Tema 1.3. Un lenguaje mínimo y su procesador: *Restricciones contextuales*

Profesor
Federico Peinado

Elaboración del material
José Luis Sierra
Federico Peinado

Ingeniería en Informática
Facultad de Informática – Universidad Complutense de Madrid
Curso 2009-2010

Restricciones contextuales y lenguajes contextuales

- Como ya hemos mencionado, la validez sintáctica de *una parte* de un programa informático suele depender de *otras partes* que *preceden* a esta
  → Su validez sintáctica ***depende del contexto***
  - Cada condición que deban cumplir estas partes interdependientes es una **restricción contextual** (ej. las constantes utilizadas deben haber sido declaradas)

- Los lenguajes de programación suelen ser, por tanto, **lenguajes contextuales** en la práctica
  → ¡No se pueden generar con *gramáticas incontextuales*!
  - Ej. Algo tan simple como esto ya es un lenguaje contextual:
    $L = \{\alpha\&\alpha$ tal que $\alpha \in \{0,1\}^*\}$

- Afortunadamente, para definir un lenguaje contextual (expresando todas sus restricciones contextuales) podemos usar una **gramática de atributos**

Definir una gramática de atributos para un lenguaje contextual

- Para definir la gramática de atributos que genera un lenguaje contextual determinado se siguen estos pasos:
  1. Se eliminan las restricciones contextuales, **relajando el lenguaje** hasta hacer otro "parecido" pero incontextual
  2. Se añaden las restricciones contextuales otra vez, pero sobre la gramática incontextual de ese nuevo lenguaje, usando **atributos y ecuaciones semánticas**

- Ej. Paso 1

  $L_{relajado} = \{\alpha\&\beta$ tal que $\alpha \in \{0,1\}^*$ y $\beta \in \{0,1\}^*\}$

  $G_{relajado}$ es incontextual:

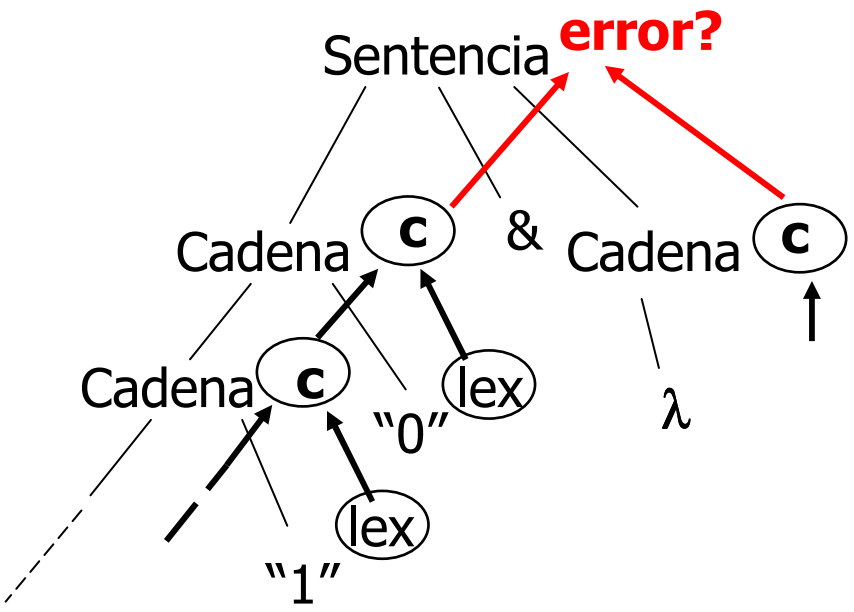
```
Sentencia ::= Cadena & Cadena
Cadena ::= Cadena 0 | Cadena 1 | λ
```

Definir una gramática de atributos para un lenguaje contextual

- Ej. Paso 2

  Debo asegurarme de que $\alpha = \beta$ para volver al lenguaje L

  Idea: Considerar α y β como dos atributos que se sintetizan a base de concatenar los 0's y 1's que los componen y comprobar que sus valores (cadenas) son iguales



* De alguna forma hay que expresar el **error** si las dos cadenas completas que componen la sentencia no son iguales al final

