UNIVERSITAT JAUME·I

5º Ingeniería Informática

E79 Procesadores de lenguaje

Examen de teoría (16 de diciembre 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 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.

Macros elementales (2 puntos)

Con esta extensión, se dota al lenguaje de un sencillo procesador de macros.

Las definiciones de macros pueden aparecer después de las directivas `#include` (si las hay) y antes de cualquier definición de otro tipo. La definición de una macro tiene la forma

`#define` *macro sustitución*

donde *macro* es el nombre de la macro, que sigue las reglas de los identificadores y *sustitución* es un componente léxico que no se omita. Tanto entre `#define` y *macro* como entre *macro* y *sustitución* hay exactamente un espacio. Tras *sustitución* y hasta el final de línea puede haber cuantos espacios se desee, pero ningún comentario.

Una vez definida una macro, cualquier aparición del identificador es sustituida por el correspondiente componente de sustitución. Si ese componente corresponde a otra macro, se repite el proceso. Está indefinido el comportamiento del compilador ante secuencias de definiciones circulares (p.ej, que la macro `a` se sustituya por la `b`, que a su vez se sustituya por `a`).

Por ejemplo, el programa

```
#include <stdio.h>
#define por *
#define abre {
#define cierra }
#define begin abre
#define end cierra
#define escribe printf

int main (void) begin
  escribe("%d\n", 3 por 4);
end
```

escribiría `12` por la salida estándar.

Nota: para simplificar la escritura de los niveles léxico y/o sintáctico, considera que sólo pueden participar en las sustituciones los siguientes componentes: identificadores, operadores aritméticos y delimitadores (paréntesis y llaves).

Cálculo de logaritmos (3 puntos)

Esta extensión permite utilizar la cabecera `math.h`. Al incluirla en un programa, se tiene acceso a la función `log10`. Esta función recibe como parámetro un real (se aplican la correspondientes promociones si es necesario) y devuelve el real que resulta de calcular la parte entera del logaritmo en base 10 del argumento. Por ejemplo, la sentencia

```
printf("%f\n", log10(2005));
```

escribiría `3.0` por la salida estándar.

No está definido el comportamiento de `log10` ante argumentos que no sean estrictamente positivos.

Además de las modificaciones, muestra qué código se generaría para la máquina virtual Stan con el ejemplo anterior.

Pregunta 2 (2 puntos)

En un lenguaje de programación se escriben los comentarios comenzando por el carácter almohadilla (`#`) y terminando con otro carácter almohadilla. Entre ambas almohadillas se admite cualquier número de caracteres que no sean almohadillas y otras almohadillas si están inmediatamente precedidas de, al menos, una almohadilla y no tocan a la que cierra el comentario. Por ejemplo, tanto "`# incremento i #`" como "`### i##j #`" son comentarios válidos. No lo son, sin embargo "`#i####`" o "`#i#j#`".

Dadas las siguientes expresiones regulares, indica para cada una de ellas si corresponde o no a los comentarios de este lenguaje. En caso negativo, proporciona un contraejemplo, en caso positivo, demuéstralo construyendo el AFD correspondiente a la expresión.

- `#`$^+$`([^#]|##)`*`#`
- `#([^#]|##`$^+$`[^#])`*`#`
- `#([#*^#])`*`#`
- `#`$^+$`[^#]`$^+$`(##`$^+$`[^#]`$^+$`)`*`#`

Pregunta 3 (1,5 puntos)

Sea G la siguiente gramática:

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

Supongamos que la primera regla tiene asociada la acción

$$\langle \mathsf{S} \rangle \rightarrow \langle \mathsf{A} \rangle\ \{\textbf{si}\ \langle \mathsf{A} \rangle.\text{alt}\ \textbf{entonces escribe}\ (\text{“Es alternante”})\ \textbf{si no escribe}\ (\text{“No es alternante”})\}$$

donde *alt* es cierto si la cadena generada no tiene dos símbolos consecutivos iguales. Por ejemplo, si la cadena generada fuera `abab`, el atributo sería cierto mientras que si fuera `aab`, el atributo sería falso. 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)

Sean G_1 y G_2 dos gramáticas. Decimos que G_1 es *LL-menor* que G_2, simbólicamente $G_1 \preceq_{LL} G_2$, si al calcular las tablas de análisis LL(1) de ambas se cumple que cualquier entrada no vacía en la tabla de G_1 es igual que la correspondiente entrada en la tabla de G_2. Por ejemplo, sean las gramáticas:

A:

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

B:

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

Se cumple que $A \preceq_{LL} B$.

Demuestra la verdad o falsedad de las siguientes afirmaciones:

- Si $G_1 \preceq_{LL} G_2$ y G_1 es LL(1), entonces G_2 es LL(1).
- Si $G_1 \preceq_{LL} G_2$ y G_2 es LL(1), entonces G_1 es LL(1).
- Si $G_1 \preceq_{LL} G_2$, entonces $L(G_1) \subseteq L(G_2)$.

Duración del examen: 4 horas

¡Buena suerte!