Autómatas y Lenguajes Formales 2019-II
Facultad de Ciencias UNAM*
Nota de Clase 2: Lenguajes y expresiones regulares

Favio E. Miranda Perea A. Liliana Reyes Cabello Lourdes González Huesca

17 de febrero de 2019

1. Lenguajes Regulares

El estudio de los lenguajes como conjuntos de cadenas debe ser susceptible a realizarse de manera formal mediante una abstracción conveniente.

Una forma para caracterizar lenguajes sencillos, de fácil descripción, es utilizar operaciones sobre cadenas, éstas permiten la *generación* de nuevos lenguajes. Los lenguajes regulares son los conjuntos de cadenas más simples.

Definición 1. *Un lenguaje $L \subseteq \Sigma^\star$ es regular si es generado a partir de los lenguajes básicos $\varnothing, \{\varepsilon\}, \{a\}$ (donde $a \in \Sigma$) y mediante las operaciones de unión, concatenación y estrella de Kleene. Recursivamente:*

- *$\varnothing$ y $\{\varepsilon\}$ son lenguajes regulares.*
- *Si $a \in \Sigma$ entonces $\{a\}$ es regular.*
- *Si L, M son regulares entonces $L \cup M, LM$ y $L^\star$ son regulares.*
- *Son todos.*

El conjunto de los lenguajes regulares será denotado por REG.

Ejemplos:

- Cualquier lenguaje *finito* es regular.
- En efecto si $L = \{w_1, \ldots, w_n\} \subseteq \Sigma^\star$ entonces $L = L_1 \cup L_2 \cup \ldots \cup L_n$ con $L_i = \{w_i\}$. Cada lenguaje L_i es regular puesto que si $w_i = a_1 \ldots a_{k_i}$ con $a_j \in \Sigma$ entonces $L_i = \{a_1\}\{a_2\}\ldots\{a_{k_i}\}$ y cada $\{a_j\}$ es regular por definición.
- En particular cualquier alfabeto Σ es un lenguaje regular.

*Material elaborado en el marco del proyecto PAPIME PE102117

- Sobre el alfabeto $\Sigma = \{a, b\}$ tenemos los siguientes lenguajes regulares *infinitos*:
  - $L_1 = \{w \in \Sigma^\star \mid w \text{ empieza con } \ b\} = \{b\}\Sigma^\star$
  - $L_2 = \{w \in \Sigma^\star \mid w \text{ contiene exactamente una } \ a\} = \{b\}^\star\{a\}\{b\}^\star$
  - $L_3 = \{w \in \Sigma^\star \mid w \text{ contiene la subcadena } \ ba\} = \Sigma^\star\{ba\}\Sigma^\star$
  - $L_4 = \{w \in \Sigma^\star \mid w \text{ contiene exactamente dos } b\} = \{a\}^\star\{b\}\{a\}^\star\{b\}\{a\}^\star$
  - $L_5 = \{w \in \Sigma^\star \mid w \text{ contiene un número par de } a\} = \{b\}^\star \cup \left(\{b\}^\star\{a\}\{b\}^\star\{a\}\{b\}^\star\right)^\star$
- No todo lenguaje infinito es regular

Problemas de interés acerca de lenguajes regulares Dado un lenguaje regular $L \subseteq \Sigma^\star$ existen diversos problemas de decisión (es decir, problemas cuya respuesta es sí o no) acerca de L, enunciamos algunos de ellos que tienen solución algorítmica.

1. Problema de vacuidad: ¿Es L el lenguaje vacío?
2. Problema de finitud: ¿Es L finito ?
3. Problema de la pertenencia: Dada $w \in \Sigma^\star$, ¿$w \in L$?
4. Problema de la equivalencia: Dado $M \subseteq \Sigma^\star$, ¿$M = L$?

1.1. Expresiones Regulares

Los lenguajes regulares se pueden describir por medio de una expresión que generaliza la forma de las cadenas en el lenguaje, es decir una representación de las cadenas. Un mecanismo de suma importancia para denotar lenguajes regulares es por medio de las llamadas expresiones regulares, que son *fórmulas* explícitas de la cadena característica del lenguaje.

