UNIVERSITAT JAUME·I

Ingeniería Informática

Procesadores de lenguaje

Examen (26 de junio 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. Rotación de vectores (3 puntos)

El objetivo de esta extensión es facilitar la rotación de vectores. Para ello, se introduce una nueva sentencia que consiste en un identificador de variable o un acceso a vector seguido del símbolo `<<`, para una rotación hacia la izquierda, o `>>`, para una rotación hacia la derecha, y un punto y coma. Una rotación hacia la izquierda consiste en asignar a la primera posición del vector el valor de la segunda, a la segunda el de la tercera y así sucesivamente hasta que a la última posición se le asigna el valor que inicialmente tenía la primera posición. Una rotación hacia la derecha es análoga salvo que la asignación es del valor de la penúltima posición a la última, de la antepenúltima a la penúltima, etc. La variable o el acceso a vector deben corresponder a un vector con tipo base elemental.

Por ejemplo, dada la declaración

```
v: vector [3] de entero;
```

y suponiendo que inicialmente `v` tiene los valores `1,2,3`, la sentencia

```
v<<;
```

dejaría `v` con `2,3,1`. Análogamente, para la declaración

```
m: vector [10] de vector [3] de entero;
```

si `m[3]` tiene los valores `2,4,6`, la sentencia

```
m[15/5]>>;
```

haría que los valores en `m[3]` fueran `6,2,4`.

Además, con las declaraciones anteriores, tanto `v[1]<<;` como `m>>;` serían erróneas por no referirse a vectores de tipos elementales.

Se garantiza que la longitud del código generado (medido en instrucciones) es independiente del tamaño del vector.

Nota: Al describir la *generación de código*, basta con que elijas una de las dos direcciones (`<<` o `>>`).

E2. Intercambios múltiples (2 puntos)

Esta extensión permite realizar de forma cómoda intercambios entre variables de tipos elementales. Para ello, se introduce una nueva sentencia que consta de: una lista de identificadores de variable o accesos a vector separados por comas; el nuevo símbolo `<->`; una segunda lista de identificadores o accesos a vector separados por comas; un punto y coma. Ambas listas deben tener la misma longitud. El efecto de la sentencia es el intercambio de los valores de las variables situadas en la misma posición en ambas listas, que deberán tener el mismo tipo y ser de tipos elementales.

Por ejemplo, dadas las declaraciones

```
a,b: entero;
v: vector [2] de cadena;
```

la ejecución de

```
a,v[0] <-> b,v[99-98];
```

provocaría que `a` y `b` intercambiaran sus valores, así como `v[0]` y `v[1]`.

Está indefinido el comportamiento de la sentencia si aparece más de una referencia a una misma variable simple o elemento de un vector.

Pregunta 2 (1,5 puntos)

Sea r_n la expresión regular $(\overbrace{\texttt{ab}\ldots\texttt{b}}^{n}|\texttt{a}\overbrace{\texttt{ab}\ldots\texttt{b}}^{n-1}|\ldots|\overbrace{\texttt{a}\ldots\texttt{a}}^{n-1}\texttt{bb}|\overbrace{\texttt{a}\ldots\texttt{a}}^{n}\texttt{b})$. 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 enteros separados por comas:

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

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

$$\langle \mathsf{S}\rangle \rightarrow \langle \mathsf{A}\rangle \ \{\textbf{escribe}(\text{“El máximo es ”}, \langle \mathsf{A}\rangle.\text{máx});\}$$

con el objetivo de escribir cuál es el mayor entero de la lista. Por ejemplo, para la lista `3,45,8` el máximo es 45.

Añade, *al final de cada una de las restantes reglas*, las acciones semánticas necesarias para que se calcule el valor del atributo máx 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: puedes serte útil la función *nd* que te da el número de dígitos de un número entero.

Pregunta 4 (2 puntos)

Sean $G = (N, \Sigma, P, \langle \mathsf{S}\rangle)$ una gramática. Definimos la gramática con permutación de los no terminales $\langle \mathsf{A}\rangle$ y $\langle \mathsf{B}\rangle$ como la gramática $G_{\langle \mathsf{A}\rangle|\langle \mathsf{B}\rangle} = (N, \Sigma, P_{\langle \mathsf{A}\rangle|\langle \mathsf{B}\rangle}, \langle \mathsf{S}\rangle)$ donde $P_{\langle \mathsf{A}\rangle|\langle \mathsf{B}\rangle}$ tiene las mismas reglas que P pero sustituyendo en la parte derecha cada $\langle \mathsf{A}\rangle$ por una $\langle \mathsf{B}\rangle$ y viceversa. Por ejemplo, la regla $\langle \mathsf{A}\rangle \rightarrow \mathbf{a}\langle \mathsf{A}\rangle\mathbf{b}\langle \mathsf{B}\rangle$ se cambiaría por $\langle \mathsf{A}\rangle \rightarrow \mathbf{a}\langle \mathsf{B}\rangle\mathbf{b}\langle \mathsf{A}\rangle$.

Por otro lado, definimos la gramática con reduplicación del terminal **a** como la gramática $G_{\mathbf{a}^2} = (N, \Sigma, P_{\mathbf{a}^2}, \langle \mathsf{S}\rangle)$ donde $P_{\mathbf{a}^2}$ tiene las mismas reglas que P pero sustituyendo en la parte derecha cada **a** por **aa**. Por ejemplo, la regla $\langle \mathsf{A}\rangle \rightarrow \mathbf{a}\langle \mathsf{A}\rangle\mathbf{b}\langle \mathsf{B}\rangle$ se cambiaría por $\langle \mathsf{A}\rangle \rightarrow \mathbf{aa}\langle \mathsf{A}\rangle\mathbf{b}\langle \mathsf{B}\rangle$.

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

- Si G es LL(1), entonces $G_{\langle \mathsf{A}\rangle|\langle \mathsf{B}\rangle}$ es LL(1).
- Si G es SLR, entonces $G_{\langle \mathsf{A}\rangle|\langle \mathsf{B}\rangle}$ es SLR.
- Si G es LL(1), entonces $G_{\mathbf{a}^2}$ es LL(1).
- Si G es SLR, entonces $G_{\mathbf{a}^2}$ es SLR.

Duración del examen: 4 horas