

Examen juni 2021-2022

Vraag1: meval

- Zet test-case om naar een afgeleide expressie en geef ook alle accessoren van zijn expressies en procedures die je nodig hebt om het in de eval aan te roepen

(test-case <test-exp> <expected-value >)

Evalueert naar:

Als het true is:

#t

Als het false is:

((<test-exp>) <expected-value> <actual-value>)

Bv.

(test-case (fac 0) 0)

Evalueert naar:

((fac 0) 0 1)

(test-case (fac 0) 1)

Evalueert naar:

#t

- Zet test-suite om naar een dedicated expressie en geef ook alle accessoren van zijn expressies en procedures die je nodig hebt om het in de eval aan te roepen

(test-suite <naam> <verwachte-waarden> <test-cases>)

Evalueert naar:

aantal gefaalde testen

Bv.

(test-suite "Mijn-test" (list 1 2 3 6)

 '((test-case (fac 0) 0)

 (test-case (fac 1) 1)

 (test-case (fac 2) 0)

 (test-case (fac 3) 6))

Evalueert naar:

1

Vraag2: geval

(post ?reaction-id ?sender ?original-id ?msg)

Met volgende feiten in databank:

(post 1 "Thierry" 1 "Hallo")

(post 2 "Ward" 1 "Dag")

(post 3 "Coen" 2 "Hoi")

(post 5 "Wolfgang" 4 "Wat is me dit nu")

- 1) Maak een regel die bepaalt wanneer een post de allereerste is
(start-post ?id ?sender ?msg)
- 2) Maak een query die als resultaat de berichten teruggeeft die een reactie zijn op berichten die zich niet meer in de database bevinden
- 3) Maak een regel die waar is indien een bericht een reactie is op een ander bericht
(is-reactie ?reactie-id ?origineel)
- 4) Breid deze regel uit zodat deze waar is indien deze post een reactie is op een oudere post
- 5) Maak een regel die voor een postid de starter van het gesprek teruggeeft
(start-gesprek ?post-id ?naam)

Vraag3: registermachine

Zet volgende code om naar registermachine code

- ```
(define (vind-grootste-munt bedrag munten)
 (if (<= (car munten) bedrag)
 (car munten)
 (vind-grootste-munt bedrag (cdr munten))))
```

  

```
(define (zoek-munten bedrag munten)
 (if (= bedrag 0)
 '()
 (let ((gezochte-munt (vind-grootste-munt bedrag munten)))
 (display gezochte-munt) (newline)
 (cons gezochte-munt (zoek-munten (- bedrag gezochte-munt) munten)))))
```
- ```
(zoek-munten 478 '(200 100 50 20 10 5 2 1))
; (200 200 50 20 5 2 1)
```
- ```
(zoek-munten 7 '(200 100 50 20 10 5 2 1))
(5 2)
```

#### Vraag 4

- A) Waarin verschilt de voorstelling van procedure-objecten in beide evaluatoren?
- B) Wat is het verschil tussen unificatie en patroonvergelijking
- C) Hoeveel posities kan de free pointer verplaatst worden in elke iteratie van gc-loop
- D) Verklaar waarom de execute procedure bij de registermachines kan stoppen en niet in een infinite loop terecht komt

```
(define (execute)
 (let ((insts (get-contents pc)))
 (if (null? insts)
 'done
 (begin
 ((instruction-execution-proc (car insts)))
 (execute))))))
```

- E) Decompileer dit stukje code en geef de 3 argumenten

```
(assign proc (op lookup-variable-value) (const x) (reg env))
(assign val (op lookup-variable-value) (const z) (reg env))
(assign argl (op list) (reg val))
(assign val (op lookup-variable-value) (const y) (reg env))
(assign argl (op cons) (reg val) (reg argl))
(test (op primitive-procedure?) (reg proc))
(branch (label primitive-branch1))
compiled-branch2
(assign continue (label after-call3))
(assign val (op compiled-procedure-entry) (reg proc))
(goto (reg val))
primitive-branch1
(assign val (op apply-primitive-procedure) (reg proc) (reg argl))
after-call3

>
```