2

Los Métodos Sintácticos

2.1 Introducción

Entre los métodos estructurales de reconocimiento de formas destacan los métodos englobados en lo que se conoce como *reconocimiento sintáctico de formas*. Estos métodos intentan aprovechar las técnicas desarrolladas por la *teoría de lenguajes formales* [Aho,73] [Eilemberg,74] [Harrison,78], las cuales proveen de una representación (las gramáticas) y de un mecanismo de interpretación ("parsing" o *análisis sintáctico*) para aquellas formas cuyos objetos se pueden describir como cadenas de subobjetos [Gonzalez,78] [Fu,82] [Miclet,86].

En su utilización en reconocimiento de formas, la teoría de lenguajes tropezó desde un principio con un problema inherente a este dominio de aplicación: la representación imprecisa de los objetos. Para solventar esta dificultad se hace necesario recurrir a métodos de análisis sintáctico tolerantes a cadenas "ruidosas" o plagadas de errores. Los más desarrollados de estos métodos son el *análisis sintáctico corrector de errores* y el *análisis sintáctico estocástico*, que utilizados simultáneamente permiten construir *analizadores sintácticos correctores de errores estocásticos*.

2.2 Los lenguajes y sus gramáticas

Dado un conjunto de símbolos o *alfabeto* V, el conjunto de todas cadenas posibles sobre ese alfabeto es V^* (el monoide libre sobre V mediante el operador concatenación) y V^+ será el mismo conjunto sin la cadena vacía. Un *lenguaje* sobre el alfabeto V es un subconjunto de V^*. Existen un

número no enumerable de lenguajes, todos ellos representables mediante *gramáticas*.

Una gramática se define como la cuádrupla G=(N,V,P,S), donde V es el *alfabeto* de G y N es un conjunto finito de símbolos ***no*** *terminales*, $V \cap N = \emptyset$. Se conoce a $W=V \cup N$ como el *vocabulario* de G, una *frase* $\zeta \in W^*$ es una cadena de símbolos de W. $P \subset W^* N W^* \times W^*$ es un conjunto de *reglas de producción* o reescritura, que transforman una frase (en la que por lo menos hay un no terminal) en otra frase del mismo vocabulario. Estas reglas se denotan como $\zeta_1 A \zeta_2 \rightarrow \zeta_3$, donde $\zeta_1, \zeta_2, \zeta_3 \in W^*$ y $A \in N$. Finalmente, $S \in N$ es el símbolo no terminal *inicial* o *axioma* de G.

Mediante la aplicación sucesiva de reglas de G se puede transformar una frase $\zeta 1$ del vocabulario en otra ζ_n. Escribiremos esto como $\zeta_1 \overset{*}{\Rightarrow} \zeta_n$; $\zeta_1, \zeta_n \in W^*$; y si $D(\zeta_n)=(\zeta_1, \zeta_2, \zeta_3, \ldots, \zeta_n)$, $\zeta_i \in W^*$, es la secuencia de frases que han llevado de ζ_1 a ζ_n, diremos que D es una *derivación* de ζ_n desde ζ_1. Se define entonces el *lenguaje generado por G* como el conjunto de todas las cadenas **del alfabeto** que se pueden derivar del axioma de G: $L(G)=\{ \alpha : \alpha \in V^*, S \overset{*}{\Rightarrow} \alpha \}$.

Al proceso de obtener una derivación de una cadena a partir del axioma de una gramática y aplicando reglas de ésta, se le denomina *análisis sintáctico*. Se dice que una gramática es *ambigua* si para alguna cadena del lenguaje existe más de una posible derivación para generarla.

Chomsky dividió las gramáticas (y por lo tanto los lenguajes) en una jerarquía que va del tipo 0 a 3, en orden decreciente de complejidad, y en base a la forma de sus reglas:

Tipo 0: Gramáticas sin restricciones en las reglas.

Tipo 1: Gramáticas *sensibles al contexto*, con reglas de la forma $\zeta_1 A \zeta_2 \rightarrow \zeta_1 \beta \zeta_2$. Es decir, el no terminal A se sustituye por la frase β del vocabulario en el *contexto* $\zeta_1 \zeta_2$ ($\zeta_1, \zeta_2 \in W^*$, $\beta \in W^+$).

Tipo 2: Gramáticas de *independientes del contexto*, en las que $A \rightarrow \beta$ independientemente del contexto.

Tipo 3: Gramáticas *regulares*, cuyas reglas son de la forma $A \rightarrow aB$ o $A \rightarrow a$; $a \in V$; $A,B \in N$.

La gramáticas de tipo 0 y 1 son las que proporcionan el mayor poder descriptivo, aunque son las gramáticas del tipo 2 y 3 las más utilizadas en aplicaciones prácticas como el reconocimiento de formas, principalmente debido a su mucho menor complejidad. Todas ellas son las llamadas *gramáticas formales*.

2.3 Reconocedores de lenguajes

Según se ha definido en el apartado anterior, para cada lenguaje existe una gramática generadora del mismo. Similarmente, existe un *reconocedor* capaz de decidir si una cadena pertenece o no a dicho lenguaje.

Para los lenguajes regulares el reconocedor es un *autómata finito*. Se demuestra que para todo autómata finito existe una gramática regular que genera el lenguaje aceptado por el autómata y viceversa [Aho,73].

Un autómata finito se define como una quíntupla $A=(V,Q,\delta,q_o,F)$, donde V es un conjunto finito de símbolos y Q un conjunto finito de *estados*. q_o es el *estado inicial* y $F \subset Q$ el conjunto de *estados finales*. δ una *función de transición* entre estados: $\delta:(Q\times V)\rightarrow \mathcal{P}(Q)$. El funcionamiento de un autómata se inicia en q_o para ir pasando, mediante la función de transición, de un estado al estado siguiente según indiquen los sucesivos símbolos de la cadena a reconocer. Si después del último símbolo de la cadena se ha llegado a un estado final, la cadena *pertenece* al lenguaje del autómata (figura 2.1). Formalmente, el *lenguaje de cadenas de V^* aceptado por el autómata A* se define como $L(A)=\{\alpha \mid \alpha\in V^* \wedge \Delta(q_o,\alpha)\cap F\neq\emptyset\}$, donde $\Delta:(Q\times V^*)\rightarrow \mathcal{P}(Q)$ es la función de transición δ extendida a cadenas de símbolos (λ es la cadena vacía):

$$\Delta(q,\lambda)=\{q\};\quad \Delta(q,\alpha a)=\bigcup_{q'\in\Delta(q,\alpha)} \delta(q',a);\qquad \alpha\in V^*;\ a\in V;\ q\in Q;$$