Definición 2. *Las expresiones regulares están definidas recursivamente como sigue:*

- $\varnothing$ *es una expresión regular.*
- ε *es una expresión regular.*
- *Si* $a \in \Sigma$ *entonces* a *es una expresión regular.*
- *Si* α, β *son expresiones regulares entonces* $\alpha\beta$, $\alpha + \beta$, $\alpha^\star$ *son expresiones regulares.*
- *Son todas.*

[START] Ejemplo

Ejemplo: Sea $\Sigma = \{0, 1\}$,

- $(0+1)^\star$ Todas las cadenas de 0's y 1's.
- $01^\star$ Un 0 seguido cualquier número de 1's.
- $0(0+1)^\star$ Todas las cadenas de 0's y 1's, empezando con un 0.
- $(0+1)^\star 01^\star 01^\star$ Todas las cadenas de 0's y 1's que contienen al menos dos 0's.

[END] Ejemplo

Toda expresión regular denota a un lenguaje definido como sigue:

Definición 3. *El lenguaje definido por una expresión regular r denotado por $\mathcal{L}[\![r]\!] \subseteq \Sigma^*$ se define como:*

$$\begin{aligned}
\mathcal{L}[\![\varnothing]\!] &= \varnothing \\
\mathcal{L}[\![\varepsilon]\!] &= \{\varepsilon\} \\
\mathcal{L}[\![a]\!] &= \{a\} \\
\mathcal{L}[\![r+s]\!] &= \mathcal{L}[\![r]\!] \cup \mathcal{L}[\![s]\!] \\
\mathcal{L}[\![rs]\!] &= \mathcal{L}[\![r]\!]\mathcal{L}[\![s]\!] \\
\mathcal{L}[\![r^*]\!] &= \mathcal{L}[\![r]\!]^*
\end{aligned}$$

A la operación $\mathcal{L}[\![\cdot]\!]$ se le conoce como denotación.

[START] Ejemplo

Ejemplos: Sea $\Sigma = \{a, b, c\}$,

$$\begin{aligned}
\mathcal{L}[\![aa^\star]\!] &= \{a\}\{a\}^\star &&= \{a, aa, aaa, aaaa, \ldots\} \\
\mathcal{L}[\![a(b+c)d]\!] &= \{a\}\{\{b\} \cup \{c\}\}\{d\} &&= \{abd, acd\} \\
\mathcal{L}[\![(a+b)^\star]\!] &= \{a \cup b\}^\star &&= \{\varepsilon, a, b, ab, ba, abb, aba, \ldots\} \\
\mathcal{L}[\![ab^\star c]\!] &= \{a\}\{b\}^\star\{c\} &&= \{ac, abc, abbc, abbbc, \ldots\} \\
\mathcal{L}[\![a^\star + b^\star + c^\star]\!] &= \{a\}^\star\{b\}^\star\{c\}^\star &&= \{\varepsilon, a, b, c, \ldots, a^k, b^k, c^k, \ldots\} \\
\mathcal{L}[\![\varepsilon]\!] &= \mathcal{L}[\![\varnothing^\star]\!] &&= \{\varepsilon\} \\
\mathcal{L}[\![(a+b)c]\!] &= (\{a\} \cup \{b\})\{c\} &&= \{ac, bc\}
\end{aligned}$$

[END] Ejemplo

Más aún todo lenguaje regular puede denotarse con una expresión regular como lo asegura la siguiente

Proposición 1. *Un lenguaje $L \subseteq \Sigma^*$ es regular si y sólo si existe $r \in ER$ tal que $L = L[\![r]\!]$*

Demostración. Para $\Rightarrow$) hacemos inducción estrucrural sobre REG. Para $\Leftarrow$) inducción estructural sobre r. □

De acuerdo a la proposición anterior en adelante identificaremos a una expresión regular r con $L\,[\![r]\!]$, por ejemplo si $L = \{\varepsilon, 0, 00, 000, \ldots, 0^n, \ldots 1\}$ escribiremos $L = 0^*+1$ en vez de $L = \mathcal{L}[\![0^*+1]\!]$.

