Introduktion til programmering, ugeseddel 3

Version 1

13. september 2014

Den tredje undervisningsuge har til formål at gøre jer fortrolige med brug af funktioner som *værdier*. Ind til nu har vi adskilt funktioner fra primitive typer af værdier (f.eks. heltal og sandhedsværdier), men som vi vil se i denne uge kan funktioner også gemmes i lister, funktioner kan gives som argumenter til andre funktioner, og en funktion kan „konstruere" andre funktioner.

Derudover skal vi lære hvordan vi med *lokale erklæringer* kan give navne til mellemresultater i en funktion, og hvordan vi kan bruge det til at skjule hjælpefunktioner.

1 Plan for ugen

Mandag

Lokale erklæringer, funktioner som værdier og introduktion til ugesedlens tema.

Pensum: HR: 3.7 → IP-2: 4.5 → `3M-kvadratrod2-local.sml`[1] → IP-2: 10.1-10.2

Tirsdag

Listekombinatorer, anonyme funktioner og virkefelter.

Pensum:
IP-2: 10.3-10.5 → HR: 9.1-9.5.1 og 9.6 → *Intuition for* `foldl`[1]

Fredag

`foldl` vs. `foldr` og effektiv sortering

Pensum: HR: 9.5.2 → IP-2: 8

[1] Findes på Absalon

2 Opgavetema: Programmering med billeder

I denne uge skal I lære om hvordan funktioner kan bruges som værdier, til det formål har vi skrevet et lille SML-bibliotek til at manipulere billeder. I denne sektion vil vi forklare hvordan det bruges. Vi vil bruge billedet "torben.bmp"som udgangspunkt:

Figur 1: torben.bmp

Start med at indlæse biblioteket:

```
use "InstagraML.sml";
```

Nu kan vi indlæse en billedfil[2] fra harddisken (kun BMP, se fodnote):

```
val torben = InstagraML.readBMP "torben.bmp";
```

Det simpleste vi kan gøre ved et billede er at rotere det 90° i urets retning:

```
val torbenPaaSiden = InstagraML.clockwise torben;
```

For at se resultatet skal vi gemme billedet til en ny fil:

```
- InstagraML.writeBMP ("torbenPaaSiden.bmp", torbenPaaSiden);
> val it = () : unit
```

Vi kan nu finde filen `torbenPaaSiden.bmp` på harddisken:

Figur 2: torbenPaaSiden.bmp

Bemærk typen af `writeBMP`:

```
InstagraML.writeBMP : string * image -> unit
```

Returværdien er værdien `()`, da det ikke er selve resultatet af funktionen vi er interesseret i, men at den laver en ændring på vores harddisk.

[2]Billeder skal være i bitmap-formattet (dvs. „.bmp“-filer). Du kan konvertere billeder online vha. `http://www.imagemagick.org/MagickStudio/scripts/MagickStudio.cgi`

Farver og `colour`-typen

For at forstå hvad et billede er, skal vi vide hvordan en computer „regner“ med farver. En farve er en blanding af rød, grøn og blå farve. I SML repræsenterer vi blandingsforholdet mellem de tre farver som en trippel bestående af tre heltal mellem 0 og 255.

Eksempel:

```
val red     = (255,  0,  0);
val green   = (  0,255,  0);
val blue    = (  0,  0,255);
val magenta = (255,  0,255);  (* magenta = bland roed og blaa *)
```

Vi har introduceret typeforkortelsen `colour` for at gøre typerne nemmere at læse. At skrive `colour` betyder altså præcist det samme som at skrive `int * int * int`. Mos-ML vil dog blive ved med at skrive `int * int * int` når den udskriver typerne.

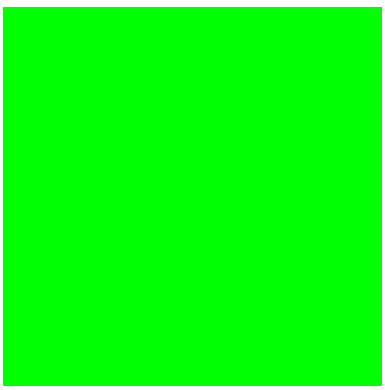
Hvad er et billede?

Et billede er en angivelse af hvilken farve, hver enkelt lille punkt på billedet har. Hvert sådan et punkt kaldes en pixel. Billedet af Torben der er blevet brugt ovenfor er 180 pixels i bredden og 180 pixels i højden, dvs. at der er $180 \times 180 = 32400$ pixels i billedet. For hver af disse punkter skal der gemmes en farve.

Vi behøver i denne opgave ikke præcist at vide hvordan et billede gemmes på harddisken, men blot at `readBMP` og `writeBMP` gør det for os.

