Introduktion til programmering, ugeseddel 7

Version 1.1

17. oktober 2014

Den syvende undervisningsuge handler om *moduler*.

1 Plan for ugen

Mandag

Strukturer og signaturer, separat oversættelse.

Pensum: HR: 11.1-11.4. Moscow ML Owner's Manual afsnit 1 og 5-7 (findes i Absalon under "Undervisningsmateriale").

Tirsdag

Strukturer, signaturer og funktorer.

Pensum: HR: 11.5-11.7.

Fredag

Repetition af ugens pensum. Se ovenfor.

Bemærk: Da eksamen ligger i kursusuge 8, er det ikke muligt at genaflevere ugeopgave 7. Desuden er afleveringsfristen for den individuelle opgave allerede lørdag aften i stedet for søndag aften.

2 Mandagsopgaver

Emner: Strukturer og signaturer.

Vi har i tidligere uger (uge 4 og 5) brugt to forskellige typer til at repræsentere farver. Vi vil nu generalisere dette med SML's modulsystem.

Et farvemodul skal definere følgende elementer:

1. en type `colour`, der bruges til at repræsentere farveværdier.
2. Værdier `black : colour`, `white : colour`, `red : colour`, `green : colour`, `blue : colour`, `cyan : colour`, `magenta : colour`, `yellow : colour`, der definerer primærfarverne.
3. En funktion `complement : colour -> colour`, der returnerer *komplementærfarven* til en farve. Se `http://en.wikipedia.org/wiki/Complementary_colors#The_RGB_model` for en forklaring af komplementærfarver.
4. En funktion `toRGB : colour -> int * int * int`, der kan konvertere en farve til et tripel af heltal i intervallet 0-255, der repræsenterer farven som en RGB-værdi (ligesom i `InstagraML`).
5. En funktion `fromRGB : int * int * int -> colour`, der kan konvertere et RGB-tripel (tre heltal i intervallet 0-255) til en farve.

Mandagsopgaverne omhandler farvemoduler af denne slags. Alle signaturer og strukturer skrives i en enkelt `.sml` fil (altså som en toplevel-mode compilation unit, som beskrevet i Moscow ML Owner's Manual afsnit 5 og 7).

7M1 Brug beskrivelsen herover af farvemoduler til at skrive en signatur for et farvemodul. Signaturen skal have navnet `Colour`.

7M2 Skriv en struktur `RGB : Colour`, der implementerer farver som RGB-tripler, således at `toRGB` bliver identitetsfunktionen. Bemærk, at vi bruger gennemsigtig (*transparent*) signaturmatching.

Farvekomplementering sker ved at hvert RGB-komponent komplementeres. Dette gøres ved at trække komponentens værdi fra 255. For eksempel er `(255, 155, 55) = complement (0, 100, 200)`.

7M3 Skriv en struktur `PrimaryColours :> Colours` (bemærk uigennemsigtig (*opaque*) signaturmatching), hvor farver er repræsenteret med den datatype, der blev brugt i uge5-opgaven:

```
datatype primaryColours = Black | Red | Green | Blue
                        | White | Cyan | Magenta | Yellow
```

Der skal gælde følgende:

- Funktionen `toRGB` skal implementeres som beskrevet i opgave 5G1.

- Funktionen `fromRGB` skal runde RGB-værdier mellem 0 og 127 (begge inklusive) til 0 og værdier mellem 128 og 255 til 255, således at hvert RGB-komponent har værdier, der enten er 0 eller 255, og derefter returnere den primærfarve, der med `toRGB` afbildes til dette RGB-tripel. Med andre ord, "rundes" en farve af til nærmeste primærfarve. **Vink:** Lav en hjælpefunktion `round : int -> int`, der afrunder et tal til enten 0 eller 255 som beskrevet herover.
- `complement` følger de almindelige regler for komplementærfarver. For eksempel er `Green = complement Magenta`.

7M4 Prøv at skrive `PrimaryColours.red = PrimaryColours.green`. Forklar resultatet.

