Handbók fyrir Schmorpho

Hörður Freyr Yngvason

23. apríl 2010

1 Inngangur

Fyrir áfangann Þýðendur (TÖL202M), vorönn 2010, átti hanna forritunarmál og búa til þýðanda frá málinu yfir í Morpho þulu. Afraksturinn var málið *Schmorpho*, (nánast) hlutmengi af **Sch**eme í **Morpho**.

Morpho þulan er forritunarmál m.a. sniðið í kring um vakningarfærslur til að keyra á Morpho vélinni [1]. Innbyggður stuðningur er við halaendurkvæmni og lokanir, sem gerir það að verkum að vélin er kjörin í keyrsluhluta fallsforritunarmáls. Markmið mitt í verkefninu var fyrst og fremst að *eyða sem minnstri vinnu í þýðandann, en fá um leið sem öflugast forritunarmál.* Það ætti því ekki að koma á óvart að stefnan hafi verið tekin á hlutmengi í Scheme. Nánar tiltekið einskorðaði ég mig við lykilorðin `if` og `define` en reyndi á móti að útfæra þau sem best.

1.1 Dæmi um forrit

Eftirfarandi kóða má þýða í keyrslueiningu með aðalfall `main`. Hjálparföll á borð við `+`, `++`, `print` og `%` eru flutt inn úr `BASIS` einingunni í Morpho.

```
; keyrslufall einingarinnar
(define (main)
  ; skilgreiningar með hliðarverkunum
  (define m1 (print (++ "fib " (fib 100))))
  (define m2 (print (++ "gcd " (gcd 10 100))))
  ; erum búin
  (exit 0))

; stærsti samdeilir náttúrlegra talna a og b
(define (gcd a b)
  (if (== b 0)
    a
    (gcd b (% a b))))

; n-ta Fibonacci talan
(define (fib n)
  ; halaendurkvæmt innra fall fyrir ítrun
  (define (iter m a b)
    (if (== m n)
      b
```

```
    (iter (+ m 1) b (+ a b))))
(iter 1 1 1))
```

2 Notkun og uppsetning

Fyrir utan Morpho útgáfuna [1] samanstendur Schmorpho af skránum í töflu 1. Til að þýða þulusmiðinn má nota nýlega útgáfu af GCC, Flex og Bison, síðan

Skrá	Hlutverk
`lesgrein.l`	Lesgreinir skrifaður í flex[3]
`thattari.y`	þáttari skrifaður í Bison [2]
`CodeGen.h/cpp`	Klasasafn fyrir milliþulu
`schmorpho-asm`	þulusmiðurinn. þarf að þýða sérstaklega.
`schmorpho`	Einföld, stillanleg Bash-skripta sem tengir saman Schmorpho þulusmiðinn og Morpho þýðandann
`makefile`	`make`-skrá sem notar GCC, flex og Bison [4, 3, 2] til að framleiða þulusmiðinn, þ.e. forritið `schmorpho-asm`

Tafla 1: Lykilskrárnar í Schmorpho

nægir að nota `make`-skrána í aðalmöppunni. Þá fæst keyrsluskráin `schmorpho-asm` sem er þulusmiðurinn fyrir Schmorpho.

2.1 Þulusmiðurinn `schmorpho-asm`

Þulusmiðurinn les Schmorpho-kóða af aðalinntaki og skrifar þulu á aðalúttak. Hægt er að þýða Schmorpho forritsskrá `skrá.schmo` á þrjá vegu

1. `schmorpho-asm < skrá.schmo`

   Skrifar hráa þulu fyrir föllin í skránni án þess að setja þau í einingu.

2. `schmorpho-asm nafn < skrá.schmo`

   Skrifar þulu fyrir hringtengda einingu með nafnið `nafn.mmod` og flytur inn `BASIS` en er án keyrslufalls.

3. `schmorpho-asm nafn aðalfall < skrá.schmo`

   Skrifar þulu fyrir hringtengda keyrslueiningu sem flytur inn `BASIS`, hefur nafnið `nafn.mexe` og keyrslufallið `aðalfall`, sem verður að vera til staðar í `skrá.schmo`.

