Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3725 – Traductores e Interpretadores
Abril – Julio 2012

Guía Corta:
Minimización (Aproximada) de AFD's usando Derivadas

1 Derivada de una Expresión Regular

Recordemos la definición de la derivada de un lenguaje. Sea Σ un alfabeto, L un Lenguaje sobre Σ y a un símbolo en Σ; se define la derivada D_aL, de L con respecto a a, como:

$$D_aL = \{w | aw \in L\}$$

Definiremos ahora el concepto de derivada de una Expresión Regular. Sea Σ una alfabeto, e una Expresión Regular sobre Σ y a un símbolo en Σ; se define la derivada D_ae, de e con respecto a a mediante las siguientes ecuaciones:

$$\begin{aligned}
D_a\emptyset &= \emptyset \\
D_a\lambda &= \emptyset \\
D_ab &= \begin{cases} \lambda & \text{si } a = b \\ \emptyset & \text{si } a \neq b \end{cases} \qquad \forall b \in \Sigma \\
D_a(s+t) &= D_as + D_at \\
D_a(st) &= (D_as)t + \Lambda(s)D_at \\
D_a(s*) &= (D_as)s*
\end{aligned}$$

En la definición anterior Λ es una función que evalúa en λ si el argumento pasado reconoce λ y $\emptyset$ de lo contrario. Podemos definir este comportamiento a través de las siguientes ecuaciones:

$$\begin{aligned}
\Lambda\emptyset &= \emptyset \\
\Lambda\lambda &= \lambda \\
\Lambda a &= \emptyset \qquad \forall a \in \Sigma \\
\Lambda(s+t) &= \Lambda s + \Lambda t \\
\Lambda(st) &= \Lambda s\ \Lambda t \\
\Lambda(s*) &= \lambda
\end{aligned}$$

Nótese que el papel que juega entonces Λ en la ecuación $D_a(st) = (D_as)t + \Lambda(s)D_at$, es la de incluir D_at si $\lambda \in sem(s)$. Esto, ya que de ser así $sem(st)$ incluiría todas las palabras en $sem(t)$ y por tanto deben derivarse estas últimas también.

La definición de derivada sobre una Expresión Regular, análogamente puede extenderse para tratar con frases de un alfabeto Σ, además de símbolos. Tal extensión vendría definida por las siguientes ecuaciones:

$$D_\lambda e = e$$

$$D_{wa} e = D_a(D_w e), \text{ con } a \in \Sigma, w \in \Sigma^*$$

Habiendo definido la derivada de una Expresión Regular, es importante notar la siguiente propiedad:

$$sem(D_w e) = D_w sem(e)$$

Lo cual nos asegura que derivar una Expresión Regular es equivalente al haber derivado sobre la semántica de la misma. Sin embargo, es más sencillo derivar la Expresión Regular, ya que su representación es única (dos Expresiones Regulares pueden ser *equivalentes* si su semántica es igual, pero solo son *iguales* si son la misma Expresión Regular).

2 Autómata Finito Determinista usando Derivadas

Construiremos ahora un Autómata Finito Determinista, basado en la noción de derivadas. Notemos primero que derivar una Expresión Regular e, bajo un símbolo $a \in \Sigma$, resulta en una nueva Expresión Regular que representa lo que quedaría por ver, habiendo leído a.

Si la semántica de una expresión resultante contiene λ, entonces está permitido no leer más y la palabra construida a través de los pasos de derivación pertenece al lenguaje. Esto último se puede ver a través de la siguiente propiedad:

$$w \in sem(e) \equiv \lambda \in sem(D_w e)$$

Esta noción nos lleva a ver una Expresión Regular directamente como un Autómata Finito Determinista, donde las transiciones son dadas por cada derivada y los estados finales son aquellos donde la semántica de la expresión resultante contenga λ.

Formalmente, dada una expresión regular e sobre un alfabeto Σ, construiremos el Autómata Finito Determinista M de la siguiente manera:

$$M_e = (\Sigma, Q, \delta, q_0, F), \text{ donde}$$

- Σ es el alfabeto sobre el cual estaba e.
- $Q = \{D_w e \mid w \in \Sigma^*\}$ es el conjunto de las derivadas sobre todas las posibles frases.

- $(\forall q, a : q \in Q \wedge a \in \Sigma : \delta(q, a) = D_a q)$. La función de transición con a se define por la derivada sobre a.
- $q_0 = D_\lambda e$, estado en el que aún no se ha leído nada.
- $F = \{q \mid q \in Q \wedge \Lambda q = \lambda\}$, estados cuya semántica incluye a λ.

