E79 Procesadores de lenguaje

Examen de teoría (11 de febrero de 2002)

Pregunta 1 (4 puntos)

Queremos hacer un traductor de `Apila` a `miniC`. Un programa en `Apila` consiste en una serie de declaraciones y sentencias mezcladas en cualquier orden. Las declaraciones constan de una lista de identificadores separados por comas y seguidos de un nombre de tipo. Los tipos posibles son `int` y `float` para enteros y flotantes. Tanto `int` como `float` son palabras reservadas. Los identificadores son secuencias de letras y dígitos que comienzan por una letra.

Hay dos tipos de sentencias: asignaciones y sentencias de escritura. Las primeras constan de una expresión seguida de la cadena `->` y un identificador. Las sentencias de escritura consisten en una lista de expresiones separadas por comas seguidas de la palabra reservada `print`. No se emplea ninguna marca para indicar fin de sentencia ni separación entre ellas. Además, los espacios en blanco no cumplen ninguna función, excepto la de separar componentes léxicos.

Las expresiones se escriben en notación polaca inversa. Así, en `Apila` la sentencia (en `C`) `i= i+1;` se escribe `i 1 + -> i`. Los operadores que se pueden emplear en `Apila` son:

Tipo	Lexemas
Aditivos	`+`, `-`.
Multiplicativos	`*`, `/`, `%`.
Lógicos	`&&`, `||`, `!`.
Desplazamiento	`<<`, `>>`

El significado de los operadores es el mismo que en `C`, con la excepción de los operadores `+` y `-`, que son siempre binarios.

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.

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.

El lenguaje `miniC` 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.

Un ejemplo de programa en `Apila` y su correspondiente traducción podría ser:

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

Se pide:

- Una gramática para el lenguaje `Apila`.
- Añadir a la gramática 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 instrucciones de escritura.
- Intenta ver cómo calcularías `0 1 - || 1` (el resultado debe ser `1`).
- Especifica claramente aquellas decisiones que tomes respecto a posibles ambigüedades en el enunciado.

Pregunta 2 (2 puntos)

Transforma la parte de la gramática de la pregunta anterior que modela las sentencias de escritura y las expresiones de modo que pueda ser analizada en tiempo lineal. Demuestra que tu nueva gramática permite el análisis en tiempo lineal.

Observaciones:

- Sólo se piden las sentencias de escritura y las expresiones, no incluyas ni declaraciones, ni asignaciones, ni listas de sentencias o similares.
- Es bastante difícil obtener una gramática RLL(1) (o LL(1)) para el lenguaje `Apila`. Se aconseja intentar obtener una SLR. Si, de todas formas, utilizas una RLL(1), explica claramente cualquier modificación que hagas respecto al método "estándar".

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 o bien no tienen ninguna `b` o bien tienen al menos una `a` y una `b`.
- Cadenas sobre el alfabeto `{a,b,c}` tales que tienen una única `b`, que delante sólo tiene `a`es y detrás sólo tiene `b`es de modo que por cada `a` hay dos `b`es.
- Números binarios mayores que `101101`.
- Números binarios de cinco bits palíndromos.

Pregunta 4 (2 puntos)

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

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

Demuestra que esta gramática es ambigua.