Procesadores de Lenguajes

Profesor. Ramón Lopez-Gozar Delgado
Correo. rlopez@ugr.es
Despacho. 3ª Planta 26
Tutorias. Miercoles y viernes 11.30-14.30

1. Traductores

Cuando el lenguaje de especificación de un programa es diferente al lenguaje máquina, es necesario traducirlo a lenguaje máquina para poder ejecutarlo. Una **traducción** relaciona dos o más modelos semánticos.

Un modelo semántico $M = (L(G), D, I)$ está formado por:

- **Lenguaje L(G).** Conjunto de construcciones simbólicas formadas por secuencias de símbolos de un alfabeto. Se utiliza para prepresentar los valores semánticos de un dominio semántico.
- **Dominio semántico D.**
- **Interpretación I.** Aplicación $I : L(G) \to D$

1.1. Requisitos para construir un traductor.

Dados dos lenguajes $L_1(G_1)$ y $L_2(G_2)$, se necesitan dos moelos semánticos M_1 y M_2 tales que

- $D_1 \subset D_2$
- $\forall \alpha \in L_1(G_1)\ \exists \beta \in L_2(G_2)$ tal que $I_1(\alpha) = I_2(\beta)$

Bajo estas condiciones un **esquema de traducción** es una función $T : L_1(G_1) \to L_2(G_2)$ que verifica $T(\alpha) = \beta$.

1.2. Aplicaciones de los traductores.

- Traducir a código máquina programas. El dominio semántico de los lenguajes de programación coincide con el dominio semántico del lenguaje máquina.
- Podemos definir un lenguaje orientado al problema que permita formular el mismo de forma más cercana a la cultura del usuario.

1.3. Fases de un traductor

1.3.1. Verificación Sintáctica.

Se puede simplificar teniendo en cuenta lo siguiente:

- Elegir lenguajes que estén definidos por gramáticas regulares o independientes del contexto.
- Diseñar lenguajes formados como concatenación depalabras y las palabras formadas como concatenación de simbolos del alfabeto.
- Si las fases obedecen a una estructura sintática, la verificación sintáctica se puede descomponer en:
  - **Análisis léxico:** Se identifican las palabras con la misma misión sintáctica.
  - **Análisis sintáctico:** Se verifica la sintaxis de la secuencia de palabras. Se pueden aplciar técnicas predictivas (sin vuelta atras).

1. Análisis léxico.

   Un **componente léxico o token** es un conjunto de palabras que hacen la misma misión sintáctica. Los tokens son definidos mediante expresiones regulares. Son un sublenguaje.

   Un **lexema** es una palabra concreta del texto fuente, asociada a un token.

   Un **patrón** es una regla mediante la cual una secuencia de caracteres del texto fuente es asociada a un token (regla de formación de un token). Cada patrón es reconocido por un AFD.

   Desde el análisis sintáctico, se obvia el lexema de cada componente léxico, sólo nos interesa la presencia de un determinado componente léxico. Pero desde la perspectica del análisis semántico y generación de código, se necesita conocer el lexema concreto.
   Esto se soluciona introduciendo los lexemas en una tabla de símbolos, y consevando para cada componente léxico un atributo que indica el lexema concreto que representa.

   Herramienta para construir el analizador léxico: LEX.

2. Análisis sintáctico.

   La complejidad del análisis sintáctico depende del tipo de gramática que define el lenguaje.

   Para comprobar que una cadena de símbolos pertenece a un lenguaje, es necesario aplicar todas las combinaciones posibles sobre las producciones de la gramática y comprobar si puede generarse. Esto podría no terminar nunca. Para solucionarlo utilizamos **análisis sintáctico predictivo**.

   La **abstracción léxica** consiste en simplificar la gramática original en otra gramática que denotaremos gramática abstracta.

   Ante un símbolo de entrada, el **análisis predictivo**, permite decidir que hacer para proseguir el análisis y obtener una cadena del lenguaje, sin tener que generar todas las posibles cadenas del lenguaje. Es un método válido si la gramática **no es ambigua**.

   Herramienta para realizar el análisis sintántico: YACC.

1.3.2. Verificación semántica

1. Análisis semántico.

   Desde el arbol sintáctico (secuencia de producciones aplicadas en el análisis) podemos identificar las secuencias de componentes léxicos con significado conjunto.

   Ya que partimos de una secuencia de componentes léxicos correcta sintecticamente, podemos analizar el valor semantico estructurado asociado con cada subsecuencia de componentes léxicos junto con sus argumentos.

1.3.3. Fase de síntesis.

1. Generación de código.

   Consiste en aplicar de forma particularizada el **Esquema de Traducción** para cada secuencia de lexemas con sentido propio.