Cada una de las posibles secuencias de estados recorridos por el autómata para reconocer una cadena, equivale a una secuencia de reglas de la gramática (regular) equivalente, y por lo tanto, representa una *derivación* de la cadena. Nótese que en general, el funcionamiento de un autómata implica el seguir simultáneamente varios posibles caminos a través del mismo (y posiblemente llegar a varios estados finales), puesto que, según la definición de δ, dado un símbolo y un estado son varios los estados siguientes (el autómata es *no determinista*). Para recorrer en paralelo varios caminos se suelen utilizar algoritmos basados en técnicas de *programación dinámica* (algoritmo de Viterbi, ver capítulo 4) [Bellman,57] [Forney,73]. Si la definición de δ se restringe a $\delta:(Q\times V)\rightarrow Q$ entonces el autómata es *determinista*. Se demuestra que dado un autómata no determinista siempre existe uno determinista que reconoce el mismo lenguaje (pero puede tener $2^{|Q|}$ estados) y viceversa (figura 2.2) [Aho,73].

Sea el Lenguaje:

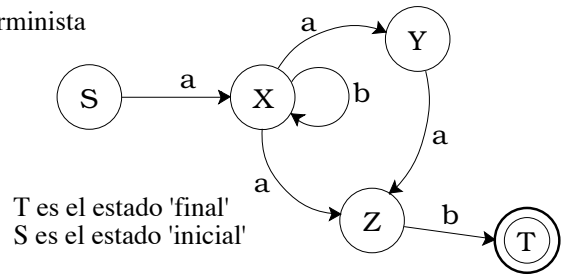
'Todas las cadenas que empiezan con 'a', seguida de un número indeterminado de 'b', y que terminan con 'ab' o 'aab'

Su Gramática:

G=(V,N,P,S); V={'a','b'}; N={S,X,Y}
P={
S→aX (axioma)
X→bX (un bucle)
X→aZ
X→aY
Y→aZ
Z→b
}

G es NO determinista

Su atómata asociado:



T es el estado 'final'
S es el estado 'inicial'

Reconocimiento de cadenas: 'abba' y 'ab'

1) Secuencia de estados:
`abbab:` SXXXZT, reconocida.
`ab:` SXX, NO reconocida.

2) Secuencia correspondiente de reglas:
`abbab:` S→aX, X→bX, X→bX, X→bX, X→aZ, Z→bT
`ab:` S→aX, X→bX, X→???

Figura 2.1 Un ejemplo de Lenguaje regular, con su Gramática y su Autómata asociado. Análisis sintáctico de 2 cadenas.

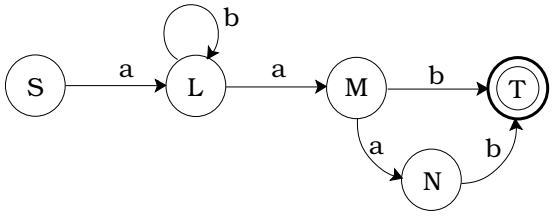


Figura 2.2 Autómata determinista equivalente al no determinista de la figura anterior. Nótese que reconoce el mismo lenguaje, pero NO representa la misma gramática (aunque sí a una equivalente).

Para los lenguajes de contexto libre los reconocedores correspondientes son los *autómatas a pila* [Fu,82], para los lenguajes dependientes del contexto se emplean los *autómatas lineales acotados*, y finalmente, para los lenguajes de tipo 0 las *máquinas de Turing* [Fu,82].

La definición de los reconocedores de lenguajes conduce inmediatamente a una posible manera de utilizar los métodos sintácticos en reconocimiento de formas. Para ello, y siempre que los objetos sean representables mediante cadenas de símbolos, basta construir un clasificador en el que se define una gramática por clase o forma. El correspondiente

reconocedor (p.e., un autómata si la gramática es regular) determinará si la cadena pertenece o no al lenguaje de la gramática (al conjunto de cadenas que éste representa) y por lo tanto, a la forma. La construcción o inferencia de los reconocedores asociados a cada clase, o equivalentemente de sus gramáticas, se puede llevar a cabo manualmente o mediante métodos de *inferencia gramatical* (ver capítulo siguiente).

2.4 El problema de la imprecisión

Es muy usual en reconocimiento de formas el que los objetos a reconocer, e incluso los ejemplos de aprendizaje, vengan distorsionados y embrollados por ruidos de diversa procedencia (medio ambiente, líneas de transmisión en mal estado, mancha en la foto,...) (figura 2.3).

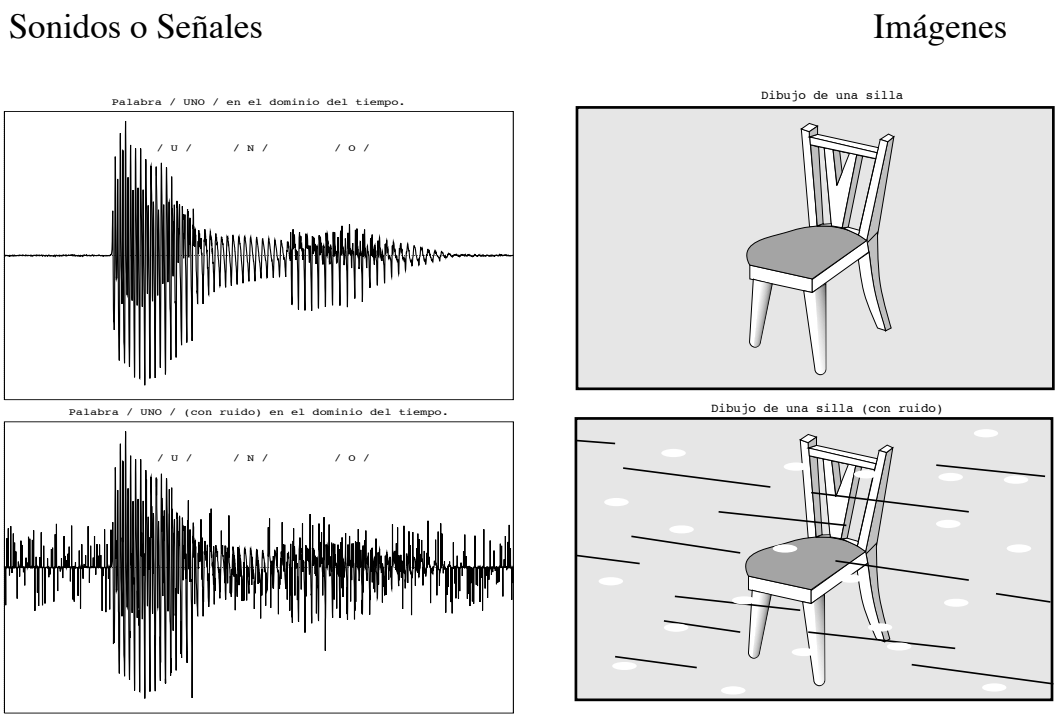


Figura 2.3 Formas "limpias" y distorsionadas por ruido.

