E79 Procesadores de lenguaje

Examen de teoría (18 de septiembre 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 ent**
⟨Expresión⟩ → ⟨Expresión⟩ **and** ⟨Expresión⟩ | ⟨Expresión⟩ **or** ⟨Expresión⟩
⟨Expresión⟩ → (⟨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 y su prioridades son, de mayor a menor:

opmul and
opad or
desp

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

- El tipo del resultado de las operaciones aritmeticas 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 para su ejecución en una máquina virtual. La máquina tiene memoria separada para instrucciones y datos y también cuenta con dos registros, `r1` y `r2`, que funcionan como una pila (`r1` es el tope de la pila). La memoria de datos está dividida en un número ilimitado de palabras que pueden albergar indistintamente un entero o un real. Las instrucciones que tiene la máquina para manipular números enteros son:

- `add`: suma `r1` con `r2`, dejando el resultado en `r1`.
- `sub`: resta `r1` de `r2`, dejando el resultado en `r1`.
- `mul`: multiplica `r1` por `r2`, dejando el resultado en `r1`.
- `div`: divide `r2` entre `r1`, dejando el resultado en `r1`.
- `not`: si `r1` es cero, pasa a valer uno, en otro caso, pasa a ser cero.
- `print`: escribe en pantalla el contenido de `r1`, interpretrado como un entero.
- `push` n: copia en `r2` el contenido de `r1` e introduce en `r1` el entero n.
- `push &`d: copia en `r2` el contenido de `r1` y en `r1` el contenido de la dirección d.
- `pop`: copia en `r1` en valor de `r2`.
- `pop &`d: copia en la dirección d el valor de `r1` y en `r1` el valor de `r2`.

Además, existen las instrucciones `fadd`, `fsub`, `fmul`, `fdiv`, `fprint`, `fpush` y `fpop` con comportamiento análogo pero interpretando sus operandos como números reales.

Las instrucciones `tofloat` y `toint` transforman `r1` de entero a real y viceversa.

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

```
push 1        mul
pop &1        push 1
push 2        sub
push &1       print
```

Se pide:

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

Observaciones:

- Es importante recordar que la pila tiene **profundidad dos**; piensa cómo traducirías `a<-1*(2-(3+4));`.
- No se exige comprobación de tipos, pero sí que necesitas saber el tipo de las expresiones para generar adecuadamente las instrucciones.
- Puedes asumir que no se utiliza ninguna variable sin que se le haya asignado previamente un valor.
- 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⟩ **opad** ⟨Expresión⟩.

Pregunta 3 (2 puntos)

¿Qué lenguajes representan las siguientes expresiones regulares?

- `0(0|1)*0`
- `(0|1)*0(0|1)(0|1)`
- `0*10*10*10*`
- `(00|11)*((01|10)(00|11)*(01|10)(00|11)*)*`

Pregunta 4 (2 puntos)

Construye la tabla de análisis SLR para la gramática siguiente:

$$\begin{aligned} \langle \mathsf{E} \rangle &\rightarrow \langle \mathsf{E} \rangle \mathtt{+} \langle \mathsf{T} \rangle \mid \langle \mathsf{T} \rangle \\ \langle \mathsf{T} \rangle &\rightarrow \langle \mathsf{T} \rangle \langle \mathsf{F} \rangle \mid \langle \mathsf{F} \rangle \\ \langle \mathsf{F} \rangle &\rightarrow \langle \mathsf{F} \rangle \mathtt{*} \mid \mathtt{a} \mid \mathtt{b} \end{aligned}$$

Utilízala para analizar la cadena `ab+ab*`.