Trabajo Practico 1 - ALP

Camilo Garcia Gonzalez

10 de Septiembre del 2019

1 Extensión de sintaxis.

1.1 Sintaxix abstracta:

```
intexp ::= ...
         | var = intexp
         | intexp ; intexp
```

1.2 Sintaxis concreta:

```
intexp ::= ...
         | var ':=' intexp
         | intexp ';' intexp
```

2 Extensión de sintaxis abstracta en Haskell.

```
data IntExp = ...
            | LetExp Variable IntExp
            | SeqExp IntExp IntExp
```

3 Parser para un programa LIS.

(Ver en Parser.hs)

4 Extensión de semantica Big-Step.

Como con $\Downarrow_{\text{intexp}}$ no es posible expresar un cambio en el entorno, definimos la relación $\Downarrow'_{\text{intexp}}$ para la cual vale:

$$\frac{\langle e, \sigma\rangle \Downarrow_{\text{intexp}} n}{\langle e, \sigma\rangle \Downarrow'_{\text{intexp}} \langle n, \sigma\rangle}$$

y además tenemos las siguientes reglas de derivación Big-Step para secuencia y asignación de expresiones enteras:

$$\frac{\langle e, \sigma\rangle \Downarrow'_{\text{intexp}} \langle n, \sigma'\rangle}{\langle v = e, \sigma\rangle \Downarrow'_{\text{intexp}} \langle n, [\sigma'|v : n]\rangle}\ \textsc{AssExp}$$

$$\frac{\langle e_1, \sigma\rangle \Downarrow'_{\text{intexp}} \langle n_1, \sigma'\rangle \qquad \langle e_2, \sigma'\rangle \Downarrow'_{\text{intexp}} \langle n_2, \sigma''\rangle}{\langle e_1; e_2, \sigma\rangle \Downarrow'_{\text{intexp}} \langle n_2, \sigma''\rangle}\ \textsc{SeqExp}$$

5 Determinismo de la relación de evaluación de un paso.

Veamos que $\rightsquigarrow$ es determinista, para esto probaremos que si t $\rightsquigarrow$ t' y t $\rightsquigarrow$ t", debe ser entonces t'=t".

Proof. Lo probaremos por inducción en la derivación t $\rightsquigarrow$ t'.

- Si la última regla utilizada fue Skip, entonces

  1. t = $\langle \textbf{skip}, \sigma\rangle$
  2. t' = σ

  Luego, si tenemos t $\rightsquigarrow$ t", por (1), la única regla aplicable para derivar t" debe ser también Skip.
  $$\therefore t' = t''$$

- Si la última regla utilizada fue Ass, entonces

  1. t = $\langle v := e, \sigma\rangle$
  2. $\langle e, \sigma\rangle \Downarrow_{\text{intexp}} n$
  3. t' = $[\sigma, v : n]$

  Luego, si tenemos t $\rightsquigarrow$ t", por (1), la única regla aplicable para derivar t" debe ser también Ass.
  $$\therefore t' = t''$$

- Si la última regla utilizada fue Seq1, entonces

  1. t = $\langle c_0; c_1, \sigma\rangle$
  2. $\langle c_0, \sigma\rangle \rightsquigarrow \sigma'$
  3. t' = $\langle c_1, \sigma'\rangle$

  Luego, si tenemos t $\rightsquigarrow$ t", por (1), t" podría haber sido derivada mediante Seq1 o Seq2. Si fué mediante Seq2 tendriamos:

  4. $\langle c_0, \sigma\rangle \rightsquigarrow \langle c'_0, \sigma'\rangle$

5. t" = $\langle c_0'; c_1, \sigma' \rangle$

Pero (2) y (4) contradicen la hipotesis inductiva, por lo que concluimos que t" debió ser derivada mediante Seq1.

$$\therefore t' = t''$$

- Si la última regla utilizada fue Seq2, seguir un razonamiento análogo al punto anterior.

- Si la última regla utilizada fue If1, entonces

  1. t = $\langle \textbf{if}(b)\textbf{then}(c_0), \sigma \rangle$
  2. $\langle b, \sigma \rangle \Downarrow_{\text{boolexp}} \textbf{true}$
  3. t' = $\langle c_0, \sigma \rangle$

  Luego, si tenemos t $\rightsquigarrow$ t", por (1), t" podría haber sido derivada mediante If1 o If2, pero por (2) no puede ser If2 (pues para esto deberia ser $\langle b, \sigma \rangle \Downarrow_{\text{boolexp}} \textbf{false}$), por lo que t $\rightsquigarrow$ t" se derivó mediante If1.

$$\therefore t' = t''$$

- Si la última regla utilizada fue If2, seguir un razonamiento análogo al punto anterior.

