Lenguaje LET

Este es un lenguaje de programacin muy simple con vinculación de variables locales y expresiones condicionales.

Sintaxis

La sintaxis concreta del lenguaje se especifica con una gramática libre de contexto, las categorías sintácticas o no-terminales del lenguaje se enfatizan en itálicas, mientras que los terminales se escriben sin énfasis, o bien, en negritas para estilizar palabras reservadas. Una producción de la forma $X \to \alpha$ establece que se puede derivar la cadena de terminales y no-terminales α a partir de la categoría sintáctica X.

La sintaxis abstracta del lenguaje se especifica a partir de la estructura de las producciones, cada una de estas se denota con un nombre de la producción, seguido de una lista de nombres para cada uno de los no-terminales del lado derecho de la produccin. Una estructura de la forma $x\ (y_1\ y_2\ \ldots\ y_n)$ representa la existencia de: (a) un constructor x que toma n argumentos y produce un árbol de sintaxis de este tipo; (b) un predicado $x?$ que toma un argumento y determina si es de tipo x; (c) n selectores x-y_i que toman un argumento de tipo x y regresan su subestructura correspondiente a y_i.

Sintaxis concreta

		Sintaxis abstracta
Program ::=	*Expression*	program (*exp1*)
Expression ::=	*Integer*	const-exp (*num*)
Expression ::=	**-**(*Expression* , *Expression*)	diff-exp (*exp1 exp2*)
Expression ::=	**zero?**(*Expression*)	zero?-exp(*exp1*)
Expression ::=	**if** *Expression* **then** *Expression* **else** *Expression*	if-exp (*exp1 exp2 exp3*)
Expression ::=	*Identifier*	var-exp (*var*)
Expression ::=	**let** *Identifier* = *Expression* **in** *Expression*	let-exp (*var exp1 body*)

Semántica

La interpretación de expresiones del lenguaje de acuerdo a su sintaxis abstracta se especifica utilizando semántica operacional de pasos grandes (también llamada semántica natural) que consiste de afirmaciones de la forma (value-of $e\ \rho) = v$ donde e es una expresión, ρ un entorno y v un valor expresado.

Existe un entorno vacío ρ_0, un mecanismo para extender un entorno ρ con la vinculación de una variable x a un valor v denotado como $[x : v]\rho$ y un mecanismo para aplicar entornos a variables $\rho(x)$. En particular, los entornos se modelan como funciones de variables a valores denotados, con ρ_0 indefinida para cualquier variable.

En este lenguaje los valores expresados *ExpVal* (resultados de expresiones) y los valores denotados *DenVal* (vinculados a variables) son los mismos y corresponden a $Int + Bool$. Los procedimientos expressed→int y expressed→bool toman valores expresados y regresan los valores codificados en el lenguaje de implementación.

Interpretación de expresiones

(value-of (const-exp n) ρ) = (int-expressed n)

(value-of (var-exp *var*) ρ) = ρ(*var*)

(value-of (diff-exp *exp1 exp2*) ρ)
  = (int-expressed (- (expressed→int (value-of *exp1* ρ))
                      (expressed→int (value-of *exp2* ρ))))

(value-of (zero?-exp *exp1*) ρ)
  = (**let** ([*val1* (value-of *exp1* ρ)])
      (bool-expressed (= 0 (expressed→int *val1*))))

(value-of (if-exp *exp1 exp2 exp3*) ρ)
  = (**if** (expressed→bool (value-of *exp1* ρ))
      (value-of *exp2* ρ)
      (value-of *exp3* ρ))

(value-of (let-exp *var exp1 body*) ρ)
  = (**let** ([*val1* (value-of *exp1* ρ)])
      (value-of *body* [*var* : *val1*]ρ))