4º Ingeniería Informática
II26 Procesadores de lenguaje

Examen (30 de junio 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 y los atributos asociados a posibles nuevas categorías; 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. Sumatorios y productorios de vectores (3 puntos)

Esta extensión introduce dos operadores que permiten calcular la suma o el producto de los elementos de parte de un vector de enteros. Para ello, se utiliza la sintaxis `+[v, i, j]`, para calcular $\sum_{k=i}^{j} v[k]$, y `*[v, i, j]`, para calcular $\prod_{k=i}^{j} v[k]$, donde *v* es un vector de enteros e *i* y *j* son expresiones de tipo entero. Ten en cuenta que el vector cuyas componentes se están sumando o multiplicando puede ser un componente de un vector mayor y que los nuevos operadores tienen como resultado un entero y pueden participar en otras expresiones, incluyendo otros sumatorios o productorios, como en:

```
m: vector [5] de vector [6] de entero;
v: vector [7] de entero;
...
escribe *[v, 2+1, 3*2] + *[v, 3, 5];
escribe +[m[3], *[m[2], 1, 2], llama f(5++[v, 0, 4])];
```

Si en el acceso al vector o en el cálculo de los límites se producen efectos secundarios, estos suceden sólo una vez. Así, en el ejemplo anterior, la función `f` es llamada una sola vez. Es responsabilidad del programador que el valor de *i* no sea superior al de *j* y que ambos estén dentro del vector.

E2. Sentencia de escritura de rangos (2 puntos)

Mediante esta extensión se permite al usuario escribir de manera cómoda una serie de expresiones para un rango de valores. Por ejemplo, la siguiente sentencia escribe la tabla del dos:

```
escribe "2*" y n y "=" y 2*n y " " para n:= 0..10;
```

Más formalmente, una sentencia de escritura de rangos se compone, en este orden, de: 1) la palabra reservada `escribe`; 2) una lista de una o más expresiones separadas por la nueva palabra reservada `y`; 3) la nueva palabra reservada `para`; 4) un acceso a una variable entera formado por un identificador, posiblemente seguido por una o más expresiones entre corchetes; 5) un símbolo de asignación; 6) dos expresiones enteras separadas por el nuevo componente léxico `..` (dos puntos seguidos) que definen los valores inicial y final; y 7) un punto y coma.

Cuando se ejecuta la sentencia, se le da a la variable todos los valores desde el inicial hasta el final, ambos incluidos. Para cada valor, se escriben las expresiones de la lista, calculadas, donde sea necesario, con el nuevo valor de la variable. En caso de que el valor inicial sea mayor que el final, no se escribe nada. No debes preocuparte si los posibles efectos secundarios de evaluar las expresiones o el acceso a la variable suceden más de una vez.

En esta extensión, no debes crear nuevos tipos de nodos ni modificar los existentes. Alternativamente, puedes renunciar a un 50 % de la calificación de la extensión y proponer una solución que no tenga en cuenta esta restricción.

Pregunta 2 (1,5 puntos)

Escribe un AFD para el lenguaje de las cadenas de dígitos que no empiezan ni terminan por 22. Por ejemplo, serían válidas las cadenas 2, 212, 314, 122223, pero no las cadenas 22, 223, 12222.

Asume que el alfabeto es el de los dígitos.

Pregunta 3 (1,5 puntos)

Para cada una de las siguientes tablas de análisis LL(1):

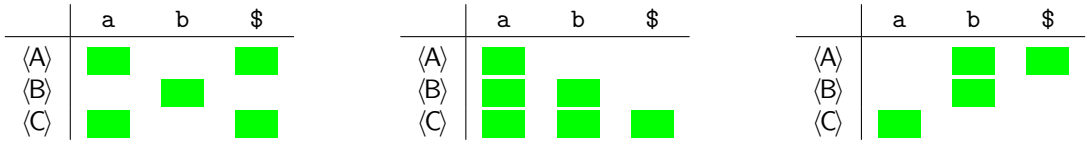

Tabla a) Tabla b) Tabla c)

escribe, o demuestra que no es posible, una gramática incontextual que sea LL(1), cuyo símbolo inicial sea ⟨A⟩, que no tenga símbolos inútiles, cuyos no terminales sean todos anulables y tal que las únicas celdas no vacías de su tabla de análisis sean las sombreadas.

Pregunta 4 (2 puntos)

Resuelve los dos apartados siguientes teniendo en cuenta que no puedes modificar las gramáticas, 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.

Apartado a)

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

$$\langle \text{Expresión} \rangle \quad \rightarrow \quad \textbf{núm} \; (\; \textbf{por núm} \;)^* (\; \textbf{más núm} \; (\; \textbf{por núm} \;)^* \;)^*$$

Añade acciones semánticas a la gramática anterior para sintetizar, mientras se lleva a cabo un análisis RLL(1) de la entrada, el atributo *valor* de ⟨Expresión⟩ teniendo en cuenta las prioridades habituales de la suma y el producto (por ejemplo, 3+4*5 tiene como resultado 23 y no 35). Asume que los componentes de la categoría **núm** son enteros positivos y que su valor está en el atributo *v*.

Apartado b)

Considera ahora la siguiente gramática, que modela secuencias formadas por una cantidad impar de expresiones separadas por dos puntos,

$$\begin{aligned} \langle \text{Sentencia} \rangle \quad &\rightarrow \quad \langle \text{Expresión} \rangle \langle \text{MásExpresiones} \rangle \\ \langle \text{MásExpresiones} \rangle \quad &\rightarrow \quad \textbf{dp} \; \langle \text{Expresión} \rangle \; \textbf{dp} \; \langle \text{Expresión} \rangle \langle \text{MásExpresiones} \rangle \, | \, \lambda \end{aligned}$$

donde el no terminal ⟨Expresión⟩ obtiene siempre un atributo sintetizado *valor* con el resultado de evaluar la expresión, según se ha visto en el apartado anterior.

Añade a la gramática las acciones semánticas necesarias para sintetizar, mientras se lleva a cabo un análisis RLL(1) de la entrada, el atributo *nésimo* de ⟨Sentencia⟩ con el resultado de evaluar la *n*-ésima expresión, contando desde 1 y siendo *n* el resultado de evaluar la primera. Si el valor de la primera expresión es mayor que el número de expresiones, debe devolverse el valor de la última.

Ten en cuenta los siguientes ejemplos:

Entrada	nésimo
3:3+4:5*8:3*2:4+5	40
12	12
5:3+4:1+2*5	11

Duración del examen: 4 horas