Vi ønsker nu at definere udvidede moduler for farver med to ekstra operationer:

1. En funktion `+++ : colour * colour -> colour`, der kan lægge to farveværdier sammen.
2. En funktion `*** : real * colour -> colour`, der kan skalere en farveværdi.

7M5 Definer en signatur `ExtendedColour`, der udvider `Colour`. Brug `include Colour` til dette.

7M6 Definer en struktur `ExtendedRGB : ExtendedColour`, der udvider `RGB` med de to operationer. Brug `open RGB` til dette.

Sørg for at RGB-værdier ligger i intervallet 0-255 selv efter addition og skalering af farver. Definer til dette en hjælpefunktion `constrain : int -> int`, der "skubber" tal ind i intervallet 0-255.

7M7 Definer en struktur `ExtendedPrimaryColours : ExtendedColour`, der udvider `PrimaryColours` med de to operationer.

Vink: Det kan være en fordel at konvertere farverne til RGB, behandle RGB-værdierne, og konvertere resultatet tilbage til en primærfarve. Du kan godt kalde funktioner fra `ExtendedRGB` til at gøre dette.

3 Tirsdagsopgaver

Emner: Strukturer og signaturer, separat oversættelse.

Det forventes, at du inden øvelserne tirsdag har forberedt dig på opgaverne ved at løse så mange som muligt på egen hånd.

Til forskel fra mandagsopgaverne bruger vi nu *structure mode compilation units* som beskrevet i Moscow ML Owner's Manual afsnit 5 og 6.

7T1 Skriv signaturen `Colour` fra mandagsøvelserne i en fil `Colour.sig`.

7T2 Fra en kommandolinje (brug evt. "Shell Command" fra menuen "Tools" i Emacs) oversæt denne med kommanden `mosmlc -c Colour.sig`.

7T3 Lav en fil `Colour.sml`, der indeholder en struktur `Colour :> Colour`, der implementere signaturen `Colour` som RGB-tripler (dvs. at den bortset fra navnet og uigennemsigtigheden er identisk med strukturen `RGB`).

7T4 Fra en kommandolinje (brug evt. "Shell Command" fra menuen "Tools" i Emacs) oversæt denne med kommanden `mosmlc -c Colour.sml`.

7T5 Start mosml med `mosml -P full`, men uden at give nogen fil eller buffer som inddata (brug "Start SML repl" fra SML menuen i Emacs). Skriv derefter kommandoen `load "Colour";` i mosml-kommandolinjen. Skriv derefter `Colour.red;` og observer svaret.

7T6 Skriv følgende kodelinjer:

```
structure Hello =
struct
  val _ = TextIO.output (TextIO.stdOut, "Hello World!\n");
end
```

i en fil `Hello.sml` og oversæt denne med kommandoen
`mosmlc Hello.sml -o Hello`. Hvis du bruger Windows, så brug i stedet
`mosmlc Hello.sml -o Hello.exe`.

7T7 Kald programmet `Hello` fra en kommandolinje. Hvis du er bruger Linux eller MacOS, så skriv `./Hello` på kommandolinjen, hvis du er i Windows, skriv `Hello`. **NB!** Sørg for at din kommandolinje bliver afviklet i det filkatalog, hvor filen `Hello` (eller `Hello.exe`) findes.

4 Fredagsopgaver

Emne: Funktorer.

Det forventes, at du inden øvelserne fredag har forberedt dig på opgaverne ved at løse så mange som muligt på egen hånd.

Fredagsopgaverne bygger videre på mandagsopgaverne og inddrager flere elementer fra uge5-opgaverne.

Alle funktorer, signaturer og strukturer skrives i en enkelt `.sml` fil (altså som en toplevel-mode compilation unit, som beskrevet i Moscow ML Owner's Manual afsnit 5 og 7). Som udgangspunkt bruges en fil med alle signaturer og strukturer fra mandagsopgaverne.