3 Málfræði

3.1 Frumeiningar málsins

3.1.1 Sértákn

Einu sértáknin eru svigar. Við notum aðeins venjulega sviga en tökum líka frá hornklofa og slaufusviga til að fyrirbyggja ruglingsleg nöfn.

3.1.2 Lykilorð

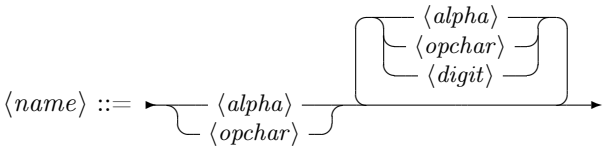
Í málinu eru aðeins tvö lykilorð, `define` og `if`. Við notum `define` til að skilgreina öll *nöfn*, en `if` er notað til að tákna val á milli tveggja ólíkra segða byggt á niðurstöðu samanburðarsegðar.

3.1.3 Athugasemdir

Allt fyrir aftan `;` í línu telst vera athugasemd. Athugasemdir eru hunsaðar við þýðingu.

3.1.4 Nöfn

Nöfn ⟨*name*⟩ eru notuð til að vísa í föll og önnur gildi. Þau mega samanstanda af ýmis konar sértáknum, bókstöfum og tölustöfum, en mega ekki hefjast á tölustaf.



⟨*alpha*⟩ ::= allir íslenskir stafir

⟨*opchar*⟩ ::= ‘`+`’ | ‘`-`’ | ‘`*`’ | ‘`/`’ | ‘`%`’ | ‘`$`’ | ‘`!`’ | ‘`_`’ | ‘`:`’ | ‘`|`’
| ‘`<`’ | ‘`=`’ | ‘`>`’ | ‘`^`’ | ‘`#`’ | ‘`&`’ | ‘`?`’ | ‘`@`’

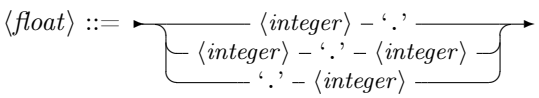
3.1.5 Frumstæð gildi

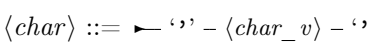
Hugmyndin er að nota nákvæmlega sömu frumstæðu gildin og Morpho þulan. Það eru Bool-gildi ⟨*bool*⟩, heiltölur ⟨*integer*⟩, fleytitölur ⟨*float*⟩, bókstafir ⟨*char*⟩ og strengir ⟨*string*⟩ og ‘`null`’. Táknum öll frumstæð gildi einu nafni ⟨*value*⟩. Bool-gildi eru ‘`true`’ og ‘`false`’; fleytitölur eru í tugakerfi, strengir eru safn stafa afmarkað af tvöföldum gæsalöppum og bókstafir eru stakir stafir innan einfaldra gæsalappa; ‘`null`’ er tóma tilvísunin í Morpho. Nánar tiltekið höfum við

⟨*value*⟩ ::= ⟨*bool*⟩ | ⟨*integer*⟩ | ⟨*float*⟩ | ⟨*char*⟩ | ⟨*string*⟩ | ‘`null`’

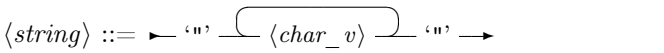
⟨*bool*⟩ ::= ‘`true`’ | ‘`false`’

⟨*digit*⟩ ::= 0 | 1 | 2 | 3 | 5 | 6 | 7 | 8 | 9

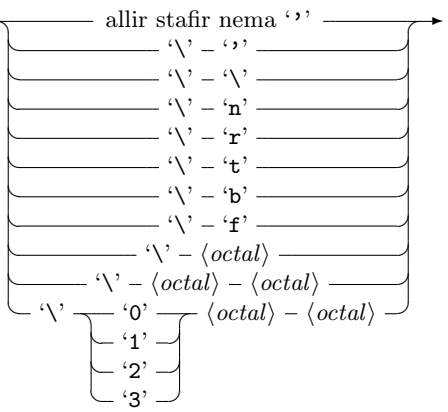




⟨*string*⟩ ::=



⟨*char_v*⟩ ::=



⟨*octal*⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

3.2 Mállýsing

Hér má sjá sömu mállýsingu og þáttarinn notar, nema á hreinu BNF-formi og án hliðarverkana sem þulusmiðin felur í sér.

