Schriftelijk Examen
Interpretatie van Computerprogramma's I

Eerste zittijd academiejaar 2024–2025
Coen De Roover
23/06/2025 (reconstructie)

Het examen bestaat uit 4 vragen, verdeeld over 5 bladzijden.
Gelieve elke vraag te beantwoorden op een afzonderlijk blad.
Gesloten boek.
Bundel met `eval` codes beschikbaar.

---

1. Meta-circulaire Evaluator

Implementeer de special form `(indices-where <var> <list-exp> <pred-exp>)`.
Deze bindt `<var>` consecutief aan elk element van de lijst waarnaar `<list-exp>` evalueert.
De index van de elementen waarvoor `<pred-exp>` slaagt
(d.w.z. naar een van **#f** verschillende waarde evalueert), wordt daarbij in een lijst verzameld.
De volledige special form levert die lijst van indices als resultaat op.
Merk op dat het bereik van `<var>` beperkt is tot `<pred-exp>`.

Toelichting gebruik:

```
;;; M-eval input
(indices-where x (list 1 -3 4 5 6 8 -9) (> 0 x))
;;; M-eval output
'(1 6)

;;; M-eval input
(indices-where y (list 1 2 3 4 5 6) (even? y))
;;; M-eval output
'(1 3 5)
```

1a) Implementeer alle hulpprocedures die de meta-circulaire evaluator nodig heeft om `indices-where` expressies te herkennen en er onderdelen uit te selecteren.

1b) Breid de meta-circulaire evaluator uit met ondersteuning voor `indices-where` expressies. Implementeer `indices-where` via een dedicated evaluator procedure (bijv. `eval-indices-where`).

1c) Breid de meta-circulaire evaluator uit met ondersteuning voor `indices-where` expressies. Implementeer `indices-where` ditmaal als een afgeleide expressie.

2. Registermachines

Procedure som-van-cijfers:

```
(define (som-van-cijfers n)
  (if (< n 10)
      n
      (+ (remainder n 10) (som-van-cijfers (quotient n 10)))))
```

Toelichting gebruik:

```
(som-van-cijfers 42)
6
```

Zet dit vervolgens om naar een subroutine in registermachinecode.
Het iteratieve/recursieve karakter moet behouden blijven.

Je mag uitgaan van volgende primitieve operaties: `<`, `>`, `=`, `-`, `+`, `remainder`, `quotient`.
Gebruik de registers: `cont`, `v1`, en `res`. Je mag geen extra registers gebruiken.
De subroutines volgen een contract: `v1` als input, `res` als output, en `cont` om uit de subroutines te springen.

```
(define som-machine
  (make-machine
    '(cont res v1)
    `(,@ops
      (>, >) (<, <) (=, =) (+, +) (-, -)
      (remainder, remainder)
      (quotient, quotient))
    '(start
      ;;; jouw code hier

      stop)))

(set-register-contents! som-machine 'v1 42)
(start som-machine)
(display (get-register-contents som-machine 'res))
```

3. Open Vragen: Inzicht

3a) Waarin verschilt de voorstelling van procedure-objecten tussen M-eval en A-eval? Verklaar elk verschil.

3b) Voorspel hoeveel `"*"` er door L-eval op het scherm komen, en motiveer.

```
(define (f a) a)
(define x (f (begin (display "*") 3)))
```

3c) In welke volgorde evalueert EC-eval de operanden van een applicatieprocedure?
Welke optimalisatie wordt hierbij toegepast?
Verwijs naar de betrokken subroutines.

3d) Met hoeveel posities kan de scan pointer verplaatst worden in elke iteratie van `gc-loop` (**stop-en-kopieer algoritme**)?
Verklaar de mogelijkheden.

3e) Reconstrueer de 3 argumenten voor de oproep van `compile` die onderstaande registermachinecode heeft gegenereerd:

```
(assign val (op make-compiled-procedure) (label entry26) (reg env))
(goto (reg continue))
entry26
(assign env (op compiled-procedure-env) (reg proc))
(assign env (op extend-environment) (const (x)) (reg argl) (reg env))
(assign val (op lookup-variable-value) (const y) (reg env))
(goto (reg continue))
after-lambda27
```

4. Logisch Programmeren

Feitenbank:

```
(wandeltocht dodentocht Bornem Bornem 100)
(wandeltocht brumech Brussel Mechelen 24)
(wandeltocht habru Halle Brussel 17)

(gewandeld Coen (dodentocht))
(gewandeld Cindy (dodentocht brumech))
(gewandeld Sarah (habru))
```

Met betekenis:

```
(wandeltocht ?naam ?van ?naar ?km)
(gewandeld ?persoon ?tochtenlijst)
```

4a) Schrijf een *logische query* om te bepalen welke wandelingen aan elkaar gekoppeld kunnen worden. Twee wandelingen kunnen gekoppeld worden als de ene begint waar de andere eindigt.

4b) Gebruik een *logische regel* om het predicaat `(tocht-inlijst ?tocht ?tochtenlijst)` te implementeren.
Dit predicaat slaagt wanneer `?tocht` een wandeltocht is die in de lijst `?tochtenlijst` voorkomt.

4c) Schrijf een *logische query* waarmee bepaald kan worden welke personen een tocht gewandeld hebben die start of eindigt in Brussel.
Je mag gebruikmaken van 4b.

4d) Gebruik een *logische regel* om het predicaat `(kampioen-wandelaar ?persoon)` te implementeren.
Dit slaagt wanneer `?persoon` de langste wandeling gewandeld heeft.
Je mag gebruikmaken van 4b.