Compiladors: Examen de laboratori

11 d'abril de 2012

Duració de l'examen: fins les 10:00h (no es podrà fer cap entrega més enllà d'aquesta hora). Cal enviar l'intèrpret complet en un fitxer `tgz` i fer l'entrega a través del racó. En el racó també es pot trobar un fitxer anomenat `proves.tgz` que conté uns jocs de proves per validar els exercicis proposats a l'examen.

1 Repeat-until (3 punts)

Afegir una nova instrucció a l'intèrpret d'Asl que permeti repetir una seqüència d'instruccions fins que es compleixi una condició. Per exemple, aquesta seria una manera de programar un factorial amb el `repeat-until`.

```
func fact(n)
  if (n <= 1) then return 1 endif;
  f = 1;
  repeat
    f = f * n;
    n = n - 1;
  until n <= 1;
  return f;
endfunc
```

Joc de proves: `fact_repeat.asl`.

2 Do-while (2 punts)

En aquest exercici cal *intentar* afegir una nova instrucció que implementi un `do-while`. La funció anterior es podria escriure de la següent manera:

```
func fact(n)
  if (n <= 1) then return 1 endif;
  f = 1;
  do
    f = f * n;
    n = n - 1;
  while n > 1;
  return f;
endfunc
```

Comentar les dificultats per introduir aquesta nova instrucció i explicar per què es produeixen aquestes dificultats. Escriure la resposta en un fitxer anomenat `Do_while.txt` i incloure'l amb l'entrega de l'examen.

3 Break (3 punts)

Afegir la instrucció `break` que fa que s'abandoni l'execució del bucle en el que s'executa (la mateixa semàntica que en els llenguatges C/C++/Java).

L'exercici consta de dues parts:

(1.5 punts) Modificar l'analitzador sintàctic per tal d'acceptar la instrucció i donar un error qual el `break` aparegui fora d'un bucle (`while` o `repeat`).

(1.5 punts) Afegir les accions necessàries a l'intèrpret per a poder executar la instrucció `break`.

Per a la primera part, només cal modificar l'analitzador sintàctic. Suggerència: fer servir alguna variable global dins de `@members` i acaba d'omplir la part de l'analitzador que es mostra a continuació:

```
@lexer::header {
package parser;
}

@members{
... Definicions globals ...
}

...
// break statement
break_stmt: BREAK {if (...) throw new RecognitionException(input);}
            ;
            catch [RecognitionException re] {
              reportError(re);
              emitErrorMessage("Syntax error: break statement out of a loop");
            }
```

Per a la segona part, només cal modificar el fitxer `Interp.java`. Suggerència: afegir una variable de la classe que indiqui si s'ha executat una instrucció `break`.

Jocs de proves: `break_error.asl`, `fact_break.asl` i `sum_product_break.asl`.

4 Avaluació estricta d'expressions booleanes (2 punts)

L'avaluació d'expressions booleanes a Asl és *mandrosa*, és a dir, l'avaluació finalitza tant aviat com es pot decidir el valor de l'expressió.

Volem incorporar operadors booleans per a realitzar avaluació estricta (s'avaluen tots els operands independentment del resultat de l'expressió). Per això definirem dos nous operadors: *e_and* (eager and) i *e_or* (eager or).

Modificar l'intèrpret per a introduir els nous operadors booleans. Només cal modificar els fitxers `Asl.g` i `Interp.java`.

Joc de proves: `eager.asl`.