⟨*program*⟩ ::= ϵ | ⟨*program*⟩ ⟨*defun*⟩

⟨*defun*⟩ ::= '`(`' '`define`' '`(`' ⟨*name*⟩ ⟨*args*⟩ '`)`' ⟨*body*⟩ '`)`'

⟨*defvar*⟩ ::= '`(`' '`define`' ⟨*name*⟩ ⟨*expr*⟩ '`)`'

⟨*args*⟩ ::= ϵ | ⟨*args*⟩ ⟨*name*⟩

⟨*body*⟩ ::= ⟨*expr*⟩
| ⟨*defuns*⟩ ⟨*expr*⟩
| ⟨*defvars*⟩ ⟨*expr*⟩
| ⟨*defuns*⟩ ⟨*defvars*⟩ ⟨*expr*⟩

⟨*defuns*⟩ ::= ⟨*defun*⟩ | ⟨*defuns*⟩ ⟨*defun*⟩

⟨*defvars*⟩ ::= ⟨*defvar*⟩ | ⟨*defvars*⟩ ⟨*defvar*⟩

⟨*exprs*⟩ ::= ϵ | ⟨*exprs*⟩ ⟨*expr*⟩

⟨*expr*⟩ ::= ⟨*cexrp*⟩ | ⟨*vexpr*⟩

⟨*cexpr*⟩ ::= '`(`' ⟨*name*⟩ ⟨*exprs*⟩ '`)`'
| '`(`' ⟨*cexpr*⟩ ⟨*exprs*⟩ '`)`'
| '`(`' '`if`' ⟨*expr*⟩ ⟨*expr*⟩ ⟨*expr*⟩ '`)`'

⟨*vexpr*⟩ ::= ⟨*name*⟩ | ⟨*value*⟩

4 Merking málsins

4.1 Schmorpho forrit

Schmorpho forrit, ⟨*program*⟩, er runa skilgreininga á föllum, ⟨*defun*⟩, sem mynda saman eina einingu í Morpho. Venjulega látum við eininguna vera hringtengda til að líkja eftir virkni Scheme, en ef engin viðföng eru send inn í þulusmiðinn fæst einfaldlega runa falla, sem er ekki sett inn í einingu (þ.e. ekki umlukin tvöföldum slaufusvigum).

4.2 Skilgreiningar

Skilgreiningar í mállýsingunni eru táknaðar með ⟨*defun*⟩ fyrir föll og ⟨*defvar*⟩ fyrir breytur. Skilgreining er ekki gildi, heldur gefur hún gildi eða útfluttu falli nafn. Skilgreiningin ⟨*defun*⟩ í ⟨*program*⟩ er t.d. skilgreining á útfluttu falli, en ⟨*defun*⟩ í stofni falls, ⟨*body*⟩, skilgreinir innra fall (lokun) í foreldri sínu.

4.2.1 Skilgreiningar á föllum

Föll eru skilgreind með

⟨*defun*⟩ ::= ‘`(`’ ‘`define`’ ‘`(`’ ⟨*name*⟩ ⟨*args*⟩ ‘`)`’ ⟨*body*⟩ ‘`)`’

hvort sem þau standa fyrir innra eða ytra fall. Munurinn á innri og ytri föllum er sá, að nöfn innri falla eru líka breytur með gildi, sem er lokunin fyrir fallið, á meðan nöfn ytri falla standa ekki fyrir nein gildi (sjá nánar í hluta 4.4). Í skilgreiningu falls er ⟨*name*⟩ nafn þess, ⟨*args*⟩ listi nafna viðfanga þess (núll eða fleiri) og ⟨*body*⟩ stofn þess. Stofninn er þannig, að fyrst koma skilgreiningar allra innri falla, síðan allar breytuskilgreiningar, og loks ein segð, sem er skilagildi fallsins, þ.e.