Un lenguaje regular puede ser denotado por más de una expresión regular por lo que necesitamos una noción de equivalencia, diremos que r y s son equivalentes si denotan exactamente al mismo lenguaje, es decir,

Definición 4. *Decimos que dos expresiones regulares $\alpha, \beta \in ER$ son equivalentes, y usamos la notación $\alpha = \beta$, si y sólo si $L[\![\alpha]\!] = L[\![\beta]\!]$.*

[START] Ejemplo

Ejemplo: Sean r y s expresiones regulares, donde $r = \varepsilon + (0+1)^\star 1$ y $s = (0^\star 1^\star)$. Por demostrar que r y s son equivalentes.

Demostración.

($\subseteq$) Sea $w \in \mathcal{L}[\![r]\!] = \varepsilon + (0+1)^\star 1$.

Si $w = \varepsilon$, entonces $w \in \mathcal{L}[\![s]\!]$ por definición.

Si $w \neq \varepsilon$ entonces w puede ser representado como $w = w'1$ donde $w' \in \mathcal{L}[\![(0+1)^\star]\!]$. Supongamos que w' contiene k instancias del símbolo 1. Esto significa que w' puede ser escrito de la forma $w' = x_1 1 x_2 \dots x_k 1 x_{k+1}$ donde $x_i \in 0^\star$, pero entonces $w = w'1 = (x_1 1)(x_2 1)\dots(x_{k+1}1) = (0^\star 1)(0^\star 1)\dots(0^\star 1)$.

Teniendo así que $w \in \mathcal{L}[\![(0^\star 1)^\star]\!]$.

($\supseteq$) Sea $w \in \mathcal{L}[\![s]\!] = (0^\star 1)^\star$.

Entonces $w = \varepsilon$ ó $w = x_1 x_2 \ \dots x_n$, con $n > 0$ tal que $x_i \in \mathcal{L}[\![0^\star 1]\!]$.

Si $w = \varepsilon$, entonces $w \in \mathcal{L}[\![r]\!]$.

Si $w = x_1 x_2 \dots x_n$ entonces se puede representar a $w = w'1 = x_1 x_2 \dots x_{n-1} 0^z 1$ con $z \geq 0$.

Sin embargo $w' = x_1 x_2 \dots x_{n-1} 0^z \in \mathcal{L}[\![(0+1)^\star]\!]$, y se sigue que $w'1 \in \mathcal{L}[\![(0+1)^\star]\!]1 \subseteq \mathcal{L}[\![r]\!]$.

□

[END] Ejemplo

2. Axiomas de Salomaa

Las siguientes equivalencias, conocidas como Axiomas de Salomaa, constituyen un sistema completo para el razonamiento ecuacional con expresiones regulares. Es decir, cualquier ecuacion $\alpha = \beta$ de expresiones regulares puede deducirse útilizando únicamente estos once axiomas.

- Asociatividad: $\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma \qquad \alpha(\beta\gamma) = (\alpha\beta)\gamma$
- Conmutatividad: $\alpha + \beta = \beta + \alpha$
- Distributividad: $\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma \qquad (\beta + \gamma)\alpha = \beta\alpha + \gamma\alpha$
- Idempotencia: $\alpha + \alpha = \alpha$
- Propiedades de $\varnothing$: $\alpha + \varnothing = \alpha \qquad \varnothing\alpha = \alpha$
- Propiedades de ε: $\varepsilon\alpha = \alpha$
- Propiedades de la estrella de Kleene: $\alpha^\star = \varepsilon + \alpha^\star\alpha \qquad \alpha^\star = (\varepsilon + \alpha)^\star$

