Capítulo 2

Autómatas finitos estocásticos y lenguajes

> *Si el universo consta de un número infinito de términos, es rigurosamente capaz de un número infinito de combinaciones –y la necesidad de un Regreso queda vencida. Queda su mera posibilidad, computable en cero.*
> Jorge Luis Borges, Historia de la eternidad.

Las máquinas de estados finitos son uno de los mecanismos más sencillos que permiten realizar una tarea y se caracterizan por que su funcionamiento requiere tan sólo una capacidad de almacenamiento que es acotable *a priori*. Desafortunadamente, no todos los problemas pueden ser resueltos utilizando autómatas finitos. Vamos a considerar como ejemplo dos tareas del ámbito de la programación de ordenadores, aparentemente de dificultad similar:

1. Un programa que calcula el cociente de una cadena numérica

por 3. Este programa puede implementarse de forma sencilla sin que se necesite almacenar toda la cadena de entrada. Basta con ir procesando ésta símbolo a símbolo, calculando el cociente y guardando el resto para sumárselo (multiplicado por tres) a la siguiente lectura. Por tanto, un programa que disponga de un sistema de almacenamiento para guardar dos bits es suficiente para realizar la tarea. Por muy larga que sea la cadena de entrada, bastará con esperar el tiempo suficiente para obtener la salida, pero jamás se producirá el desbordamiento del programa. Los estados de la memoria utilizada pueden entenderse como los estados de un autómata finito y los cambios que se producen con la entrada pueden interpretarse como la llamada función de transición (véase la figura 2.1).

2. Un programa que invierte el sentido de la cadena. En este caso es necesario optar entre:

   (a) leer la cadena completa y después escribir el resultado, con lo que es de esperar que para una cadena lo suficientemente larga se agoten las reserva de memoria, o

   (b) utilizar una función recursiva.

   En esta segunda opción, el desbordamiento se producirá cuando se agote la pila de almacenamiento de llamadas recursivas, lo cual es inevitable si la cadena es lo suficientemente larga.

La tarea 2 pertenece a un grupo de problemas más difíciles de resolver que los del tipo 1. Mientras que éstos son resolubles mediante una máquina de estados finitos, los primeros requieren una pila de almacenamiento virtualmente ilimitada.

Precisamente por encontrarse los autómatas finitos entre los sistemas más sencillos que se pueden utilizar en la teoría de lenguajes, éstos han sido aplicados con éxito en numerosas tareas de aprendizaje y de reconocimiento de patrones. Toda la información que nos

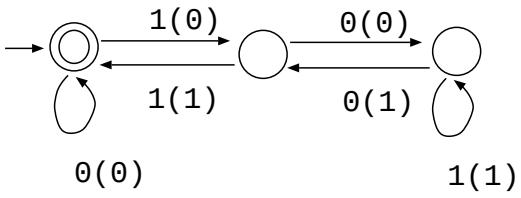

Figura 2.1: Máquina de estados finitos capaz de realizar el cociente de una cadena binaria por tres. El estado inicial aparece marcado por una flecha y la salida se forma a partir de los números entre paréntesis.

puede proporcionar un autómata finito es el estado en el que se encuentra después de haber procesado la entrada. Por ello, los estados se clasificarán en estados de aceptación y de no aceptación y servirán para clasificar las entradas en correctas e incorrectas. Los autómatas estocásticos aparecen cuando los cambios de estado del autómata se producen de forma aleatoria. En lo que sigue se introducen más detalladamente estos conceptos.