7F1 Skriv en funktor `Figure`, der er parameteriseret med en struktur `C`, der matcher signaturen `Colour`, og som implementerer følgende elementer:

1. En type `colour`, der er identisk med den farvetype, der er defineret i parameteren `C`.
2. En funktion `toRGB`, der er identisk med den funktion, der er defineret i parameteren `C`.
3. En type `figure`, der ligner den, der er beskrevet i opgavetemaet fra uge 5, pånær at den i stedet for at være polymorf i farven bruger farvetypen `colour`.
4. En funktion `colourOf : figure -> point -> colour option`, svarende til den funktion, der blev implementeret i opgave 5G3.
5. En funktion
   `toInstagraML : figure * point * int * int * real -> InstagraML.image`
   svarende til den funktion, der blev implementeret i opgave 5G6.

   Bemærk, at denne funktion selv skal konvertere fra `colour` til RGB.

7F2 Definer en struktur `RGBFigure` ved at anvende funktoren `Figure` på strukturen `RGB`.

7F3 Definer en struktur `PrimaryFigure` ved at anvende funktoren `Figure` på strukturen `PrimaryColours`.

7F4 Skriv en funktor `ExtendedFigure`, der er parameteriseret over en struktur `C : ExtendedColour`. Den skal implementere de samme typer og funktioner som `Figure`, pånær at datatypen `figure` implementerer en ekstra konstruktor `Blend of real * figure * figure`.

Ideen er, at `Blend` (b, f, g) blander farverne fra de to figurer f og g i forholdet b, hvor $0 < b < 1$ efter følgende regler:

1. Hvis et punkt ligger i den ene figur, men ikke i dem begge, har det farven fra denne figur.
2. Hvis et punkt ligger i begge figurer, farven fra f er c_f og farven fra g er c_g, gives farven `+++(***(`b`,` c_f`), ***(1−`b`,` c_g`))`.

7F5 Definer en struktur `ExtendedRGBFigure` ved at anvende funktoren `ExtendedFigure` på strukturen `ExtendedRGB`.

7F6 Definer en struktur `ExtendedPrimaryFigure` ved at anvende funktoren `Figure` på strukturen `ExtendedPrimaryColours`.

5 Opgavetema:

Ugens opgavetema er *multimængder*. En *multimængde* er en variant af mængder, hvor antallet af forekomster af et element har betydning. Derfor kan man ikke blot spørge, *om* et element findes i multimængden, men også *hvor mange gange* elementet findes. $\mathit{forekomster}(m,\ x)$ vil være antallet af forekomster af x i multimængden m. Dette vil altid være større end eller lig med 0.

Foreningsmængde, fællesmængde og mængdedifferens for multimængder defineres således:

- Foreningsmængden $m_1 \uplus m_2$ af to multimængder m_1 og m_2 er en multimængde, der indeholder elementerne fra m_1 og m_2 lige så mange gange, som de forekommer i m_1 og m_2 tilsammen. For eksempel er

$$\{1,2,2,3\} \uplus \{1,1,4\} \quad = \quad \{1,1,1,2,2,3,4\}$$

- Fællesmængden $m_1 \cap m_2$ af to multimængder m_1 og m_2 er en multimængde, der indeholder elementerne, der findes i både m_1 og m_2 lige så mange gange, som de forekommer i den af m_1 og m_2, hvor de forekommer færrest gange. For eksempel er

$$\{1,2,2,3\} \cap \{1,1,4\} \quad = \quad \{1\}$$

- Differensmængden $m_1 \setminus m_2$ af to multimængder m_1 og m_2 er en multimængde, der indeholder elementerne, der findes i m_1 flere gange end i m_2. Antallet af forekomster i differensmængden er da differensen mellem antal forekomster i m_1 og m_2. For eksempel er

$$\{1,2,2,3\} \setminus \{1,1,4\} \quad = \quad \{2,2,3\}$$

Dette kan udtrykkes med formlerne

$$\begin{array}{lcl} \mathit{forekomster}(m_1 \uplus m_2,\ x) & = & \mathit{forekomster}(m_1,\ x) + \mathit{forekomster}(m_2,\ x) \\ \mathit{forekomster}(m_1 \cap m_2,\ x) & = & \min(\mathit{forekomster}(m_1,\ x),\ \mathit{forekomster}(m_2,\ x)) \\ \mathit{forekomster}(m_1 \setminus m_2,\ x) & = & \max(0,\ \mathit{forekomster}(m_1,\ x) - \mathit{forekomster}(m_2,\ x)) \end{array}$$

max-operationen i ligningen for multimængdedifferens er for at undgå et negativt antal forekomster.