En el caso del reconocimiento sintáctico de formas, ello quiere decir que las cadenas presentadas al reconocedor pueden no ser correctas, es decir, pueden presentar símbolos equivocados y símbolos de más o de menos. Esto a su vez conlleva el que las gramáticas, que a menudo son construídas (automática o manualmente) a partir de un subconjunto representativo de cadenas de la forma, puedan ser inexactas. Esta imprecisión, que afecta tanto en la descripción estructural de los objetos como en la de las formas, no puede deshacerse mediante los procedimientos usuales de análisis

sintáctico, por lo que se han desarrollado soluciones alternativas, las cuales se enmarcan en dos aproximaciones bien diferenciadas:

- Aquellas en las que la interpretación se lleva a cabo mediante el uso de métodos derivados de la teoría de la decisión, definiendo una distancia entre cadenas, a partir de información estadística relativa a los distintos errores debidos al ruido u otras causas [Aho,72] [Aho,73] [Thomason,74] [Bahl,75] [Tanaka,86] [Tanala,87]. Las técnicas correspondientes se engloban en el llamado *análisis sintáctico corrector de errores*.

- Aquellas que asocian a la información estructural una información probabilística, sobre la frecuencia de uso/aparición de las distintas partes de la estructura. Este suplemento de información flexibiliza notablemente la capacidad de representación del modelo, y ha conducido a la extensión de los lenguajes formales con el fin de incluir en ellos los llamados *lenguajes estocásticos*. Estos lenguajes se representan mediante *gramáticas estocásticas*, con las que se lleva a cabo un *análisis sintáctico estocástico* [Gonzalez,78] [Fu,82].

2.4.1 Análisis sintáctico corrector de errores

La idea básica del análisis sintáctico corrector de errores reside en la estimación de una *distancia* o *medida de similitud* entre una cadena distorsionada y su original perteneciente a una gramática dada.

La distancia entre dos cadenas se define a partir del "número mínimo" de *transformaciones de error* necesarias para pasar de una cadena a la otra. Se consideran tres tipos posibles de error: *borrado*, *inserción* o *sustitución* de un símbolo, pudiéndose pasar de una cadena a otra cualquiera simplemente mediante combinación de estos errores [Thomason,74] [Fu,82]. Formalmente, se definen tres transformaciones:

borrado: $\alpha a \beta \overset{T_b}{\vdash} \alpha\beta$;

inserción: $\alpha\beta \overset{T_i}{\vdash} \alpha a \beta$;

sustitución: $\alpha a \beta \overset{T_s}{\vdash} \alpha b \beta$

$T_b, T_i, T_s : V^* \rightarrow V^*$; $\alpha,\beta \in V^*$; $a,b \in V$; $a \neq b$;

a partir de las cuales se define la *distancia de Levenshtein* $d_L(\alpha,\beta)$ entre dos cadenas $\alpha,\beta \in V^*$ como el mínimo número de transformaciones necesarias para convertir α en β (figura 2.4). Es decir, si C es cualquier secuencia de

transformaciones que convierta α en β y S_C, B_C, I_C son respectivamente el número de sustituciones, borrados e inserciones en C:

$$d_L(\alpha,\beta)=\min_{\forall C} \{ S_C + B_C + I_C \}$$

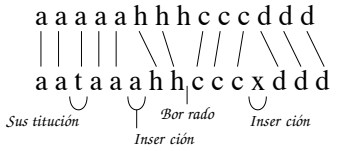


Figura 2.4 Distancia de Lenvenshtein entre dos cadenas.

Si es necesario, es posible extender esta definición dando más o menos importancia a un tipo de error frente a otro, obteniéndose entonces la *distancia de Levenshtein ponderada*:

$$d_{LP}(\alpha,\beta)=\min_{\forall C} \{ w_s \cdot S_C + w_b \cdot B_C + w_i \cdot I_C \}$$

donde w_s, w_i, w_b son respectivamente los pesos (costes) de sustitución, borrado e inserción. Incluso cabe introducir una dependencia según el símbolo que se borra (se inserta o sustituye) (*distancia de Fu ponderada*) [Fu,82].

Una vez definida la distancia, un algoritmo *analizador sintáctico corrector de errores* es un algoritmo que busca una cadena β perteneciente al lenguaje de la gramática G tal que la distancia (d) entre β y la cadena a analizar α sea mínima:

$$\beta = \underset{\forall \omega \in L(G)}{\operatorname{argmin}} \{ d(\alpha,\omega) \}$$

a la distancia que proporciona el mínimo se la puede considerar como la *distancia entre la cadena α y el lenguaje L(G)*:

$$D(\alpha,L(G)) = d(\alpha,\beta) = \min_{\forall \omega \in L(G)} \{ d(\alpha,\omega) \}$$

Generalmente, los algoritmos de análisis sintáctico con corrección de errores se subdividen en dos etapas:

- Construcción de la *gramática expandida* G^e de la gramática G original, añadiendo a cada regla de G sus correspondientes *reglas de error*. Existirá una regla de error asociada a cada posible transformación de error (figura 2.5).

Concretamente, las reglas de error a añadir a una gramática **regular** G=(N,V,P,S) para construir su gramática expandida serán, $\forall x \in V$, (ε es el símbolo nulo *nil*):

Inserción (de x): $A \rightarrow xA \quad \forall (A \rightarrow bB) \in P$

Sustitución (de b por x): $A \rightarrow xB \quad \forall (A \rightarrow bB) \in P$;
$A \rightarrow x \quad \forall (A \rightarrow b) \in P$

Borrado (de b)[1]: $A \rightarrow B \quad \forall (A \rightarrow bB) \in P$; $A \rightarrow \varepsilon \quad \forall (A \rightarrow b) \in P$

