UNIVERSITAT JAUME·I

4º Ingeniería Informática

II26 Procesadores de lenguaje

Examen (28 de junio de 2010)

PREGUNTA 1 (5 PUNTOS)

Explica cómo debería modificarse la versión 9 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 y los atributos asociados a posibles nuevas categorías, **no utilices categorías sin nombre con metacomp**; las de modificaciones del nivel sintáctico o semántico deben aclarar las reglas, acciones semánticas y/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. Suma de listas con vectores (2,5 puntos)

Esta extensión introduce un nuevo operador que se aplica a una lista no vacía de expresiones y devuelve su suma. Utilizamos la sintaxis `+{`e_1`,` e_2`,...,` e_n`}` donde las e_i son expresiones de tipo real o vector unidimensional de reales. Para cada e_i, si es de tipo real, hay que sumar su valor y, si es de tipo vector de reales, hay que sumar todos sus componentes. No puede haber ni espacios ni comentarios entre el `+` y la `{`. Por ejemplo, la salida de

```
globales
  v: vector[3] de real;
  m: vector[3] de vector[2] de real;
fin
secuencia
  v[0] = 1.0; v[1] = 2.0; v[2]= 3.0;
  escribe +{v}, " ", +{v, 1.1}, " ", +{2.2, v, v}; nl;
  m[0][0] = 30.0; m[0][1] = 12.0;
  escribe +{m[0]}, " ", +{m[0], m[0][1]}, " ", +{m[0], 0.3*2*5, m[0][0]}; nl;
fin
```

es

```
6.0 7.1 14.2
42.0 54.0 75.0
```

Con esas mismas declaraciones, la expresión `+{m, 5.0}` es errónea ya que `m` no es unidimensional.

Al usar este operador, la cantidad de instrucciones del código generado debe ser independiente de la talla de los vectores. También se garantiza que los efectos secundarios de evaluar las expresiones suceden una sola vez y en el orden en que aparecen en la lista.

E2. Cadenas binarias (2,5 puntos)

Esta extensión permite utilizar la palabra reservada `es` como un nuevo operador que tiene una expresión de tipo cadena como operando izquierdo y una expresión de tipo entero como operando derecho. El resultado del operador es de tipo lógico y vale cierto si la cadena contiene la representación en binario del entero y falso en caso contrario; en particular, si la cadena es vacía o contiene algún carácter distinto de `0` (48 en código ASCII) y de `1` (49 en código ASCII), el resultado es falso.

El operador es binario e infijo. Es menos prioritario que los operadores relacionales y más que el operador de conjunción.

Por ejemplo, el siguiente programa escribiría `Conseguido`:

```
globales
   vc: vector[10] de vector[20] de vector[30] de cadena;
fin
funcion Trece() : cadena es
secuencia
   devuelve "001101"; // 13 en binario
fin
secuencia
   vc[1][3][5] = "101010"; // 42 en binario
   si vc[1][3][5] es 40 + 2  /\ llama Trece() es 26 / 2 entonces
      escribe "Conseguido";
   fin
fin
```

Pregunta 2 (2 puntos)

Los analizadores de lenguaje XML ignoran las denominadas secciones `CDATA`. Cada una de dichas secciones se abre exactamente con la secuencia de caracteres `<![CDATA[` y se cierra con la primera secuencia de tres caracteres `]]>` que siga a la secuencia de apertura.

Escribe tanto un AFD como una expresión regular para modelar dichas secciones.

Pregunta 3 (1,5 puntos)

Escribe una gramática incontextual, sin partes derechas regulares, que sea LL(1) y genere el mismo lenguaje que la siguiente:

$$\langle \text{Agenda} \rangle \quad \rightarrow \quad (\textbf{nombre teléfono})^* (\textbf{nombre email})^*$$

Demuestra que tu solución es LL(1).

Pregunta 4 (1,5 puntos)

Considera la siguiente gramática para expresiones formadas por sumas de productos de números:

$$\begin{aligned} \langle \text{Expresión} \rangle &\quad \rightarrow \quad \textbf{número}\ \langle \text{RestoExpresión} \rangle \\ \langle \text{RestoExpresión} \rangle &\quad \rightarrow \quad (\ \textbf{más} \mid \textbf{por}\ )\ \textbf{número}\ \langle \text{RestoExpresión} \rangle \mid \lambda \end{aligned}$$

Añade a la gramática las acciones semánticas necesarias para sintetizar, mientras se lleva a cabo un análisis RLL(1) de la entrada, el atributo v de ⟨Expresión⟩ con el resultado de evaluar la expresión, siguiendo las reglas habituales de prioridad. Por ejemplo, para `0*1*2+3*4+5*6` el valor de v es `42`. Asume que los componentes de la categoría **número** son enteros y que su valor está en el atributo v.

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 simple (entero, lógico o carácter) 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