Ingeniería Informática
Procesadores de lenguaje
Examen (14 de septiembre de 2007)

PREGUNTA 1 (5 PUNTOS)

Explica cómo debería modificarse `minicomp` (en su versión para Stan o para Rossi, pero sin ninguna de las extensiones planteadas en las prácticas) de modo que aceptara las extensiones que se presentan a continuación. Las extensiones son independientes entre sí; no hace falta que consideres sus posibles interacciones.

Procura que tu descripción sea clara, escueta y precisa. Para ello, puede facilitarte la exposición una estructura que siga las distintas etapas del compilador.

Puedes optar por descripciones algorítmicas o en lenguaje natural para lograr una mayor sencillez en la explicación, pero dejando claro qué se modificaría y cómo. Como mínimo: 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. Recorrido de vectores (3 puntos)

Esta modificación introduce una nueva estructura de control que permite recorrer vectores fácilmente. Para ello, se emplea la sintaxis

```
para variable en vector[exp1:exp2] hacer
  sentencias
fin
```

donde *variable* y *vector* son identificadores, posiblemente seguidos de expresiones entre corchetes; *vector* tiene tipo vectorial con tipo base igual al tipo de *variable*, que debe ser elemental. Tanto *exp1* como *exp2* son expresiones de tipo entero.

La ejecución de la estructura supone la ejecución de las *sentencias* una vez por cada uno de los elementos de *vector* con índices desde *exp1* hasta *exp2*, ambos inclusive, y con el valor correspondiente almacenado en *variable*. Por ejemplo, suponiendo la declaración `v: vector [50] de entero;`, podemos escribir los valores de `v[13]` a `v[40]` mediante el fragmento

```
para x en v[8+5:8*5] hacer
  escribe x; nl;
fin
```

También podemos escribir la segunda fila de `matriz: vector [5] de vector [5] de entero;` mediante

```
para m[0][0] en m[1][0:4] hacer
  escribe m[0][0]; nl;
fin
```

Notas:

- Los posibles efectos secundarios de evaluar *exp1*, *exp2*, *variable* y *vector* se producen sólo una vez.
- Está indefinido el valor que tendrá *variable* al salir del bucle.
- Es responsabilidad del programador que *exp1* sea menor o igual que *exp2* y que ambas estén dentro de los límites del vector.

E2. Autoasignaciones sobre vectores (2 puntos)

Mediante esta extensión, se permite la modificación cómoda de todos los elementos de un vector. Para ello, se define una nueva sentencia:

```
vector op exp ;
```

donde: ***vector*** es un identificador, posiblemente seguido de expresiones entre corchetes, de tipo vector de enteros; `op` es un componente léxico con lexema `+=`, `-=`, `*=` o `/=`; y ***exp*** es una expresión de tipo entero.

La ejecución de la sentencia hace que a cada elemento del vector se le asigne un nuevo valor calculado aplicando ***op*** a su valor anterior y al resultado de ***exp***. Por ejemplo, para multiplicar los elementos de la segunda fila de la variable `matriz: vector [5] de vector [5] de entero;` por el resultado de `f(3)`, podemos hacer

```
matriz[1]*= llama f(3);
```

Notas:

- Los posibles efectos secundarios de evaluar ***vector*** y ***exp*** se producen sólo una vez.
- El código generado (medido en instrucciones) es independiente del tamaño del vector.

Pregunta 2 (1,5 puntos)

Sea r_n la expresión regular $\texttt{b}^*(\texttt{ab})^*(\texttt{aab})^*\ldots(\overbrace{\texttt{a}\ldots\texttt{a}}^{n}\texttt{b})^*$. Escribe el autómata finito determinista obtenido para r_3 mediante el algoritmo de construcción usando ítems. Para un n cualquiera mayor que cero, ¿cuántos estados tiene el autómata finito determinista obtenido para r_n mediante el algoritmo de construcción usando ítems?

