CUP: Generadores de analizadores sintácticos (II)

Jesús Gallardo Casero
Asignatura *Teoría de la Computación*
E.U. Politécnica de Teruel
Universidad de Zaragoza

Contenidos

- Tratamiento de errores en Cup.
- Análisis semántico.
- Integración de aspectos semánticos en Cup.

Tratamiento de errores en Cup

- CUP presenta soporte para la recuperación de errores sintácticos mediante el uso del símbolo *error*.
- Este símbolo se incluirá en las producciones de la gramática como un símbolo más, con el objetivo de reconocer secuencias de entrada erróneas.
- Ejemplo:

```
expr_stmt ::= expr PYC | for_stmt PYC | error PYC;
```

- Si ninguna producción puede ser casada para *expr_stmt* por la entrada, entonces se declara un error de sintaxis e irán saltando tokens hasta que encuentre el punto y coma.

Tratamiento de errores en Cup

- Es posible sobreescribir los métodos de gestión de errores para cambiar la manera en la cual se tratan:

```
public void report_error(String message, Object info)
public void report_fatal_error(String message, Object
  info)
```

Análisis semántico

- Después de las fases de análisis léxico y sintáctico, la siguiente fase que integra un procesador de lenguajes es el **análisis semántico**.
- Dicha fase tiene dos funciones básicas:
  - Dar significado a las construcciones del lenguaje fuente.
  - Realizar comprobaciones semánticas.
    - No todas las sentencias correctas sintácticamente son válidas en el lenguaje.

Análisis semántico

- El análisis semántico puede realizarse a la vez que el sintáctico o en una segunda pasada tras la ejecución de aquel.
- En nuestro caso, la manera de implementarlo será añadir a la definición del lenguaje en Cup los aspectos semánticos.
- De esta forma, se generará un analizador sintáctico y semántico del lenguaje en cuestión.

Análisis semántico

- El mecanismo para realizar el análisis sintáctico y el semántico a la vez son las **gramáticas con atributos**.
- Se trata de añadir a la gramática libre de contexto:
  - **Atributos** a los símbolos de la gramática (terminales y no terminales).
  - **Reglas semánticas** a las producciones de la gramática, que operen con dichos atributos.

