4º Ingeniería Informática

II26 Procesadores de lenguaje

Control de teoría, grupo TE-1, 22 de mayo de 2008 (solución)

PREGUNTA 1 (0,5 PUNTOS)

Sea G la siguiente gramática de parte de un lenguaje de programación:

$$
\begin{array}{rcl}
\langle \text{Programa} \rangle & \rightarrow & \langle \text{LSentencias} \rangle \\
\langle \text{LSentencias} \rangle & \rightarrow & \langle \text{Sentencia} \rangle \langle \text{LSentencias} \rangle \mid \lambda \\
\langle \text{Sentencia} \rangle & \rightarrow & \textbf{bucle} \ \langle \text{LSentencias} \rangle \ \textbf{hasta} \ \langle \text{Expresión} \rangle \ \textbf{fin bucle} \\
\langle \text{Sentencia} \rangle & \rightarrow & \textbf{si} \ \langle \text{Expresión} \rangle \ \textbf{entonces} \ \langle \text{LSentencias} \rangle \ \textbf{fin si} \\
\langle \text{Sentencia} \rangle & \rightarrow & \textbf{interrumpe} \ ; \\
\langle \text{Sentencia} \rangle & \rightarrow & \textbf{escribe} \ \langle \text{Expresión} \rangle \ ; \\
\langle \text{Expresión} \rangle & \rightarrow & \textbf{id} \\
\langle \text{Expresión} \rangle & \rightarrow & \textbf{cte}
\end{array}
$$

Diremos que un bucle es degenerado si tiene en su interior una sentencia **interrumpe** que no esta dentro de ninguna sentencia condicional. Esto implica que el cuerpo del bucle se ejecutará como máximo una vez y sólo hasta la sentencia, de ahí el calificativo de degenerado.

Añade a la gramática las acciones semánticas necesarias para calcular el número de bucles degenerados en un programa. Este número se almacenará en el atributo *nbd* (número de bucles dengenerados) de ⟨Programa⟩.

Por ejemplo, en el programa siguiente

```
1     escribe a;
2     bucle
3       bucle
4         escribe b;
5         interrumpe;
6       hasta c;
7       si a entonces
8         b:= c;
9         interrumpe;
10      fin si
11    hasta d;
```

el valor de *nbd* es 1, debido a que el primer **interrumpe** (línea 5) no está dentro de ninguna sentencia condicional. El segundo **interrumpe** (línea 9) sí está guardado por una condición, por lo que el bucle no es degenerado.

Puedes añadir los atributos que consideres necesarios, tanto heredados como sintetizados, con dos restricciones: deben ser de tipo simple (entero o lógico) y no puedes emplear como heredados atributos sintetizados y viceversa.

Incluye una tabla con los atributos que crees similar a la siguiente:

Atributo	Clase	Tipo	Significado
nbd	Sintetizado	Entero	Número de bucles degenerados

SOLUCIÓN

Usaremos la siguiente aproximación: un bucle será degenerado si en la lista de sentencias de su cuerpo hay un interrumpe sin guardar. Esto es muy similar al problema de saber si hay un interrumpe fuera de un bucle (basta con preguntarse si hay uno fuera de un sí). Por eso, utilizamos el atributo *isg* (interrumpe sin guardar) para controlarlo.

La tabla de atributos es

Atributo	Clase	Tipo	Significado
nbd	Sintetizado	Entero	Número de bucles degenerados
isg	Sintetizado	Lógico	Hay interrumpes sin guarda

Y las acciones son:

⟨Programa⟩ → ⟨LSentencias⟩ {⟨Programa⟩.nbd:= ⟨LSentencias⟩.nbd;}

⟨LSentencias⟩ → ⟨Sentencia⟩⟨LSentencias$_1$⟩ {⟨LSentencias⟩.nbd:= ⟨Sentencia⟩.nbd+ ⟨LSentencias$_1$⟩.nbd;
⟨LSentencias⟩.isg:= ⟨Sentencia⟩.isg **o** ⟨LSentencias$_1$⟩.isg;}

⟨Lsentencias⟩ → λ {⟨LSentencias⟩.nbd:=0; ⟨LSentencias⟩.isg:= falso;}

⟨Sentencia⟩ → **bucle** ⟨LSentencias⟩ **hasta** ⟨Expresión⟩ **fin bucle**
{**si** ⟨LSentencias⟩.isg **entonces**
⟨Sentencia⟩.nbd:= ⟨LSentencias⟩.nbd+1 **si no**
⟨Sentencia⟩.nbd:= ⟨LSentencias⟩.nbd **fin si**;
⟨Sentencia⟩.isg:= falso;}

⟨Sentencia⟩ → **si** ⟨Expresión⟩ **entonces** ⟨LSentencias⟩ **fin si** {⟨Sentencia⟩.nbd:= ⟨LSentencias⟩.nbd;
⟨Sentencia⟩.isg:= falso;}

⟨Sentencia⟩ → **interrumpe ;** {⟨Sentencia⟩.nbd:= 0;
⟨Sentencia⟩.isg:= cierto;}

⟨Sentencia⟩ → **escribe** ⟨Expresión⟩ **;** {⟨Sentencia⟩.nbd:= 0;
⟨Sentencia⟩.isg:= falso;}

⟨Expresión⟩ → **id**

⟨Expresión⟩ → **cte**

Pregunta 2 (0,5 puntos)

Escribe el esquema de generación de código para el operador máximo. Este operador tiene dos operandos i y d y devuelve el máximo de ambos, garantizando que sólo se evalúan una vez.

Asume que ambos operandos son de tipo entero.

Solución

En primer lugar, generaremos código para ambos operandos y después compararemos los registros resultantes para asegurarnos de que en el izquierdo se guarda el máximo de ambos.

Objeto NodoMáximo:
...
Método generaCódigo()
izda:= i.generaCódigo();
dcha:= d.generaCódigo();
l :=*nuevaEtiqueta*()
emite(`si` izda `>=` dcha `salta` l);
emite(izda`:=` dcha);
emite(l:);
liberaReg(dcha);
devuelve izda;
fin generaCódigo
...
fin NodoMáximo