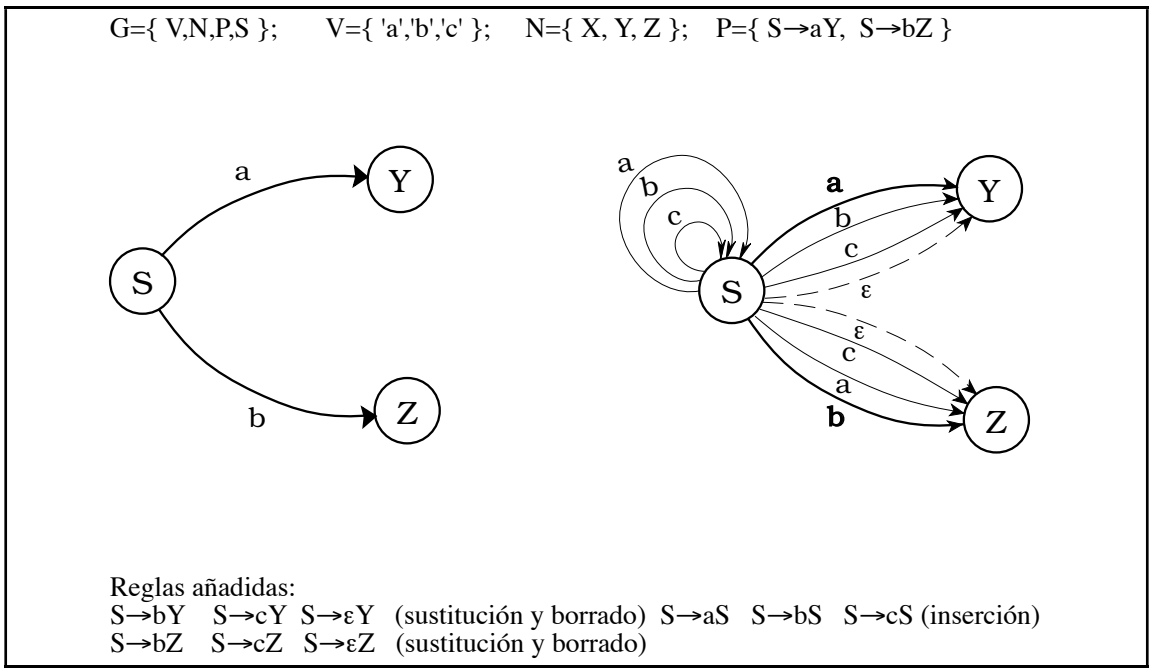


Figura 2.5 Autómata expandido: transiciones que se añaden al hacer la expansión de dos transiciones procedentes del mismo estado. Obsérvese la redundancia de transiciones de inserción.

- Analizar sintácticamente la cadena desconocida con G^e y luego, de entre todas las derivaciones obtenidas (G^e es siempre ambigua), buscar aquella cuyas reglas (transformaciones) de error produzcan la distancia mínima. Esta derivación[2] proporciona no sólo la cadena que más se asemeja a la muestra, sino también la distancia a la misma.

  Típicamente, los algoritmos que se emplean para esta búsqueda y minimización son los mismos que los utilizados para gramáticas no deterministas, por lo que se basan también en las técnicas de

[1] Según la definición más corriente, estas reglas NO SON REGULARES. Esto se discute con detalle en el Capítulo 5.

[2] La derivación óptima puede no ser única.

programación dinámica. Sin embargo, la particular forma de las reglas de borrado obliga a utilizar algoritmos especiales, algunos de los cuales se proponen en el capítulo 5.

2.4.2 Gramáticas estocásticas

Una gramática estocástica se forma añadiendo probabilidades a las reglas de una gramática no estocástica (la *gramática característica* de la gramática estocástica). El *lenguaje estocástico* generado por tal gramática está formado no sólo por las cadenas del lenguaje, sino también por su probabilidad de producción, que se obtiene mediante el producto de las probabilidades de las reglas utilizadas.

La teoría de las gramáticas estocásticas está perfectamente asentada, con una teoría matemática que complementa a la de los lenguajes formales. Por ejemplo, una gramática estocástica regular es equivalente (o se puede modelizar como) un *generador de Markov* [Casacuberta,90b], el cual destaca por ser uno de los procesos estocásticos más tratables y mejor comprendidos [Rabiner,89a].

2.4.2.1 Definiciones

Una gramática *ponderada* es una quintupla $G_s=(N,V,P,S,D)$. Donde $G=(N,V,P,S)$ es la gramática característica de la gramática ponderada y D es el conjunto de pesos asociado a las reglas de P. Para cada regla $\zeta_{1i}A_i\zeta_{2i}\rightarrow\zeta_{ij}$ el peso se escribe p_{ij} ($i=1,\ldots,K$; $j=1,\ldots,n_i$; K es el número de partes izquierdas distintas y n_i el número de reglas con la parte izquierda i). Se dice que G_s es *estocástica* si los pesos se comportan como probabilidades, es decir, si $p_{ij}\subseteq]0,1]$ y la suma de las probabilidades de todas las reglas con la misma parte izquierda es igual a la unidad:

$$\forall\ \zeta_{1i}A_i\zeta_{2i}\in W^*NW^*:\ \zeta_{1i}A_i\zeta_{2i}\rightarrow\zeta_{ij},\ j=1,\ldots,n_i,\ \sum_{i=1}^{ni} p_{ij}=1$$

La tipología 0,1,2,3 así como otros conceptos que dependen de las formas de las producciones (forma normal de Greinbach, etc...) se extienden por similitud con las no estocásticas.

Una gramática estocástica es *no-restringida* si la probabilidad de una regla no depende de la secuencia de reglas anteriormente aplicada. En una gramática no-restringida (y no ambigua), la probabilidad de generación de una cadena $\alpha=a_1a_2...a_m$; $a_1,a_2,...,a_m\in V$, a partir del axioma S, aplicando la

secuencia de reglas $r_1,r_2,r_3,...,r_m$, viene dada por $p(\alpha)=p(r_1)\cdot p(r_2)\cdot\ldots\cdot p(r_m)$, donde $p(r_i)$ es la probabilidad de la regla r_i. Escribiremos esto $S \overset{*}{\underset{p(\alpha)}{\Rightarrow}} \alpha$.