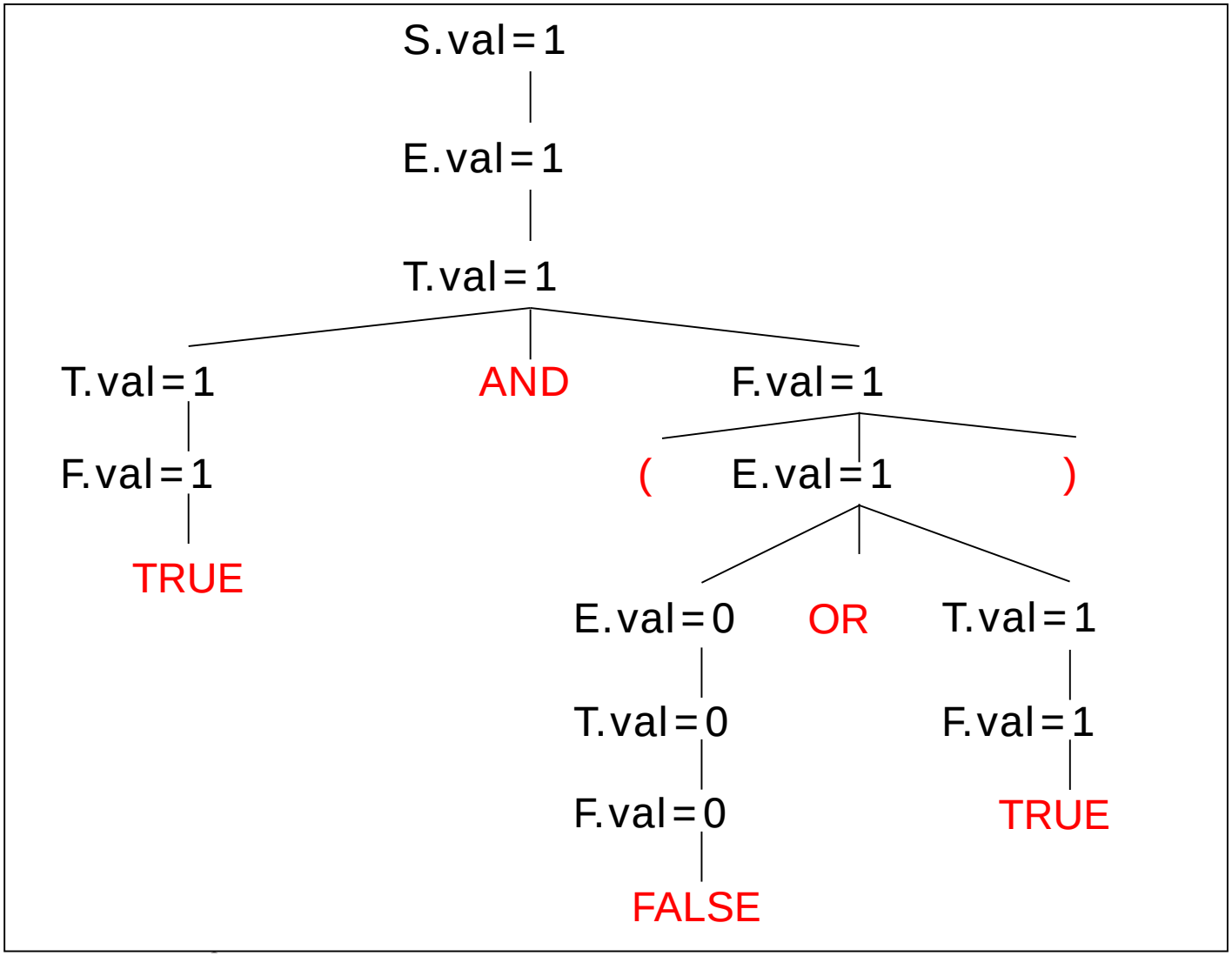
Análisis semántico

- Ejemplo:

Atributos: S, E, T, F, ID → val: entero

Producciones	Reglas Semánticas
S → E	S.val = E.val
E → E OR T	E.val = E_1.val $\vee$ T.val
E → T	E.val = T.val
T → T AND F	T.val = T_1.val $\wedge$ F.val
T → F	T.val = F.val
F → (E)	F.val = E.val
F → NOT E	F.val = $\neg$E.val
F → ID	F.val = ID.val
F → TRUE	F.val = 1
F → FALSE	F.val = 0

Análisis semántico



Integración de aspectos semánticos en Cup

- La integración de aspectos semánticos en Cup se lleva a cabo mediante la asignación de tipos a cada símbolo de la gramática.
- Esos tipos pueden ser tipos primitivos de Java o cualquier otra clase, existente o definida por nosotros.
- En la lista de símbolos se especifica el tipo:

```
terminal Integer OPERANDO;
non terminal Integer expr, factor, term;
```

Integración de aspectos semánticos en Cup

- Una vez que cada atributo tiene un tipo, es posible definir las reglas semánticas que manipulan dichos atributos.
- Las reglas son código Java que se añade en cualquier lugar de las partes derechas de las producciones (habitualmente al final).
- Para acceder a los distintos símbolos de la producción, suelen utilizarse *alias*.