Eksempel: For at lave et billede hvor alle pixels har samme farve kan vi bruge funktionen `InstagraML.pixel`, der givet en farve returnerer et billede med størrelse 1×1. Hvis man har behov for det kan man skalere 1-pixel billedet til de ønskede dimensioner. Se `InstagraML.scale` nedenfor.

```
- InstagraML.writeBMP ("green.bmp", InstagraML.pixel green);
> val it = () : unit
```



Figur 3: green.bmp skaleret til størrelse 512x512 pixels

2.1 InstagraML dokumentation

Til at starte med behøver I ikke forstå alt i denne sektion, gå i stedet i gang med opgaverne og slå op her undervejs.

`pixel : colour -> image`

Returnerer et billede med 1 pixel i bredden og 1 pixel i højden, af den angivne farve.

```
width : image -> int
height : image -> int
```

Returnerer henholdsvis bredden og højden af billedet i antal af pixels.

Eksempel:

```
- InstagraML.width torben;
> val it = 180 : int
- InstagraML.height (InstagraML.pixel green);
> val it = 1 : int
```

`clockwise : image -> image`

Roterer et billede 90° i urets retning.

Eksempel: `InstagraML.clockwise torben;`

`beside : image * image -> image`

Sætter to billeder ved siden af hinanden.

Eksempel: `InstagraML.beside (torben, torben);`

`scale : real -> real -> image -> image`

Skalerer størrelsen af et billede. Det første argument ganges på bredden af billedet, det andet argument ganges på højden af billedet.

Eksempel: `InstagraML.scale 2.0 0.5 torben;`

`recolour : (colour -> colour) -> image -> image`

Vi kan aflæse af typen af recolour at dens første argument er en funktion der ændrer en '`colour`' til en anden '`colour`'. Idéen er at vi kan „omfarve" et billede ved at give en funktion der fortæller hvordan farven af hver pixel skal ændres.

Eksempel:

```
(* Goer et billede mere blaat (mere brugervenligt) *)
fun makeBlue (r,g,b) = (r,g,255);
val blueTorben = InstagraML.recolour makeBlue torben;
```

2.2 Afprøvning

Opgaverne til denne uge der omhandler InstagraML-billeder er afprøvning ikke helt så lige til. Det er svært at konstruere det forventede output og InstagraML har pt. ikke en funktion til at sammenligne to billeder. Vi forventer derfor at en del af jeres afprøvning beror på *visuel inspektion* af resultaterne.

To ting er det dog muligt at skrive testtilfælde for: at de resulterende billeder har de forventede dimensioner og at jeres funktioner kaster undtagelser på de tidspunkter det er specificeret i opgaveteksten.

3 Mandagsopgaver

Mål: Blive endnu bedre til rekursion over lister. Intro til InstagraML biblioteket.

De følgende opgaver løses vha. funktionerne gennemgået i afsnit 2.1. Ingen af opgaverne i denne uge kræver at I forstår eller redigerer hvad der sker internt i InstagraML.sml

3M1 I denne opgave skal du skrive en funktion der summerer de tal i en liste der opfylder et givent kriterie. Som eksempel bruger vi kriteriet at *tallet skal være positivt*:

```
fun isOkay x = x > 0
```

Skriv funktionen `sumIfOkay : int list -> int` der summerer alle de tal i listen hvor `isOkay` returnerer `true`. Afprøv med andre definitioner af `isOkay`.

3M2 HR 5.14

3M3 Skriv en funktion:

```
counterClockwise : image -> image
```

Der roterer et billede 90° mod urets retning.

3M4 Skriv en funktion der placerer et billede oven på et andet, sådan så højden af det nye billede er summen af højden af de to argument-billeder. Første argument skal placeres over andet argument. Kast undtagelsen `Domain` hvis to billeder er af forskellig bredde.

```
above : image * image -> image
```

3M5 Skriv en rekursiv funktion:

```
imageConcat : image list -> image
```

Der sætter alle billederne i en liste ved siden af hinanden. Funktionen skal kaste undtagelsen `Empty` hvis listen er tom. Tirsdag vil vi se hvordan imageConcat kan skrives uden brug af rekursion.

3M6 Omskriv `counterClockwise` så den skrives uden brug af "`fun`", men som en værdierklæring.

```
val counterClockwise = ...
```

Hint: brug funktionssammensætning (f `o` g), forklaret i HR afsnit 9.6.

Hvis du har mere tid, gå da i gang med at forberede dig til tirsdagsøvelserne.

4 Tirsdagsopgaver

Mål: Videre med InstagraML og brug af listekombinatorerne map, filter og foldl.

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

3T1 Skriv en funktion der tager gennemsnittet af to farver (gennemsnittet af hvert komponent):

```
colourAverage : colour -> colour -> colour
```

3T2 Erklær en funktion der gør et billede rødere ved at for hver pixel tager gennemsnittet af dens nuværende farve og postkasserød, (255,0,0).

```
fadeRed : image -> image
```

3T3 Overvej hvorfor vi ikke har givet `colourAverage` typen:

```
colourAverage : colour * colour -> colour
```

3T4 Funktioner af typen `image -> image` kalder vi effekter, find flere eksempler i `Effects.sml`.