Pregunta 3 (1,5 puntos)

Sea G la siguiente gramática, que genera listas de identificadores separados por comas:

$$\begin{array}{lcl}\langle \mathsf{S}\rangle & \rightarrow & \langle \mathsf{A}\rangle \\ \langle \mathsf{A}\rangle & \rightarrow & \textbf{letra coma letra } \langle \mathsf{A}\rangle \,|\, \textbf{letra } \langle \mathsf{B}\rangle \textbf{ letra} \,|\, \langle \mathsf{B}\rangle \textbf{ coma } \langle \mathsf{B}\rangle \\ \langle \mathsf{B}\rangle & \rightarrow & \textbf{letra } \langle \mathsf{B}\rangle \textbf{ coma } \langle \mathsf{A}\rangle \textbf{ letra} \,|\, \langle \mathsf{B}\rangle \textbf{ coma } \langle \mathsf{A}\rangle \,|\, \textbf{letra}\end{array}$$

donde **letra** es un terminal que representa una letra con valor **letra**.v y **coma** representa una coma. Supongamos que la regla inicial tiene la acción semántica siguiente:

$$\langle \mathsf{S}\rangle \rightarrow \langle \mathsf{A}\rangle \ \{\textbf{escribe}(\text{“Hay ”}, \langle \mathsf{A}\rangle.\text{nm}, \text{“ identificadores que comienzan por mayúscula”});\}$$

que escribe el número de identificadores de la lista que comienzan por mayúscula.

Añade, *al final de cada una de las restantes reglas*, las acciones semánticas necesarias para que se calcule el valor del atributo nm de $\langle \mathsf{A}\rangle$. Puedes utilizar los atributos adicionales que consideres necesarios, pero ninguna variable global. Además, los atributos que añadas deben ser de tipo entero o lógico.

Nota: puede serte útil la función *may* que devuelve cierto si el carácter que se le pasa es una letra mayúscula.

PREGUNTA 4 (2 PUNTOS)

Sean $G = (N, \Sigma, P, \langle \mathsf{S} \rangle)$ una gramática y δ una derivación $\langle \mathsf{S} \rangle \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$. Llamaremos *firma izquierda* de δ a la secuencia $X_1 X_2 \ldots X_n$ formada por los primeros símbolos (terminales o no terminales) de cada una de las cadenas $\alpha_1, \alpha_2, \ldots, \alpha_n$. Análogamente, llamaremos *firma derecha* de δ a la secuencia formada por los últimos símbolos de $\alpha_1, \alpha_2, \ldots, \alpha_n$. Por ejemplo, para la gramática

$$\begin{array}{lcl} \langle \mathsf{A} \rangle & \rightarrow & \langle \mathsf{B} \rangle \langle \mathsf{C} \rangle \,|\, \langle \mathsf{B} \rangle \\ \langle \mathsf{B} \rangle & \rightarrow & \mathbf{a} \mid \mathbf{b} \\ \langle \mathsf{C} \rangle & \rightarrow & \mathbf{c}\ \langle \mathsf{B} \rangle \end{array}$$

y la derivación $\langle \mathsf{A} \rangle \Rightarrow \langle \mathsf{B} \rangle \langle \mathsf{C} \rangle \Rightarrow \mathbf{a}\ \langle \mathsf{C} \rangle \Rightarrow \mathbf{a\ c}\ \langle \mathsf{B} \rangle \Rightarrow \mathbf{a\ c\ b}$, la firma izquierda es $\langle \mathsf{B} \rangle\ \mathbf{a\ a\ a}$ y la firma derecha es $\langle \mathsf{C} \rangle \langle \mathsf{C} \rangle \langle \mathsf{B} \rangle\ \mathbf{b}$.

Para cada una de las afirmaciones siguientes, da un contraejemplo si es falsa o justifica su veracidad:

1. Si G es LL(1), entonces existe un número n tal que la firma izquierda de cualquier derivación canónica por la izquierda tiene menos de n no terminales.
2. Si G es SLR, entonces existe un número n tal que la firma izquierda de cualquier derivación canónica por la izquierda tiene menos de n no terminales.
3. Si G es LL(1), entonces existe un número n tal que la firma derecha de cualquier derivación canónica por la derecha tiene menos de n no terminales.
4. Si G es SLR, entonces existe un número n tal que la firma derecha de cualquier derivación canónica por la derecha tiene menos de n no terminales.

Duración del examen: 4 horas