

Schriftelijk Examen: "Interpretatie van Computerprogramma's I"

Eerste zittijd, academiejaar 2015-2016

Coen De Roover

Vrije Universiteit Brussel

20 juni 2016

Het examen bestaat uit 5 vragen, verdeeld over 6 bladzijden.
Gelieve elke vraag te beantwoorden op een afzonderlijk blad.
Zet je naam, studierichting en rolnummer op *elk* blad dat je afgeeft.

De vragen zijn geformuleerd ten opzichte van de gepubliceerde slides.
Deze mogen, samen met *persoonlijke* nota's en met het referentieboek van
Abelson en Sussman, geraadpleegd worden tijdens deze test.
Het gebruik van elektronische media is echter *niet* toegestaan.
Veel succes!

1 Meta-circulaire evaluator

Veronderstel volgende globale definities in een programma:

```
1 (define x 0.66)
2 (define y 0.77)
3 (define speed 0.5)
4 (define dt 0.5)
5 (define (clamp val)
6   (min 1 (max 0 val)))
```

Onderstaande expressie zal dan 0.91 printen als waarde voor de variabele `new-x`, en 1 als waarde voor de variabele `new-y`:

```
1 (let ((new-x (clamp (+ x (* speed dt))))  
2     (new-y (clamp (+ y (* speed dt)))))  
3     (display new-x) ;; print 0.91  
4     (display new-y) ;; print 1
```

De oproepen van procedure `clamp` zorgen ervoor dat de waarden van `new-x` en `new-y` altijd tot het interval $[0, 1]$ behoren. Zonder deze oproepen zou er namelijk 0.91 en 1.02 geprint worden.

Om zulke repetitieve oproepen van `clamp` te vermijden, stellen we `let-clamped` als nieuw type expressie voor. Deze variant van `let` past `clamp` automatisch toe op de waarden waaraan de te binden variabelen gebonden worden:

```
1 (let-clamped ((new-x (+ x (* speed dt)))  
2              (new-y (+ y (* speed dt)))))  
3     (display new-x) ;; print 0.91  
4     (display new-y) ;; print 1
```

- 1.a) Implementeer alle hulpfuncties die je nodig hebt om de meta-circulaire evaluator uit te breiden met ondersteuning voor `let-clamped` expressies.
- 1.b) Breid de meta-circulaire evaluator uit met ondersteuning voor `let-clamped` expressies. Implementeer `let-clamped` als afgeleide expressie.
- 1.c) Breid de meta-circulaire evaluator uit met ondersteuning voor `let-clamped` expressies. Implementeer `let-clamped` als een special form met een "dedicated evaluator".

2 Logisch programmeren

Beschouw volgende feitenbank over e-mailberichten:

```
1 (email "Afspraak volgende week" "Beste A, ...")
2 (email "Re: Afspraak volgende week" "Beste B, ...")
3 (email-met-bijlagen "Re: Re: Afspraak volgende week"
4                       "OK. Zie bijlage. Mvg, B"
5                       (("fig-1.jpg" 10) ("fig.gif" 24)))
6 (email-met-bijlagen "Fwd: Examenvoorstel"
7                       "Beste A, in bijlage ..."
8                       (("examen.pdf" 1002)))
```

Van elke email worden het onderwerp en de inhoud bijgehouden. Van elke email-met-bijlagen wordt daarnaast eveneens de lijst van bijlagen bijgehouden. Merk op dat elke bijlage voorgesteld wordt als een lijst met als eerste element de naam van de bijlage en als tweede element de grootte van de bijlage.

