4º Ingeniería Informática

II26 Procesadores de lenguaje

Control de teoría, grupo TE-1, 29 de mayo de 2008 (solución)

Introducción

En determinado lenguaje, se puede buscar el máximo de una serie de variables mediante lo que llamaremos una expresión máximo. Ésta consiste en una lista de uno o más identificadores separados por comas y encerrados entre `<+` y `+>`. Por ejemplo, el máximo de las variables `a1`, `b2` y `c3` se representaría con la expresión `<+ a1, b2, c3 +>`. Opcionalmente, si la lista tiene más de un elemento, el último (y sólo el último) elemento de la lista puede ser un literal entero, que también entra en el cálculo del máximo.

A continuación se presentan cuatro cuestiones relacionadas con la expresión máximo. Son independientes entre sí, no debes preocuparte por sus posibles interacciones.

Pregunta 1 (0,1 puntos)

Escribe una expresión regular que represente la expresión máximo descrita en la introducción. Asume

- que los identificadores son secuencias de dígitos y letras minúsculas que comienzan por letra,
- que los literales enteros son secuencias de dígitos y
- que no hay blancos.

Solución

Podemos utilizar la siguiente expresión regular:

`<\+[a–z][a–z0–9]*(,[a–z][a–z0–9]*)*(,[0–9]+)?\+>`

Además de los delimitadores, encontramos la estructura estándar para describir una lista de uno o más elementos separados por comas seguida de un entero opcional.

Pregunta 2 (0,1 puntos)

Escribe una gramática RLL(1) que acepte únicamente expresiones máximo correctamente formadas. Usa el no terminal ⟨Máximo⟩ como símbolo inicial. Utiliza las siguientes categorías léxicas:

- **cm** y **fm**: comienzo y fin de máximo (los delimitadores `<+` y `+>`).
- **id**: identificador.
- **en**: literal entero.
- **coma**: la coma.

Solución

Una primera idea sería utilizar directamente la expresión anterior:

$$\langle\text{Máximo}\rangle \rightarrow \textbf{cm id } (\textbf{coma id })^*(\textbf{coma en })?\ \textbf{fm}$$

Sin embargo, tenemos un conflicto con la coma: en la clausura, ante una coma el analizador no sabe si continuar en la clausura o terminarla y pasar al elemento opcional.

Para resolver el conflicto, hacemos lo siguiente. Escribimos el interior ⟨Máximo⟩ como un **id** seguido de ⟨MásId⟩, que generará el resto de **id** y, posiblemente el **en** final:

$$\begin{aligned}\langle\text{Máximo}\rangle &\rightarrow \textbf{cm id } \langle\text{MásId}\rangle\ \textbf{fm}\\ \langle\text{MásId}\rangle &\rightarrow \lambda|\textbf{coma } (\textbf{en} \mid \textbf{id } \langle\text{MásId}\rangle)\end{aligned}$$

De manera intuitiva, la segunda regla dice que después de un **id** podemos terminar o poner una coma seguida de un entero o una coma seguida de un entero y volver a empezar el ciclo.

Una manera un poco más formal de derivar la gramática anterior es comenzar por escribir la primera gramática como

$$\begin{aligned} \langle \text{Máximo} \rangle &\rightarrow \textbf{cm id} \langle \text{MásId} \rangle \textbf{ fm} \\ \langle \text{MásId} \rangle &\rightarrow (\textbf{coma id })^* (\textbf{coma en })? \end{aligned}$$

Ahora expandimos la clausura sabiendo que (**coma id**)* es equivalente a $\lambda|$(**coma id**)(**coma id**)*, aplicamos la distributiva y expandimos la opcionalidad:

$$\begin{aligned} \langle \text{Máximo} \rangle &\rightarrow \textbf{cm id} \langle \text{MásId} \rangle \textbf{ fm} \\ \langle \text{MásId} \rangle &\rightarrow \lambda|\textbf{coma en}|\textbf{coma id } (\textbf{coma id })^* (\textbf{coma en })? \end{aligned}$$

Aquí podemos extraer el prefijo común **coma** y darnos cuenta de que (**coma id**)*(**coma en**)? es justamente ⟨MásID⟩:

$$\begin{aligned} \langle \text{Máximo} \rangle &\rightarrow \textbf{cm id } \langle \text{MásId} \rangle \textbf{ fm} \\ \langle \text{MásId} \rangle &\rightarrow \lambda|\textbf{coma } (\textbf{en} \mid \textbf{id } \langle \text{MásId} \rangle) \end{aligned}$$

que es lo que queríamos encontrar.