Se evt. også `http://en.wikipedia.org/wiki/Multiset` og `http://theory.stanford.edu/~arbrad/pivc/sets.pdf`. Bemærk dog, at sidstnævnte bruger $-$ i stedet for $\setminus$ til at angive multimængdedifferens.

En mulig signatur for multimængder er:

```
signature MSET =
sig
  type 'a mset  (* typen af multimængder med elementer af type 'a *)
  val multiplicity : ''a mset * ''a -> int (* antal forekomster af element *)
  val empty : 'a mset  (* den tomme multimængde *)
  val singleton : ''a -> ''a mset  (* laver multimængde med et element *)
  val union : ''a mset * ''a mset -> ''a mset (* foreningsmængde *)
  val intersect : ''a mset * ''a mset -> ''a mset (* fællesmængde *)
  val minus : ''a mset * ''a mset -> ''a mset (* mængdedifferens *)
end
```

6 Gruppeaflevering

Gruppeafleveringen obligatorisk. Alle delspørgsmål skal besvares. Opgaven afleveres i Absalon. Der afleveres en fil pr. gruppe, men den skal angive alle deltageres fulde navne i kommentarlinjer øverst i filen. Filens navn skal være af formen `7G-`*`initialer`*`.sml`, hvor initialer er erstattet af gruppemedlemmernes initialer. Hvis f.eks. Bill Gates, Linus Torvalds, Steve Jobs og Gabe Logan Newell afleverer en opgave sammen, skal filen hedde `7G-BG-LT-SJ-GLN.sml`. Brug gruppeafleveringsfunktionen i Absalon.

Gruppeopgaven giver op til 2 point, som tæller til de 20 point, der kræves for eksamensdeltagelse.

NB! Der er af tidshensyn *ikke* mulighed for genaflevering af ugeopgave 7.

Alle signaturer og strukturer skrives i den SML-fil, der afleveres. Denne fil skal kunne åbnes med `mosml -P full` uden fejl. Der bruges altså *toplevel mode compilation units*.

7G1 Kopier ovenstående signatur til SML-filen.

7G2 Skriv en struktur `Mset : MSET`, som implementerer en multimængde som en uordnet liste af elementer, hvor antallet af forekomster af et element i en liste angiver antallet af forekomster af elementet i multimængden.

7G3 Definer følgende multimængder:

```
val one = Mset.singleton 1
val two = Mset.singleton 2
val three = Mset.singleton 3
val four = Mset.singleton 4
val m1223 = Mset.union (one,
                        Mset.union(two,
                                   Mset.union (two, three)))
val m114 = Mset.union (one,
                       Mset.union(one,
                                  Mset.union (four, Mset.empty)))
```

Ser de ud som forventet?

7G4 Definer følgende multimængder:

```
val forening = Mset.union (m1223, m114)
val faelles = Mset.intersect (m1223, m114)
val minus = Mset.minus (m1223, m114)
```

Sammenlign resultaterne med eksemplerne i opgavetemaet.

7G5 Definer følgende værdier:

```
val forekomster1 = Mset.multiplicity (forening, 1)
val forekomster2 = Mset.multiplicity (faelles, 1)
val forekomster3 = Mset.multiplicity (minus, 1)
```

Er resultaterne som forventet?

Vi laver nu en signatur:

```
signature EMSET =
sig
  include MSET
  val twice : ''a mset -> ''a mset
end
```

der udvider `MSET` med en funktion `twice`, der fordobler forekomsten af alle elementer i en multimængde.

7G6 Skriv en funktor `UMset(mm : MSET) : EMSET`, der tager en multimængdestruktur og laver en udvidet multimængdestruktur.

7G7 Anvend `UMset` på strukturen `Mset` for at få en struktur `Eset`, der implementerer `EMSET`.

