;;; examples from
A Hacker’s introduction to Partial Evaluation

Jianshi Huang

21 10 2011

1 Introduction

#

(defun emit-+ (x y) (cond ((eql x 0) y) ((eql y 0) x) ((and (equal x y) (not (symbolp x))) ‘(let ((y ,x)) (+ y y))) (t ‘(+ ,x ,y))))

(defun emit-* (x y) (cond ((or (eql x 0) (eql y 0)) 0) ((eql y 1) x) ((eql x 1) y) ((and (equal x y) (not (symbolp x))) ‘(let ((y ,x)) (* y y))) (t ‘(* ,x ,y))))

(defun emit-power (x n) (locally (declare (disable-package-locks cl:*)) (macrolet ((* (&rest a) ‘(emit-* ,@a))) ;; original code (labels ((square (x) (* x x)) (power (x n) (cond ((= n 0) 1) ((oddp n) (* x (power x (- n 1)))) (t (square (power x (/ n 2))))))) (power x n)))))

(defun emit-poly-value (coeffs x) (locally (declare (disable-package-locks cl:* cl:+)) (macrolet ((* (&rest a) ‘(emit-* ,@a)) (+ (&rest a) ‘(emit-+ ,@a))) (reduce (lambda (value coeff) (+ (* value x) coeff)) coeffs :initial-value 0))))

(defun call-with-cse (receiver) (let ((bindings ‘())) (labels ((cseify (emitter) (lambda (&rest operands) (let ((exp (apply emitter operands))) (cond ((or (symbolp exp) (numberp exp)) exp) ((assoc exp bindings) (cadr (assoc exp bindings))) (t (let ((name (gensym))) (push (list exp name) bindings) name))))))) (let ((exp (funcall receiver #’cseify))) ‘(let* ,(reverse (mapcar #’reverse bindings)) ,exp)))))

(defmacro with-cse ((receiver) &body body) ‘(call-with-cse (lambda (,receiver) (locally ,@body))))

(defun emit-poly-value (coeffs x) (with-cse (cseify) (declare (disable-package-locks * +)) (macrolet ((* (&rest args) ‘(funcall cseify #’emit-*

,@args)) (+ (&rest args) ‘(funcall cseify #’emit-+ ,@args))) (reduce (lambda (value coeff) (+ (* value x) coeff)) coeffs :initial-value 0))))