El conjunto de estados propuesto, $Q = \{D_w e \mid w \in \Sigma^*\}$, parece ser infinito ya que hay infinitas posibles frases $w \in \Sigma^*$. Sin embargo, éste no es el caso. Las derivadas sobre las frases se repiten, asegurando que la cantidad de derivadas resultantes sea finita. Esto, ya que si no fueran finitas, en realidad la Expresión Regular original no tenía una forma finita y por ende no era realmente Regular. [1]

2.1 Construcción Sistemática

Una vez sabemos la estructura que tendrá el Autómata Finito M_e, sabiendo que la cantidad de estados es finita, procedemos a descubrir cuáles son dichos estados. Para ello, se dará una forma de construcción sistemática basada en prefijos.

La propiedad más importante a considerar para el correcto funcionamiento de este proceso es la siguiente:

$$(\forall x, y, z : x, y, z \in \Sigma^* \wedge D_x e = D_{yz} e : D_{xz} e = D_{yz} e)$$

En otras palabras, si una derivada sobre una frase w es igual a otra derivada ya encontrada, entonces calcular las derivadas sobre frases que tengan como prefijo a w sería redundante, pues ya se habrían calculado.

El algoritmo entonces sería el siguiente:

1. Se crean dos conjuntos $Q = \emptyset$ y $S = \{D_\lambda e\}$
2. Mientras $S \neq \emptyset$
   - 2.1) Tomar un elemento $q \in S$:
   - 2.2) $Q = Q \cup \{q\}$
   - 2.3) $S = S - \{q\}$
   - 2.4) $S = S \cup \{D_a q \mid a \in \Sigma \wedge D_a q \notin Q\}$
3. Devolver Q

[1] De hecho, si la Expresión original era infinita, esta máquina aún es válida. Sin embargo, el resultado ya no es un Autómata Finito sino una generalización del mismo conocido como una *Máquina de Estados*. Estas máquinas liberan la restricción de la finitud del conjunto de estados. Estos no tienen representación gráfica simple, pero si diversas aplicaciones prácticas (Por ejemplo, moverse a través de un plano en algún espacio cartesiano buscando un punto que cumpla con alguna propiedad en particular). Sin embargo, no son parte del alcance del curso de Traductores e Interpretadores.

En el algoritmo anterior, Q es el conjunto de estados definitivos y S es el conjunto de estados candidatos. (Puede verse como análogo a la noción de *caminos cerrados* y *caminos abiertos* del Modelo General de Etiquetamiento.)

Este algoritmo nos devuelve el conjunto de estados del autómata deseado. La función de transición, los estados finales y el estado inicial se pueden conseguir ahora a partir de la definición de M_e, presentada anteriormente.

2.2 Ejemplo detallado

Usaremos el algoritmo visto, ahora sobre un ejemplo concreto. Sea $e = (0+1)*11$ la Expresión Regular, sobre el alfabeto $\Sigma = \{0, 1\}$, cuya semántica es la de todas las frase compuestas de 0's y 1's, tal que terminen en al menos dos 1's.

Construiremos ahora las derivadas sobre todos los posibles prefijos que den resultados diferentes, tomando los mismos en orden lexicográfico. Primeramente el caso base y el estado inicial del Autómata resultante:

$$D_\lambda e = e$$

Ahora, se derivará sobre la frase que solo contiene al símbolo a:

$$\begin{aligned} D_0 e &= D_0((0+1)*11) \\ &= D_0((0+1)*) + \Lambda((0+1)*)D_0(11) \\ &= D_0(0+1)(0+1)*11 + D_0(11) \\ &= (D_0 0 + D_0 1)(0+1)*11 + \emptyset \\ &= (\lambda + \emptyset)(0+1)*11 \\ &= (0+1)*11 \\ &= e \\ &= D_\lambda e \end{aligned}$$

Nótese que $D_0 e = D_\lambda e$, por lo que ambos estados son equivalente y continuar con frases que tengan a a como prefijo sería redundante. Se derivará ahora sobre la frase que solo contiene al símbolo b:

$$\begin{aligned} D_1 e &= D_1((0+1)*11) \\ &= D_1((0+1)*) + \Lambda((0+1)*)D_1(11) \\ &= D_1(0+1)(0+1)*11 + D_1(11) \\ &= (D_1 0 + D_1 1)(0+1)*11 + 1 \\ &= (\emptyset + \lambda)(0+1)*11 + 1 \\ &= (0+1)*11 + 1 \\ &= e + 1 \end{aligned}$$