Atributo de error

- Normalmente se asocia un atributo sintetizado **err** con el axioma de la gramática (raíz del árbol)
  - Este atributo vale *true* si la sentencia contiene algún **error** (*violación de una restricción contextual*) y *false* en otro caso
  - Este **mecanismo básico de detección de errores**, puede enriquecerse haciendo que el atributo contenga *más información* sobre *todos los errores producidos,* por ejemplo
- Ej. Gramática de atributos resultante para L

```
Sentencia ::= Cadena & Cadena
  Sentencia.err = (Cadena_0.c ≠ Cadena_1.c)
Cadena ::= Cadena 0
  Cadena_0.c = Cadena_1.c || "0"
Cadena ::= Cadena 1
  Cadena_0.c = Cadena_1.c || "1"
Cadena  ::= λ
  Cadena.c = ""
```

Concatenación **||** de cadenas de caracteres

¡Se descarta toda sentencia en la que el atributo **err** se sintetiza a *true*!

Atributo de error y tabla de símbolos

- En el ejemplo anterior era trivial comprobar si la sentencia era errónea, pero normalmente en un lenguaje de programación **se necesita conocer la tabla de símbolos** para evaluar si hay error en algún punto del programa
  - Y de hecho los errores no se evalúan solamente en la raíz del árbol sintáctico, sino en muchos nodos más profundos
- La tabla de símbolos , que se *sintetiza* en la *sección de declaraciones* de un programa (atributo *ts*), pasa a ser un **atributo heredado (ts_h) <u>en la sección de código</u>**
  → Hasta las categorías sintácticas de los niveles más bajos del árbol tendrán *acceso a la tabla de símbolos* para consultarla en la evaluación de posibles errores

Ejemplo: Restricciones contextuales sobre las asignaciones

4. Definición sintáctica contextual

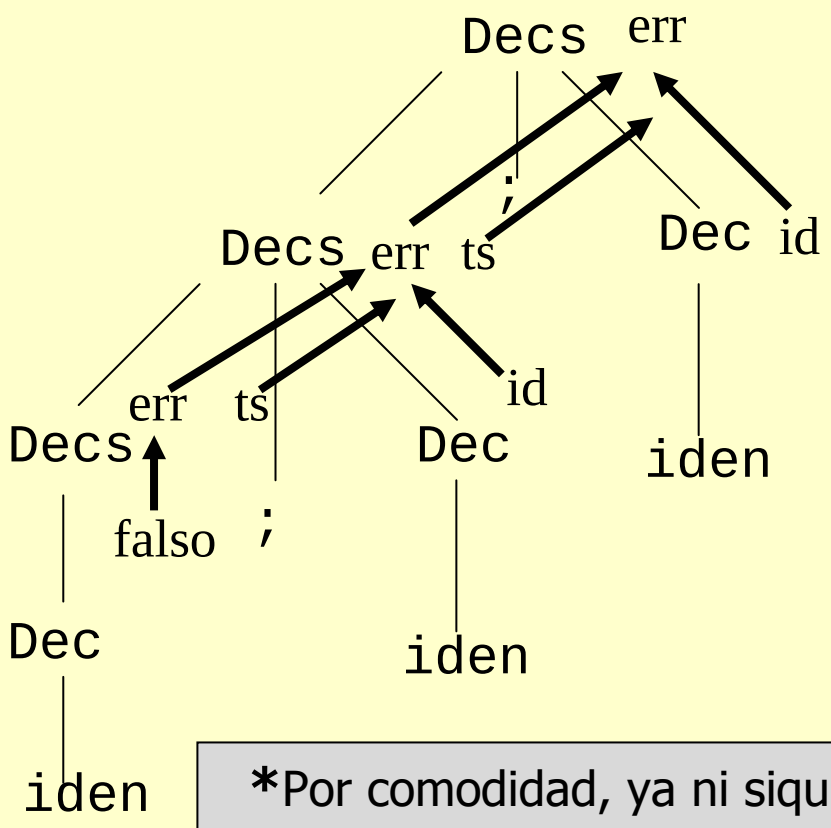
- **Descripción informal** de restricciones contextuales
  - Vamos a considerar dos de las más básicas:

    "Los identificadores utilizados en las instrucciones de la sección de código (tanto a la izquierda como a la derecha de la asignación) deben haber sido declarados previamente en la sección de declaraciones"

    "Un mismo identificador sólo puede aparecer una vez en la sección de declaraciones"

Ejemplo: Restricciones contextuales sobre las asignaciones

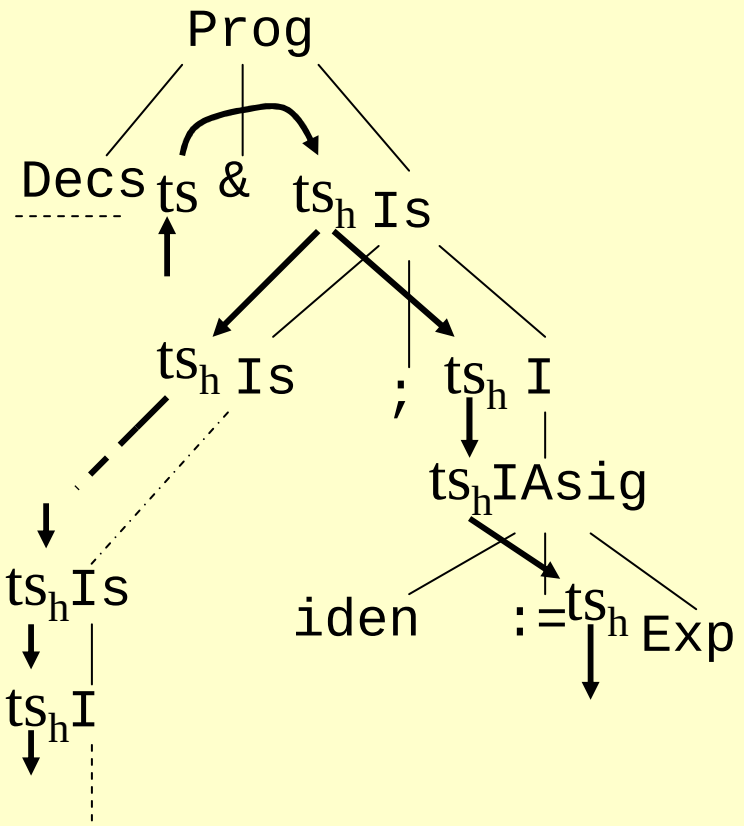
4. Definición sintáctica contextual
   - Error y tabla de símbolos *sintetizados* en las declaraciones



*Por comodidad, ya ni siquiera ponemos elipses alrededor de los atributos en el grafo de dependencias:
ts debería ser (ts) , **err** debería ser (err) , etc.

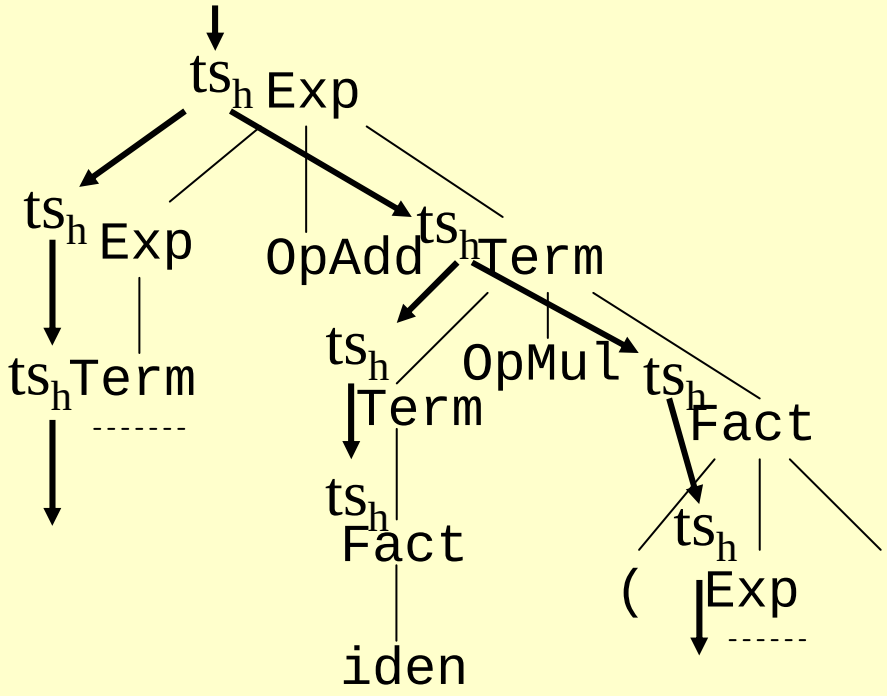
Ejemplo: Restricciones contextuales sobre las asignaciones