2.1 Alfabetos y lenguajes

Un *alfabeto* $\mathcal{A}$ es un conjunto finito y ordenado de símbolos. Por ejemplo: $\mathcal{A} = \{a, b, c, ..., z\}$ o el alfabeto binario $\mathcal{A} = \{0, 1\}$. El orden de sus elementos se denomina *ordenación alfabética*. Una secuencia de un número arbitrario (finito) de símbolos del alfabeto forma una *cadena, palabra* o *frase*. Por ejemplo, $w = aca$ significa que la cadena w está formada por una secuencia de tres símbolos del alfabeto: a, c y a. Dos secuencias compuestas de los mismos símbolos pero en diferente orden son distintas: $aca \neq caa$. El número de símbolos que componen la cadena es su longitud, y se representa habitualmente por $|w|$.

Al conjunto de todas las cadenas generables a partir del alfabeto $\mathcal{A}$ lo llamaremos *lenguaje universal* $\mathcal{A}^*$ y a la operación por la que se genera una cadena a partir de los símbolos que la componen, *con-*

catenación. Es posible formar cadenas de longitud arbitraria, y en particular, cabe la posibilidad de que la cadena contenga 0 símbolos. En ese caso, se trata de una *cadena vacía*, que se representará de forma especial mediante el símbolo λ y que satisface $\forall w \in \mathcal{A}^*$:

$$w\lambda = \lambda w = w. \tag{2.1}$$

Con la incorporación de λ, cuyas características son las de un elemento neutro, la concatenación dota de estructura de *monoide* a $\mathcal{A}^*$, es decir, se trata de una operación interna asociativa, con elemento neutro pero no conmutativa.

Dado que la concatenación es no conmutativa, se han de definir dos operaciones inversas, una por la izquierda y otra por la derecha. Así, si $w = xy$, siendo x, y, w palabras de $\mathcal{A}^*$, diremos que x es un *prefijo* de w mientras que y es un *sufijo* de w, y escribiremos $x = wy^{-1}$ o, alternativamente, $y = x^{-1}w$.

Por último, un *lenguaje* es un subconjunto cualquiera de cadenas de $\mathcal{A}^*$. Por ejemplo, las frases del castellano son un subconjunto de todas las cadenas que se pueden formar a partir del alfabeto habitual, y los signos de puntuación. También llamaremos, por analogía, lenguajes a conjuntos como el de las cadenas binarias que expresan un número par, o aquél que no contiene ninguna cadena o *conjunto vacío*, representado como $\emptyset$.

Los lenguajes que se pueden generar a partir de alfabetos finitos sólo pueden ser finitos o de cardinal infinito *numerable*. En efecto, si un lenguaje es infinito, siempre es posible establecer una ordenación efectiva en él, de forma que a cada cadena se le asigne un número de orden siguiendo la llamada *ordenación lexicográfica* o *canónica*. Ésta consiste en recorrer las palabras del lenguaje de menor a mayor longitud de las cadenas y dentro de cada longitud por el orden alfabético definido para el alfabeto $\mathcal{A}$. Por ejemplo, en el caso del alfabeto binario seguiría la sucesión $\mathcal{A}^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$.

Por último, la concatenación de palabras puede generalizarse para englobar también la de lenguajes: la *concatenación de dos lenguajes* L

y L' es un nuevo lenguaje formado por las cadenas obtenidas mediante concatenación de una cadena de L con otra de L', es decir,

$$LL' = \{w = xy : x \in L \wedge y \in L'\} \tag{2.2}$$

2.2 Autómatas finitos deterministas

Un autómata finito puede servir para discriminar palabras, observando el estado final en el cuál se encuentra. Por ejemplo, el autómata de la figura 2.1 permite clasificar las cadenas binarias de entrada que son múltiplo de tres y las que no, simplemente mirando cuál es el estado final después de la lectura. Los estados válidos son llamados *estados de aceptación* y aparecen marcados con doble círculo. El estado inicial (aquel en el que se encuentra el autómata antes de analizar la cadena) se señala con una flecha. Esta función como clasificadores de palabras permite que los autómatas finitos sean utilizados con frecuencia para el análisis léxico de programas o textos, es decir, para determinar qué palabras están correctamente escritas y pertenecen al lenguaje en cuestión, o para buscar determinados patrones en un texto.

