Lenguajes Regulares

- Son aquellos generados por gramáticas regulares (**GR**)
- En los lenguajes de programación se usan para los diferentes **tokens** (LR dentro del lenguaje de programación)
  - Ejemplo: los identificadores
- Si un lenguaje formal es finito entonces SIEMPRE es un lenguaje regular (**LR**)

Expresiones Regulares
(Básicas)

- Una expresión regular (**ER**) es un patrón que describe un conjunto de cadenas que forman un LR
- Las ER se forman con
  - caracteres del alfabeto del LR representado
  - ε para representar la cadena vacía
  - Operadores
    - Unión
    - Concatenación
    - Estrella de Kleene
  - Paréntesis para agrupar o cambiar precedencias

Expresiones Regulares

- Precedencia
  - Potenciación (Estrella de Kleene), concatenación y por último unión.
- Los LR son cerrados respecto de estas operaciones, por eso podemos aplicarlos sabiendo lo obtenido sigue siendo un LR
- La unión es conmutativa, por tanto la ER a+ab y la ER ab+a representan el mismo lenguaje
- Dos ER se dicen equivalentes si representan el mismo lenguaje

Expresiones Regulares

- Un carácter solo es una ER, ε es una ER
  - La ER a representa el LR {a}
- Concatenación (no conmutativa)
  - La ER a.b o simplemente ab representa el LR {ab}
- Unión (conmutativa)
  - La ER a+b (también a|b pero...!!) representa el LR {a, b}
- Potenciación (NO es parte de las ER básicas)
  - La ER a^3 representa el LR {aaa}
- Estrella de Kleene
  - La ER a* representa el LR {ε, a, aa, aaa, …}

Factorización y Clausuras

- Dado que la operación de concatenación no es conmutativa tendremos factorizaciones a izquierda y a derecha
  - a(a+b) = aa + ab
  - aab+abb = a(ab+bb) = a(a+b)b
- Clausura positiva (ya vista)
  - Como la de Kleene pero sin potencia cero
  - $a^* = \varepsilon, a, aa, aaa \ldots.$
  - $a^+ = aa^* = a^*a$

Algunas igualdades útiles

- $a^*a^* = a^*$
- $\varepsilon^* = \varepsilon$
- $a^* = aa^* + \varepsilon$
- $(ab)^*a = a(ba)^*$
- $(R^*)^* = R^*$
- $(R^*+S^*)^* = (R^*S^*)^* = (R+S)^*$

Definición formal de ER

1. $\varnothing$ es la ER que representa la conjunto vacío
2. ε es una ER que representa al lenguaje L={ε}
3. $x \in \Sigma$ es la ER que representa el Lenguaje L={x}
4. $s \in \Sigma^*$ es la ER que representa el Lenguaje L={s}
   (p.ej: s = abf)
5. R_1 y R_2 son ER entonces R_1+R_2 es también una ER
6. R_1 y R_2 son ER entonces R_1R_2 es también una ER
7. R es ER entonces R^* es también una ER
8. R es ER entonces (R) es también una ER

Ejemplos

- La ER a(b+c)*a representa todas las cadenas que comienzan y terminan exactamente con una **a** y en el medio puede tener una cantidad arbitraria (incluso ninguna) de **b** o **c**.
- La ER (x+y+z)(ab)* representa todas cadenas que comienzan con **x** o **y** o **z** y continúan con una cantidad arbitraria (eventualmente cero) de la cadena "**ab**"
- Expresión Regular Universal (**ERU**)
  - Es aquella que representa todas las cadenas que se pueden formar con un alfabeto dado. Por ejemplo, si el alfabeto es
    Σ = {a, b, c} entonces la ERU correspondiente es (a+b+c)*

MetaER

- Además de los operadores básicos muchos programas permiten usar otros operadores que hacen más fácil armar las ER y no cambian el poder expresivo de las mismas. Se las conoce como ER extendidas o MetaER
- Las extensiones dependen un poco de que programa se use, sin embargo algunas son muy comunes y las detallamos en la diapositiva siguiente
- A la tabla de la diapositiva siguiente debemos agregar el uso de \ para poder usar algunos de los símbolos que se usan como operadores sin su significado especial, así * no es la clausura de Kleene sino un asterisco tomado literalmente como tal

MetaER

Metacaracteres Operadores	Comentario
. (punto)	Se corresponde con cualquier carácter (solo uno) salvo \n dado que los programas de ER suelen analizar por línea
| (pipe o barra vertical)	Operador de unión (corresponde al + en ER básicas)
[ ] (corchetes)	Enumera conjunto de caracteres (alternativa a usar unión)
[^]	Complemento del conjunto enumerado.
[-]	Da un rango de caracteres [0-9] es la unión de todos los dígitos
{ } (llaves)	Para indicar una potenciación. b{3} corresponde a bbb
{ , }	Similar pero todas las potencias entre la primera y la última. b{2,3} corresponde a bb+bbb
?	Indica opción (cero o una vez) b? Corresponde a (b+ε)
*	Clausura de Kleene
+	Claursura Positiva
( ) (paréntesis)	Se utiliza para agrupar subexpersiones

Estándar y herramientas

- Nos guiamos por el estándar POSIX.1-2017 https://pubs.opengroup.org/onlinepubs/9699919799/ que en su capítulo 9 trata el tema de ER https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html
  - Agregan otros elementos como
    - ^ para indicar que el patrón encontrado debe estar al principio de la línea
    - $ para indicar que debe encontrarse al final.
    - Clases de caracteres, por ejemplo [:digit:]
- Como herramienta de referencia usaremos grep https://www.gnu.org/software/grep/manual/
  - Ademas de la expresiones básicas y extendidas de POSIX también implementan las PCRE (Perl Compatible Regular Expresions)

Ejemplos

- [a-zA-Z][a-zA-Z0-9_-]{3,11}
  - Cadena que comienza con una letra y continúa con letras o dígitos o guión bajo o guión medio en cantidad de 3 a 11
- 0[xX][0-9a-fA-F]+
  - Constantes hexadecimales “similares” a las de lenguaje C
- (\+|-)?[0-9]+
  - Nros enteros con un signo previo opcional. Notar que al + debimos anteponerle \
- [0-9]*\.[0-9]+
  - Nros reales donde que pueden comenzar directamente con el . decimal

Operaciones entre Lenguajes

- En tanto conjuntos los lenguajes pueden operarse con
  - Unión
  - Concatenación
  - Intersección
  - Complemento (con respecto a su ERU)

Tabla de Clausuras sobre operaciones

Operación	LR	LIC	LSC	LRE
Unión	Si	Si	Si	Si
Concatenación	Si	Si	Si	Si
Intersección	Si	No	Si	Si
Complemento	Si	No	Si	No

Notación: a los Lenguajes generados por gramáticas irrestrictas se los suele llamar Lenguajes Recursivamente Enumerables (LRE)

Definición Regular

- Para simplificar a veces se construye una “definición regular” que no es otra cosa que una ER auxiliar que luego se usa en otra más compleja
- Ejemplo para constantes Hexadecimales en lenguaje C
  - Definiciones
    - PRE = 0[xX]
    - DH = [0-9A-Fa-f]
    - SU = [uU]
    - SL = [lL]
  - ER
    - PRE DH+ (SU SL | SL SU | SU | SL)?

Licencia