Ingeniería Informática
Procesadores de lenguaje
Examen (11 de diciembre 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. Sentencia iterativa para-cada-tal-que (3 puntos)

Mediante esta extensión se introduce una nueva sentencia iterativa que permite ejecutar una lista de sentencias para cada uno de los resultados de una lista de expresiones que cumplan una determinada condición. Por ejemplo, la sentencia:

```
para cada x en [3 * 4, 5 + 6, -7, 8 + 9] tal que x % 2 == 1 /\ x > 0 hacer
   escribe x;
   nl;
fin
```

escribiría los números impares positivos `11` y `17`, seguido cada uno de un fin de línea.

Más concretamente, esta nueva sentencia tiene la siguiente sintaxis:

```
para cada variable en [expresiones] tal que condición hacer
   sentencias
fin
```

donde:

- *`variable`* debe ser de tipo elemental y puede ser un identificador, posiblemente seguido de expresiones entre corchetes para acceder a una componente de un vector;
- *`expresiones`* contiene una o más expresiones, separadas por comas, del mismo tipo que la variable; y
- *`condición`* puede ser cualquier expresión de tipo lógico.

La ejecución de la sentencia requiere evaluar cada una de las expresiones en el orden en que aparecen de izquierda a derecha; tras evaluar cada expresión, se debe asignar el resultado a la variable, a continuación se debe evaluar la condición, y finalmente se deben ejecutar las sentencias si se cumple la condición (utilizando en ellas, si interviene, el valor de la variable asignado previamente, tal como ilustra el ejemplo).

En este ejercicio no debes preocuparte por el tamaño del código que se genera.

E2. Operador es-prefijo (2 puntos)

Esta extensión introduce el operador `|>` que permite averiguar si una cadena es prefijo de otra. Este operador es binario, infijo y asociativo por la izquierda. Su prioridad es superior a la del operador de negación e inferior a la de los operadores relacionales. Sus dos operandos deben ser expresiones de tipo cadena. El resultado es de tipo lógico y vale cierto si la primera cadena es prefijo de la segunda y falso si no. Si la primera cadena es vacía, el resultado siempre es cierto.

Por ejemplo, el siguiente programa escribiría `Conseguido`:

```
globales
   vc: vector[10] de vector[20] de vector[30] de cadena;
fin
funcion Ejemplo() : cadena es
secuencia
   devuelve "Ejemplo";
fin
secuencia
   vc[1][3][5] := "Eje";
   si vc[1][3][5] |> llama Ejemplo() /\ vc[1][3][5] |> "Eje" entonces
      escribe "Conseguido";
   fin
fin
```

Pregunta 2 (2 puntos)

Escribe tanto un AFD como una expresión regular para reconocer las cadenas de dígitos, posiblemente vacías, que **no contienen** la subcadena `1112` (formada por esos 4 caracteres seguidos, sin ninguna separación entre ellos). Puedes suponer que el alfabeto es el de los dígitos.

Por ejemplo, `11132112`, `1121` y `2111` pertenecen al lenguaje que debes modelar, mientras que `1112`, `1111122222` y `3211123` no pertenecen a él.

Pregunta 3 (1.5 puntos)

Para cada una de las dos gramáticas siguientes, escribe una gramática incontextual, sin partes derechas regulares, que sea LL(1) y que genere el mismo lenguaje:

(a) $\langle$CódigoMorse1$\rangle \rightarrow$ (**punto punto** | **punto punto punto**)$*$

(b) $\langle$CódigoMorse2$\rangle \rightarrow$ (**punto punto** | **punto punto punto**) $*$ (**punto raya**)?

Demuestra en cada caso que tu gramática es LL(1).

Pregunta 4 (1.5 puntos)

La siguiente gramática

$$
\begin{array}{lcl}
\langle \mathsf{A} \rangle & \rightarrow & \langle \mathsf{B} \rangle \\
\langle \mathsf{B} \rangle & \rightarrow & \langle \mathsf{C} \rangle \ (\ \textbf{op1}\ \langle \mathsf{B} \rangle\ )? \\
\langle \mathsf{C} \rangle & \rightarrow & \textbf{entero}\ \langle \mathsf{D} \rangle \\
\langle \mathsf{D} \rangle & \rightarrow & \textbf{op2}\ \langle \mathsf{C} \rangle \\
\langle \mathsf{D} \rangle & \rightarrow & \lambda
\end{array}
$$

modela expresiones construibles utilizando números y dos operadores, uno asociado al componente léxico **op1** y otro, más prioritario, asociado al componente léxico **op2**. Ambos operadores son binarios, infijos, asociativos por la izquierda y con resultado y operandos de tipo entero. Posiblemente tú escribirías una gramática diferente para modelar eso; no obstante, este ejercicio consiste en trabajar con esa versión.

Añade a la gramática las acciones semánticas necesarias para sintetizar, mientras se lleva a cabo un análisis LL(1) de la entrada, el atributo *resultado* de $\langle$A$\rangle$ con el resultado de evaluar la expresión.

Asume que los componentes de la categoría **entero** son enteros positivos cuyo valor está en el atributo *valor*, y que dispones, en el lenguaje de las acciones semánticas, de los mismos operadores con esta representación: $\oplus$ para **op1** y $\otimes$ para **op2**.

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.

Duración del examen: 4 horas