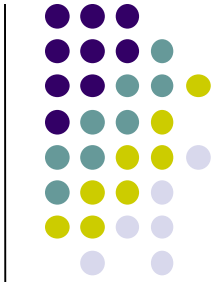
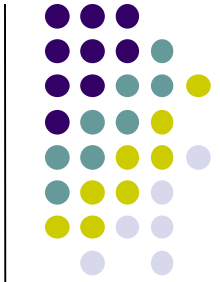


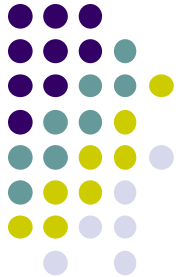
- **Capítulo 4**
- **Análisis Léxico y Sintáctico**

Tópicos del Capítulo



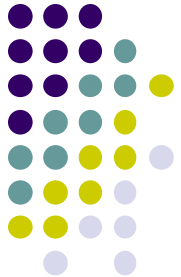
- Introducción
- Análisis léxico
- El problema del Análisis Sintáctico
- Análisis Sintáctico descendente recursiva
- Análisis Sintáctico ascendente

Introducción



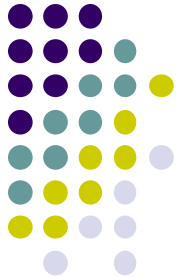
- Sistemas de implementación de lenguaje deben analizar el código de origen, independientemente del abordaje de implementación específica
- Casi todos los análisis sintácticos son basados en una descripción formal de la sintaxis del lenguaje de origen (BNF)

Análisis Sintáctico



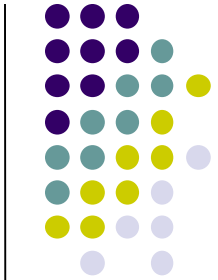
- La porción de análisis sintáctico de un procesador de lenguaje casi siempre consiste en dos partes distintas:
  - *Analizador Léxico* (construcciones de pequeña escala del lenguaje, como nombres, literales numéricos, símbolos)
  - *Analizador sintáctico* o *parser* (construcciones de gran escala del lenguaje como expresiones, sentencias y unidades del programa);