   Un aspecto importante de los traductores de lenguajes de programación consiste en que deben generar códigos de una eficiencia comparable al generado directamente en lenguaje máquina. Estacualidad deseable de los traductores ha dado luhar al desarrollo de técnicas de optimizacion de código.

2. Optimización de código.

   Toda secuencia de programa obtenido en código máquina u objeto se caracteriz apor la presencia de:

   - Instrucciones de evaluación.
   - Instrucciones de control.

   En ocasiones es posible reorganizar el código para reducir el número de instrucciones.

1.4. Compiladores

Un compilador obtiene una especificación en lenguaje máquina equivalente. Tduce la especificación de entrada a lenguaje máquina incompleto y con instrucciones máquina incompletas.
El **enlazador** enlaza los programas objetos y completas las instrucciones máquina incompletas generando un ejecutable.

1.5. Intérprete

Un **intérprete** carga la especificación de un programa en lenguaje no máquina y la interpreta y la ejecuta, instrucción a intrucción.

2. Analizador léxico

En analizador léxico se ecarga de leer, carácter a carácter, el documento de entrada y generar una secuencia de patrones léxicos denominados **tokens** y, en su caso, asocia atributos a los tokens.
El analizador léxico simplifica el diseño del analizador sintáctico, confiriendole mayor eficacia. Además aumenta la portabilidad del traductor.

2.1. Conceptos básicos

- **Token.** Conjunto de secuencias de caracteres con la misma misión sintáctica.
- **Lexema.** Secuencia de caracteres que forman un **token**.
- **Patrón.** Regla o reglas que describen a los lexemas asociados a un **token**.

Ejemplo: «repeat», «while» y «if» son **lexemas** del **token** «CondCiclo».

- **Gramática abstracta.** Gramática resultante de considerar los tokens com osímbolos terminales y eliminar aquellas producciones en las que derivan los tokens.
- **Tabla de símbolos.** Lugar de almacenamiento temporal (durante la fase de compilación y, en lenguajes orientados a objetos, durante la ejecución) debido a la necesidad de identificar los lexemas en la ocurrencia de cada token durante las fases posteriores de traducción.
- **Tabla de tokens.** Tabla formada por tantas filas como tokens se haya identificado. Las columnas tienen la siguiente información: «Token», «Código», «Atributos» y «Patrón (expresion regular)».

2.2. Ejercicios tipo.

Dado la siguiente gramáticam, determinar el conjunto de tokens con el máximo nivel de abstracción para la construcción de un traductor.

1. Encontrar todas las palabras/símbolos/signos de la gramática descritos por medio de su expresión regular.
2. Agruparlos atendiendo a la misión sintáctica, es decir, que desempeñen el mismo papel a nivel sintáctico.
3. Repetir el paso anterior con el resto hasta alcanzar la máxima abstracción.

Ejemplo. Tema 2. D. 19.

3. Análisis sintáctico

El **objetivo** del análisis sintáctico es analizar las secuencias de tokens y comprobar que son correctas sintácticamente. A partir de la secuencia de tokens, el analizador sintáctico nos devuelve:

1. Si la secuencia es correcta o incorrecta sintácticamente. Es decir, existe un conjunto de reglas gramaticales aplicables para poder estructurar la secuencia de tokens.
2. El orden en que hay que aplicar las producciones de la gramática pra obtener la seuencia de entrada (árbol sintáctico).

Si no se encuentra un árbol sintáctico para una secuencia de entrada, entonces la secuencia de enrada es incorrecta sintácticamente (tiene errores sintácticos).

3.1. Gramáticas libres del contexto

Una **gramática** se dice libre del contexto cuando las producciones son de la forma $A \rightarrow \alpha$, es decir, en la parte izquierda solo puede haber **un** símbolo y debe ser **no terminal**. Se denomina libre dle contexto debido a que podemos cambiar A por α independientemente del contexto en el que se encuentre.

3.2. Ambiguedad

Un árbol sintáctico es una representación gráfica donde aparecen las producciones de la gramática aplicadas y en el orden que son aplciadas para obtener una secuencia se símbolos de un lenguaje. En las hojas aparecen símbolos terminales y en los nodos interioreslos no terminales.
Una gramática se dice **ambigua** cuando admite más de un arbol sintáctico para una misma secuencia de símbolos de entrada.

3.3. Formas normales.

3.3.1. Forma normal de Chomsky (CNF)

Toda gramática libre del contexto cuyas producciones son de la forma:

$$A \to Bc \tag{1}$$
$$A \to a \tag{2}$$

3.3.2. Forma normal de Greibach (GNF)

Toda gramática libre del contexto cuyas producciones son de la forma

$$A \to a\alpha$$

Donde α es una cadena de simbolos no terminales o la cadena vacía.