2.1. Otras equivalencias de Expresiones Regulares

Las siguientes son algunas equivalencias útiles de expresiones regulares

$$\varepsilon^\star = \varepsilon \qquad \varnothing^\star = \varepsilon$$
$$\alpha\alpha^\star = \alpha^\star\alpha \qquad \alpha^\star = \alpha^\star\alpha^\star = (\alpha^\star)^\star$$
$$\alpha^\star = \varepsilon + \alpha\alpha^\star \qquad (\alpha+\beta)^\star = (\alpha^\star + \beta^\star)^\star$$
$$(\alpha+\beta)^\star = (\alpha^\star\beta^\star)^\star = (\alpha^\star\beta)^\star\alpha^\star = \alpha^\star(\beta\alpha^\star)^\star \qquad (\alpha\beta)^\star = \varepsilon + \alpha(\beta\alpha)^\star\beta$$
$$\alpha(\beta\alpha)^\star = (\alpha\beta)^\star\alpha$$

3. Propiedades de cerradura

Las propiedades de cerradura nos permiten construir nuevos lenguajes regulares a partir de lenguajes ya conocidos por medio de algunas operaciones entre lenguajes.
Si L, M son lenguajes regulares entonces:

- $L \cup M$ es regular.
- LM es regular.
- $L^\star$ es regular.
- L^+ es regular.
- $\overline{L}$ es regular
- $L \cap M$ es regular.
- $L - M$ es regular.

TODO

[START] Demostración

[END] Demostración

4. Aplicaciones de las expresiones regulares

- Búsqueda y sustitución en editores de texto (vi, emacs, etc.)
- Caza de patrones (pattern matching) y procesamiento de textos y conjuntos de datos, por ejemplo en minería de datos (grep, sed, awk, perl, etc.)
- Implementación de lenguajes de programación (fase de análisis léxico): conversión del programa fuente en una sucesión de elementos léxicos (lexemas o tokens), en esta sucesión se identifican las distintas componentes de un programa, como son palabras reservadas, identificadores, tipos de datos, etc.(lex, flex, etc.)

[START] Ejemplo

- Usando `grep` y expresiones regulares para cazade patrones de texto en Linux.
  El programa `grep` significa *glabal regular expression print*. Esto significa que se puede hacer

uso de `grep` para ver si la entrada recibida es cazada por un patrón especificado. Tomando como ejemplo el texto de la licencia[1] **GNU General Public License version 3**.

Bracket Expressions: Poniendo un grupo de caractetes dentro de los corchetes (`[` y `]`) se puede especificar que el caracter en esa posición puede ser cualquier caracter que se encuentre dentro de los corchetes.

Por ejemplo, para encontrar las líneas que contienen `too` ó `two`, se puede especificar de manera clara usando el siguiente patrón[2].

```
$ grep "t[wo]o" GPL-3
```

Y la *salida* mostrará las posibles dos variantes en el archivo:

```
your programs, too.
freedoms that you received.  You must make sure that they, too, receive
  Developers that use the GNU GPL protect your rights with two steps:
a computer network, with no transfer of a copy, is not conveying.
System Libraries, or general-purpose tools or generally available free
    Corresponding Source from a network server at no charge.
...
...
```

[END] Ejemplo

5. La derivada de un lenguaje

El concepto de derivada de un lenguaje con respecto a una cadena fue introducido en 1964 por Brzozowski, aunque se origina en trabajos de Rabin, Raney y Elgot, véase [3, 4, 2]. Curiosamente, hasta donde sabemos, este concepto no figura en ningún libro de texto actual, siendo este uno de los motivos principales para discutirlo en este artículo.

Definición 5. *La derivada de un lenguaje $L \subseteq \Sigma^*$ con respecto a una cadena $u \in \Sigma^*$, denotada $\partial_u L$, es el lenguaje definido como:*

$$\partial_u L = \{v \mid uv \in L\}$$

Es decir la derivada de L con respecto a u es el conjunto de todas las cadenas tales que al anteponerles u están en L.

