E79 Procesadores de lenguaje

Examen de teoría (17 de diciembre de 2002)

PREGUNTA 1 (6 PUNTOS)

A continuación, se presentan tres posibles extensiones del lenguaje 2KS. Elige dos de ellas y explica claramente qué modificaciones se tendrían que hacer en un compilador de 2KS 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.

Inicialización automática de parámetros

Mediante esta extensión se permite la escritura de nuevas funciones basadas en otras ya existentes de modo que el último parámetro se calcule automáticamente en función del resto. La definición de una nueva función se hace mediante la palabra reservada `SUBRUTINA` seguida de: el nombre de la nueva función; un signo menor que (`<`); el nombre de la función en la que se basa la nueva; un signo mayor que (`>`); entre paréntesis, el nombre del último parámetro, el símbolo de asignación (`<-`) y una expresión en la que no aparezcan variables distintas de los nombres de los parámetros; finalmente, un punto y coma (`;`).

Por ejemplo: sea `f` una función con tres parámetros enteros, `a`, `b` y `c`. Podemos definir las nuevas funciones `g` y `h` de la siguiente manera:

```
SUBRUTINA g<f> (c<- 2*a+b); // g tiene dos parámetros enteros (a y b)
SUBRUTINA h<g> (b<- a+1);   // h tiene un parámetro entero (a)
```

Con esta definición, la llamada `g(2,3)` es equivalente a `f(2,3,7)` y `h(5)` es equivalente a `g(5,6)` y, por lo tanto, equivalente a `f(5,6,16)`.

Nota: la pseudo-asignación al último parámetro debe seguir las reglas de tipos de 2KS.

Nuevos operadores lógicos

Esta extensión dota a 2KS de dos nuevos operadores lógicos: `&&` y `||`. Estos operadores tienen la misma precedencia y asociatividad que los ya existentes (`&` y `|`) y difieren de ellos en su forma de evaluación. Tanto `&&` como `||` garantizan que su segundo operando sólo se evaluará si es necesario para conocer el resultado de la operación. En cuanto a su significado, es similar al que da Python para los operadores `and` y `or`:

- El resultado del operador `&&` es su primero operando si es falso y el segundo si el primero es cierto.
- El resultado del operador `||` es su primer operando si es cierto y el segundo si el primero es falso.

Así `n!=0 && 1/n` devuelve `0` (sin provocar error de ejecución) si `n` es nula y `1/n` si no lo es. Por otro lado `!llamar || g(b)` devuelve `1` si `llamar` es falso y el valor de `g(b)` si `llamar` es cierto (y sólo se ejecuta `g` en este caso).

Creación de variables equivalentes

Esta modificación permite crear nuevos nombres (alias) para variables ya declaradas en el programa. Para ello, la sintaxis de las declaraciones de variables se modifica de modo que tras un identificador se puede incluir opcionalmente la secuencia[1] "`= alias(` *id* `)`". Por ejemplo, la declaración

```
enter i= alias(j);
```

[1] Observa que esta extensión introduce una nueva palabra reservada, `alias`, con las reglas habituales de las palabras reservadas en 2KS.

hace que cualquier uso de `i` en el ámbito de esta declaración sea equivalente a un uso de `j`.

Los alias que se declaren deben tener el mismo tipo que la variable original (que puede a su vez ser un alias). En cuanto al ámbito de la declaración, puede ser distinto, aplicándose a los alias las mismas reglas de ámbito que a las variables.

Pregunta 2 (2 puntos)

Escribe expresiones regulares para los siguientes lenguajes:

- Comentarios de un lenguaje de programación que comienzan por una secuencia de tres asteriscos y terminan por otra secuencia de tres asteriscos sin ninguna de tales secuencias entre ellas. Por ejemplo:

  ```
  *** Esto es un comentario ***
  ```

- Identificadores formados por letras minúsculas que no coincidan ni con la palabra reservada `fi` ni con `fins`.
- Números tales que en la secuencia de sus dígitos no aparece el número doce (`13124` no valdría y `54213` sí).

Pregunta 3 (1 puntos)

Sea $G = (N, \Sigma, P, \langle \mathsf{S} \rangle)$ una gramática incontextual tal que si $\langle \mathsf{A} \rangle \to \alpha \in P$ y $\langle \mathsf{A} \rangle \to \beta \in P$, entonces $\alpha = \beta$. Se pide:

- Demostrar que G es LL(1).
- Decir qué podemos afirmar acerca de si G es RLL(1) en caso de que G sea una GPDR.

Pregunta 4 (1 puntos)

Sea G una gramática en la que la sentencia x tiene la siguiente derivación:

$$\langle \mathsf{S} \rangle \stackrel{*}{\Rightarrow} \alpha\beta_1\gamma \Rightarrow \alpha\beta_2\gamma \Rightarrow \ldots \Rightarrow \alpha\beta_n\gamma \stackrel{*}{\Rightarrow} x$$

¿Puede ser G una gramática LR(1) en los siguientes casos?

- Si existen i y j diferentes tales que $\beta_i = \beta_j$.
- Si la derivación anterior es una derivación canónica por la izquierda.

Justifica las respuestas.