TÖL304G
Forritunarmál
Verkefnablað 4

Snorri Agnarsson

10. september 2024

Verkefni — Assignments

Þið skuluð prófa öll verkefnin ykkar í einhverju Scheme kerfi, til dæmis DrRacket, og sýna niðurstöður.

Hér eru nokkur Scheme föll og lykilorð sem mögulegt er að þið viljið nota til að leysa þessi verkefni: define, lambda, if, and, or, car, cdr, cons, null?, list, =, *, +.

Athugið að Scheme report[1] inniheldur lýsingar á öllum þessum föllum og lykilorðum og einnig má finna nákvæma skjölun í fylgigögnum fyrir DrRacket og önnur Scheme kerfi.

Athugið að halaendurkvæmni má vera þannig að viðkomandi fall noti hjálparfall til að leysa verkefnið og að hjálparfallið sé halaendurkvæmt.

You should test all your solutions in some Scheme system, for example DrRacket, and show results.

Here are a few Scheme functions and keywords you might possibly want to use for solving these problems: define, lambda, if, and, or, car, cdr, cons, null?, list, =, *, +.

Note that the Scheme report[2] contains descriptions of all these functions and keywords and you can also find precise documentation in the documentation for DrRacket and other Scheme systems.

Note that tail recursion can be achieved by using a tail recursive helper function.

[1] http://www.hi.is/~snorri/downloads/r5rs.pdf

[2] http://www.hi.is/~snorri/downloads/r5rs.pdf

Hópverkefni — Group Assignments

1. Klárið að forrita *halaendurkvæmt* Scheme fall `myiota` miðað við eftirfarandi beinagrind. Athugið að faldaða fallið `hjalp` þarf að vera halaendurkvæmt.

   Finish programming a *tail recursive* Scheme function `myiota` according to the following skeleton. Note that the nested function `hjalp` needs to be tail recursive.

```
;; Notkun: (myiota n)
;; Fyrir:  n er heiltala, n>=0
;; Gildi:  Listi allra heiltalna i, þannig að
;;         0 < i <= n, í vaxandi röð,
;;         þ.e. listinn (1 2 ... n)

;; Use:    (myiota n)
;; Pre:    n is an integer, n>=0
;; Value: The list of all integers i, such that
;;        0 < i <= n, í ascending order,
;;        i.e. the list (1 2 ... n)
(define (myiota n)
  ;; Notkun: (hjalp r x)
  ;; Fyrir:  r er heiltala, 0 <= r <= n.
  ;;         x er listinn (r+1 r+2 ... n)
  ;; Gildi:  Listinn (1 2 ... n)

  ;; Use:    (hjalp r x)
  ;; Pre:    r is an integer, 0 <= r <= n.
  ;;         x is the list (r+1 r+2 ... n)
  ;; Value:  The list (1 2 ... n)
  (define (hjalp r x)
    ...
  )
  (hjalp ...)
)
```

   Til dæmis skal Scheme segðið `(myiota 0)` skila `()` og Scheme segðin `(myiota 5)` skal skila `(1 2 3 4 5)`. Sýnið prófanir, eins og endranær.

   For example the Scheme expression `(myiota 0)` should return `()` and the Scheme expression `(myiota 5)` should return `(1 2 3 4 5)`. Show tests as always.

2. Skrifið *halaendurkvæmt* Scheme fall `myfoldl` sem uppfyllir eftirfarandi lýsingu.

Write a *tail recursive* Scheme function `myfoldl` that fulfills the following description.

```
;; Notkun: (myfoldl f u x)
;; Fyrir:  f er tvíundarfall, þ.e. fall
;;         sem tekur tvö viðföng af einhverju
;;         tagi, x er listi (x1 ... xN)
;;         gilda af því tagi, u er gildi
;;         af því tagi.
;; Gildi:  (f (f ...(f (f u x1) x2) ...) xN)
;; Aths.:  Með öðrum orðum, ef við skilgreinum
;;         tvíundaraðgerð ! með a!b = (f a b),
;;         þá er útkoman úr fallinu gildið á
;;           u ! x1 ! x2 | ... ! xN
;;         þar sem reiknað er frá vinstri til
;;         hægri

;; Use:    (myfoldl f u x)
;; Pre:    f is a binary funtion, i.e. a function
;;         that takes two arguments of some type,
;;         x=(x1 ... xN) is a list of values of
;;         that type, u is a value of that type.
;; Value: (f (f ...(f (f u x1) x2) ...) xN)
;; Note:   In other words, if we define a binary
;;         operation ! with a!b = (f a b), then
;;         then the value returned is the value of
;;           u ! x1 ! x2 | ... ! xN
;;         where we compute from left to right
```

Til dæmis skal segðin `(myfoldl + 3 (list 1 2))` skila tölunni 6, þ.e. $3 + 1 + 2$, reiknað frá vinstri. Athugið að það er nauðsynlegt að tryggja að röð reikniaðgerða sé nákvæmlega rétt þannig að, til dæmis, segðin `(myfoldl - 3 (list 1 2))` skili réttu gildi samkvæmt lýsingunni að ofan, þ.e. útkomunni úr segðinni `(- (- 3 1) 2)`, sem er `0`. Til hliðsjónar er gagnlegt að kíkja á Dafny föllin `Fold_L` og `FoldL_loop` sem eru í skránni `Analogues.dfy` í Canvas.