Ejemplo 1. *Consideremos el lenguaje definido por la expresión regular $r = b(a+c)^*$, la derivada de r con respecto a la cadena b es $(a+c)^*$, mientras que la derivada de r con respecto a la cadena a es $\varnothing$, puesto que ninguna cadena de r empieza con a.*

Esta operación tiene diferentes propiedades de interés que discutiremos más adelante, pero antes veamos como calcular la derivada.

[1] Se puede consultar en el sitio `https://www.gnu.org/licenses/gpl-3.0.txt`.
[2] Se asume que la licencia fue guardada con el nombre `GPL-3`.

5.1. Cálculo de la Derivada

De manera similar a la derivada en el Cálculo Diferencial, es posible dar reglas inductivas para calcular la derivada de una expresión (lenguaje) regular. Para esto se requiere primero determinar cuando una expresión regular es anulable, esto es, cuando ε pertenece al lenguaje de r. Esta tarea la realiza la función de *nulidad* ν, descrita a continuación:

Definición 6. *Sea $\nu : ER \to ER$ la función de nulidad cumple con la siguiente propiedad*

$$\nu(r) = \begin{cases} \varepsilon & \textit{Si } r \textit{ es anulable} \\ \varnothing & \textit{otro caso} \end{cases}$$

y se define recursivamente como sigue

$$\begin{aligned} \nu(\varepsilon) &= \varepsilon \\ \nu(\varnothing) &= \varnothing \\ \nu(a) &= \varnothing \\ \nu(r+s) &= \nu(r) + \nu(s) \\ \nu(rs) &= \nu(r)\nu(s) \\ \nu(r^*) &= \varepsilon \end{aligned}$$

Empleando a la función de nulidad como auxiliar Brzozowski define, en [1], la derivada de una expresión regular con respecto a un símbolo de la siguiente manera.

Proposición 2. *Sean r una expresión regular y $a \in \Sigma$, la derivada de r con respecto al símbolo a denotada por $\partial_a r$, se calcula recursivamente mediante las siguientes reglas*

$$\begin{aligned} \partial_a \varnothing &= \varnothing \\ \partial_a \varepsilon &= \varnothing \\ \partial_a a &= \varepsilon \\ \partial_a b &= \varnothing \textit{ si } b \neq a \\ \partial_a (r+s) &= \partial_a r + \partial_a s \\ \partial_a (rs) &= (\partial_a r)s + \nu(r)\partial_a s \\ \partial_a (r^*) &= (\partial_a r)r^* \end{aligned}$$

Demostración. Véase [1]. □

Se observa que estas reglas son reminiscentes de las reglas de derivación del Cálculo Diferencial, por lo que es probable que esta halla sido la razón para elegir el nombre de derivada para esta operación en lenguajes regulares.

Para calcular la derivada con respecto a una cadena arbitraria nos servimos de la siguiente caracterización.

Proposición 3. *Sea r una expresión regular y $u \in \Sigma^*$ la derivada de r con respecto a la cadena u denotada por $\partial_u r$ se calcula mediante las siguientes reglas:*

$$\partial_\varepsilon r = r$$
$$\partial_{ua} r = \partial_a(\partial_u r)$$

Demostración. Es directo de las definiciones. □

Obsérvese que de la proposición anterior se puede concluir que si $u = a_1 \dots a_n$ entonces

$$\partial_u r = \partial_{a_n} \partial_{a_{n-1}} \dots \partial_{a_1} r$$

Por lo que basta calcular derivadas de símbolos de manera iterada, utilizando las reglas de derivación, para obtener la derivada con respecto a cualquier cadena.

5.2. Propiedades de la derivada

Veamos ahora algunas propiedades importantes de la derivada, en el caso de los lenguajes regulares.

Proposición 4. *Si $L \subseteq \Sigma^*$ es regular, entonces $\partial_u L$ es regular para todas las cadenas $u \in \Sigma^*$.*

Demostración. Pendiente.