Pregunta 3 (0,1 puntos)

Supongamos la siguiente gramática:

$$\begin{aligned} \langle \text{Máximo} \rangle &\rightarrow \textbf{cm } \langle \text{IdOEn} \rangle \langle \text{MásIdOEn} \rangle \textbf{ fm} \\ \langle \text{MásIdOEn} \rangle &\rightarrow \textbf{coma } \langle \text{IdOEn} \rangle \langle \text{MásIdOEn} \rangle \mid \lambda \\ \langle \text{IdOEn} \rangle &\rightarrow \textbf{id} \mid \textbf{en} \end{aligned}$$

Añade las acciones semánticas necesarias para que el atributo *bien* de ⟨Máximo⟩ sea cierto sí y sólo si la cadena derivada es correcta para un máximo. Puedes añadir los atributos que consideres necesarios, tanto heredados como sintetizados, con dos restricciones: deben ser de tipo simple (entero o lógico) y no puedes emplear como heredados atributos sintetizados y viceversa.

Solución

Añadiremos dos atributos sintetizados más:

- *esId*: un atributo lógico de ⟨IdOEn⟩ que es cierto si se ha derivado de él un **id**.
- *vacía*: un atributo lógico de ⟨MásIdOEn⟩ que es cierto si la cadena derivada es vacía.

En la primera regla, comprobaremos que el primer elemento es un **id** y que el resto es correcto. En la primera parte de la segunda regla, habrá que comprobar que o bien el primer elemento tras la coma es un **id** y el resto está bien o bien que el resto está vacío; además, rellenamos el atributo para indicar que la cadena generada no está vacía. En cuanto a la segunda parte, vemos que la cadena es vacía y correcta. En la reglas de ⟨IdOEn⟩, nos limitamos a informar sobre si es o no un **id** lo que se ha derivado.

La gramática con las acciones queda así:

$$\begin{aligned} \langle \text{Máximo} \rangle &\rightarrow \textbf{cm } \langle \text{IdOEn} \rangle \langle \text{MásIdOEn} \rangle \textbf{ fm } \{\langle \text{Máximo} \rangle.\text{bien} := \langle \text{IdOEn} \rangle.\text{esId; } \textbf{y } \langle \text{MásIdOEn} \rangle.\text{bien;}\} \\ \langle \text{MásIdOEn} \rangle &\rightarrow \textbf{coma } \langle \text{IdOEn} \rangle \langle \text{MásIdOEn} \rangle \\ &\quad \{\langle \text{MásIdOEn} \rangle.\text{bien} := \langle \text{IdOEn} \rangle.\text{esId } \textbf{y } \langle \text{MásIdOEn}_1 \rangle.\text{bien } \textbf{o } \langle \text{MásIdOEn}_1 \rangle.\text{vacía;} \\ &\quad\ \langle \text{MásIdOEn} \rangle.\text{vacía} := \textbf{falso;}\} \\ &\mid \lambda \ \{\langle \text{MasIdOEn} \rangle.\text{bien} := \textbf{cierto; } \langle \text{MasIdOEn} \rangle.\text{vacía} := \textbf{cierto; } \} \\ \langle \text{IdOEn} \rangle &\rightarrow \textbf{id } \{\langle \text{IdOen} \rangle.\text{esId} := \textbf{cierto;}\} \\ &\mid \textbf{en } \{\langle \text{IdOen} \rangle.\text{esId} := \textbf{falso;}\} \end{aligned}$$

Pregunta 4 (0,2 puntos)

Supongamos que el nodo NodoMáximo tiene como atributos la lista l de accesos a variables, el booleano hayEntero que es cierto sí y sólo si en la expresión había un entero, y el nodoEntero e con ese valor.

Escribe el método de generación de código para NodoMáximo.

Solución

Podemos dividir la generación de código en tres partes:

- Generamos código para evaluar el primer elemento de l, que provisionalmente será el máximo. Utilizaremos el registro con el resultado, r, para guardar el resultado final.
- Por cada elemento de l desde el segundo hasta el final, generamos código para evaluar ese elemento y para que se actualice r si fuera necesario. Habrá que liberar el registro empleado para el cálculo.
- Finalmente, si es necesario, emitimos código para evaluar el entero y actualizar r, siguiendo el mismo esquema que antes.

Con esta idea, el método queda:

```
Objeto NodoMáximo
    ...
    Método generaCódigo()
        r := l[1].generaCódigo();
        para n en l[2:] hacer:
            r2 := n.generaCódigo();
            et:= nuevaEtiqueta();
            emite(si r >= r2 salta et);
            emite(r := r2);
            emite(et:);
            liberaReg(r2);
        fin para
        si hayEntero entonces
            r2 := e.generaCódigo();
            et:= nuevaEtiqueta();
            emite(si r >= r2 salta et);
            emite(r := e);
            emite(et:);
            liberaReg(r2);
        fin si
        devuelve r;
    fin generaCódigo
    ...
fin NodoMáximo
```