Munið að halaendurkvæmt fall er hliðstætt lykkju.

Ekki má skrifa nein hjálparföll, enda eru þau algerlega ónauðsynleg í þessu tilviki.

Sýnið prófun á segðunum

```
(myfoldl - 3 '(1 2))
```

og

```
(myfoldl (lambda (a b) (cons b a)) '() '(1 2 3)).
```

For example the expression `(myfoldl + 3 (list 1 2))` should return the number 6, i.e. $3 + 1 + 2$, computed from left. Note that it is necessary to ensure that the sequence of operations is exactly correct so that, for example, the expression `(myfoldl - 3 (list 1 2))` returns the correct value according to the description above, i.e. the result from the expression `(- (- 3 1) 2)`, which is `0`. For reference it is useful to look at the Dafny functions `Fold_L` and `FoldL_loop` which are in the file `Analogues.dfy` in Canvas.

Remember that tail recursion is analogous to a loop.

No helper functions should be written, they are completely unnecessary in this case.

Show tests for the expressions

```
(myfoldl - 3 '(1 2))
```

and

```
(myfoldl (lambda (a b) (cons b a)) '() '(1 2 3)).
```

3. Notið föllin að ofan (`myiota` og `myfoldl`) til að skrifa tvær Scheme segðir til að reikna summu og margfeldi talnanna $1, \ldots, 30$.

   Use the functions above (`myiota` and `myfoldl`) to write two Scheme expressions to compute the sum and the product of the numbers $1, \ldots, 30$.

Einstaklingsverkefni — Individual Assignments

1. Skrifið halaendurkvæmt Scheme fall `sum1` sem uppfyllir eftirfarandi lýsingu.

   Write a tail recursive Scheme function `sum1` that fulfills the following description.

```
;; Notkun: (sum1 n)
;; Fyrir:  n er heiltala, n>=0
;; Gildi:  Summan 0+1+...+n

;; Use:    (sum1 n)
;; Pre:    n is an integer, n>=0
;; Value: The sum 0+1+...+n
```

2. Skrifið halaendurkvæmt Scheme fall `sum2` sem uppfyllir eftirfarandi lýsingu.

Write a tail recursive Scheme function `sum2` that fulfills the following description.

```
;; Notkun: (sum2 i n)
;; Fyrir:  i og n eru heiltölur, i <= n+1
;; Gildi:  Summan i+(i+1)+...+n, summa þeirra
;;         heiltalna k þannig að i <= k <= n.

;; Use:   (sum2 i n)
;; Pre:   i and n are integers, i <= n+1
;; Value: The sum i+(i+1)+...+n, the sum of the
;;        integers k such that i <= k <= n.
```

Athugið að segðin `(sum2 11 10)` verður að skila 0.

Note that the expression `(sum2 11 10)` must return 0.

3. Skrifið Scheme fall `sum3` sem uppfyllir eftirfarandi lýsingu.

   Write a Scheme function `sum3` that fulfills the following description.

```
;; Notkun: ((sum3 i) n)
;; Fyrir:  i og n eru heiltölur, i <= n+1
;; Gildi:  Summan i+(i+1)+...+n

;; Use:   ((sum3 i) n)
;; Pre:   i and n are integers, i <= n+1
;; Value: The sum i+(i+1)+...+n
```

   Athugið að segðin `((sum3 11) 10)` verður að skila 0.

   Note that the expression `((sum3 11) 10)` must return 0.

4. Skrifið halaendurkvæmt fall `reviota`, ásamt lýsingu, sem tekur heiltölu $n \geq 0$ sem viðfang og skilar listanum $(n \ \ldots \ 3\ 2\ 1)$. Þið munuð þurfa halaendurkvæmt hjálparfall, svipað og í `myiota` að ofan. Athugið að hjálparfall sem er faldað inn í annað fall getur notað breytur (viðföng) úr ytra fallinu. Þið þurfið að skrifa lýsingu (Notkun/Fyrir/Gildi) bæði fyrir `reviota` og fyrir hjálparfallið.

   Write a tail recursive function `reviota`, with a description, that takes an integer $n \geq 0$ argument and returns the list $(n \ \ldots \ 3\ 2\ 1)$. You will need a tail recursive helper function, similar to `myiota` above. Note that a helper function that is nested inside another function can access variables (arguments) in the outer function. You need to write a description (Use/Pre/Value) for both `reviota` and for the helper function.