4º Ingeniería Informática
II26 Procesadores de lenguaje

Examen (11 de septiembre de 2009)

PREGUNTA 1 (5 PUNTOS)

Explica cómo debería modificarse la versión 10 de `minicomp` de modo que aceptara las siguientes extensiones, que son independientes entre sí.

Las descripciones de modificaciones del nivel léxico deben explicitar, si las hay, las nuevas expresiones regulares, los atributos asociados a posibles nuevas categorías y sus funciones de tratamiento; las de modificaciones del nivel sintáctico o semántico deben aclarar las reglas, acciones semánticas o métodos de comprobación afectados; las de modificaciones de la generación de código deben dejar claro qué instrucciones de máquina virtual se emitirían con la modificación. Por ejemplo: sería aceptable una frase como "se comprobaría que el hijo izquierdo es de tipo entero" (es trivial transformar la comprobación en un test), pero no una como "se generaría código para sumar los dos números" (no sabemos si se generaría una instrucción o varias, cuál o cuáles serían, dónde están los números —¿en registros, en memoria, en la pila?—, etc.).

Explicita cualquier asunción que hagas acerca del enunciado propuesto.

E1. Búsqueda en vectores (3 puntos)

Esta extensión permite calcular la posición de la primera aparición de un elemento *e* en un vector *v* utilizando la notación *v*`{`*e*`}`, donde *e* debe ser una expresión de tipo entero o real y *v* debe ser un vector cuyo tipo base debe coincidir con el de *e*. El resultado es de tipo entero. Si no se encuentra *e* en *v*, el resultado es -1. Por ejemplo, el siguiente código escribiría 20:

```
v: vector [4] de real;
...
v[0] := 1.1;
v[1] := 2.2;
v[2] := 4.4;
v[3] := 4.4;
escribe v{4 * 1.1} * 10;
```

Ten en cuenta que el vector puede ser una componente de un vector multidimensional. Por ejemplo:

```
m: vector [10] de vector [20] de vector[30] de entero;
...
escribe m[2][3]{4};
escribe m[5][6]{v{7.7} + 8} - 9;
```

Si en el acceso al vector o en el cálculo de la expresión se producen efectos secundarios, estos suceden sólo una vez. Es responsabilidad del programador que el vector haya sido inicializado previamente.

El número de instrucciones generadas debe ser independiente del tamaño del vector.

E2. Alias (2 puntos)

Mediante esta extensión, se pueden utilizar alias en `minicomp`. Por ejemplo:

```
#alias m1 matriz_1
```

define el alias `m1` que se sustituye por `matriz_1`. En general, una definición de alias comienza por la secuencia `#alias` seguida del nombre del alias y su expansión, y termina con un fin de línea. Todos estos elementos pueden estar separados por uno o más espacios, pero no por comentarios ni tabuladores. Así, su forma es la siguiente:

```
#alias alias expansión
```

Las definiciones de alias pueden aparecer en cualquier punto donde lo pueda hacer un comentario. Tanto ***alias*** como ***expansión*** siguen las reglas de los identificadores.

Si ***alias*** no está definido, se define en este punto; si ya lo estaba, se redefine. Desde la definición de ***alias*** hasta el final del programa, o hasta su posible redefinición, cada una de sus apariciones es sustituida, *a nivel léxico*, usando ***expansión***. Si ***expansión*** es otro alias, se repite el proceso. Es responsabilidad del programador que no haya definiciones circulares (por ejemplo, que `a` se defina como `b` y que a su vez `b` se defina como `a`) en este proceso de expansión y que ***alias*** no sea una palabra reservada.

Por ejemplo, el siguiente programa escribe los números del 1 al 10:

```
#alias begin secuencia
#alias end fin
#alias while mientras

globales
  numero: entero; #alias contador numero
fin
begin
  contador := 1;
  while contador <= 10 #alias do hacer
  do
    escribe numero, "  ";
    contador := contador + 1;
  end
end
```

Pregunta 2 (2 puntos)

Suponiendo que el alfabeto es el de los dígitos `0` a `9`:

1. Escribe una expresión regular para reconocer las cadenas de dos o más dígitos impares tales que el último dígito es mayor que todos los anteriores. Por ejemplo, serían válidas las cadenas `135`, `1317` o `357` pero no las cadenas `235` (2 es par), `735` (5 no es mayor que 7), `5` (sólo tiene un dígito) ni `11` (1 no es mayor que 1).

2. Dibuja un AFD para reconocer el sublenguaje formado por cadenas que cumplen las mismas restricciones del apartado anterior y en las que además no aparecen los dígitos `7` ni `9`.

Pregunta 3 (1,5 puntos)

Sea G una gramática LL(1) en la que todos los símbolos no terminales son no anulables. Supongamos que elegimos una producción cualquiera, $\langle \mathsf{A} \rangle \rightarrow \alpha$, y añadimos a la parte derecha una secuencia arbitraria de símbolos terminales y no terminales, β, obteniendo la regla $\langle \mathsf{A} \rangle \rightarrow \alpha\beta$. ¿Podemos asegurar que la gramática resultante sigue siendo LL(1)? Justifica tu respuesta.

Pregunta 4 (1,5 puntos)

En un determinado lenguaje se trabaja con órdenes que permiten imprimir páginas individuales y/o rangos de páginas, con la restricción de que no se aceptan páginas individuales después de la aparición del primer rango.

La siguiente gramática incontextual modela este tipo de órdenes:

$$\begin{aligned}
\langle \mathsf{Imprimir} \rangle &\rightarrow \textbf{imprime número } \langle \mathsf{MásPáginas} \rangle \\
\langle \mathsf{MásPáginas} \rangle &\rightarrow \textbf{número } \langle \mathsf{MásPáginas} \rangle | \textbf{rango número } \langle \mathsf{MásRangos} \rangle | \lambda \\
\langle \mathsf{MásRangos} \rangle &\rightarrow \textbf{número rango número } \langle \mathsf{MásRangos} \rangle | \lambda
\end{aligned}$$

Añade a la gramática las acciones semánticas necesarias para sintetizar, mientras se lleva a cabo un análisis RLL(1) de la entrada, el atributo *cp* de ⟨Imprimir⟩ con el resultado de contar la cantidad total de páginas a imprimir. Si en algún rango el número de la primera página es mayor que el de la última, ese rango aporta 0 páginas al total. Estos son algunos ejemplos:

Entrada	*cp*
`imprime 5`	1
`imprime 5 7 5`	3
`imprime 12..15`	4
`imprime 12..15 3..10 7..5`	12
`imprime 5 7 5 12..15 3..10`	15

Ten en cuenta que no puedes modificar la gramática, que no está permitido utilizar variables globales, que todos los atributos que utilices han de ser de tipo entero o lógico y que no puedes utilizar como heredados atributos sintetizados y viceversa. Indica para cada atributo si es heredado o sintetizado y qué representa.

Asume que los componentes de la categoría **número** son enteros positivos y que su valor está en el atributo *v*.