□

Esta proposición muestra que la clase de lenguajes regulares es cerrada bajo derivación. A continuación mostramos otra propiedad de gran importancia para nuestros propósitos.

Lema 1. *Para toda r expresión regular y $a \in \Sigma$, se cumple que*

$$L[\![\partial_a r]\!] = \partial_a(L[\![r]\!]),$$

es decir, las operaciones de denotación y derivada con respecto a un símbolo de una expresión regular conmutan.

Demostración. Inducción sobre r.

Si $r = \varnothing$ tenemos que $L[\![\partial_a \varnothing]\!] = L[\![\varnothing]\!] = \varnothing$. Por otro lado, $\partial_a(L[\![\varnothing]\!]) = \partial_a(\ \varnothing) = \varnothing$, luego entonces la propiedad se cumple para este caso.

Si $r = \varepsilon$, sabemos que $L[\![\partial_a \varepsilon]\!] = L[\![\varnothing]\!] = \varnothing$, así mismo $\partial_a(L[\![\varepsilon]\!]) = \partial_a(\varepsilon) = \varnothing$ y nuevamente se cumple la propiedad.

Para el caso en que $r = b \in \Sigma$, hay dos opciones, si $b \neq a$ se tiene que $L[\![\partial_a(b)]\!] = L[\![\varnothing]\!] = \varnothing = \partial_a(b) = \partial_a(L[\![b]\!])$. Cuando $b = a$, tenemos que $L[\![\partial_a a]\!] = L[\![\varepsilon]\!] = \varepsilon = \partial_a(a) = \partial_a(L[\![a]\!])$, con lo cual concluimos que la propiedad se cumple para todos los casos base.

Para $r = s + t$ tenemos lo siguiente

$$\begin{aligned}
L[\![\partial_a(s+t)]\!] &= L[\![\partial_a s + \partial_a t]\!] \quad \text{por la proposición 2} \\
&= L[\![\partial_a s]\!] + L[\![\partial_a t]\!] \quad \text{por la definición 3} \\
&= \partial_a(L[\![s]\!]) + \partial_a(L[\![t]\!]) \quad \text{por hipótesis inductiva} \\
&= \partial_a(L[\![s]\!] + L[\![t]\!]) \quad \text{por la definición 3} \\
&= \partial_a(L[\![s+t]\!]) \quad \text{por la proposición 2}
\end{aligned}$$

Para el caso $r = st$ tenemos

$$\begin{aligned}
L[\![\partial_a(st)]\!] &= L[\![\partial_a s + \nu(s)\partial_a t]\!] \quad \text{por la proposición 2} \\
&= L[\![(\partial_a s)t]\!] + L[\![\nu(s)\partial_a t]\!] \quad \text{por la definición 3} \\
&= L[\![\partial_a s]\!]L[\![t]\!] + L[\![\nu(s)]\!]L[\![\partial_a t]\!] \quad \text{por la definición 3} \\
&= \partial_a(L[\![s]\!])L[\![t]\!] + L[\![\nu(s)]\!]\partial_a(L[\![t]\!]) \quad \text{por hipótesis inductiva} \\
&= \partial_a(L[\![s]\!])L[\![t]\!] + \nu(L[\![s]\!])\partial_a(L[\![t]\!]) \quad \text{probando que } \nu(L[\![s]\!]) = L[\![\nu(s)]\!] \\
&= \partial_a(L[\![s]\!]L[\![t]\!]) \quad \text{por la proposición 2} \\
&= \partial_a(L[\![st]\!]) \quad \text{por la definición 3}
\end{aligned}$$

Por último tenemos $r = s^*$,

$$\begin{aligned}
L[\![\partial_a s^*]\!] &= L[\![(\partial_a s)s^*]\!] \quad \text{por la proposición 2} \\
&= L[\![\partial_a s]\!]L[\![s^*]\!] \quad \text{por la definición 3} \\
&= (\partial_a L[\![s]\!])L[\![s^*]\!] \quad \text{por hipótesis inductiva} \\
&= (\partial_a L[\![s]\!])L[\![s]\!]^* \quad \text{por la definición 3} \\
&= \partial_a(L[\![s]\!]^*) \quad \text{por la proposición 2} \\
&= \partial_a(L[\![s^*]\!]) \quad \text{por la definición 3}
\end{aligned}$$