Como $D_1 e$ es una expresión que aún no se había obtenido, se agrega el mismo al conjunto de estados definitivo y se continúa derivando sobre las frases que tengan 1 como prefijo.

$$\begin{aligned} D_{10}e &= D_0(D_1) \\ &= D_0(e+1) \\ &= D_0e + D_01 \\ &= e + \emptyset \\ &= e \\ &= D_\lambda e \end{aligned}$$

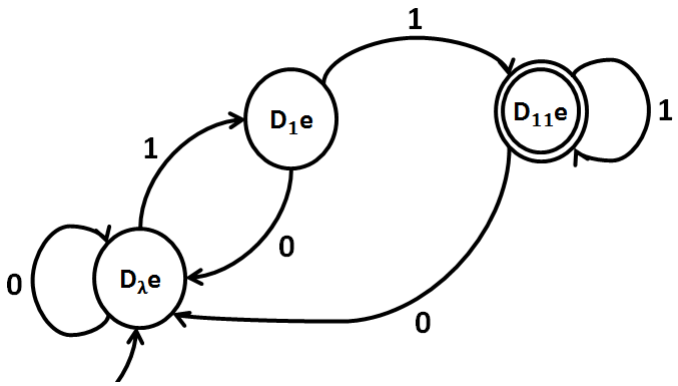
$$\begin{aligned} D_{11}e &= D_1(D_1) \\ &= D_1(e+1) \\ &= D_1e + D_11 \\ &= e + 1 + \lambda \end{aligned}$$

$$\begin{aligned} D_{110}e &= D_0(D_11) \\ &= D_0(e+1+\lambda) \\ &= D_0e + D_01 + D_0\lambda \\ &= e + \emptyset + \emptyset \\ &= e \\ &= D_\lambda e \end{aligned}$$

$$\begin{aligned} D_{111}e &= D_1(D_11) \\ &= D_1(e+1+\lambda) \\ &= D_1e + D_11 + D_1\lambda \\ &= e + 1 + \lambda + \emptyset \\ &= e + 1 + \lambda \\ &= D_{11}e \end{aligned}$$

Como ya no quedan prefijos posibles, el algoritmo termina. El conjunto de estados resultante es $Q = \{D_\lambda e, D_1 e, D_{11} e\}$, el estado inicial es $D_\lambda e$ y el conjunto de estados finales es $F = \{D_{11}e\}$, ya que es el único donde $\Lambda D_{11} e = \lambda$. La función de transición viene de derivar cada uno de los estados, con cada símbolo en Σ y ver a que otro estado lo lleva.

El autómata resultante para nuestro ejemplo sería:



2.3 Minimalidad del Autómata Conseguido

Como lo dice el título de la presente guía, el Autómata Finito construido a partir de este método, para la Expresión Regular e, es una *aproximación* al Autómata Finito Determinista Mínimo para $sem(e)$. Sin embargo, el método otorga en general aproximaciones cercanas aplicando un método mucho menos elaborado que el algoritmo tradicional (también conocido como algoritmo de Hopcroft).

Por lo tanto, si se pide que se halle el Autómata Finito Determinista Mínimo para $sem(e)$, no basta con aplicar el presente algoritmo. Una vez terminado, se debe aplicar el algoritmo tradicional de minimización sobre el resultado de este. Muy posiblemente, sin embargo el autómata dado ya sea mínimo o la cantidad de pasos de separación de clases sea muy pequeña, por lo que el esfuerzo requerido en esta segunda fase es *relativamente pequeño*.

3 Ejercicios sugeridos

1. Sobre el alfabeto $\Sigma = \{0, 1\}$, sea $e = 0(0 + 1)* 1(0 + 1)* 0$. Halle el Autómata Finito Determinista con el método visto, usando derivadas.
2. Sobre el alfabeto $\Sigma = \{a, b\}$, sea $e = a(ab + ba)* b$. Halle el Autómata Finito Determinista con el método visto, usando derivadas.
3. Demuestre que $\mathcal{L}(M_e) = sem(e)$.

   Pista: *La propiedad* $sem(D_w e) = D_w sem(e)$, podría serle util.