⟨*body*⟩ ::= ⟨*defun*⟩ ⟨*defvar*⟩ ⟨*expr*⟩

Jafngild, en aðeins flóknari lýsing er í mállýsingunni (3.2), þar eð taka þurfti tillit til viðvarana frá Bison um árekstra.

4.2.2 Breytuskilgreiningar

Skilgreiningar af gerðinni

⟨*defvar*⟩ ::= ‘`(`’ ‘`define`’ ⟨*name*⟩ ⟨*expr*⟩ ‘`)`’

skilgreina breytunafnið ⟨*name*⟩ þannig að það standi fyrir gildi segðarinnar ⟨*expr*⟩. Þær geta ekki verið endurkvæmar.

4.3 Sýnileiki nafna

Innri föll eru líka kölluð *földuð föll* og fjöldi skilgreininga utan um faldað fall er *földunarhæð þess*. Í skilgreiningu falls lítum við svo á, að viðföngin ⟨*args*⟩ og innihald stofnsins ⟨*body*⟩ séu á sömu földunarhæð og einni földunarhæð fyrir neðan nafn fallsins, ⟨*name*⟩. Ystu föll eru á földunarhæð 0 og við lítum svo á að þar séu engin nöfn. Við leyfum skilgreiningu nafns, svo lengi sem það er ekki þegar til á földunarhæð sinni, en nöfn eru tekin frá í þeirri röð sem þau koma

fyrir á hverri földunarhæð. Aftur á móti er leyfilegt að endurskilgreina nafn af ytri földunarhæð, en þá er engin leið að nálgast það aftur á þeirri og innri hæðum. [1]

Örlítið flóknari (en afskaplega eðlilegar) reglur gilda, um notkun nafna. Í hvert sinn sem við sjáum vísað í nafn, þá lítum við svo á að nafnið standi fyrir *nálægustu sýnilegu skilgreiningu nafnsins.*

1. Við sjáum ekkert af innri hæðum.
2. Nöfn viðfanga, þ.e. ⟨*args*⟩, eru sýnileg alls staðar á sinni földunarhæð
3. Öll föll, sem skilgreind eru á sömu földunarhæð með skilgreiningu af gerðinni ⟨*defun*⟩ eru sýnileg alls staðar á sinni földunarhæð, sér í lagi sín á milli.
4. Aðrar breytur, þ.e. allt skilgrent í ⟨*defvar*⟩ hluta stofns, eru sýnilegar í þeirri röð sem þær eru skilgreindar en ósýnilegar sjálfum sér og öllu á undan sér.
5. Ef nafn er sýnilegt á földunarhæð tilvísunarinnar með tilliti til uppruna hennar (úrskurðað með 1-4), þá er sú skilgreining nálægust. Annars lítum við á umlykjandi fall sem *uppruna* tilvísunarinnar og leitum aftur á földunarhæð þess. Ef við lendum á földunarhæð 0, þá þýðir það að nafnið er innflutt og verður að vera notað sem fall (þ.e. ekki gildi).

Reglur 3 og 5 gefa saman að föll á sömu földunarhæð geta verið innbyrðis endurkvæm, sér í lagi getur fall kallað endurkvæmt á sjálft sig. Aftur á móti geta breytuskilgreiningar af gerðinni ⟨*defvar*⟩ ekki verið endurkvæmar.

4.4 Gildi og segðir

Nú þegar við höfum rætt um skilgreiningar, þá er nauðsynlegt að vita hvað telst til gilda í málinu. Gildi og segðir eru sami hluturinn í Schmorpho. Þ.e. allar segðir eru gildi, og öll gildi eru segðir. Við táknum segðir með ⟨*expr*⟩ í mállýsingunni og skiptum þeim upp í tvo flokka, ⟨*vexpr*⟩ og ⟨*cexpr*⟩. Við getum hugsað okkur að ⟨*cexpr*⟩ séu segðir sem þarf að reikna og ⟨*vexpr*⟩ séu gildi sem búið er að reikna. Meginástæðan fyrir þessari flokkun er þó sú, að kall á fall getur verið tvenns konar, þ.e. kall á innflutt fall eða kall á lokun. Það þýðir að fyrir kall af gerðinni ⟼ '`(`' – ⟨*name*⟩ – ⟨*exprs*⟩ – '`)`' ⟶ þurfum við að kanna hvort ⟨*name*⟩ er innflutt nafn. Með flokkuninni á ⟨*expr*⟩ getum við merkt þessi vafatilfelli strax í þáttun, sem getur einfaldað þýðingu til muna.