□

El resultado anterior se extiende para cadenas de acuerdo al siguiente

Lema 2. *Para toda expresión regular r y para toda cadena $u \in \Sigma^*$, se cumple que*

$$\partial_u(L[\![r]\!]) = L[\![\partial_u(r)]\!].$$

Demostración. Por inducción sobre u

Si $u = \varepsilon$, tenemos que $L[\![r]\!] = L[\![\partial_\varepsilon r]\!]$, por lo tanto la propiedad se cumple para el caso base.

Ahora bien, si $u = wa$ tenemos

$$\begin{aligned}
\partial_u(L[\![r]\!]) &= \partial_a(\partial_w(L[\![r]\!])) \quad \text{por la proposición 3} \\
&= \partial_a(L[\![\partial_w r]\!]) \quad \text{por hipótesis de inducción} \\
&= L[\![\partial_a(\partial_w r)]\!] \quad \text{por el lema 1} \\
&= L[\![\partial_u r]\!] \quad \text{por la proposición 3}
\end{aligned}$$

□

Proposición 5. *El operador derivada preserva equivalencia, es decir, si $r = s$ entonces $\partial_a r = \partial_a s$*

Demostración. Es directo de las definiciones. □

5.3. El problema de la pertenencia

Dada una expresión regular r y una cadena $u \in \Sigma^*$, determinar si $u \in L[\![r]\!]$.
Resolver el problema es equivalente a resolver:

$$u \in L[\![r]\!] \Leftrightarrow \varepsilon \in L[\![\partial_u r]\!]$$

Es decir, se ha reducido la pertenencia de u al lenguaje L a verificar la pertenencia de ε al lenguaje $\partial_u r$ y esto último sucede si y sólo si

$$\nu(\partial_u r) = \varepsilon$$

Usando lo anterior y la definición de ∂_u, generamos un algoritmo que verifique si $u \in L[\![r]\!]$. Expresandolo en términos de la relación $r \sim u$ (u casa[3] con r).

Definición 7. *La relación de* casa *se define como sigue:*

$$r \sim \varepsilon \Leftrightarrow_{def} \nu(r) = \varepsilon$$
$$r \sim aw \Leftrightarrow_{def} \partial_a r \sim w$$

Proposición 6. *Para cualesquiera $u \in \Sigma^\star$ y $r \in ER$, se cumple que:*

$$r \sim u \Leftrightarrow u \in L[\![r]\!]$$

Demostración. Inducción sobre u □

6. Lenguajes no regulares

No todos los lenguajes son regulares, por ejemplo:

$$L = \{a^n b^n \mid n \geq 1\}$$

no es definible mediante una expresión regular.

- ¿Cómo decidir cuando un lenguaje no es regular?
- Mediante propiedades de cerradura.
- Mediante el lema del bombeo (pendiente).

En el siguiente tema discutiremos más formas de decidir cuándo un lenguaje es regular o no.

Referencias

[1] J. Brzozowski. Derivatives of Regular Expressions. Journal of the Association for Computing Machinery, Vol. 11, No. 4. 1964.

[2] C. C. Elgot, J. D. Rutledge. Operations on Finite Automata. Procc. AIEE Second Ann. Symp. on Switching Circuit Theory and Logical Design, Detroit. 1961.

[3] match

[3] M. O. Rabin, D. Scott. Finite Automata and their Decision Problems. IMB Journal Res. Develop, No. 3. 1959.

[4] G. N. Raney. Sequential Functions. Journal ACM, Vol. 5 No. 177. 1958.

[5] Using Grep & Regular Expressions to Search for Text Patterns in Linux `https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux`

Ejemplo uso de grep