Los elementos de un autómata finito determinista (DFA) son $A = (Q, \mathcal{A}, \delta, q_I, F)$, siendo:

- $Q = \{q_1, q_2, ..., q_N\}$ el conjunto finito de estados posibles del autómata;
- $\mathcal{A} = \{a_1, a_2, ..., a_L\}$ su alfabeto finito de entrada;
- $q_I \in Q$ su estado inicial;
- $F \subset Q$ el subconjunto de estados de aceptación;
- δ una *función de transición* entre estados de la forma $\delta : Q \times \mathcal{A} \rightarrow Q$ que dicta el comportamiento del autómata.

Un autómata finito opera ejecutando secuencias de movimientos o pasos según va leyendo los símbolos de la cadena de entrada. Cada

paso viene determinado por el estado actual del autómata y por el símbolo que se lee de la entrada. El movimiento consiste en leer ese símbolo (de manera que el siguiente símbolo de la cadena será leído en el siguiente movimiento) y cambiar al estado determinado por la función de transición δ.

Para describir formalmente el comportamiento del autómata sobre una cadena, debemos extender la función de transición $\delta : Q \times \mathcal{A} \to Q$ para que pueda actuar sobre un estado y una cadena y no sólo sobre un estado y un símbolo. Así, la nueva función es $\delta : Q \times \mathcal{A}^* \to Q$, de manera que $\delta(q, w)$ es el estado en el que se encontrará el autómata después de haber leído la cadena w empezando en el estado q y puede ser definida:

- $\delta(q, \lambda) = q$ (si no se lee ninguna cadena, es decir, se lee la cadena vacía, el autómata permanece en el mismo estado); y
- $\forall w \in \mathcal{A}^*, \forall a \in \mathcal{A}$ se cumple $\delta(q, wa) = \delta(\delta(q, w), a)$ (es decir, el estado al que llega el autómata al leer la cadena no vacía wa puede obtenerse por pasos: primero se calcula el estado $p = \delta(w, q)$ al que se llega con w desde q, y después se obtiene el estado al que se llega con el símbolo a desde p: $\delta(p, a)$).

Los autómatas definidos de esta forma son *deterministas*, en el sentido de que siempre está unívocamente definido cuál es el estado en que se encuentra el autómata después de haber leído una parte de la cadena de entrada. Si en algún caso no aparece definida la imagen de una transición entenderemos que la cadena no es válida o, alternativamente, que se dirige a un *estado de absorción* que recoge todas las transiciones no definidas y que no es un estado de aceptación.

Podemos definir el *lenguaje aceptado* por un DFA como:

$$L(A) = \{w \in \mathcal{A}^* : \delta(q_I, w) \in F\} \tag{2.3}$$

Por tanto, el DFA nos permite determinar si una cierta cadena w pertenece o no a $L(A)$ simplemente observando si el estado final al que se

llega mediante w partiendo de q_I es uno de los estados del subconjunto F. Todos los lenguajes reconocibles mediante autómatas finitos deterministas reciben el nombre de *lenguajes regulares* o también *lenguajes racionales*. Por tanto, dado un lenguaje L, si existe un DFA A tal que $L = L(A)$, entonces L es un lenguaje regular.

2.3 Gramáticas regulares

Una forma diferente de introducir el concepto de lenguaje es a través del uso de gramáticas generativas, siguiendo las ideas de Chomsky. Admitamos que cualquier lenguaje está constituido por unidades de información completas que reciben el nombre de oraciones o frases[1], lo cual es una aproximación en el caso de los lenguajes naturales, debido a la gran importancia del contexto. Por ejemplo, la mayoría de las *oraciones* en castellano constan de un *sujeto* y de un *predicado*. A estos términos abstractos como *oración*, *sujeto* o *predicado* les llamaremos *variables* del lenguaje y nunca aparecen en una frase construida como tales. La abstracción de las características de estos términos o variables es lo que se llama una *regla de derivación*, *regla de producción*, o simplemente *producción*. Por ejemplo, el hecho de que una *oración* consta de *sujeto* y *predicado* es una regla de producción y lo escribiremos:

$$\textit{oración} \rightarrow \textit{sujeto} \quad \textit{predicado}$$

A la variable *oración* se le denomina *símbolo inicial* o *axioma*, ya que debe estar en el origen de cualquier generación de una frase. Todas estas variables son abstracciones, y al final, deben ser sustituidas por las palabras concretas del lenguaje. A estos símbolos o palabras que pueden aparecer en una frase terminada les llamamos *terminales*, y al proceso por el cual se llega desde el concepto abstracto de *oración* hasta la frase particular se le denomina *derivación*. Por ejemplo:
oración $\Rightarrow$ *sujeto predicado* $\Rightarrow$ *modificador núcleo predicado* $\Rightarrow$ *modificador núcleo verbo complemento* $\Rightarrow$ el *núcleo verbo complemento* $\Rightarrow$

[1] que en la notación formal usada aquí, serán cadenas de símbolos.

el ordenador *verbo complemento* $\Rightarrow$ el ordenador está *complemento* $\Rightarrow$ el ordenador está averiado
es una derivación de la frase "el ordenador está averiado" donde se ha aplicado un subconjunto de reglas de producción (entre las muchas posibles del castellano):

- *oración* $\rightarrow$ *sujeto predicado*
- *sujeto* $\rightarrow$ *modificador núcleo*
- *predicado* $\rightarrow$ *verbo complemento*
- *modificador* $\rightarrow$ el
- *núcleo* $\rightarrow$ ordenador
- *verbo* $\rightarrow$ está
- *complemento* $\rightarrow$ averiado

Como se puede apreciar, se ha hecho la suposición de que las variables se sustituyen por otros términos independientemente del lugar donde aparezcan. Por ello, a este tipo de mecanismos de generación se les denomina gramáticas independientes del contexto. Una *gramática independiente del contexto* está definida por $G = (V_N, V_T, P, S)$, siendo $V_N = \{A, B, C, ...\}$ un conjunto finito de símbolos variables (no terminales), $V_T = \mathcal{A} = \{a, b, ...\}$ un conjunto finito de símbolos terminales o alfabeto, $S \in V_N$ el símbolo o variable inicial y P un conjunto finito de reglas de derivación. Al conjunto de símbolos, tanto variables como terminales, lo representaremos como $V = V_N \cup V_T$. Las reglas de derivación $r \in R$ indican cómo sustituir variables y se pueden expresar como aplicaciones $r : V_N \rightarrow V^*$, siendo V^* todas las cadenas formadas por símbolos de V.

Las cadenas generadas por una gramática son aquellas que se pueden obtener partiendo de S mediante la aplicación sucesiva de reglas de derivación de R. A este conjunto de cadenas derivables desde S se le llama *lenguaje generado por la gramática* G:

$$L(G) = \{w \in \mathcal{A}^* : S \stackrel{\star}{\Rightarrow} w\} \tag{2.4}$$

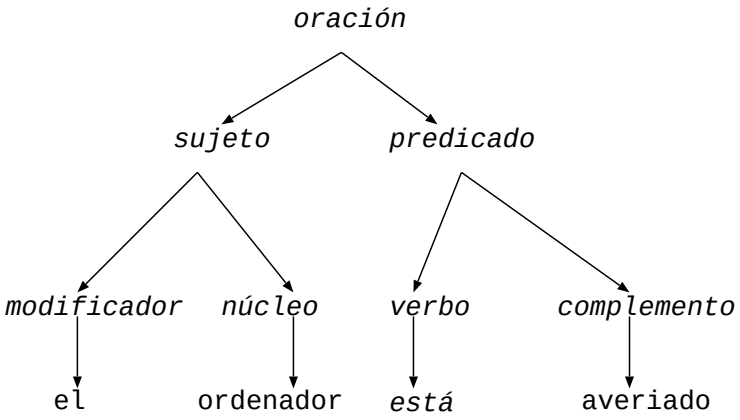

Figura 2.2: Árbol de derivación.