4. Definición sintáctica contextual
   - Tabla de símbolos *heredada* (ts_h) en el código



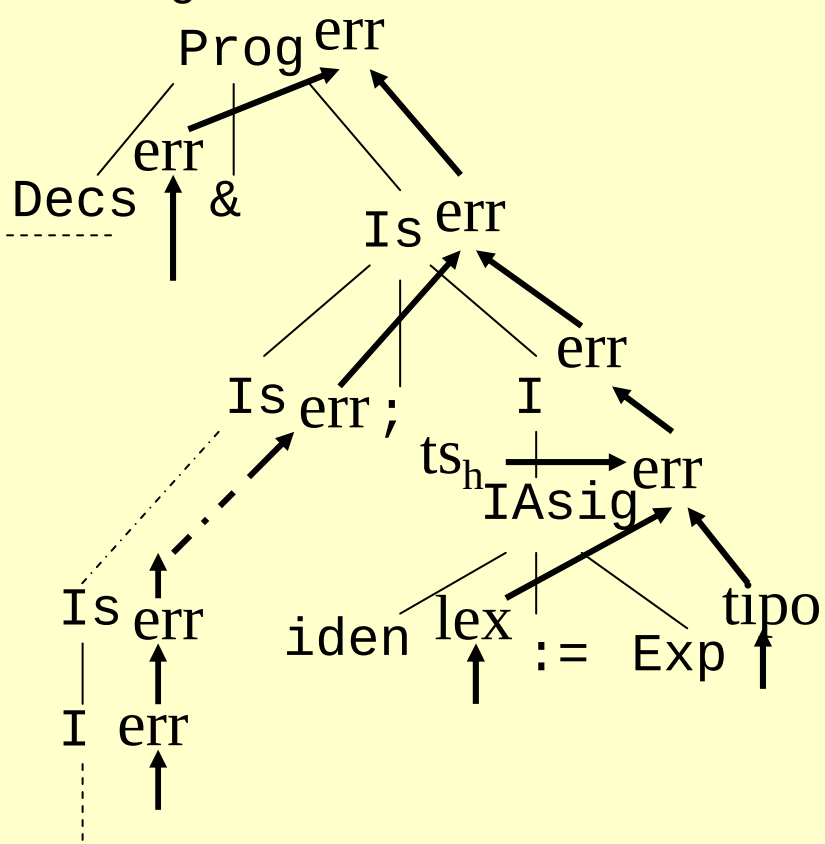
Ejemplo: Restricciones contextuales sobre las asignaciones