En una gramática ambigua, existirán n_α secuencias de reglas (derivaciones) para generar la misma cadena α, cada una con su respectiva probabilidad $p_i(\alpha)$, $i=1,...,n_\alpha$. En este caso la *probabilidad de generación de* α se puede definir de dos posibles maneras:

- suma de la probabilidad de un número finito de eventos excluyentes (*aproximación aditiva*):

$$p(\alpha) = \sum_{i=1}^{n_\alpha} p_i(\alpha)$$

- máxima verosimilitud (*aproximación maximal*):

$$p(\alpha) = \max_{i=1,..,n_\alpha} p_i(\alpha)$$

Una gramática estocástica G_s genera un *lenguaje ponderado* , es decir, un lenguaje en el que cada cadena lleva asociada un peso $p(\alpha)$ y que se define como (utilizando la aproximación aditiva):

$$L(G_s) = \{\ [\alpha,p(\alpha)] \mid \alpha\in V^*,\ S \overset{*}{\underset{p_i(\alpha)}{\Rightarrow}} \alpha,\ i=1,...,n_\alpha,\ p(\alpha) = \sum_{1=i}^{n_\alpha} p_i(\alpha)\ \}$$

si la gramática es ambigua. Si no lo es:

$$L(G_s)=\{\ [\alpha,p(\alpha)] \mid \alpha\in V^*, S \overset{*}{\underset{p(\alpha)}{\Rightarrow}} \alpha\ \}$$

Para poder considerar a $L(G_s)$ como un espacio muestral, en el que un suceso es la ocurrencia de una cadena con probabilidad dada por la gramática G_s, es necesario que la suma de las probabilidades de todas las cadenas que pueda generar la gramática sea igual a la unidad [Gonzalez,78]:

$$\sum_{x\in L(G_s)} p(\alpha)=1$$

A un lenguaje que cumple esta propiedad se le denomina un *lenguaje estocástico*, y a una gramática que genera un lenguaje estocástico es una gramática *consistente* :

$$G_s \text{ es consistente} \Leftrightarrow L(G_s) \text{ es estocástico}$$

Las condiciones de consistencia para gramáticas de contexto libre se pueden consultar en [Fu,82] [Wetherell,90]. Para que una gramática regular (de tipo 3) sea consistente basta con que sea *propia* (es decir, que no existan símbolos inútiles, ver [Fu,74] y [Miclet,86]).

2.4.2.2 Autómatas estocásticos

Del mismo modo en que se extienden las gramáticas para obtener gramáticas estocásticas, se pueden extender los reconocedores para obtener reconocedores estocásticos. En concreto, para las gramáticas estocásticas de tipo 3 o regulares, los reconocedores designados serán los *autómatas finitos estocásticos*, que se derivan de los autómatas finitos añadiendo probabilidades a las transiciones.

Un autómata finito estocástico (AFE) es una quíntupla $A_s(Q,V,\delta,I,F)$, donde Q es un conjunto finito de estados y V el conjunto de símbolos terminales. $\delta: Q \times E \times Q \rightarrow 0,1]$ es una función parcial que asigna las probabilidades a todas las posibles transiciones. $I:Q \rightarrow [0,1]$, $F:Q \rightarrow \{0,1\}$ definen para cada estado su probabilidad de ser inicial y su pertenecia o no al conjunto de estados finales, respectivamente. Para asegurar la estocasticidad, la suma de probabilidades de las transiciones que salen de un mismo estado debe ser igual a la unidad:

$$\forall q \in Q; \sum_{q_i \in Q, a \in V} \delta(q,a,q_i)=1 \quad \wedge \quad \sum_{q \in Q} I(q)=1$$

Se pueden dar toda una serie de definiciones similares a las formuladas para gramáticas estocásticas:

Si el AFE es *no-restringido* (la probabilidad de una transición es independiente de las de las demás transiciones) la probabilidad de aceptar una cadena $\alpha=\alpha_1\alpha_2...\alpha_n$ siguiendo la secuencia de estados $q_o,q_1,...,q_n$ se escribe como:

$$I(q_o)\cdot\delta(q_o,\alpha_1,q_1)\cdot...\cdot\delta(q_{n-1},\alpha_n,\alpha_n)\cdot F(q_n)$$

Dado que para una determinada cadena α pueden haber varios caminos a través del autómata que lleguen al estado final, cada uno con su probabilidad $p_c(\alpha)$, la probabilidad de generación de una cadena se puede definir también mediante la aproximacíon aditiva o la maximal:

$$p(\alpha) = \sum_{\forall c} p_c(\alpha) \ \textit{(aditiva)} \quad \text{o bien} \quad p(\alpha) = \max_{\forall c} \{ p_c(\alpha) \} \ \textit{(maximal)}$$

Al igual que en las gramáticas, el *lenguaje aceptado por un AFE* está formado por las cadenas que a partir del estado inicial llevan a un estado final, **junto** con su probabilidad de aceptación. Se dice que dos AFE son *equivalentes* si aceptan el mismo lenguaje ponderado (mismas cadenas y mismas probabilidades).

Dada una gramática estocástica regular, existe un AFE que acepta un lenguaje idéntico al generado por la gramática (pero lo contrario no es cierto) [González,78] (figura 2.6).

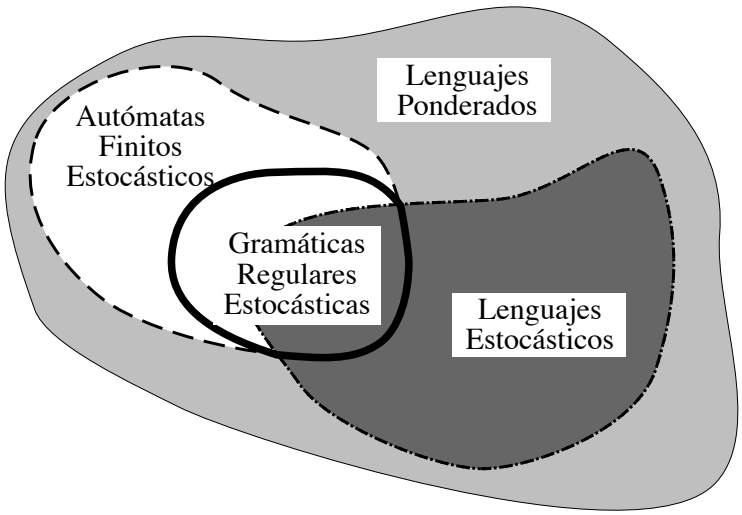


Figura 2.6 Relación de inclusión entre los lenguajes ponderados (P), los estocásticos (E), los generados por gramáticas regulares estocásticas (GRE) y los autómatas finitos estocásticos (AFE).

Como se recordará, en el caso general de los autómatas finitos no deterministas, al llevar a cabo el análisis sintáctico es necesario utilizar métodos que permitan evaluar en paralelo varios caminos en el autómata. Cuando se trata de autómatas estocásticos ello se complica debido a la presencia de las probabilidades, y si bien el *algoritmo de Viterbi* sigue siendo válido cuando se trabaja con la aproximación maximal, es necesario emplear el algoritmo *"Forward"* para el caso de la aproximación aditiva [Rabiner,89]. Ambos algoritmos son de complejidad similar (ver capítulo 4).