donde el asterisco indica una derivación en un número cualquiera de pasos. A la representación gráfica de este proceso mediante un árbol, se le llama *árbol de derivación*. La figura 2.2 representa el árbol de derivación correspondiente al ejemplo anterior.

Llamaremos *gramáticas regulares* a un subconjunto de las gramáticas independientes del contexto cuyas reglas de derivación pueden ser escritas usando únicamente los prototipos siguientes:

$$\begin{aligned} r: \quad & V_N \quad \rightarrow V_T V_N \\ r: \quad & V_N \quad \rightarrow \lambda \end{aligned} \tag{2.5}$$

Por ejemplo, $A \longrightarrow a$, o también $A \longrightarrow aB$. Como su propio nombre indica, todo lenguaje generable mediante una de estas gramáticas es regular y recíprocamente, todos los lenguajes regulares pueden generarse mediante una gramática regular (véase, por ejemplo, Hopcroft & Ullman 1979).

2.4 Autómatas finitos deterministas estocásticos

Un autómata finito determinista estocástico A asocia a cada cadena de $\mathcal{A}^*$ una cierta probabilidad de aparición. Para ello, se incorpora a la definición de A una función que asigna una probabilidad a cada una de las transiciones. Formalmente $A = (Q, \mathcal{A}, \delta, q_I, p)$, donde p es una función probabilística $p : Q \times \mathcal{A} \to [0, 1]$. Por ejemplo, en el autómata de la figura 2.3 las probabilidades de transición aparecen representadas sobre el arco correspondiente por los valores entre paréntesis. Debe notarse que a diferencia del caso de los autómatas no estocásticos, no se incluye en la definición una lista de estados de aceptación. La probabilidad de generar una cadena $w \in \mathcal{A}^*$ viene dada por el producto de las probabilidades de las transiciones que ejerce el camino, incluida la probabilidad de terminación en el último nodo, que es el resto hasta uno de la suma de las probabilidades de transición. En el ejemplo de la figura, el autómata asigna una probabilidad 0.075 a la cadena 11 y una probabilidad 0 a la cadena 101.

Para calcular las probabilidades anteriores es necesario conocer la probabilidad $p(q_i, \lambda)$ de terminación en cada nodo q_i. Esta probabilidad puede obtenerse a partir de la función $p(q_i, a)$ definida anteriormente. En efecto, si el autómata se encuentra en un estado q_i, el siguiente paso se determina aleatoriamente según las probabilidades $p(q_i, a)$ y $p(q_i, \lambda)$. Como las probabilidades de los sucesos que pueden ocurrir a continuación han de sumar uno, podemos escribir:

$$p(q_i, \lambda) + \sum_{a \in \mathcal{A}} p(q_i, a) = 1 \tag{2.6}$$

Por tanto, los valores de $p(q_i, \lambda)$ se pueden obtener a partir de la función p mediante la relación anterior. En el ejemplo de la figura 2.3, $p(q_1, \lambda) = 0.5$ y $p(q_2, \lambda) = p(q_3, \lambda) = 0$. Con estos valores se obtiene, por ejemplo, que la probabilidad de la cadena 11 es $p(11|A) = p(q_1, 1)\ p(q_2, 1)\ p(q_1, \lambda) = 0.075$.

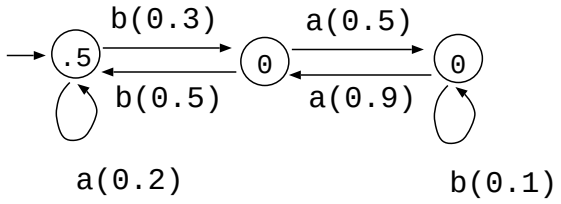

Figura 2.3: Autómata estocástico.