4. Definición sintáctica contextual
   - Tabla de símbolos *heredada* (ts_h) en el código *(continuación…)*



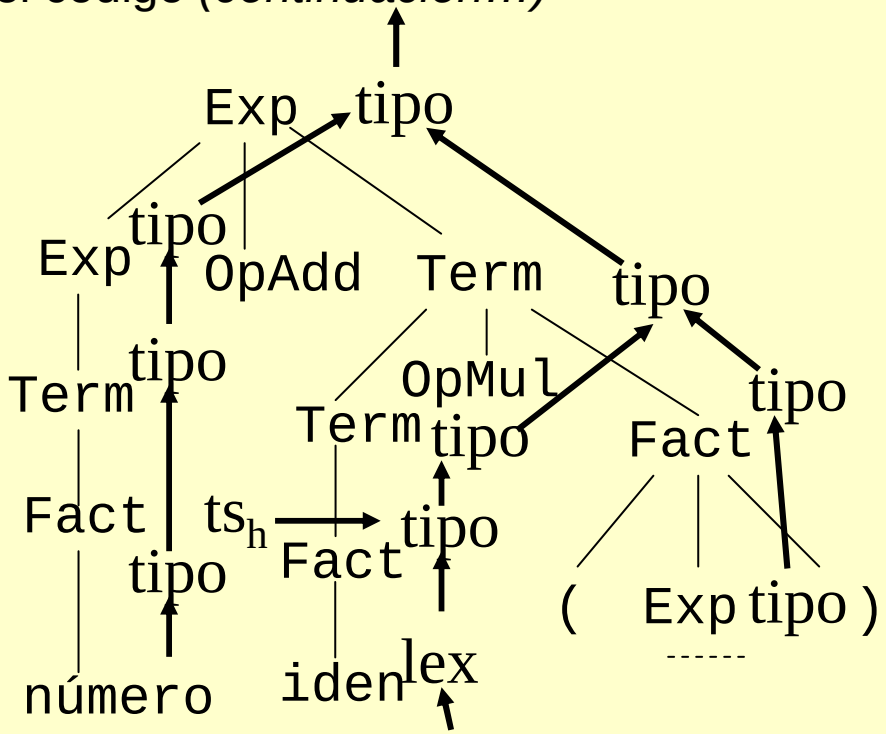
Ejemplo: Restricciones contextuales sobre las asignaciones

4. Definición sintáctica contextual
   - Error y tipo *sintetizados* (y tabla de símbolos *heredada*) en el código



Ejemplo: Restricciones contextuales sobre las asignaciones

4. Definición sintáctica contextual
   - Error y tipo *sintetizados* (y tabla de símbolos *heredada*) en el código *(continuación…)*



*Se aprovecha el atributo **tipo** de las expresiones para guardar un "tipo erróneo" en caso de incompatibilidad de tipos, y así no tener que usar el atributo **err** en esta parte

Ejemplo: Restricciones contextuales sobre las asignaciones

4. Definición sintáctica contextual
   - Error y tabla de símbolos en las declaraciones

```
Decs ::= Dec
   Decs.err = false
 Decs ::= Decs ; Dec
   Decs0.err = Decs1.err ∨ existeID(Decs1.ts,Dec.id)
```

Ejemplo: Restricciones contextuales sobre las asignaciones

4. Definición sintáctica contextual
   - Error y tipo (y tabla de símbolos) en el código

```
Prog ::=  Decs & Is
   Prog.err = Decs.err ∨ Is.err
   Is.ts_h = Decs.ts
Is ::= I
   I.ts_h = Is.ts_h
   Is.err = I.err
 Is ::= Is ; I
    Is_1.ts_h = I.ts_h = Is_o.ts_h
    Is_o.err = Is_1.err ∨ I.err
 I ::= IAsig
   IAsig.ts_h = I.ts_h
   I.err = IAsig.err
 IAsig ::= iden := Exp
   Exp.ts_h = IAsig.ts_h
   IAsig.err = (Exp.tipo = terr) ∨
                ¬ existeID(IAsig.ts_h,iden.lex)
...
```

Ejemplo: Restricciones contextuales sobre las asignaciones

4. Definición sintáctica contextual

- Error y tipo (y tabla de símbolos) en el código *(continuación…)*

```
Exp ::= Exp OpAd Term
   Exp1.tsh = Term.tsh = Expo.tsh
   Expo.tipo = si (Exp1.tipo = terr ∨ Term.tipo = terr) terr
               si no tnum
 Exp  ::= Term
   Term.tsh = Exp.tsh
   Exp.tipo = Term.tipo
 Term ::= Term OpMul Fact
   Term1.tsh = Fact.tsh = Termo.tsh
   Termo.tipo = si (Term1.tipo = terr ∨ Fact.tipo = terr) terr
                si no tnum
 Fact ::= num
   Fact.tipo = tnum
 Fact ::= iden
   Fact.tipo = si ¬existeID(Fact.tsh,iden.lex) terr
               si no tnum
 Fact ::= ( Exp )
   Exp.tsh = Fact.tsh
   Fact.tipo = Exp.tipo
```

Críticas, dudas, sugerencias...

Federico Peinado
www.federicopeinado.es