Ventajas de usar BNF para describir una sintaxis



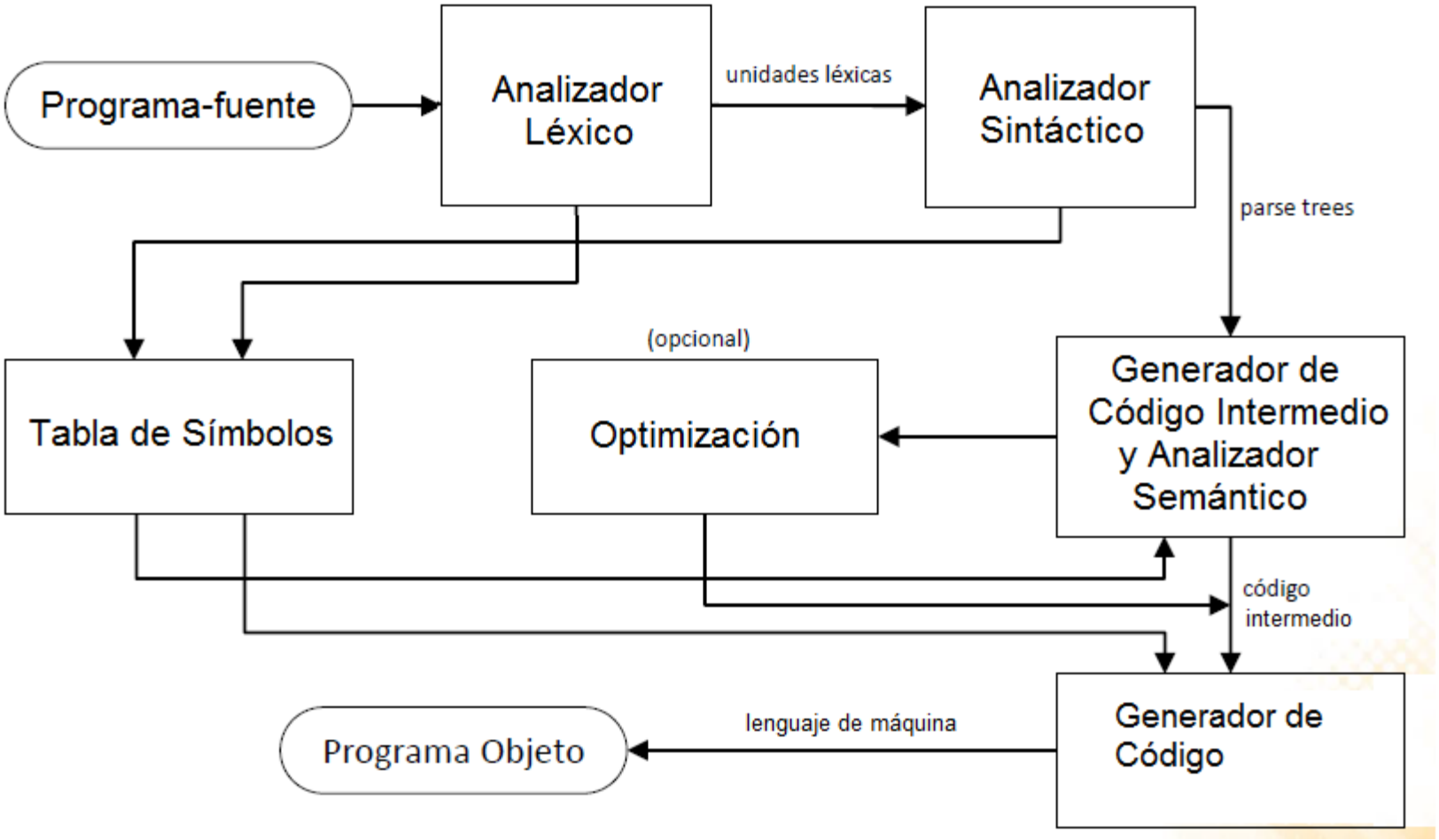
- Provee una descripción clara y concisa, tanto para humanos cuanto para los sistemas de software
- Puede ser usada como la base directa para el analizador sintáctico
- Implementaciones basadas en BNF son relativamente fáciles de mantener

Razones para separar el análisis léxico del análisis sintáctico

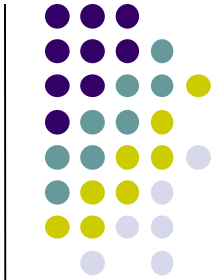


- Simplicidad: técnicas para análisis léxico son menos complejas de aquellas para análisis sintáctico; el proceso puede ser más simple si fuera realizado separadamente
- Eficiencia: separación permite optimización del analizador léxico;
- Portabilidad: el analizador léxico lee los archivos de entrada que contienen el código fuente del programa, por lo que depende en cierta medida de la plataforma. Pero el analizador sintáctico puede ser independiente de la plataforma.

Proceso de Compilación

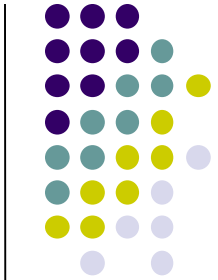


Análisis Léxico



- Un analizador léxico es esencialmente un cazador de patrones para cadenas de caracteres.
- Un analizador léxico sirve como el paso inicial de una analizador sintáctico
- Colecta caracteres y los agrupa lógicamente, asignando códigos internos de acuerdo con su estructura – lexemas.
  - Lexema tiene correspondencia con un patrón de caracteres, que es asociado a la categoria léxica llamada *token*
  - `sum` es un lexema; su token puede ser `IDENT`

