KARAN TANWAR
2017CS50411

31/1/19

PROOF OF CORRECTNESS

eval : exptree → int
compiler: exptree → list of opcodes
stackmc : List of opcodes → head (list of bigint)

To prove :- $\forall t \in$ exptree of $ht(t) \geq 1$

eval (t) = stackmc [] (compile t)

Proof : By induction on ht (t)

Base case : $ht(t) = 1$
$\Rightarrow t := N(i)$

P(1)

$\Rightarrow$ eval (t) = eval (N (i)) = i
stackmc [] (compile t) = stackmc [] [CONST i]
= CONST i

$\Rightarrow$ eval (t) = stackmc [] (compile t) for $ht(e) = 1$

INDUCTION HYPOTHESIS

P(k): $\forall \dot{t} \in$ exptrees s.t $ht(\dot{t}) \leq k$
eval ($\dot{t}$) = stackmc [] (compile $\dot{t}$)

INDUCTIVE STEP

P(k+1) Let $ht(t) = k+1$ $k > 0$
$\Rightarrow t \neq N(i)$ Thus t := Plus (t1, t2)
or Minus (t1, t2)
or Mult (t1, t2)
:
and so on

Reason for this is, compile of t1 and t2 gives lists of opcodes and stackmc (compile t1) gives eval of t1 (Induction hypothesis) By definition, eval (t1) and eval (t2) should be added to stack.

w.l.g. we choose any operator say Plus

thus $t = Plus(t_1, t_2)$

Thus by definition of ht(t),

$$ht(t_1) \leq k \text{ and } ht(t_2) \leq k \quad \text{— (a)}$$

~~Eval stackmc [] comp~~

compile t = compile (Plus t1 t2)

= [compile t1] @ [compile t2] @ [PLUS]

stackmc [] (compile t)

= ~~add (Compile t1) (compile t2)~~

= add ((stackmc [] compile t1) (stackmc [] compile t2))

eval(t) = eval (Plus t1, t2)

= add (eval t1 eval t2)

Since using Induction hypothesis and (a),

eval t1 = stackmc [] (compile t1)

eval t2 = stackmc [] (compile t2)

⇒ stackmc [] (compile t) = eval (t)

Thus P(k+1) is true when P(k) is true and P(1) is true.

Thus eval computes same as compiler + stack machine calculates.

(The proof for t = UNARYMINUS( ) or ABS( ) will be same as above; instead there will be only t1 instead of t1 and t2.)