5º Ingeniería Informática

E79 Procesadores de lenguaje

Examen de teoría (1 de septiembre de 2004)

PREGUNTA 1 (5 PUNTOS)

A continuación, se presentan dos posibles extensiones del lenguaje *r-LaST*. Explica claramente qué modificaciones se tendrían que hacer en un compilador de *r-LaST* a Stan para que las aceptara. Las modificaciones son independientes entre sí; no hace falta que consideres sus posibles interacciones.

En tu descripción de las modificaciones, 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.

Escritura de sufijos (2 puntos)

Esta modificación introduce la nueva palabra clave `SPRINT` (por *suffix print*) y una nueva construcción para escribir un sufijo de una cadena. Esta construcción tiene dos parámetros: el primero es una expresión de tipo cadena y el segundo una expresión de tipo entero. Sean c el resultado de evaluar la cadena e i el de evaluar la expresión entera. El resultado de la evaluación de `SPRINT` es c y, como efecto secundario de esa evaluación, se escribe la subcadena $c_i \dots c_{\|c\|}$.

Así, el fragmento siguiente:

```
s:= SPRINT("James Bond\n", 7);
PRINT(s)
```

Escribiría por pantalla:

```
Bond
James Bond
```

Siendo `s` una variable de tipo cadena.

Nota: no está definido qué sucede si la expresión no corresponde a una posición de la cadena.

Operadores definidos por el usuario (3 puntos)

Mediante esta extensión, el programador puede hacer que determinadas funciones binarias se utilicen como operadores. Para ello, se introduce la nueva palabra reservada `OPERATOR` y una serie de nuevos operadores. Estos operadores constan de un paréntesis abierto, uno o más operadores aritméticos iguales entre sí y un paréntesis cerrado. Por ejemplo: `(++)`, `(---)`, `(*)`, `(/)` y `(%%)` están correctamente escritos. Sin embargo, `()` y `(+-)` son erróneos. El primero por no tener operador aritmético y el segundo por tener operadores distintos.

Los nuevos operadores se declaran en la zona de constantes globales mediante la sintaxis:

`OPERATOR` *operador* `:` *función anónima*

A la función anónima se le exige que tenga exactamente dos parámetros, ambos de tipo numérico (real o entero) y que el tipo devuelto sea también de tipo numérico.

Estos nuevos operadores son binarios, infijos, asociativos por la izquierda y crean dos nuevos niveles de precedencia entre el tres (más unario, cambio de signo y negación) y el cuatro (multiplicación, división, etc.). Los menos prioritarios de los nuevos operadores son los formados por repeticiones de los signos más y menos. Los más prioritarios son los restantes.

La utilización de los operadores definidos por el usuario se traduce en el equivalente a una llamada a la correspondiente función, en particular, se siguen las normas habituales respecto a los tipos.

Por ejemplo, dadas las declaraciones

```
const
  operator (+): func (a: integer, b: integer)-> real is
    real(a+b)
  end,
  operator (**): func (a: real, n: integer)-> real is
    if n<0 then 1.0/self(a, -n)
    else if n=0 then 1.0
         else a*self(a, n-1)
         end
    end
  end
...
```

tendríamos

```
print ( 2(+)3, << 5.0 >>
        (2(+)3)(**)-1, << 0.2 >>
        2(+)3(**)-1, << Error de tipos en (+) >>
        2(+)3(+)4, << Error de tipos en el segundo (+) >>
        2.0(**)2(**)3 << 64.0 >> )
```

Pregunta 2 (1,5 puntos)

Sean r y s dos expresiones regulares. Definimos el operador *lista* ($\uparrow$) de modo que la expresión regular $(r\uparrow s)$ representa el lenguaje $L_{(r\uparrow s)} = L_r(L_sL_r)^*$. Este operador nos permite especificar cómodamente listas de elementos con un separador. Por ejemplo, las listas de enteros separados por barras las podemos especificar como $[0\text{–}9]^+\uparrow\backslash|$.

Explica cómo se debe modificar el algoritmo de cálculo de construcción de autómatas a partir de expresiones regulares para tener en cuenta este nuevo operador. Para ello, basta con que expliques cómo modificar el cuadro de los apuntes que explica las transformaciones que se deben aplicar a los ítems no básicos para calcular la clausura de los estados (este es el cuadro 1 de la página 13 de los apuntes).

Utiliza el algoritmo para construir el autómata correspondiente a la expresión regular $[0\text{–}9]^+\uparrow\backslash|$.

Pregunta 3 (2 puntos)

Diremos que un carácter es un *extremo derecho* si es el último de la cadena o es distinto del carácter siguiente. Por ejemplo, en la cadena `aabcccaa`, hay cuatro extremos derechos: la segunda `a`, la `b`, la tercera `c` y la última `a`.

Dada la gramática G con las reglas

$$\begin{aligned}
\langle \mathsf{A}\rangle &\rightarrow \langle \mathsf{B}\rangle\langle \mathsf{C}\rangle|\mathsf{a}\\
\langle \mathsf{B}\rangle &\rightarrow \langle \mathsf{B}\rangle\mathsf{a}\langle \mathsf{B}\rangle|\mathsf{b}|\langle \mathsf{C}\rangle\\
\langle \mathsf{C}\rangle &\rightarrow \mathsf{a}\langle \mathsf{C}\rangle\langle \mathsf{B}\rangle|\mathsf{b}
\end{aligned}$$

añade a G las reglas semánticas necesarias para que el atributo sintetizado *nae* de $\langle \mathsf{A}\rangle$ contenga el número de aes que son extremos derechos. Por ejemplo, para las cadenas `babb` y `aabbbabb` los valores de *nae* serían 1 y 2, respectivamente.

Puedes utilizar los atributos *sintetizados* 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)

Responde, justificadamente, a las siguientes preguntas:

- ¿Puede ser SLR una gramática que tenga las reglas $\langle \mathsf{A}\rangle \rightarrow \mathsf{a}\langle \mathsf{B}\rangle$ y $\langle \mathsf{A}\rangle \rightarrow \mathsf{a}$?
- Sea G una gramática que tiene las reglas $\langle \mathsf{A}\rangle \rightarrow \langle \mathsf{B}\rangle\mathsf{a}$ y $\langle \mathsf{A}\rangle \rightarrow \mathsf{a}$:
  - ¿Qué podemos decir acerca de $\langle \mathsf{B}\rangle$ si sabemos que G es LL(1)?
  - Si $\langle \mathsf{B}\rangle$ es anulable, ¿puede G ser SLR?, ¿y LR(1)?

Duración del examen: 4 horas
¡Buena suerte!