E79 Procesadores de lenguaje

Examen de teoría (19 de diciembre de 2001)

PREGUNTA 1 (4 PUNTOS)

Supongamos que tienes la siguiente gramática para un lenguaje de expresiones:

⟨Programa⟩ → (⟨Sentencia⟩)*
⟨Sentencia⟩ → ⟨Expresión⟩ ; | **id**<-⟨Expresión⟩ ;
⟨Expresión⟩ → ⟨Expresión⟩ **opad** ⟨Expresión⟩ | ⟨Expresión⟩ **opmul** ⟨Expresión⟩ | ⟨Expresión⟩ **desp** ⟨Expresión⟩
⟨Expresión⟩ → ⟨Expresión⟩ **and** ⟨Expresión⟩ | ⟨Expresión⟩ **or** ⟨Expresión⟩
⟨Expresión⟩ → (⟨Expresión⟩) | **not** ⟨Expresión⟩ | **id** | **ent** | **real**

Es decir, un programa en este lenguaje consta de una serie de asignaciones y expresiones. Las primeras tienen como efecto cambiar el valor de la variable en la parte izquierda; las segundas escriben el resultado de la expresión. Los operadores son asociativos a izquierdas, excepto el de desplazamiento, que lo es a derechas, y su prioridades son, de mayor a menor:

opmul and
opad or
desp

El operador **desp** (representado con `<<`) corresponde a un desplazamiento a la izquierda del primer operando, que debe tener tipo entero, tantas veces como indique el segundo, que también debe ser de tipo entero. Así, `5<<3` da como resultado `40`. Los tipos de las expresiones siguen las reglas habituales:

- El tipo del resultado de las operaciones aritméticas es el más general de los tipos de los operandos.
- Las operaciones lógicas y de desplazamiento únicamente admiten operandos enteros.

Además, el tipo de una variable es el tipo de la última expresión que se le asignó.

En el caso de las operaciones lógicas, se considera como verdadero cualquier número distinto de cero y se obtiene como resultado el valor cero o uno, según corresponda.

Quieres hacer un compilador que permita transformar esos programas en programas del lenguaje `miniC`. Este lenguaje es similar a `C`, con la siguientes diferencias:

- Los programas constan únicamente de la función `main`.
- Los únicos tipos permitidos para las variables son `float` e `int`.
- No hay sentencias compuestas.
- Las sentencias de escritura son `printint` y `printfloat`.
- Las partes derechas de las asignaciones sólo pueden ser constantes o variables.
- Se pueden utilizar los operadores de asignación `+=`, `-=`, `*=` y `/=`.
- Sólo se permiten dos instrucciones de control de flujo:
  - Instrucciones condicionales con condiciones que sean una constante o una variable.
  - Instrucciones `for` de la forma
    `for ( `v` =0 ; `v` < `c` ; `v` ++ )`
    Con v una variable y c una constante o una variable.

Así, la traducción de: `a<-1; 2*a-1;` podría ser:

```
int main ()
  {
    int a, _temp;
    a=1;
    _temp=2;
    _temp*=a;
    _temp-=1;
    printint(_temp);
    return 0;
  }
```

Se pide:

- Añadir a la gramática original las acciones semánticas necesarias para efectuar las traducciones al lenguaje `miniC`;
- o bien, añadir las acciones semánticas necesarias para obtener una representación intermedia y describir los algoritmos de paso de la representación intermedia al lenguaje `miniC`

Observaciones:

- No se exige comprobación de tipos, pero sí que necesitas saber el tipo de las expresiones para generar adecuadamente las declaraciones de las variables y las instrucciones de escritura.
- Puedes asumir que no se utiliza ninguna variable sin que se le haya asignado previamente un valor.
- Puedes asumir que ninguna variable de usuario comienza por `_` (subrayado).
- Intenta ver cómo calcularías `-1 or 1` (el resultado debe ser `1`).

Pregunta 2 (2 puntos)

Transforma la gramática de la pregunta anterior de modo que pueda ser analizada en tiempo lineal. Demuestra que tu nueva gramática permite el análisis en tiempo lineal y escribe las acciones semánticas asociadas a las producciones en que se ha transformado la regla ⟨Expresión⟩ → ⟨Expresión⟩ **desp** ⟨Expresión⟩.

Pregunta 3 (2 puntos)

Escribe expresiones regulares, o demuestra que no existen, para los siguientes lenguajes:

- Cadenas sobre el alfabeto `{a,b,c}` tales que si aparece al menos una `b`, debe haber al menos una `a`.
- Cadenas sobre el alfabeto `{a,b,c}` tales que no contienen la subcadena `ba`.
- Números binarios mayores que `101001`.
- Números binarios de seis bits palíndromos.

Pregunta 4 (2 puntos)

Nos han propuesto la siguiente gramática para resolver el problema de la ambigüedad del if-then:

$$
\begin{array}{lcl}
\langle \mathsf{S} \rangle & \rightarrow & \textbf{if } \langle \mathsf{E} \rangle \textbf{ then } \langle \mathsf{S} \rangle \\
\langle \mathsf{S} \rangle & \rightarrow & \langle \mathsf{M} \rangle \\
\langle \mathsf{M} \rangle & \rightarrow & \textbf{if } \langle \mathsf{E} \rangle \textbf{ then } \langle \mathsf{M} \rangle \textbf{ else } \langle \mathsf{S} \rangle \\
\langle \mathsf{M} \rangle & \rightarrow & \textbf{otro} \\
\langle \mathsf{E} \rangle & \rightarrow & \textbf{id}
\end{array}
$$

Demuestra que esta gramática es ambigua.