DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico: tlengrep parte 2

15 de diciembre de 2023 — Teoría de lenguajes

Grupo: BTS

Estudiante	LU	Correo electrónico
Luciana Skakovsky	131/21	lucianaskako@gmail.com
Florencia Rosenzuaig	118/21	f.rosenzuaig@gmail.com
Alan Roy Yacar	174/21	alanroyyacar@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

1. Introducción

El propósito de esta segunda parte es ampliar el trabajo práctico inicial al incluir la funcionalidad de parsing de expresiones regulares. Este cambio nos permitirá parsear la cadena de texto de entrada, y traducirla a una instancia de la clase RegEx, para luego buscar todas sus apariciones en el archivo. En este informe mostraremos los pasos para la construcción de dicho parser.

2. Solución

La primera parte de la solución consiste en idear la gramática del parser. A continuacion se presenta dicha gramática.

G = ⟨{P, S, R, T, F, SYMB}, { OR, LPAREN, RPAREN, STAR, PLUS, POWER, POWER_RANGE, LCORCHETE, RCORCHETE, _W, _D, CHAR, NUMBER, MINUS, LLLAVE, RLLAVE, RANGE}, δ, P⟩

donde δ:

Gramática

P → S | λ
S→ R OR S | R
R → TR | T
T → K STAR | K PLUS | K OPTIONAL | K POWER | K POWER_RANGE | K
K → (P) | [F | RANGE | SYMB | _W | _D
F → SYMB | RANGEF |]
SYMB → CHAR | NUMBER | MINUS | LLLAVE | RLLAVE

2.1. Tokens utilizados

- OR: denota la operación |.
- STAR: denota la operación $*$
- PLUS: denota la operación $+$
- OPTIONAL: denota la operación ?
- POWER: denota la operación "$\{n\}$". También tenemos como símbolo las llaves y los números (de forma individual), sin embargo el lexer se encarga de reconocer que "$\{n\}$" es power. Además, el lexer extrae el valor de n.
- RANGE: denota un rango "$a-b$"donde a y b son alfanuméricos. Este token se comporta distinto según si está dentro de corchetes o si está "suelto". Si está dentro de corchetes, representa el rango entre "a" y "b". Si está suelto, representa la concatenación entre "a", " $-$ " y "b"
- POWER_RANGE: denota la operación $\{n, m\}$. El lexer se encarga de reconocerlo y luego el parser extrae los valores de n y m.
- (R/L)PAREN: denota los símbolos (y) respectivamente.

- (R/L)CORCHETE: denota los símbolos [y] respectivamente.
- (R/L)LLAVE: denota los símbolos { y }.
- CHAR: denota letras (en mayúscula y minúscula), el guión bajo, el espacio, y los símbolos escapeados (+, *, |, ? , (,) , [,]).
- NUMBER: denota valores numericos.
- MINUS: denota el caracter " − ".
- _W: denota el conjunto de caracteres [a-zA-Z]. El parser se encarga de construir la unión entre todos los caracteres.
- _D: denota el conjunto de caracteres [0-9]. En este caso, también es el parser quién se encarga de construir la unión de todos los caracteres.

3. Prioridad de tokens

Por cómo funciona `yacc`, los tokens tienen prioridad según qué tan complejas son las reglas que los definen. En nuestro lexer, aquel con más prioridad es CHAR. Sin embargo, nosotros necesitábamos que RANGE tenga mayor prioridad que CHAR y MINUS. Para resolverlo, escribimos la definición de RANGE otra forma para que el lexer de menor prioridad a CHAR y MINUS.

4. Consideraciones

- Como tomamos el rango como un token [a-b] donde a y b son alfanuméricos, si tenemos el caso $[a--]$ la interpretación de nuestro parser no sera el rango entre el caracter "a" y el caracter " − " sino que sera interpretado como los 3 caracteres por separado "a" , " − " y " − ".
  Esta decisión la tomamos debido a que en el enunciado se pide que el " − " sea interpretado como un caracter normal salvo que se esté utilizando para definir un rango dentro de una clase de caracteres y consideramos que en el caso explicado anteriormente no tiene sentido considerarlo como parte del rango. Aun así si se quisiera interpretar el caso $[a--]$ como el rango entre "a" y " − " esto se puede lograr con pequeños simples cambios en el código. Bastaría con definir RANGE en el lexer como la concatenación de:
  - todos los alfanuméricos (incluyendo a " − ")
  - el símbolo " − "
  - todos los alfanuméricos (incluyendo a " − ").

- Para las operaciones {n,m} (power_range) y [a-b] (range), decidimos que el rango debe ser válido: es decir, n tiene que ser menor que m y a tiene que ser lexicográficamente menor que b. De no cumplirse, la compilación va a dar `syntax error`.
- En el parser, todas las producciones retornan `Regex`, salvo por las producciones que parten de SYMB que retornan `chr`.

Figura 1: Implementación del token POWER_RANGE