2.4.2.3 Aprendizaje de gramáticas estocásticas

En el aprendizaje de gramáticas estocásticas, no sólo se debe de tener en cuenta la estructura de los ejemplos, sino también la frecuencia de aparición de los mismos. Para ello, el método generalmente empleado se basa en la adquisición independiente del modelo estructural (de la gramática no estocástica), añadiéndose con posterioridad las probabilidades de las reglas mediante un análisis estadístico de los ejemplos y de sus derivaciones por la gramática [Maryanski,77] [Chandhuri,86]. Más raramente, es el propio

proceso de inferencia de las reglas (estructura) el que se ve guiado por el comportamiento estadístico de los ejemplos [Thomason,86].

Consideraremos aquí únicamente el proceso de inferir las probabilidades de las reglas de una gramática característica G=(V,N,P,S) ya conocida. Dicho de otro modo, se trata de inferir el conjunto de probabilidades D de la gramática estocástica G_s correspondiente a G, a partir de un conjunto de m ejemplos $X=\{\alpha_1,...,\alpha_m\}$ con frecuencias respectivas de aparición $f_1,...,f_m$. Entonces, y siempre que G no sea ambigüa, la probabilidad por máxima verosimilitud p_{ij} de la regla $\zeta_{1i}A_i\zeta_{2i}\rightarrow\zeta_{ij}$ se puede estimar en base a:

$$\hat{p}_{ij} = \frac{n_{ij}}{\sum_{\forall j} n_{ij}}; \qquad n_{ij} = \sum_{\alpha_k \in X} f_k \cdot N_{ij}(\alpha_k)$$

donde $N_{ij}(\alpha_k)$ representa el número de veces que la regla se utiliza para analizar α_k, y por lo tanto n_{ij} simboliza la frecuencia de utilización de la regla por parte de todas las cadenas de X. Para gramáticas independientes del contexto, se puede demostrar que esta estimación $\hat{p}_{ij}$ se aproxima a p_{ij} cuando el número m de ejemplos tiende a infinito, siempre que simultánemente X se aproxime a L(G) y f_k en X se aproxime a $p(\alpha_k)$ [Maryanski,77]. Esta estimación además garantiza que la gramática sea consistente [Fu,74].

Obsérvese que el procedimiento de estimación descrito no es válido en el caso de gramáticas ambiguas, pues no se ha definido ningún método para "distribuir" las probabilidades cuando hay más de una derivación. En el caso concreto de las gramáticas estocásticas ambigüas regulares existen métodos adecuados, entre los que cabe destacar, por su uso generalizado, el algoritmo de «Baum-Welch» y el de reestimación por Viterbi [Levinson,83] [Rabiner,89].

2.4.3 Análisis sintáctico corrector de errores estocástico

A la hora de efectuar el análisis sintáctico de cadenas muestras deformadas, es posible combinar en un solo proceso las dos estrategias presentadas en los apartados anteriores, obteniéndose entonces la metodología conocida como el*análisis sintáctico corrector de errores estocástico*. La idea de este método consiste simplemente en aplicar a las gramáticas estocásticas el mismo procedimiento de corrección de errores que se utilizó para las gramáticas no estocásticas; para lo cual, a una gramática estocástica se le añadirá el *modelo de corrección de errores* (las reglas de error) [Thomason,75]. Como la gramática expandida resultante deberá ser estocástica también, es necesario complementar cada transformación de error, no sólo con un peso como en la distancia de Levenshtein y la de Fu,

sino con una *probabilidad*. En estas condiciones, la similitud entre cadenas no se mide mediante una distancia entre ellas, sino por **la probabilidad de que una se transforme en otra**. El análisis sintáctico corrector de errores no minimizará una distancia, sino que maximizará una probabilidad.

Utilizando un modelo de error similar[3] al utilizado como ejemplo en el apartado 2.4.1, las transformaciones de error posibles serán: inserción de un símbolo 'b' (**antes de otro** 'a', a,b∈V), borrado del mismo (substitución de a por ε, el símbolo nil), y sustitución de 'a' por otro símbolo 'b'. A cada uno de estos errores se le asocia una probabilidad, respectivamente $p_i(a \mid b)$, $p_b(\varepsilon \mid a)$ y $p_s(b \mid a)$. Denotaremos $p_s(a \mid a)$ a la probabilidad de sustituir 'a' por sí mismo, es decir, la probabilidad de que NO se produzca error en 'a' (*probabilidad de no error*). Nos referiremos a una cualquiera de estas*probabilidades de deformación*, como **pd**. A partir de estas definiciones, y por extensión, pueden definirse la probabilidad de transformar un símbolo en una cadena y, aún con más generalidad, la de una cadena α en otra cadena β: $p(\beta \mid \alpha)$ [Fu,82].

Un analizador sintáctico corrector de errores estocástico buscará la cadena β del lenguaje de la gramática G que maximize a la vez la probabilidad de que la cadena α (que no pertenece a dicho lenguaje) se pueda transformar en β y la probabilidad de que β sea del lenguaje:

$$\beta = \underset{\forall \omega \in L(G^e)}{\operatorname{argmax}} \{p(\alpha \mid \omega) \cdot p(\omega)\}$$

donde el que ω pertenezca al lenguaje de la gramática expandida equivale a decir que se prueban todas las deformaciones permitidas de α. Los algoritmos utilizados para construir estos analizadores son similares a los utilizados en los autómatas analizadores sintácticos correctores de errores no estocásticos.

La probabilidad que maximiza la expresión anterior:

$$p^e(\alpha) = \max_{\forall \omega \in L(G^e)} \{p(\alpha \mid \omega) \cdot p(\omega)\}$$

se puede considerar como la *probabilidad de que α sea una cadena deformada del lenguaje de G*, o dicho de otro modo, la probabilidad de que α pertenezca a $L(G^e)$. También en este caso es necesario asegurar la consistencia de la gramática, es decir, asegurarse de que su lenguaje es estocástico:

---

[3] En este caso, el coste de inserción depende del símbolo antes del cual se realiza la inserción.

$$\sum_{\forall \alpha \in L(G^e)} p^e(\alpha) = 1$$

Con el modelo de error que se está utilizando, es posible demostrar que ello será cierto siempre que las *probabilidades de deformación sean consistentes* [Fu,82], es decir:

$$\sum_{\forall b \in V} p_s(b \mid a) + p_b(\varepsilon \mid a) + \sum_{\forall b \in V} p_i(b \mid a) = 1; \quad \forall a \in V$$

En el caso de una gramática **regular**, en que se utilize un modelo de error en el que la probabilidad de inserción **no** depende del símbolo siguiente (ejemplo del apartado 2.4.1), la condición de consistencia es diferente y se estudia con detalle en el capítulo 7.

