Práctica 2

Marco Antonio Orduña Avila y Enrique Sanchez Lara

Facultad de Ciencias, UNAM

11/Sep/2019

1. Descripción del programa

Para la implementación lo que se hizo fue primero un intento de las funciones evals pero se tenia una duda que fue resuelta en el laboratorio

Para la semantica dinamica, lo que se hizo y se hace mediante un sistema de trancisisones, sabemos que los estados son arboles bien formados y los estados iniciales son expresiones cerradas, definimos valores que son un subconjunto de expresiones que ya se terminaron de evaluar, se puede observar que:

Las variables son estados bloqueados, pero no son finales. En particular las variables libres no pueden evaluarse.

Las reglas sin premisas que involucren a la relacion de transición se conocen como instrucciones puesto que corresponden a pasos primitivos de ejecución, mientras que las reglas restantes definen transiciones condicionales que determinan el orden en el que se ejecutan las instrucciones.

A veces utilizaremos la sintaxis concreta para facilitar el manejo de las transiciones en el papel o pizarrón, sin embargo debemos recordar que cualquier discusión se refiere a la sintaxis abstracta.
recordemos que la estrategia es por evaluacion glotona, ademas para la implementación se tomaron en cuenta las propiedades de la semantica dinamica propuestas en la seccion 4.2 de la pagina 8 de las notas de clase

La semántica dinámica debe implementarse mediante las siguientes funciones: $eval1 : EAB- > EAB$ tal que eval1 $e = e'sysse \rightarrow e'$ $evals : EAB- > EAB$ tal que evals $e = e'sysse \rightarrow *e'ye'$ está bloqueado. $eval : EAB- > EAB$ tal que eval $e = e'sysse \rightarrow *e'ye'$ es un valor. La diferencia con evals es que deben manejarse los errores de ejecución.

Para la implementación de la semantica estatica

Para la relación del tipado debe de implementarse como sigue:
Definir un tipo Tipo para los tipos de EAB
Definir un tipo Ctx para los contextos $\Gamma = x_1 : T_1, ..., x_n : T_n$
lo más simple es una lista de pares de variables y tipos. Definir la función de verificación de tipado $vt :: Ctx- > EAB- > Tipo- > Bool$ tal que
vt Γ e T = True syss $\Gamma \vdash e : T$
Ademas de utilizar las propiedades de la semantica estatica

$$\frac{}{not(B[b]) \to B[\neg b]}\ (notv)$$

$$\frac{e \to e'}{not(e) \to not(e')}\ (notc)$$

$$\frac{}{and(B[b_1], B[b_2]) \to B[b_1 \wedge b_2]}\ (andv)$$

$$\frac{e_1 \to e_1'}{and(e_1, e_2) \to and(e_1', e_2)}\ (andr)$$

$$\frac{e_2 \to e_2'}{and(B[b], e_2) \to and(B[b], e_2')}\ (andr)$$

$$\frac{}{and(B[b_1], B[b_2]) \to B[b_1 \vee b_2]}\ (orv)$$

$$\frac{e_1 \to e_1'}{and(e_1, e_2) \to and(e_1', e_2)}\ (orr)$$

$$\frac{e_2 \to e_2'}{and(B[b], e_2) \to and(B[b], e_2')}\ (orr)$$

$$\frac{}{Gt(I[n], I[m]) \to B[b_1 > b_2]}\ (gtv)$$

$$\frac{e_1 \to e_1'}{Gt(e_1, e_2) \to Gt(e_1', e_2)}\ (gtr)$$

$$\frac{e_2 \to e_2'}{Gt(I[n], e_2) \to Gt(I[n], e_2')}\ (gtr)$$

Para Lt y Eq son casos análogos a Gt pero cambia el caso base a menor que y la iguadad

$T ::= Nat|Bool$

La relación de tipado $\Gamma \vdash t : T$ se define inductivamente como sigue: Tipado de variables

$$\frac{}{\Gamma, x : T \vdash x : T}\ (tvar)$$

Tipado de valores numéricos:

$$\frac{}{\Gamma \vdash num[n] : Nat}\ (tnum)$$

Tipado de valores booleanos:

$$\frac{}{\Gamma \vdash bool[true] : Bool} \ (ttrue)$$

$$\frac{}{\Gamma \vdash bool[false] : Bool} \ (tfalse)$$

Tipado de expresiones aritméticas:

$$\frac{\Gamma \vdash t_1 : Nat \Gamma \vdash t2 : Nat}{\Gamma \vdash prod(t_1, t_2) : Nat} \ (tprod)$$

$$\frac{\Gamma \vdash t : Nat}{\Gamma \vdash suc(t) : Nat} \ (tsuc)$$

$$\frac{\Gamma \vdash t : Nat}{\Gamma \vdash pred(t) : Nat} \ (tpred)$$

$$\frac{\Gamma \vdash t_1 : Nat \Gamma \vdash t_2 : Nat}{\Gamma \vdash suma(t_1, t_2) : Nat} \ (tsum)$$

Tipado de expresiones booleanas:

$$\frac{\Gamma \vdash t_1 : Bool \Gamma \vdash t_2 : T \Gamma \vdash t_3 : T}{\Gamma \vdash if(t_1, t_2, t_3) : T} \ (tif)$$

$$\frac{\Gamma \vdash t : Nat}{\Gamma \vdash iszerot : Bool} \ (tisz)$$

Tipado de expresiones let:

$$\frac{\Gamma \vdash e_1 : T \Gamma, x : T \vdash e_2 : S}{\Gamma \vdash let(e_1, x.e_2 : S} \ (tlet)$$

2. Entrada y ejecución

Para ejecccutar el programa se debe de encontrar en la ruta del archibo BAE y dentro de esta carpeta deben de estar los tres archivos llamados Semantic.hs Sintax.hs y Parser.hs basta con compliar con el comando ghci Semantic.hs que ademas Semantic importa Sintax ahora para las funciones pondremos algunos ejemplos

función eval1.
$evals(Add(Mul(I2)(I6))(BTrue))Add(I12)(BTrue)$

función evals.

$evals(Let"x"(Add(I1)(I2))(Eq(V"x")(I0)))$

Debe mandar B False

función evale

$evale(Or(Eq(Add(I0)(I0))(I0))(Eq(I1)(I10)))$

debe de regresar B True

función vt

$vt[("x", Integer), ("y", Integer), ("z", Boolean), ("w", Integer)](Add(Mul(V"y")(V"w"))$
$(Add(V"x")(V"x")))(Integer)$

debe regresar True

función eval

$eval(Let"x"(BTrue)(If(BTrue)(BFalse)(Var"x")))Boolean$

debe regresar B False

3. Conclusiones

Con esto concluido podemos decir que le damos cierta seguridad al lenguaje, esto quiere decir que no le daremos oportunidad al lenguaje de que falle durante la ejecución. En general la seguridad por tipos expresa la coherencia entre la semántica estática y la semántica dinámica. La semántica estática predice que el valor de una expresión tendrá cierta forma de manera que la semántica dinámica de dicha expresión está bien definida. En consecuencia, la evaluación no puede bloquearse en un estado no final para el cual no haya transición posible, lo cual corresponde en una implementación a la ausencia de errores causados por una instrucción ilegal en tiempo de ejecución.