Integración de aspectos semánticos en Cup

- Al símbolo de la parte izquierda de la producción se le referencia con el alias RESULT.
- Ejemplo:

```
NUMEROS ::= NUMEROS:tot NUM:n {: RESULT = n+tot; :}
```

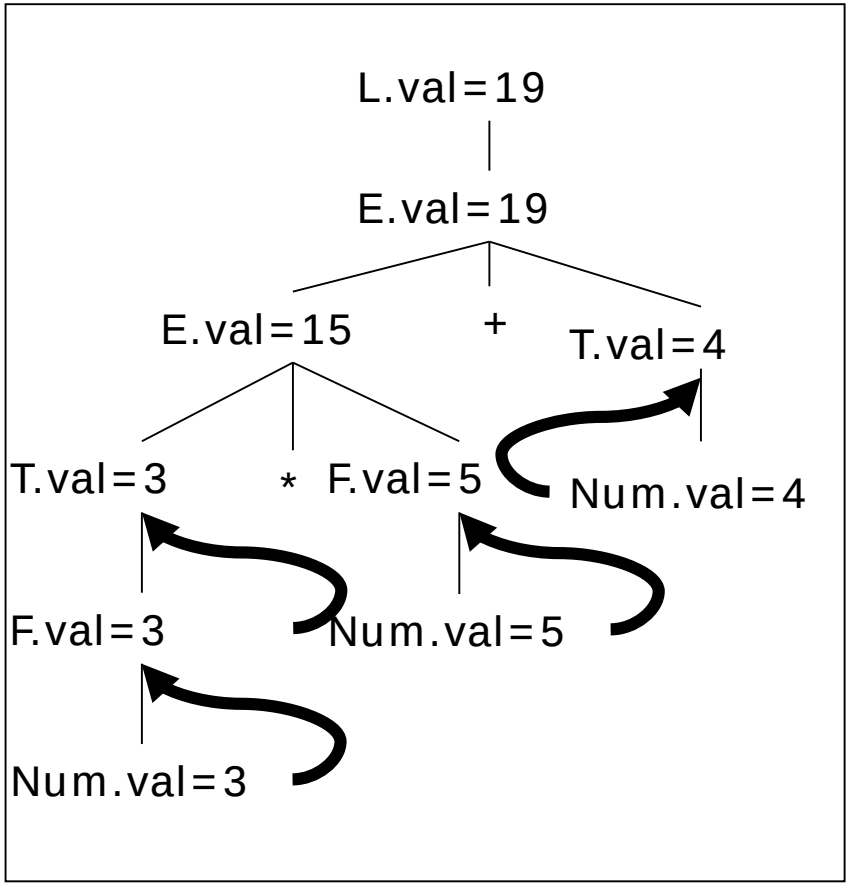
  - Esta regla almacenaría en el NUMEROS de la izquierda la suma de los valores del NUMEROS de la derecha y de NUM.
  - Todos ellos deberían ser de tipo Integer.

Integración de aspectos semánticos en Cup

- Para realizar el correcto seguimiento de la ejecución de las reglas, hay que recordar que el análisis es **ascendente**.
- Por lo tanto, primero se reducen las producciones formadas sólo por terminales, y luego se van reduciendo las de nivel superior hasta llegar a la del símbolo inicial.

Integración de aspectos semánticos en Cup

Producciones	*Reglas Semánticas*
L → E	L.val=E.val
E → E + T	E.val=E_1.val + T.val
E → T	E.val=T.val
T → T * F	T.val=T_1.val * F.val
T → F	T.val=F.val
F → (E)	F.val=E.val
F → Num	F.val=Num.val



Integración de aspectos semánticos en Cup

- En cuanto a los errores semánticos, para informar de ellos se puede *forzar* la llamada al método *report_error*.
- Ejemplo:

```
OP_DIV ::= NUM:n1 DIV NUM:n2
{: if (n2==0) parser.report_error
  (“División entre cero”, n1); :}
```

Integración de aspectos semánticos en Cup

- Otro aspecto que suele considerarse en el análisis semántico es la gestión de la **tabla de símbolos**.
- Se trata de una estructura de datos en la cual se insertan todos los elementos del lenguaje que reciben un nombre en el texto de entrada:
  - En un lenguaje de programación: variables, funciones, clases, etc.
- Si el lenguaje exige que estos elementos se declaren antes de usarlos, habrá que realizar esa comprobación semántica.
- Además, la tabla puede incluir información adicional sobre los símbolos: línea en que se declaró, etc.