2.5 Gramáticas SANSAT

En los restantes capítulos de este trabajo se tratará exclusivamente con gramáticas regulares (y en su mayoría estocásticas). En particular, en las implementaciones prácticas se ha recurrido a una subclase de las gramáticas de tipo 3: las gramáticas SANSAT ("**SA**me **N**on-terminal, then **SA**me **T**erminal"), de las cuales el autor no ha encontrado en la literatura ninguna mención explícita. Una gramática G=(N,V,P,S) será una SANSAT, si cumple la condición de que todas las reglas que tienen el mismo **no** terminal a la derecha tienen también el mismo terminal:

$$\text{si } (B \rightarrow aC) \in P \wedge (A \rightarrow bC) \in P \Rightarrow a=c; \forall A,B,C \in N, \forall a,b \in V$$

Tal como se verifica a continuación, toda gramática regular tiene una gramática SANSAT equivalente, y un *autómata de estados etiquetados* (o autómata LAS: "**LA**belled **S**tate automata") igualmente equivalente.

2.5.1 Gramática SANSAT equivalente a una gramática.

Dada una gramática regular G=(N,V,P,S), una gramática SANSAT equivalente a ella, $G_t=(N_t,V,P_t,S)$, se construye de la siguiente manera (ver figura 2.7):

- Se genera un no terminal por cada parte derecha de regla diferente:

$$N_t=\{ X_{aA} : \text{ si } (B \rightarrow aA) \in P,\ a \in V,\ A,B \in N \}$$

- Para cada no terminal X_{aA}, y cada no terminal X_{bB} que corresponde a una regla r de G que tiene A como parte izquierda r=(A→bB), se genera una regla que reescribe X_{aA} como X_{bB} precedido del terminal de la regla r, es decir X_{aA}→bX_{bB}:

$$P_t=\{ X_{aA}\rightarrow bX_{bB} : \text{si } (A\rightarrow bB)\in P \} \cup \{ X_{aA}\rightarrow b : \text{si } (A\rightarrow b)\in P \} \cup \{ S\rightarrow bX_{bB} : \text{si } (S\rightarrow bB)\in P \}$$

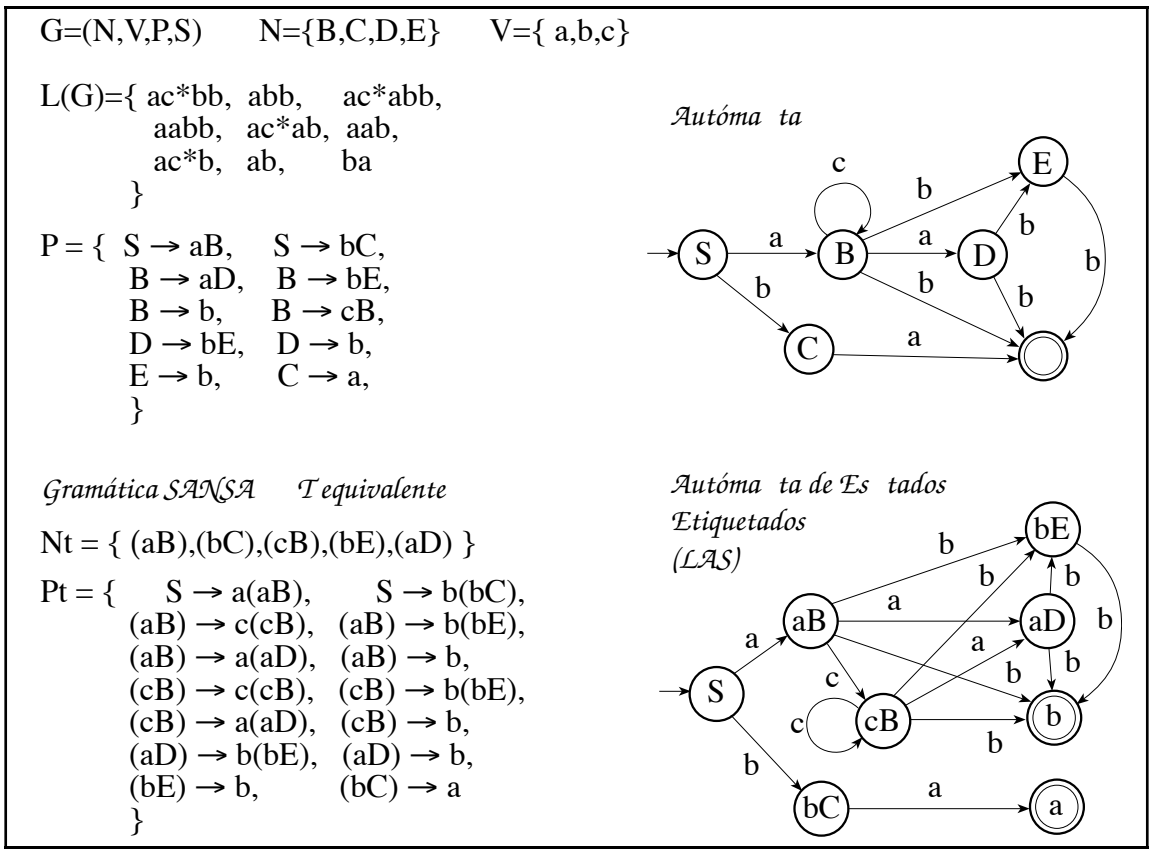


Figura 2.7 Una gramática regular G y su autómata equivalente. La gramática SANSAT equivalente a G, y su LAS (autómata de estados etiquetados) correspondiente (no se muestran las etiquetas de los estados).

Es inmediato comprobar que ambas gramáticas son equivalentes, o lo que es lo mismo que $L(G)=L(G_t)$. Para ello, basta tener en cuenta que toda derivación por G de una cadena $a_1a_2,\ldots a_n\in L(G)$ se puede reescribir mediante reglas de G_t (y por lo tanto pertenece al lenguaje de ésta) y viceversa. En efecto:

$$D=r_1r_2,\ldots r_n=$$
$$(S\rightarrow a_1A_1)(A_1\rightarrow a_2A_2)\ldots(A_{n-1}\rightarrow a_n)=$$
$$(S\rightarrow a_1X_{a1A1})(X_{a1A1}\rightarrow a_2X_{a2A2})\ldots(a_{n-1}X_{an-1An-1}\rightarrow a_n)$$

2.5.2 Autómata de estados etiquetados (LAS)

Un *autómata de estados etiquetados* LAS=(V_t,Q,δ,q_o,F,E), es un autómata equivalente a una gramática SANSAT G=(N_t,V,P_t,S), construído de manera similar a la seguida usualmente para obtener el autómata equivalente a una gramática, pero:

- Se añade una función de etiquetado E:Q→V que asigna a cada estado el símbolo terminal asociado a todas las reglas con ese no terminal a la derecha. Por completitud de E, se etiqueta arbitrariamente el estado inicial con el símbolo "~", (símbolo inicial) que se añade al conjunto de terminales.