3.4. Estrategias

Dada una secuencia de símbolos, existen dos estrategias para verificar si se trata de una secuencia de símbolos de un lenguaje.

3.4.1. Análisis descendente (Top-Down).

Partir del símbolo inicial de la gramática y generar los arboles sintácticos hasta que se alcance la secuencia de simbolos.
Este análisis se puede realizar de diferentes formas, una de ellas es utilizar la fuerza bruta, sin embargo, esto presenta un problema cuando existen producciones definidas recursivamente a la izquierda ($A \to AcB$).
Este tipo de producciones se pueden eliminar haciendo la siguiente tansformación. Supongamos que tenemos

$$A \to A\alpha \tag{3}$$
$$A \to \beta \tag{4}$$

Realizamos el cambio

$$A \to \beta A' \tag{5}$$
$$A' \to \alpha A' \tag{6}$$
$$A' \to \varepsilon \tag{7}$$

Para aplicar un método predictivo descendente, es necesario que la gramática esté **factorizada**,es decir, con tan solo leer un símbolo de la entrada se puede decidir que producción es la adecuada para aplciar la derivación.

3.4.2. Análisis ascendente (Down-Top).

Partir de la propia secuencia y buscar las suncadenas que coincidan con las partes derechas de las producciones y reescribirlas por la parte izquierda, hasta alcanzar el símbolo inicial de la gramática.

4. Métodos de análisis descendiente.

La tabla de análisis nos indica que producción de la gramática se han de aplicar ante un determinado símbolo terminal de entrada (incluyendo $). Para construir la tabla de análisis se utilizan los conjuntos de INICIALES y SEGUIDORES.

Los **INICIALES(A)** son aquellos símbolos terminales que resultan al principio de la cadena resultante de derivar A.
Los **SEGUIDORES(A)*** son aquellos símbolos terminales que siguen a una cadena derivada de A, es decir, despues de una cadena generada por A, encontramos un seguidor de A.

Para construir la tabla de análisis seguimos el siguiente algoritmo.

1. Creamos una tabla donde cada fila corresponde a un símbolo no terminal, y cada columna a uno terminal (junto con $).
2. Para cada producción de la gramática $A \to \alpha$.
   a) Añadimos la producción en las posiciones [A, INICIALES(α)].
   b) Si $\varepsilon \in INICIALES(\alpha)$ entonces también añadimos la producción a las posiciones [A, SEGUIDORES(A)].
3. Las posiciones vacías simbolizarán posiciones de «Error».

Diremos que una gramática es **LL(1)** cuando no hay mas de una producción en cada celta de la tablad de análisis. En caso contrario se dice que la gramática es ambigua LL(1).
Una gramática LL(1) se dice **simple** si todas las producciones que posean alternativas comienzan por símbolos terminales tistintos y además no aparecen producciones a la cadena vacía.
Cuando se alcanza una celda vacía en la tabla de análisis, entonces se puede afirmar que la secuencia de entrada es errónea sintacticamente. En este caso, un traductor de lenguajes debe

- Anunciar el mensaje, de forma que puedalocalizarse con facilidad.
- Recuperarse para poder seguir analizando el resto de los símbolos de entrada.

4.1. Tratamiento a nivel de frase.

Consiste en rellenar las celdas vacías de la tabla de análisis con marcas que nos permitan seleccionar un procedimiento concreto para el tratamiento del error según la celda vacía alcanzada. Para cada celda vacia de la tabla será necesario estudiar cuáles pueden ser las causas para alcanzarla, así como establecer los medios adecuados para recuperarse del error.

4.2. Tratamiento en modo pánico (Panic)

Detectado un error, el analizador se recupera del mismo saltando símbolos de la entrada hasta que aparezca un símbolo tal que con el símbolo situado en el tope de la pila, nos haga seleccionar una celda no vacía en la tabla de análisis.

Para optimizar el proceso de recuperación en modo pánico, será necesario delimitar los conjuntos de **símbolos de sincronización**.
Para ello, debemos concretar los tipos de errores que se desean tratar y, para cada caso concreto, adoptar los símbolos de sincronización adecuados.
Uno de los casos más conocidos consiste en, incluir como símbolos de sincronización a los seguidores de cada símbolo no terminal.

5. Métodos de análisis ascendiente

El **objetivo** es comprobar si hay una secuencia de símbolos pertenece a un lenguaje tal que aplicando reducciones sobre la cadena de entrada en base a lasproducciones de la gramática nos permita alcanzar el símbolo inicial de la gramática.

