Universidad Rey Juan Carlos – Escuela Técnica Superior de Ingeniería Informática
Grado en Ingeniería Informática (Móstoles) – Procesadores de Lenguajes
Prueba tema 4 – Convocatoria ordinaria Mayo 2015

Se quiere diseñar una calculadora de escritorio con capacidad para operaciones lógicas NAND, NOR y NOT. A continuación se proporciona su gramática, así como las tablas de verdad de los operadores NAND y NOR:

```
S ::= A D
D ::= nand A D | λ
A ::= B R
R ::= nor B R | λ
B ::= cte OPR cte | not B
OPR ::= < | > | =
```

A	B	A nand B	A nor B
T	T	F	F
T	F	T	F
F	T	T	F
F	F	T	T

El diseño del traductor que implemente dicha calculadora debe tener en cuenta que:

- Cuando termine el cálculo debe imprimir el resultado, “true” o “false”.
- Para los cálculos **sólo** se pueden usar (aunque no es obligatorio usar todo):
  - sentencias de asignación de valores.
  - sentencias de control de flujo tipo “if-then”, “if-then-else” o “switch-case”.
  - los operadores lógicos and, or y not.
  - los operadores de comparación menos, mayor e igual.

Se pide:

- **[6 ptos]** Especificar el traductor dirigido por la sintaxis descrito anteriormente.
- **[2 ptos]** Explicar de forma justificada si el traductor especificado podría implementarse mediante un traductor descendente recursivo:
  - Si se pudiera, hacer el traductor descendente recursivo del símbolo **S** de la gramática.
  - Si no se pudiera, hacer las modificaciones pertinentes en la gramática y hacer el traductor descendente recursivo del símbolo **S** de la gramática.
- **[1 pto]** Explicar de forma justificada si se ha usado una definición dirigida por la sintaxis o un esquema de traducción.
- **[1 pto]** Por cada atributo utilizado en el traductor, explicar de forma justificada si es heredado o sintetizado.

Solución

A continuación se ofrece una posible solución al examen, indicando las respuestas a cada pregunta planteada:

1. Especificar el traductor dirigido por la sintaxis descrito anteriormente.

```
S ::= A {D.vh = A.vs;} D {imprime(D.vs);}

D ::= nand A {D1.vh = !D.vh or !A.vs;} D1 {D.vs = D1.vs;}

      | λ {D.vs = D.vh;}

A ::= B {R.vh = B.vs;} R {A.vs = R.vs;}

R ::= nor B {R1.vh = !R.vh or !B.vs;} R1  {R.vs = R1.vs;}

      | λ {R.vs = R.vh;}

B ::= cte1 OPR cte2 {

            switch(OPR.ops){

                  case 'l': B.vs = cte1.valex < cte2.valex;

                  case 'g': B.vs = cte1.valex > cte2.valex;

                  case 'e': B.vs = cte1.valex == cte2.valex;

            }

      }

      | not B1  {B.vs = !B1.vs;}

OPR ::= < {OPR.ops = 'l';}

      | >  {OPR.ops = 'g';}

      | =  {OPR.ops = 'e';}
```

2. Explicar de forma justificada si el traductor especificado podría implementarse mediante un traductor descendente recursivo:
   a. Si se pudiera, hacer el traductor descendente recursivo del símbolo **S** de la gramática.
   b. Si no se pudiera, hacer las modificaciones pertinentes en la gramática y hacer el traductor descendente recursivo del símbolo **S** de la gramática.

La gramática es LL(1), por lo tanto se puede implementar un traductor descendente recursivo. A continuación se muestra la especificación del no terminal S y el código correspondiente:

```
S ::= A {D.vh = A.vs;} D {imprime(D.vs);}
```

```
void S ( void ) {
      Int D_vh, D_vs;
      D_vh = A();
      D_vs = D(D_vh);
      Imprime (D_vs);
}
```

3. Explicar de forma justificada si se ha usado una definición dirigida por la sintaxis o un esquema de traducción.

Se ha utilizado un esquema de traducción, puesto que hemos especificado las acciones semánticas en puntos concretos del consecuente. Por ejemplo, en la siguiente producción, la acción:

{D_1.vh = !D.vh or !A.vs;}

se ejecuta después de procesar el símbolo **A** y antes de procesar el símbolo **D_1**.

D ::= nand A {D_1.vh = !D.vh or !A.vs;} D_1 {…}

4. Por cada atributo utilizado en el traductor, explicar de forma justificada si es heredado o sintetizado.

Para facilitar su reconocimiento, los atributos sintetizados tienen un nombre terminado en "s" y los heredados en "h". Los atributos sintetizados corresponden a símbolos del antecedente y se les asigna valor en función del consecuente. En el siguiente ejemplo el atributo sintetizado es **R.vs** (del antecedente) al que asignamos valor en el consecuente, en este caso usando un atributo **R_1.vs** de un símbolo del consecuente:

R ::= nor B {…} R_1 {R.vs = R_1.vs;}

Los atributos heredados corresponden a símbolos del consecuente y toman valor en ese mismo consecuente, pudiendo usar:

- valores procedentes del antecedente como en este ejemplo **D.vh**:
  D ::= nand A {D_1.vh = !D.vh or !A.vs;} D_1 {…}

- o valores sintetizados de símbolos del mismo consecuente, como **A.vs** en el ejemplo:
  D ::= nand A {D_1.vh = !D.vh or !A.vs;} D_1 {…}

Criterios de corrección

El traductor

Fallos muy graves:

- Usar atributos sin valor asignado.

Fallos graves:

- Asignar valores a atributos con posteriormente no se usan o no se pueden usar.

Fallos leves:

- Cálculo erróneo de las operaciones.

Cuestiones

Fallos muy graves:

- No justificar las respuestas.
- Justificar lo contrario de lo que se responde.