- Si la última regla utilizada fue If3, entonces

  1. t = $\langle \textbf{if}(b)\textbf{then}(c_0)\textbf{else}(c_1), \sigma \rangle$
  2. $\langle b, \sigma \rangle \Downarrow_{\text{boolexp}} \textbf{true}$
  3. t' = $\langle c_0, \sigma \rangle$

  Luego, si tenemos t $\rightsquigarrow$ t", por (1), t" podría haber sido derivada mediante If3 o If4, pero por (2) no puede ser If4 (pues para esto deberia ser $\langle b, \sigma \rangle \Downarrow_{\text{boolexp}} \textbf{false}$), por lo que t $\rightsquigarrow$ t" se derivó mediante If3.

$$\therefore t' = t''$$

- Si la última regla utilizada fue If4, seguir un razonamiento análogo al punto anterior.

- Si la última regla utilizada fue While1, entonces

  1. t = $\langle \textbf{while}(b)\textbf{do}(c), \sigma \rangle$
  2. $\langle b, \sigma \rangle \Downarrow_{\text{boolexp}} \textbf{true}$
  3. t' = $\langle c; \textbf{while}(b)\textbf{do}(c), \sigma \rangle$

Luego, si tenemos t $\rightsquigarrow$ t", por (1), t" podría haber sido derivada mediante While1 o While2, pero por (2) no puede ser While2 (pues para esto deberia ser $\langle b, \sigma\rangle \Downarrow_{\text{boolexp}} \textbf{false}$), por lo que t $\rightsquigarrow$ t" se derivó mediante While1.

$$\therefore t' = t''$$

- Si la última regla utilizada fue While2, seguir un razonamiento análogo al punto anterior.

□

6 Árbol de derivación.

$$\dfrac{\dfrac{\dfrac{\langle x-1, [\sigma|x:2]\rangle \Downarrow_{\text{intexp}} 1}{\langle x = x-1, [\sigma|x:2]\rangle \rightsquigarrow [\sigma|x:1]}\ \textsc{Ass}}{\langle x = x-1; \textbf{while}(x>0)\textbf{do}(x=x-2), [\sigma|x:2]\rangle \rightsquigarrow \langle \textbf{while}(x>0)\textbf{do}(x=x-2), [\sigma|x:1]\rangle}\ \textsc{Seq1}$$

$$\dfrac{\langle (x>0), [\sigma|x:1]\rangle \Downarrow_{\text{boolexp}} \textbf{true}}{\langle \textbf{while}(x>0)\textbf{do}(x=x-2), [\sigma|x:1]\rangle \rightsquigarrow \langle (x=x-2); \textbf{while}(x>0)\textbf{do}(x=x-2), [\sigma|x:1]\rangle}\ \textsc{While1}$$

$$\dfrac{\dfrac{\dfrac{\langle x-2, [\sigma|x:1]\rangle \Downarrow_{\text{intexp}} -1}{\langle (x=x-2), [\sigma|x:1]\rangle \rightsquigarrow [\sigma|x:-1]}\ \textsc{Ass}}{\langle (x=x-2); \textbf{while}(x>0)\textbf{do}(x=x-2), [\sigma|x:1]\rangle \rightsquigarrow \langle \textbf{while}(x>0)\textbf{do}(x=x-2), [\sigma|x:-1]\rangle}\ \textsc{Seq1}$$

$$\dfrac{\langle (x>0), [\sigma|x:-1]\rangle \Downarrow_{\text{boolexp}} \textbf{false}}{\langle \textbf{while}(x>0)\textbf{do}(x=x-2), [\sigma|x:-1]\rangle \rightsquigarrow [\sigma|x:-1]}\ \textsc{While2}$$

7 Interprete para LIS sin manejo de errores.

(Ver en Eval1.hs)

8 Interprete para LIS con manejo de errores.

(Ver en Eval2.hs)

9 Interprete para LIS con manejo de errores y cálculo de costo.

(Ver en Eval3.hs)

10 Introducción del comando For.

```
comm ::= ...
       | for intexp ; boolexp ; intexp { comm }
```

$$\frac{\langle e_1, \sigma\rangle \Downarrow'_{\text{intexp}} \langle n, \sigma'\rangle \qquad \langle e_2, \sigma'\rangle \Downarrow_{\text{boolexp}} \textbf{true}}{\langle \textbf{for} e_1; e_2; e_3\{c\}, \sigma\rangle \rightsquigarrow \langle c; \textbf{for} e_3; e_2; e_3\{c\}, \sigma'\rangle}\ \textsc{For1}$$

$$\frac{\langle e_1, \sigma\rangle \Downarrow'_{\text{intexp}} \langle n, \sigma'\rangle \qquad \langle e_2, \sigma'\rangle \Downarrow_{\text{boolexp}} \textbf{false}}{\langle \textbf{for} e_1; e_2; e_3\{c\}, \sigma\rangle \rightsquigarrow \sigma'}\ \textsc{For2}$$