Si el autómata A no contiene nodos inútiles (estados desde los cuales no es posible llegar con probabilidad no nula a algún estado tal que $p(q_i, \lambda) > 0$), el autómata define una distribución de probabilidad $p(w|A)$ para las cadenas de $\mathcal{A}^*$, de forma que:

$$\sum_{w \in \mathcal{A}^*} p(w|A) = 1. \tag{2.7}$$

2.5 Autómatas finitos de árboles

Los árboles constituyen una estructura de orden superior a las cadenas, en el sentido de que se establecen relaciones más complejas entre sus componentes elementales. En una cadena, una vez definido un sentido de lectura privilegiado (habitualmente de izquierda a derecha) cada símbolo puede relacionarse con un antecesor único o con un sucesor también único. En el caso de los árboles, cada nodo puede tener un sólo *antecesor*, pero varios *descendientes*. El único nodo sin antecesores es la *raíz* del árbol. A los nodos que no tienen descendientes se les denomina *hojas* del árbol. Al igual que cada posición de la cadena lleva asociado un símbolo, los nodos del árbol pueden contener etiquetas. La figura 2.4 representa un árbol con 6 nodos. El nodo superior etiquetado con a es la raíz del árbol, y otros tres nodos son del tipo hoja (uno de los etiquetados con b y los dos con c).

Una forma compacta de codificar estos árboles etiquetados es mediante la llamada *notación funcional*. En ella, cada nodo se representa

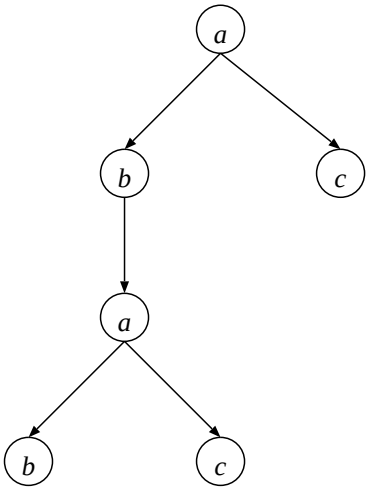

Figura 2.4: Árbol etiquetado.

como una función, cuyo nombre es la etiqueta que contiene y cuyos argumentos son los subárboles que genera cada descendiente. Por ejemplo, el árbol 2.4 se representaría en la notación funcional como $a(b(a(bc))c)$.

En un autómata de cadenas como los definidos anteriormente, se procesa la cadena de entrada de izquierda a derecha y para cada lectura y cada estado en que se encuentra el autómata se produce una transición a un nuevo estado. Cuando se ha agotado la cadena de entrada, el estado final en el que se encuentra el autómata determina la validez o no de la cadena.

En el caso de los árboles, comenzaremos a analizar el árbol por sus hojas, es decir, estableceremos un orden de lectura ascendente. En función de la etiqueta del nodo y de los estados de los descendientes se producirá una transición a un nuevo estado. Al llegar a la raíz del árbol, el estado en que se encuentra el autómata determina si el árbol es aceptado o no.

Por tanto, nos encontramos ante un mecanismo muy similar al de los autómatas finitos deterministas para cadenas al que llamare-

mos *autómata finito determinista de árboles*. Cada autómata $A = (Q, V, \delta, F)$ de este tipo constará de:

- un conjunto finito de estados $Q = \{q_1, q_2, ..., q_N\}$;
- un alfabeto finito de etiquetas para los nodos de los árboles $V = \{a_1, a_2, ..., a_L\}$;
- un subconjunto de estados de aceptación $F \subset Q$;
- un conjunto de funciones de transición $\delta = \{\delta_0, \delta_1, ...\delta_n\}$.