Análisis Léxico



- El analizador léxico es normalmente una función llamada por el analizador sintáctico cuando este necesita un token.
- Tres abordajes para construir un analizador léxico:
  - Escribir una descripción formal de los *tokens* y usar una herramienta de software que construye analizadores léxicos dirigidos por tabla que reciben esa descripción
  - Diseñar un diagrama de transiciones de estado que describa los patrones de *tokens* del lenguaje y escribir un programa que implemente el diagrama.
  - Describir un diagrama de transiciones de estado que describa los patrones de *tokens* del lenguaje y construir manualmente una implementación dirigida por tabla del diagrama de estados

Análisis Léxico

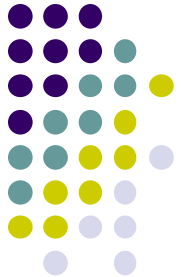
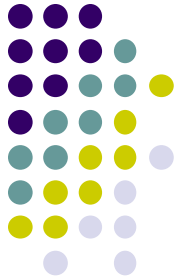
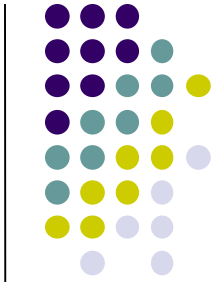


Diagrama de estados



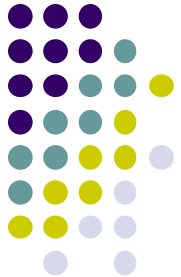
- Un diagrama de transición de estados, o apenas diagramas de estados es un grafo dirigido
- Los nodos de un diagrama de estados son rotulados con nombres de estados
- Los arcos son rotulados con los caracteres de entrada que causan las transiciones entre los estados
- Un arco puede también incluir acciones que el analizador léxico debe realizar cuando la transición ocurre

Análisis Léxico



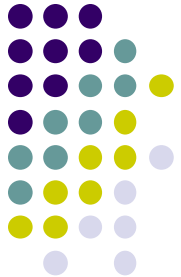
- En muchos casos, transiciones pueden ser combinadaas para simplificar el diagrama de estado
  - Al reconocer un identificador, todas las letras mayusculas y minúsculas son equivalentes.
    - Use una clase de caracteres que incluye todas las letras
  - Al reconocer un literal entero, todos los dígitos son equivalentes – usar una clase dígitos