2.a) Implementeer een *regel* email-met-of-zonder-bijlagen die abstraheert over e-mails met en zonder bijlagen, zodat volgende interactie mogelijk wordt:

```
1 ;; Query input:
2 (email-met-of-zonder-bijlagen ?onderwerp ?inhoud ?bijlagen)
3 ;; Query results:
4 (email-met-of-zonder-bijlagen "Fwd: Examenvoorstel"
5                               "Beste A, in bijlage ..."
6                               (("examen.pdf" 1002)))
7 (email-met-of-zonder-bijlagen "Re: Afspraak volgende week"
8                               "Beste B, ..."
9                               ())
10 (email-met-of-zonder-bijlagen "Re: Re: Afspraak volgende week"
11                               "OK. Zie bijlage. Mvg, B"
12                               (("fig-1.jpg" 10) ("fig.gif" 24)))
13 (email-met-of-zonder-bijlagen "Afspraak volgende week"
14                               "Beste A, ..."
15                               ())
```

2.b) Geef een *query* die de e-mail met de meeste bijlagen achterhaalt. Je mag ervan uitgaan dat er een Scheme-procedure (lijst-langer? 11 12) bestaat die aangeeft of lijst 11 langer is dan lijst 12.

2.c) Implementeer een *regel* bestandsnamen-voor-e-mail die alle bestandsnamen van de bijlagen van een e-mail met een bepaald onderwerp verzamelt:

```
1 ; Query input:
2 (bestandsnamen-voor-e-mail ?onderwerp ?bestandsnamen)
3 ; Query results:
4 (bestandsnamen-voor-e-mail "Fwd: Examenvoorstel" ("examen.pdf"))
5 (bestandsnamen-voor-e-mail "Re: Afspraak volgende week" ())
6 (bestandsnamen-voor-e-mail "Re: Re: Afspraak volgende week"
  - ("fig-1.jpg" "fig.gif"))
7 (bestandsnamen-voor-e-mail "Afspraak volgende week" ())
```

3 Programmeren van registtermachines

Voor deze vraag mag je een registtermachine veronderstellen met de registers $v1$, $v2$, res en $cont$. De registtermachine voorziet tevens de primitieve operaties $<$, $+$ en $-$ zoals je ze kent uit Scheme.

Wat volgt zijn twee versies van de Fibonacci functie in Scheme:

```
1 (define (fib1 n)
2   (if (< n 2)
3       1
4       (+ (fib1 (- n 1))
5          (fib1 (- n 2)))))
```

```
1 (define (fib2 n)
2   (define (fib fib-i fib-i+1 i)
3     (if (< i n)
4         (fib fib-i+1 (+ fib-i fib-i+1) (+ i 1))
5         fib-i))
6   (fib 1 1 0))
```

- 3.a)** Welke van de twee versies zal het minste geheugen nodig hebben om het 100ste Fibonacci getal uit te rekenen en waarom? (*kort antwoord, maximaal één zin*)
- 3.b)** Vertaal deze "geheugenvriendelijke" versie van Scheme naar registtermachinecode. Vermijd overbodige instructies.

4 Automatisch geheugenbeheer

Beschouw de Scheme versie van het stop-and-copy garbage collection algoritme uit de slides. In de procedure `gc-loop` worden de `car`-waarde en `cdr`-waarde op adres `scan` in het "nieuwe" geheugen met behulp van `vector-set!` aangepast.¹ Wanneer deze waarde geen pointer naar een `cons`-cel voorstelt, zijn deze operaties echter overbodig.

- 4.a) Leg uit waarom deze operaties overbodig zijn wanneer de waarde op `scan` geen pointer naar een `cons`-cel voorstelt.
- 4.b) Herschrijf de code van `relocate-old-result-in-new` en `gc-loop` om dit te verbeteren.
- 4.c) Heeft het zin een analoog initiatief te nemen wanneer de `car`-waarde (`cdr`-waarde) op adres `scan` wijst naar een `broken-heart`?

5 Compilatie

- 5.a) Toon met twee korte fragmenten uit de implementatie van de compiler aan dat Scheme een taal is met "lexical scoping".
- 5.b) Pas de compiler zo aan dat ze code genereert voor een variant van Scheme met "dynamic scoping".

¹Dit aan de hand van het Scheme equivalent van subroutines `update-car` en `update-cdr` uit de registermachinecode versie.