sa}^*$ y por medio de h_{s+} la transforma en la muestra S^+ de un lenguaje $L \subset \Sigma^*$. El problema de inducir un lenguaje situación/acción queda reducido al de inducir un lenguaje regular estándar que se ataja con el algoritmo KR.. Si se utilizase el algoritmo de inducción KC obtendríamos el algoritmo SKC.

- **Pasos del algoritmo**

a) Transforma la muestra S_{sa}^+ de un lenguaje $L_{sa} \subset \Sigma_{sa}^*$, en la muestra S^+ de un lenguaje $L \subset \Sigma^*$.
b) Aplicación del algoritmo de inducción KR comenzando con k = 0.
c) Incrementar sucesivamente la k hasta obtener un aceptor de estado finito A sobre Σ, tal que $h_u^{-1}(A)$ corresponda a una máquina de Mealy determinística.
d) Si k superior a la longitud de la mayor de las cadenas de S^+, devolver fallo.

- **Pseudocódigo**

Entrada: S_{sa}^+, Muestra positiva de situación/acción no vacía.
Salidas: M_f Máquina Mealy mínima k-reversible determinística.
f (valor final de k) si la Maquina Mealy es determinística.
FALLO si la Maquina Mealy no es determinística.

Procedimiento **INICIALIZAR**

Comenzar

k = 0
$S^+ = h_{s+}(S_{sa}^+)$

Fin

.....

Mientras [k ≤ (máx (longitud de secuencia en S_{sa}^+))]
y
[M_k No determinística] **hacer**
$A_k = KR(S^+. k)$
$M_k = h_a^{-1}(A_k)$
Si M_k No determinística **entonces** k = k + 1 **fin_si**

fin_mientras

Si M_k No determinística **entonces** FALLO
sino f = k
retorna (M_f, f)

fin_si

2.2.5. Algoritmo SKC

Todo el desarrollo de este algoritmo es semejante al del SKR pero utilizando el algoritmo KC como núcleo.