Análisis Léxico



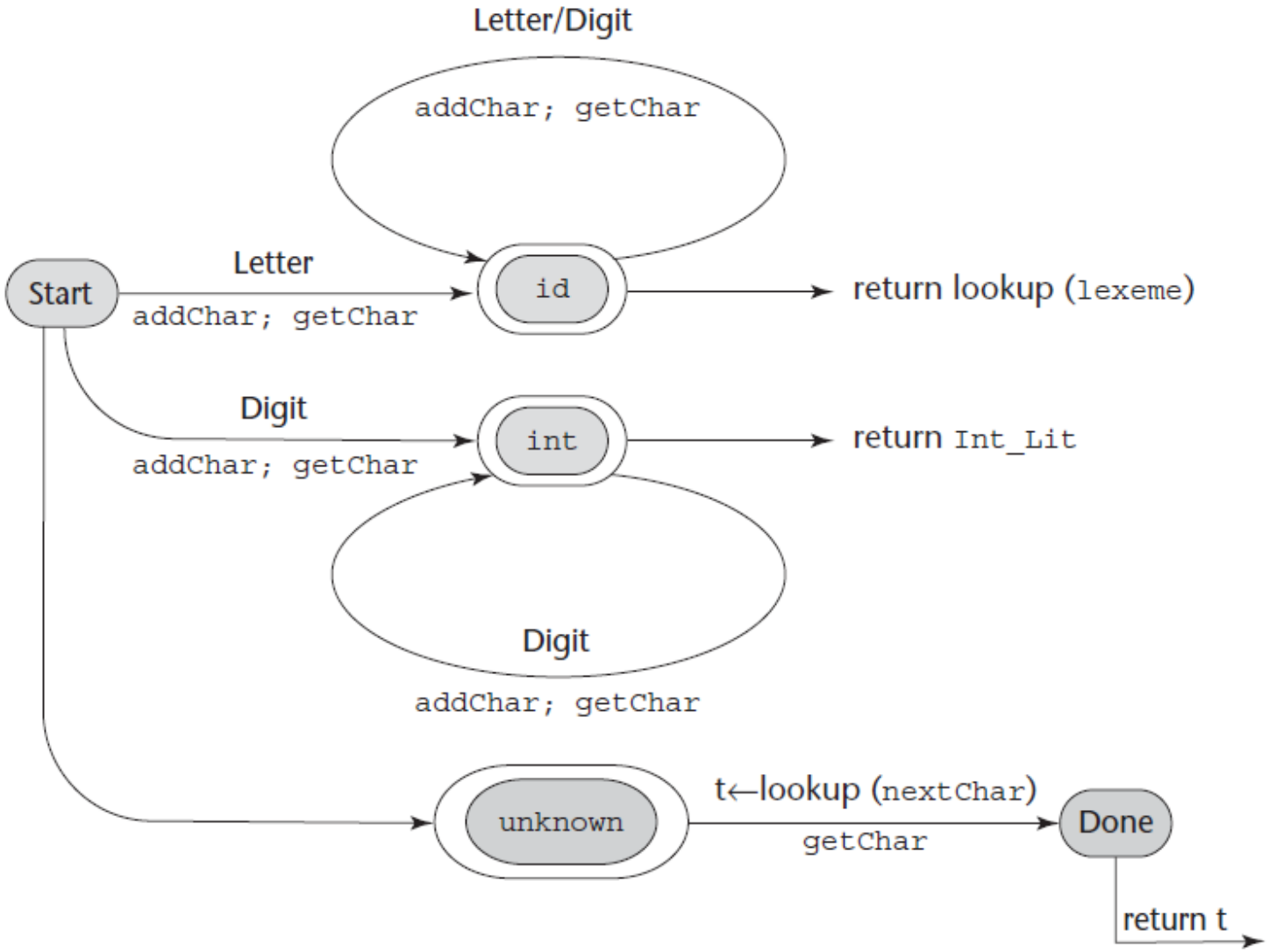
- Las palabras reservadas e identificadorse pueden ser reconocidos juntos (en vez de haber una parte del diagrama para cada palabra reservada)
  - Use una tabla para determinar si un posible identificador es un palabra reservada

Análisis Léxico

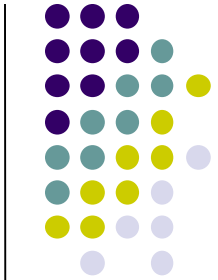


- Subprogramas utilitarios:
  - **getChar** – obtiene el próximo caracter de entrada del programa de entrada y lo coloca en la variable **`nextchar`**, determina la clase del caracter de entrada y la coloca en la variable **`charClass`**
  - **addChar** – coloca el caracter del **`nextChar`** en el lugar en que el lexema está siendo acumulado, **`lexeme`**
  - **lookup** - determina si la secuencia de caracteres **`lexeme`** es una palabra reservada.

Diagrama de estados



Análisis Léxico



- Implementación:
- → Mostrar front.c
- A seguir, tenemos la salida del analizador léxico de `front.c` cuando usado com `(sum + 47) / total`
- `Next token is: 25 Next lexeme is (`
- `Next token is: 11 Next lexeme is sum`
- `Next token is: 21 Next lexeme is +`
- `Next token is: 10 Next lexeme is 47`
- `Next token is: 26 Next lexeme is )`
- `Next token is: 24 Next lexeme is /`
- `Next token is: 11 Next lexeme is total`
- `Next token is: -1 Next lexeme is EOF`