4.4.1 Frumstæð gildi

Frumstæð gildi ⟨*value*⟩, þ.e. bool-gildi, heiltölur, fleytitölur, strengir og stafir ásamt `null` eru sér í lagi gildi/segðir. Sjá nánar í hluta 3.1.

[1] Reyndar er ekkert gert til að fyrirbyggja tvískilgreiningu nafns á sömu földunarhæð. Tilvísunin týnist í breytutöflu *þýðandans*, en gildið lifir enn í tilheyrandi vakningarfærslu og *lekur minni*, þar sem ruslasafnarinn mun halda að það sé í notkun lengur en þarf.

4.4.2 Breytur

Breytur eru nöfn, ⟨*name*⟩, sem standa fyrir gildi. Allar breytur eru nöfn (sjá hluta 3.1), en sum nöfn eru ekki breytur (sjá nánar 4.4.4 og 4.4.5). Þótt við tölum um *breytur* þá bjóðum við ekki sérstaklega upp á neinar aðgerðir til að breyta gildinu sem breytunöfn standa fyrir. Mælst er til þess að forrita sem mest án slíkra hliðarverkana, en það þó ekki bannað og auðvelt að flytja inn föll sem valda hliðarverkunum.

4.4.3 If-segðir

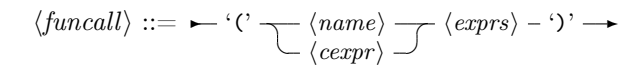
If-segðir í Schmorpho eru á forminu

⟨*ifexpr*⟩ ::= '`(`' '`if`' ⟨*cond*⟩ ⟨*then*⟩ ⟨*else*⟩ '`)`'

þar sem ⟨*cond*⟩, ⟨*then*⟩ og ⟨*else*⟩ eru ⟨*expr*⟩. Segðin ⟨*cond*⟩ er ákvarðanatökusegð: Ef útkoma hennar er önnur en `false` eða `null`, þá fær ⟨*ifexpr*⟩ gildið ⟨*then*⟩ en annars ⟨*else*⟩.

4.4.4 Köll á föll

Köll á föll gefa alltaf eitthvert gildi, þótt útkoman geti verið óskilgreind; t.d. látum við liggja á milli hluta hvert gildi segðarinnar `(print "Halló")` er. Köll á föll eru af gerðinni



þar sem ⟨*name*⟩ er nafn fallsins (lokun eða innflutt), ⟨*cexpr*⟩ er lokun fyrir fallið og ⟨*exprs*⟩ er runa þeirra gilda sem fallið tekur sem viðföng. Þetta þýðir sér í lagi að *öll viðföng eru gildisviðföng.* Við segjum að nafn eða fall sé *innflutt* ef það er notað eins og fall, en er hvergi skilgreint þar sem kallið getur *séð* það. Til eru óleyfileg köll, sem þó uppfylla málritið, t.d. getur útkoman úr ⟨*cexpr*⟩ verið `1`, sem er ekki fall. Slíkar villur koma ekki fram fyrr en hugsanlega í Morpho-hluta þýðingar/keyrslu.

Halaendurkvæmni Seinasta segðin í stofni falls er alltaf skilagildi þess, svo ef hún er kall á fall, þá má það kall vera halaendurkvæmt. Þetta gefur okkur skilvirka leið til að ítra án þess að nota lykkjur. Dæmi um þetta má sjá í greinum 1.1 og 5.

