UNIVERSITAT JAUME·I

Ingeniería Informática
Procesadores de lenguaje
Examen de teoría (20 de junio de 2005)

PREGUNTA 1 (5 PUNTOS)

A continuación, se presentan dos posibles extensiones para los lenguajes *Sé* y *Sé-*. Explica claramente qué modificaciones se tendrían que hacer en un compilador de estos lenguajes a Stan o Rossi para que las aceptara. Las extensiones son independientes entre sí; no hace falta que consideres sus posibles interacciones.

En tu descripción, procura ser claro, escueto y preciso. En particular, no es necesario que describas partes del compilador que no estén afectadas por las modificaciones. Puedes optar por descripciones algorítmicas o en lenguaje natural para lograr una mayor sencillez en la explicación. También puede facilitarte la exposición una estructura que siga las distintas etapas del compilador.

Explicita cualquier asunción que hagas acerca del compilador o del enunciado propuesto.

Operando vacío en el condicional (2 puntos)

Con esta extensión, se permite que el segundo operando de un operador condicional esté vacío. Es decir, la sentencia siguiente sería válida:

```
e= c ? : r;
```

La evaluación se hace de manera similar al operador ternario completo: se evalúa la condición y, si esta es cierta, se devuelve el valor de la condición, en caso contrario, se evalúa el tercer operando y se devuelve su valor. En general, tenemos que e_1`? :`e_2 es equivalente a e_1`?`e_1`:`e_2 salvo por el hecho de que e_1 se evalúa exactamente una vez.

Además de las modificaciones, muestra qué código se generaría para tu máquina virtual (Stan o Rossi) con el ejemplo anterior suponiendo que `e`, `c` y `r` son variables globales de tipo entero, de tipo carácter y de tipo real, respectivamente. Asígnales las direcciones que creas conveniente.

Aplicación de funciones a vectores (3 puntos)

Mediante esta modificación, se puede aplicar fácilmente una función a todos los elementos de un vector. Para ello, se emplea la sintaxis `f[`v`]` donde `f` es un identificador de función y v una expresión de tipo vectorial. La función `f` debe aceptar un parámetro de tipo elemental y devolver un resultado de tipo elemental (no puede ser de tipo `void`). La evaluación consiste en llamar a `f` con cada uno de los elementos del vector almacenando en el mismo el resultado de la llamada. Como resultado final se devuelve el valor almacenado en el último elemento del vector.

Por ejemplo, sea la línea

```
x= 2*doble[v]+1;
```

donde `doble` es una función que recibe un entero y devuelve un entero, `v` un vector de reales de talla 3 y `x` una variable entera. La ejecución de esa línea sería equivalente a

```
x= 2*(v[0]= doble(v[0]), v[1]= doble(v[1]), v[2]= doble(v[2]))+1;
```

si en *Sé* o *Sé-* tuviéramos el operador coma de C (que evalúa sus operandos izquierdo y derecho y devuelve el valor del derecho). La equivalencia incluye también las promociones necesarias para la llamada a `f` o el almacenamiento de los resultados en el vector.

Además de las modificaciones, muestra qué código se generaría para tu máquina virtual (Stan o Rossi) en el ejemplo anterior suponiendo que `x` y `v` son variables globales. Asígnales las direcciones que creas conveniente.

Notas:

- Si te resulta más sencillo, puedes permitir que el identificador de función aparezca entre paréntesis. Indica claramente si has optado o no por esta posibilidad.
- Debes generar código de modo que el tamaño del programa generado no crezca linealmente con el tamaño de los vectores sobre los que se aplica esta extensión (es decir, el código generado para un vector de tamaño cien no puede contener el equivalente a cien asignaciones).

Pregunta 2 (2 puntos)

Queremos incluir en nuestro lenguaje de expresiones regulares el operador de repetición. Este operador es postfijo y se escribe como un exponente que indica el número de repeticiones de la expresión a la que afecta. Así $L_{r^n} = \overbrace{L_r \cdots L_r}^{n \text{ veces}}$.

Escribe, utilizando el operador de repetición, una expresión regular para secuencias de entre uno y cien enteros separados por comas.

Explica cómo deberíamos modificar el algoritmo de construcción de autómatas a partir de expresiones regulares, su tabla o ambos para incluir este operador *sin transformar la expresión en su equivalente mediante concatenaciones*. Aplica tu versión modificada a la expresión regular $((\texttt{ab})^2\texttt{a})^*$.

Pregunta 3 (1,5 puntos)

Sea G la siguiente gramática:

$$\begin{array}{lcl} \langle \mathsf{S} \rangle & \rightarrow & \langle \mathsf{A} \rangle \\ \langle \mathsf{A} \rangle & \rightarrow & \langle \mathsf{A} \rangle \langle \mathsf{B} \rangle \langle \mathsf{A} \rangle \mid \mathbf{a}\ \langle \mathsf{A} \rangle \mid \mathbf{a}\ \mathbf{b} \\ \langle \mathsf{B} \rangle & \rightarrow & \langle \mathsf{B} \rangle\ \mathbf{a} \mid \langle \mathsf{B} \rangle\ \mathbf{b} \mid \lambda \end{array}$$

Supongamos que la primera regla tiene asociada la acción

$$\langle \mathsf{S} \rangle \rightarrow \langle \mathsf{A} \rangle\ \{\textbf{escribe}(\text{“La subcadena de aes más larga tiene ”}, \langle \mathsf{A} \rangle.\text{ma}, \text{“ aes.”});\}$$

donde *ma* tiene el número de aes contenido en la subcadena más larga formada únicamente por aes. Por ejemplo, para la cadena `abaaaab`, el valor de *ma* será 4. Añade, *al final de cada una de las restantes reglas*, las acciones semánticas necesarias para que se calcule el valor de este atributo.

Puedes utilizar los atributos adicionales que consideres necesarios, pero ninguna variable global. Además, los atributos que añadas deben ser de tipo entero o lógico.

Pregunta 4 (1,5 puntos)

Sea G una gramática SLR y P el conjunto de todos los caminos en su autómata de prefijos viables que parten del estado inicial y llegan hasta un estado que tiene asociada la acción de reducir con, al menos, un símbolo de entrada.

Demuestra la verdad o falsedad de las siguientes afirmaciones:

- Para todo $p \in P$, existe un prefijo de p que es parte derecha de alguna regla.
- Para todo $p \in P$, existe un sufijo de p que es parte derecha de alguna regla.
- Si existe un número n tal que todos los $p \in P$ tienen longitud inferior a n, entonces el lenguaje generado por G es finito.

Duración del examen: 4 horas
¡Buena suerte!