5.1. Método de reducción y desplazamiento.

Se trata de un método no predictivo que utiliza una máquina de pila donde

- El **alfabeto de la pila** está formado por símbolos terminales y no terminales.
- El **alfabeto de entrada** está formado por los símbolos terminales.
- La **función de transición** se define en base a las siguientes acciones sobre la pila, «Desplazar», «Reducir», «Aceptar» y «Error».

El proceso consiste en ir explorando el tope de la pila y los elementos siguientes en la pila hasta encontrar una subcadena τ que coindica con la parte derecha de una produccion dada $A \to \tau$.
Si se encuentra τ en la pila, entonces se sustituye τ por A en el tope de la pila. A este proceso se le denomina **reducción**. Y el proceso de pasar símbolos de entrada al tope de la pila se llama **desplazamiento**.
Posibles **problemas** de este método son:

- Existe más de una producción para aplicar reducción (conflicto reduce/reduce).
- Incertidumbre, si es posible aplicar desplazamiento en lugar de reduccion (conflicto desplaza/reduce).

6. Análisis sintáctico ascendente LR

Los analizadores LR(k) surgen como alternatica a los problemas que presentan las restricciones de gramáticas LL(n) para análisis descendente y las gramáticas de precedencia en análisis ascendente. En el siguiente ejemplo se ilustra el problema del error de reducción en análisis de precedencia simple.

- **L**. Analiza los símbolos de entrada de izquierda a derecha.
- **R**. Reduce a la derecha.
- **k**. Número de símbolos de anticipación de la entrada.

  Existen tres técnicas para la confección de la tabla de análisis. Cada una de ellas caracteriza a tres tipos de gramáticas LR.

  1. **LR Simple (SLR)**. Es la más facil de implementar pero la menos potente.
  2. **LR Canónica o LR(1)**. Es la más potente pero la más compleja.

3. **LALR (LR con anticipación o look-ahead).** Es la técnica intermedia y la más usada. Se trata de una simplificación del método general LR(1) y usado en las implementaciones de los generadores de analizadores sintácticos basados en YACC.

La técnica de análisis LR es válida para gramáticas libres del contextono ambiguas.

$$LL(k) \subset SLR \subset LALR \subset LR(k)$$

6.1. Analizadores LR(k)

Algunas de las ventajas de estos analizadores son

1. Puede reconocer virtualmente todos los lenguajes que obedecen a una gramática libre del contexto.
2. Es el método conocido más general sin retroceso, así como el más fácil de implementar de los conocidos de Reducción y Desplazamiento.
3. Las clases de gramáticas que pueden analizarse mediante un analizador LR son el conjunto de las gramáticas que peuden analziarse por analizadores predictivos.
4. El analizador LR puede detectar errores conforme se realiza el análisis, así como aplicar recuperación del análisis con pérdida controlada.

6.2. Tratamiento de la ambigüedad.

Todos los métodos de análisis sintáctico predicrivos requieren que la gramática no sea ambigua. Sin embargo, hay algunos casos de conflictor que pueden resolverse adoptando algún criterio o en base a informaciónadicional sobre el lenguaje (que no queda reflejado en la propia gramática), como ocurre con los operadores.

6.2.1. Conflictos reduce/desplaza

Si optamos por **reducir**, significa que estamos aceptando como válida la secuencia. En particular debemos asegurarnos que, bajo esa secuencia inicialmente aceptada, se alcance el símbolo inicial de la gramática, en el caso de que dicha secuencia fuese correcta.
Si optamos por **desplazar**, actuamos de forma contraria (acción por defecto en YACC).

6.2.2. Conflictos reduce/reduce.

Estos conflictos se resuelven facilmente eliminando producciones que resulten equivalentes.

6.3. Detección y recuperación de errores.

6.3.1. Modo pánico (Panic)

Saltar símbolos de la entrada y/o sacar símbolos de la pila hasta alcanzar una situación valida y continuar el análisis. El problema es decidir si se saca de la pila o se avanza en la entrada. Como solución sería posible la inserción de símbolos de sincronización.

6.3.2. Tabla de análisis

Completamos las celdas vacias de la tabla de análisis con procedimientos concretos para el tratamiento y recuperación de errores de forma que se pueda continuar el análisis.

6.3.3. Métodos complejos basados en el algoritmo de Graham-Rhodes

Permite recuperarse ante un error y además, en algunos casos, reparar el mismo. El método tiene dos fases.

- **Fase de condensación.** Se acota el contexto del error.
- **Fase de corrección.** Se intenta corregir el error.

Basándose en el análisis de los símbolos próximos en el tope de la pila y en la entrada, la fase de corrección decide si hay que sacar de la pila o saltar en la entrada, actuando como insertado o ignorado símbolos de la entrada.