4.4.5 Lokanir

Í Schmorpho eru öll innri föll lokanir, en *innflutt föll eru það ekki.* Það þýðir að ekki er hægt að senda innflutt fall sem viðfang í annað fall, né láta innri breytu standa fyrir innflutt fall. Aftur á móti er hægt að búa til lokun, sem kallar á innflutt fall og láta innflutt fall skila lokun. T.d.

```
(define (kósínus)
  ;; Dæmi 1
  (define g cos)           ;; Rangt! cos er innflutt.
  (define (g x) (cos x))   ;; Rétt! g er lokun jafngild cos.
  ;; Dæmi 2
```

```
  (define (e fun) (fun x)) ;; Gildistökuvörpun
  (define h (e cos))       ;; Rangt! cos hefur ekki gildi
  (define h (e g))         ;; Rétt! g er lokun
  g)                       ;; skilum g, sem er lokun

;; Dæmi 3
(define (kósínus-af-0)
  (define f (kósínus)) ;; gildið er lokun sem er jafngild cos
  (f 0))
```

4.4.6 Hlutir og önnur gildi úr Morpho

Schmorpho er langt því frá að nýta allar mögulegar aðgerðir Morpho þulunnar. Til dæmis styður Morpho þulan hlutbundna forritun en ekki Schmorpho. Hins vegar væri hægt að flytja inn slíka hluti og fela þá á bak við föll, þannig að Schmorpho geti notið góðs af þeim.

5 Fleiri dæmi um forrit

5.1 Listaföll eins og í Scheme

Í venjulega Schmorpho notum við föllin í `BASIS` beint. Hér er dæmi um hvernig nota mætti aðeins eitt fall úr `BASIS`, nefnilega `==`, til þess að útfæra listaföllin `cons`, `car`, `cdr`, `map` og `foldl`. Við erum örlítið takmörkuð af því að hafa ekkert `list` fall; í staðinn notum við `cons` ítrekað og samsömum '`null`' við tóman lista.

```
;; Ein leið til að búa til listaföll í Schmorpho sem
;; hegða sér svipað og tilsvarandi Lisp/Scheme föll.

(define (null? x) (== null x)) ; er tómur?

; Gildið er lokun f þ.a. (f true) = x, (f false) = xs og (f null) = lengdin
; Krefjumst þess að xs sé svona par eða null, þar sem
; null stendur fyrir tóman lista.
(define (cons x xs)
  (define (get i) (if i x xs))
  get)

(define (car xs) (xs true))  ; fremsta stak lista
(define (cdr xs) (xs false)) ; hali lista
(define (length xs)          ; lengd lista
  (define (iter n ys)
    (if (null? ys)
      n
      (iter (+ n 1) (cdr ys))))
  (iter 0 xs))

; viðsnúinn listi
(define (reverse xs)
  (define (iter ys zs)
```

```
    (if (null? ys)
        zs
        (iter (cdr ys) (cons (car ys) zs))))
  (iter xs null))

; framkvæma fall f á öll stök lista xs
(define (map f xs)
  (define (iter ys zs)
    (if (null? ys)
      zs
      (iter (cdr ys) (cons (f (car ys)) zs))))
  (iter (reverse xs) null))

; framkvæma tvíundaraðgerð ítrekað með vinstri tengingu
(define (foldl f u xs)
  (if (null? xs) u (foldl f (f u (car xs)) (cdr xs))))

;; Lok útfærslu. Eftirfarandi eru prófunarföll

; Hjálparfall til að fá strengframsetningu lista
(define (show xs)
  (define (concat a b) (++ a (++ " " b)))
  (concat (foldl concat "(" xs) ")"))

; Skrifar strengina "( 1 2 3 4 )" og "( 1 4 9 16 )",
; hvorn í sína línuna, á aðalúttak.
(define (main)
  (define (sqr x) (* x x))
  (define xs (cons 1 (cons 2 (cons 3 (cons 4 null)))))
  (define ys (map sqr xs))
  (define m1 (print (show xs)))
  (define m2 (print (show ys)))
  (define m3 (writeln (length xs)))
  (exit 0))
```

Heimildir

[1] Snorri Agnarsson. Morpho þýðandinn og keyrsluumhverfið.
http://morpho.cs.hi.is/

[2] Bison - GNU parser generator.
http://www.gnu.org/software/bison/

[3] flex: The Fast Lexical Analyzer.
http://flex.sourceforge.net/

[4] GCC, the GNU Compiler Collection.
http://gcc.gnu.org/