Procesadores de Lenguajes

Índice

Capítulo 1

ANÁLISIS LÉXICO

Consiste en la identificación de los caracteres de entrada así como su evaluación. Se realiza mediante *scanners* (o *lexers*).

- Un ***token*** es un componente léxico: palabras reservadas, variables, operadores, constantes, símbolos de puntuación…
- Los *tokens* pueden contener valores además de su identificador, llamados atributos.
- Cada *token* está compuesto por **lexemas**, es decir, secuencias de caracteres.
- Los lexemas se pueden describir de forma compacta con patrones (**expresiones regulares**).
- 1 *token* $\rightarrow$ 1 patrón $\rightarrow$ 1 o más lexemas.

1.1 Expresiones regulares

Las **expresiones regulares** sirven para definir de forma compacta un **lenguaje regular**.

- Un **alfabeto** (Σ) es un conjunto de símbolos finito.
- Los símbolos del alfabeto forman **cadenas**, las cuales tienen una **longitud**.
- El conjunto de cadenas que puede formar el alfabeto se denomina **lenguaje**.

Dos lenguajes regulares (y por tanto, expresiones regulares) disponen de las operaciones:

Operación	Lenguaje	Expresión
Unión	$L \bigcup M = \{c \| c \in L \vee c \in M\}$	r | s
Concatenación	$LM = \{st \| s \in L \wedge t \in M\}$	rs
Cerradura de Kleene	$L* = \bigcup_{i=0}^{\infty} L^i$	r*
Cerradura positiva	$L+ = \bigcup_{i=1}^{\infty} L^i$	r+

1.2 Autómatas finitos

Los autómatas finitos son máquinas abstractas que reconocen cadenas de un lenguaje regular. Pueden ser deterministas o no deterministas.

Los autómatas finitos no deterministas están formados por:

- Un conjunto de **estados** (S).
- Un **alfabeto** de entrada (Σ).
- Una **función de transición** ($\delta : S \times E \rightarrow P(s)$).
- Un **estado inicial** (s_o).
- Un **conjunto de estados finales o de aceptación** ($F \subseteq S$).

Los autómatas deterministas además cumplen:

- ϵ (cadena vacía) no etiqueta ningún arco. Si no, se podría cambiar de estado sin consumir entrada.
- La función de transición devuelve un único estado sucesor. En caso contrario, habría varios estados sucesores para la misma entrada.

En un AFN, una cadena es aceptada si hay algún camino que llegue a un estado final. En un AFD, solo existe un camino si es aceptado.

1.3 Conversión de ER a AFN (método de Thomson)

TODO: Meter dibujitos

1.4 Conversión de AFN a AFD (ϵ-clausura)

TODO: Algoritmo

1.5 Minimización de AFD

TODO: ???

Capítulo 2

ANÁLISIS SINTÁCTICO

Consiste en la agrupación de los *tokens* obtenidos durante el análisis léxico y su reconocimiento de acuerdo a unas **reglas de producción** que definen una **gramática libre de contexto**. Se realiza mediante *parsers*.

2.1 Gramáticas

Las gramáticas se definen como:

- Un conjunto de **no terminales** (N).
- Un conjunto de **terminales** (T, $N \cap T =$). Generan una única cadena.
- Un **símbolo no terminal inicial** (S). A partir de él se generan todos las cadenas.
- Un **conjunto de producciones** (P). Establecen transformaciones de formas de frase, y se describen utilizando la forma Backus-Naur.

Las operaciones de una gramática son:

Operación	Descripción	Símbolo
Derivación directa	Se sustituye un símbolo en un solo paso	$\Rightarrow$
Derivación	Se sustituye un símbolo en cero o más pasos	$\overset{*}{\Rightarrow}$

- Una **forma de frase** es cualquier cadena que se pueda derivar del símbolo inicial.
- Una **frase** es cualquier forma de frase con sólo elementos terminales.
- El conjunto de todas las frases forma un lenguaje.

2.1.1 Clasificación de gramáticas

Las gramáticas se pueden clasificar de la siguiente forma:

- Gramáticas libres (tipo 0)
- Gramáticas dependientes de contexto (tipo 1)
- Gramáticas libres de contexto (tipo 2)
- Gramáticas regulares (tipo 3)

TODO: Rellenar

2.2 Árboles de análisis sintáctico

TODO

2.3 Derivación (derecha-izquierda)

TODO

2.4 Ambigüedad y transformación de gramáticas

Una gramática es ambigua si existe al menos una frase ambigua, es decir, que exista más de un árbol de derivación para ella. Esto genera indeterminismo. Cuando se pueda, se debería eliminar dichas ambigüedades convirtiendo el lenguaje ambiguo en otro no ambiguo. Existen gramáticas que son inherentemente ambiguas.

Es imposible determinar si una gramática es ambigua o si dos GLC son equivalentes. Lo que si que se puede hacer es simplificarlas mediante transformaciones.

Capítulo 3

ANÁLISIS SEMÁNTICO

Capítulo 4

ENTORNO DE EJECUCIÓN

Capítulo 5

GENERALIZACIÓN DE CÓDIGO

Capítulo 6

OPTIMIZACIÓN DE CÓDIGO