Eric Wan – [ezw23@drexel.edu](mailto:ezw23@drexel.edu)

(define (is-simplified? expr)

(if (constant? expr)

#t

(and (noconstant-arith? expr) (nozeros? expr) (nomult1? expr))

)

)

(plus-simp 2 5) = (+ 2 5) = 7

(eval (plus-simp a b) env) = (eval (+ a b) env)

case 1:

let c1 be a constant int

let c2 be a constant int

let (+ c1 c2) = c3

(eval (plus-simp c1 c2) env)

by def:

= (eval (+ c1 c2) env)

= c3

(eval (+ c1 c2) env)

= c3

let c2 be any value

then (eval c2 env) = c2

and (+ c2 0) = c2

(eval (plus-simp 0 c2) env)

= (eval c2 env)

= c2

(eval (+ 0 c2) env)

= c2

**Question 2:**

(arith-eval (arith-simp expr) env) = (arith-eval expr env)

expr = (\* E1 E2)

(arith-simp expr)

= (arith-simp (\* E1 E2))  
= (mult? (\* E1 E2))

= (let ([simpexpr1 (arith-simp (op1 expr))] [simpexpr2 (arith-simp (op2 expr))])

(mult-simp simpexpr1 simpexpr2))

= (mult-simp (arith-simp (op1 expr)) (arith-simp (op2 expr)))

= (multi-simp (arith-simp E1) (arith-simp E2))

= (multi-simp (constant? E1) (constant? E2))

= (multi-simp E1 E2)

= (make-mult expr1 expr2)

= (\* E1 E2)

**Question 3:**

Prove that (is-simplified? (arith-simp expr)) = #t

1. Prove k=0, 1 case is true

assume expr is constant:

(arith-simp expr)

= (arith-simp constant)

= (constant? constant)

= constant

(is-simplified? constant)

= (constant? constant) = #t

assume expr is variable:

(arith-simp expr)

= (arith-simp variable)

= (variable? variable)

= variable

(is-simplified? variable)

= (constant? variable) = #f

= (and (noconstant-arith? expr) (nozeros? expr) (nomult1? expr))))

= (and (variable? expr) (variable? expr) (variable? expr))

= (and #t #t #t) = #t

2. Assume k=n case is true

expr is (+ E1 E2):

(arith-simp expr)

= (plus? expr) => #t

= (let ([simpexpr1 (arith-simp (op1 expr))] [simpexpr2 (arith-simp (op2 expr))])

(plus-simp simpexpr1 simpexpr2))

= (plus-simp simpexpr1 simpexpr2)

= (make-plus expr1 expr2)

= (+ E1 E2) = E3

= (is-simplified? expr)

= (is-simplified E3)

= (constant? expr) = #t

expr is (\* E1 E2):

(arith-simp expr)

form previous steps

= expr = E3

= (is-simplified? expr)

= (constant? expr) = #t

3. Prove k=n+1 case is true

expr is (+ (+ E1 E2) E3):

(arith-simp expr)

= (arith-simp (+ (+ E1 E2) E3))

= (plus? expr)

= (let ([simpexpr1 (arith-simp (op1 expr))] [simpexpr2 (arith-simp (op2 expr))])

(plus-simp simpexpr1 simpexpr2))

= (plus-simp (arith-simp (+ E1 E2)) (arith-simp E3))

from previous

= (plus-simp (arith-simp E3) (arith-simp E3))

= E6 = expr

= (is-simplified? expr)

= (constant? expr) = #t

expr is (\* (+ E1 E2) E3):

(arith-simp expr)

= (arith-simp (\* (+ E1 E2) E3))

= (mult? expr)

= (let ([simpexpr1 (arith-simp (op1 expr))] [simpexpr2 (arith-simp (op2 expr))])

(mult-simp simpexpr1 simpexpr2))

= (mult-simp (arith-simp (+E1 E2)) (arith-simp E3))

from preious

= (mult-simp E3 E3)

= E9 = expr

= (is-simplified? expr)

= (constant? expr) = #t