Dado que el número de descendientes de un nodo es variable, para describir cómo se producen las transiciones es necesario proporcionar no una sino una colección de funciones de transición $\delta = \{\delta_0, \delta_1, ...\delta_n\}$, donde n es el máximo número de descendientes admitido en el lenguaje. Cada función δ_k toma como argumentos un símbolo de V y k estados (uno por cada descendiente del nodo), es decir, $\delta_k : V \times Q^k \to Q$.

Por ejemplo, si t es una subárbol hoja con etiqueta a, entonces $k = 0$ y el estado asociado a t es $\delta(t) = \delta_0(a)$. Sin embargo, si t es un subárbol etiquetado f y con tres descendientes que generan respectivamente los subárboles t_1, t_2 y t_3, entonces $t = f(t_1, t_2, t_3)$ y el estado asociado a la raíz del subárbol es $\delta(t) = \delta_3(f, t_1, t_2, t_3)$. Como en el caso de los lenguajes de cadenas, las transiciones no contenidas en la definición conducen a árboles no aceptados. El lenguaje reconocido por el autómata A es entonces:

$$T(A) = \{t \in V^T : \delta(t) \in F\} \tag{2.8}$$

donde V^T representa todos los árboles que se pueden construir con nodos etiquetados con V.

De forma análoga al caso de las cadenas, llamamos *lenguajes racionales de árboles* a todos aquellos para los que existe un autómata finito determinista de árboles A que reconoce el lenguaje, esto es, T es racional si existe A tal que $T = T(A)$.

2.6 Autómatas estocásticos de árboles

Al igual que los autómatas finitos deterministas de cadenas pueden incorporar una función de probabilidad para generar lenguajes estocásticos, también es posible asignar una probabilidad a cada transición de un autómata finito determinista de árboles utilizando para ello un conjunto de funciones de probabilidad $p_k : V \times Q^k \rightarrow [0,1]$. En este caso, la condición de normalización es que las probabilidades de las transiciones que conducen a un mismo estado q han de sumar 1. Si denotamos, por simplicidad los estados del autómata mediante su ordinal, $Q = \{1, 2, ..., N\}$, entonces, para todo $q \in Q$

$$\sum_{k=0}^{n} \sum_{f \in V} \sum_{\substack{q_1, q_2, ..., q_k \in Q \\ q = \delta(f, q_1, q_2, ..., q_k)}} p_k(f, q_1, ..., q_k) = 1. \tag{2.9}$$

Dadas las funciones p_k, la probabilidad de generar un árbol t se obtiene efectuando el producto de las probabilidades de transición utilizadas en el análisis de t. Sin embargo, con esta definición no está garantizado que la suma de todas las probabilidades para todos los árboles sumen 1. Para que esta propiedad se satisfaga, es necesario además asignar una probabilidad $r(q)$ a cada nodo de que éste aparezca como raíz. De esta forma la probabilidad global del árbol t aparece además multiplicada por $r(q)$ si $q = \delta(t)$ es el estado asociado a la raíz, es decir, $p(t|A) = r(\delta(t))\ p(t|\delta(t))$, donde $p(t|\delta(t))$ representa la probabilidad de generar t a partir del estado $q = \delta(t)$, que debe calcularse recursivamente. Por ejemplo, para un árbol $t = f(t_1, t_2)$, obtendríamos

$$p(t|\delta(t)) = p_2(f, \delta(t_1), \delta(t_2))\ p(t_1|\delta(t_1))\ p(t_2|\delta(t_2)). \tag{2.10}$$

Siempre que

$$\sum_{q \in Q} r(q) = 1 \tag{2.11}$$

el autómata estocástico de árboles $A = (Q, V, \delta, p, r)$ establece un distribución de probabilidad sobre el lenguaje V^T, es decir,

$$\sum_{t \in V^T} p(t|A) = 1. \tag{2.12}$$