- Se obliga a que haya tantos estados finales como símbolos terminales puedan terminar una cadena del lenguaje.

Todo lo cual queda expresado formalmente como sigue:

$V_t=V \cup \{\sim\}; \quad \sim \notin V$

$Q=\{ q_{aA}: \forall X_{aA} \in N_t \} \cup F; \; F=\{ q_a : \forall a \in V, \; (X_{bA} \rightarrow a) \in P_t \}; \quad q_o=S;$

$\delta: Q \times V \rightarrow Q;$

$\delta=\{ (q_{aA}, b, q_{bB}) : \text{si } (X_{aA} \rightarrow bX_{bB}) \in P_t \} \cup \{ (q_o, a, q_{aA}) : \text{si } (S \rightarrow aA) \in P \};$

$E: Q \rightarrow V; \; E(q_o)=\sim; \; E(q_{aB})=a; \; E(q_a)=a;$

En un LAS, los terminales asociados a las transiciones son **siempre iguales a la etiqueta el estado destino de la transición**. Esto permitiría definir las transiciones sin etiquetas (terminales asociados), al ser éstas redundantes, pero se las ha conservado para permitir que un LAS siga siendo un autómata en el sentido clásico. Por otra parte, gracias a esta propiedad, un LAS puede ser representado esquemáticamente de idéntica manera que un autómata clásico, pero etiquetando los estados en lugar de las transiciones (ver figura 2.8). También aprovechando esta propiedad es posible dar una definición alternativa del lenguaje del LAS: *una cadena es reconocida por un LAS* (y por lo tanto pertenece a su lenguaje) si existe un camino en el LAS que partiendo del estado inicial llegue a uno final, tal que la secuencia de **etiquetas de estados** se corresponda con la secuencia de símbolos de la cadena:

$$a_1 a_2 \ldots a_n \text{ es reconocida por LAS} \Leftrightarrow$$
$$\exists \; q_o, q_1, .. \, q_n \mid E(q_i)=a_i; \quad i=1..n; \quad q_i \in Q; \quad q_n \in F$$

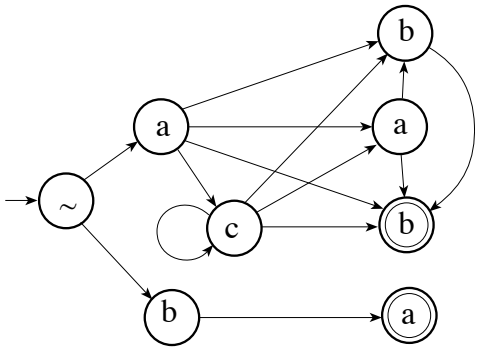


Figura 2.8 LAS de la figura anterior, representado con etiquetas en los estados en vez de en las transiciones.

Una gramática SANSAT $G_t=(N_t,V,P_t,S)$ (y por lo tanto su LAS) tiene generalmente más estados que su equivalente convencional G=(N,V,P,S). Del procedimiento de construcción de la gramática SANSAT se observa que puede llegar a tener $N_t=|N|\cdot|V|$ no terminales y $P_t=|P|\cdot|V|$ reglas[4] (ver figura 2.9). De ello se deduce que la complejidad espacial de la representación de un lenguaje mediante una gramática SANSAT (o su LAS) puede llegar a ser |V| veces mayor que su equivalente convencional, lo cual es especialmente significativo si el número de terminales es muy superior al de no terminales.

$P = \{ S \rightarrow aA,\ S \rightarrow bA,\ S \rightarrow cA,$
$A \rightarrow aA,\ A \rightarrow bA,\ A \rightarrow cA,$
$A \rightarrow a,\ A \rightarrow b,\ A \rightarrow c \}$

$N = \{ A \}$ $V = \{ a,b,c \}$

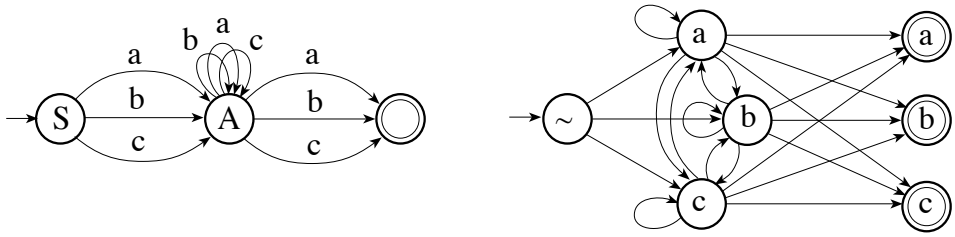


Figura 2.9 Caso extremo de incremento de transiciones y estados al pasar de un autómata a un autómata de estados etiquetados (LAS).

2.5.3 LAS y Modelos Ocultos de Markov (HMM)

Es extremadamente interesante notar cómo, a partir de la equivalencia entre una gramática regular y un autómata de símbolos etiquetados, es inmediato relacionar las gramáticas con los*Modelos Ocultos de Markov*

[4] Basta notar que, en realidad, lo que se está haciendo es desdoblar los no terminales (estados) en tantos como sea necesario para que a cada uno de ellos llegue siempre el mismo terminal.

(Hidden Markov Model: HMM [Rabiner,89a]) con estados finales. En efecto, si se define un *autómata estocástico de estados etiquetados*, como un LAS, con:

- Probabilidades de transición $p(q \mid q')$ (la probabilidad de una transición no depende del símbolo).
- Una función de etiquetado $E(q,a):Q \times V \rightarrow [0..1]$ probabilística (probabilidad de que la etiqueta de $q \in Q$ sea $a \in V$).

de forma que:

$$\sum_{\forall q} p(q \mid q')=1 \text{ y } \sum_{\forall a} E(q \mid a)=1$$

se tiene una definición equivalente a la de un Modelo Oculto de Markov con estados finales, en el que la distribución inicial de probabilidad de los estados $P_o(q_o)$, $P_o(q_1)..P_o(q_{|Q|})$ es [1,0,..,0] (sólo hay un estado inicial).