7 Individuel aflevering

Den individuelle opgave er obligatorisk. Alle delspørgsmål skal besvares. Opgaven afleveres i Absalon som en fil med navnet `7I-`*`navn`*`.sml`, hvor *navn* er erstatttet med dit navn. Hvis du fx hedder Anders A. And, skal filnavnet være `7I-Anders-A-And.sml`. Skriv også dit fulde navn som en kommentar i starten af filen.

Den individielle opgave giver op til 3 point, som tæller til de 20 point, der kræves for eksamensdeltagelse.

NB! Af tidshensyn er afleveringsfristen for den individuelle opgave allerede *lørdag aften* i stedet for søndag aften. Endvidere er der *ikke* mulighed for genaflevering af ugeopgave 7.

Alle signaturer og strukturer skrives i den SML-fil, der afleveres. Denne fil skal kunne åbnes med `mosml -P full` uden fejl. Der bruges altså *toplevel mode compilation units*.

7I1 Kopier signaturen `MSET` fra opgavetemaet til din SML-fil.

7I2 Skriv en struktur `MsetF :> MSET`, som implementerer en multimængde som en funktion fra elementer til antallet af forekomster af disse.

Din struktur skal blandt andet altså indeholde linjerne

```
type 'a mset = 'a -> int

fun multiplicity (m, x) = m x
    (* anvend multimængden på elementet *)

fun singleton x = fn y => if y = x then 1 else 0
    (* x har 1 forekomst, alle andre elementer har 0 *)

fun union (m1, m2) = fn x => m1 x + m2 x
    (* bruger formlen fra beskrivelsen i opgavetemaet *)
```

Vink: Brug formlerne vist i opgavetemaet for antal forekomster af x i $m_1 \cap m_2$ og $m_1 \setminus m_2$ til at definere `intersect` og `minus` på samme måde som `union` er defineret herover.

8 Ugens nød

De fleste kender nok spillet „Minestryger“ (`http://da.wikipedia.org/wiki/Minestryger_%28spil%29`).

Vi laver nu en variant af dette spil, hvor det for alle felter på forhånd er kendt, hvor mange bomber, der i alt er i de otte nabofelter samt feltet selv. Det g<E6>lder både felter med og felter uden bomber.

Et eksempel på et sådant spil er

```
221
232
232
```

Opgaven går nu ud på at finde ud af, i hvilke felter bomberne er. Svaret skal gives som en tekst, hvor tomme felter angives med `.` og bomber med `B`. Eksempelvis skal det ovenstående inddata give følgende uddata:

```
...
BB.
..B
```

Der er ikke altid en entydig løsning. For eksempel vil inddataet

```
11
11
```

have fire løsninger, blandt andet:

```
.B  og  B.
..      ..
```

Der er heller ikke altid løsninger, idet for eksempel

```
10
00
```

ikke har en løsning.

7N1 Lav i SML en funktion `minestryger : string -> string -> unit`, sådan at `minestryger` *in out* læser et spil fra filen *in* og skriver en løsning i filen *out*. Hvis der er flere mulige løsninger, er det ligegyldigt, hvilken af dem, der skrives til *out*. Hvis ingen løsning findes, kastes undtagelsen `Uloeselig`. Et spil består af m linjers tekst med hver n cifre fra 0 til 9. Hvis inddatafilen ikke har dette format, kastes undtagelsen `Domain`.

Et par eksempelspil er angivet herunder:

```
4421332
4543543
3433432
2434321
2312110

233322234320
233333335541
012333335652
112244324763
334455445763
554234444653
554233333543
332012221322
111013442211
001234442100
111233332100
111232111100

22122235444654334453
33133458654776546685
23144557754665457674
24343456644666556685
14454565657765567785
36554565668754556685
36454677766443556675
36454555654222333354
24344655553123433465
23345743354213332465
33224643365424443465
33113542244323444465
44102331344334445464
34223554444455545575
35432345666766565565
36553456667987653355
47553455656655564455
58553455644334442244
46432466755223442355
34310134544223320133
```

I bedømmelsen af løsninger, gives programmet større og større opgaver og stoppes når den samlede køretid overstiger et minut. Det program, der har løst flest opgaver inden da, vinder. Bemærk: Bedømmerens PC kører måske ikke helt så hurtigt som din.