Skriv en funktion `iterate` der anvender den samme effekt n gange på et billede, og sætter alle mellemresultaterne ved siden af hinanden:

```
iterate : (image -> image) -> image -> int -> image
```

Eksempel: `iterate fadeRed torben 5;`

3T5 Omskriv funktionen `sumIfOkay` fra opgave 3M1, så `erOkay` gives som et argument: `sumIfOkay2 : (int -> bool) -> int list -> int`

3T6 Omskriv `imageConcat` fra 3M5, så den bruger `foldr` i stedet for rekursion.

3T7 HR 9.2(a)

5 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 `3G-`*`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 `3G-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. Genaflevering kan hæve pointtallet fra første aflevering med højest 1 point, så sørg for at gøre jeres bedste allerede i første aflevering.

Denne opgave handler om problemløsning, som det blev gennemgået tirsdag i uge 2. Se noterne „Hvordan man angriber et programmeringsproblem“ på Absalon.

Funktionen `InstagraML.beside` kan ikke håndtere at sætte to billeder ved siden af hinanden, der ikke er lige høje. I gruppeopgaven er det jeres opgave at skrive en funktion `safeBeside` der centrerer det laveste af de to billeder ved siden af det andet og indsætter sort farve i det tomrum der opstår over og under det laveste billede.

Eksempel: `safeBeside ralf torben`

3G1 *Forstå problemet:* Skriv (i en kommentar) de forskellige tilfælde der skal håndteres og beskriv hvordan en løsning ser ud i hvert tilfælde. Tænk blandt andet over hvad der skal ske når et af billederne har et ulige antal pixels i højden og det andet har et lige antal.

3G2 *Planlæg afprøvning:* Ud fra observationerne i opgave 3G1, skriv en plan for hvordan I vil afprøve `safeBeside`.

Opdel problemet i delproblemer: Vi kan opdele problemet i tre:

a) En funktion der laver et ensfarvet billede af en specificeret bredde og højde.

b) En funktion der sætter en sort bjælke over og under et billede, så det opnår den ønskede størrelse.

c) Selve funktionen `safeBeside`.

Eksempel: `padVertical 300 torben`

3G3 *Kommentarer før kode:* Vi har medvilje været lidt uklare om punkt (a) og punkt (b). Beskriv i en kommentar hvad de to funktioner skal gøre.

3G4 *Udfør planen, start med det nemme:* Skriv en funktion der udfører punkt (a) i planen:

```
solid : colour -> int * int -> image
```

3G5 *Udfør planen:* Skriv en funktion der udfører punkt (b) i planen:

```
padVertical : int -> image -> image
```

Hvis heltallet der angiver den *nye* højde er mindre end højden af billedet, skal funktionen rejse undtagelsen Domain. Det er kun meningen at funktionen skal kunne gøre et billede større, ikke mindre.

3G6 *Udfør planen:* Skriv funktionen

```
safeBeside : image * image -> image
```

Der placerer to billeder ved sidan af hinanden og sørger for at det mindste midterjusteres i forhold til det andet vha. `padVertical`.

6 Individuel aflevering

Den individuelle opgave er obligatorisk. Alle delspørgsmål skal besvares. Opgaven afleveres i Absalon som en fil med navnet `3I-`*`navn`*`.sml`, hvor *navn* er erstatttet med dit navn. Hvis du fx hedder Anders A. And, skal filnavnet være `2I-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. Genaflevering kan hæve pointtallet fra første aflevering med højest 1 point, så sørg for at gøre dit bedste allerede i første aflevering.

3I1 I filen `Effects.sml` finder du en række effekter, alle med typen `image -> image`. Skriv en rekursiv funktion der, en efter en, anvender alle transformationerne i en liste på et givet billedet og returnerer billedet hvor alle effekterne er anvendt. Den første effekt i listen anvendes først.

```
effects : (image -> image) list -> image -> image
```

Eksempel: `effects [clockwise, fadeRed, fadeRed] torben`

3I2 Man kan også anse funktionen `effects` som en funktion der sætter effekter sammen og returnerer en ny effekt. Det er måske nemmere at se hvis vi opsætter typen på denne måde:

```
effects : (image -> image) list ->
          (image -> image)
```

Hvad udskriver MosML hvis du spørger om typen af `effects`?

```
- effects;
```

Og hvorfor?

3I3 Vi kan repræsentere et klokkeslæt som antal minutter siden midnat. For eksempel svarer 120 til kl. 2:00, 0 er midnat og 1439 er kl. 23:59. Vi kan nu opskrive butikkers åbningstider som et par af to sådanne tidsangivelser:

```
val openHours =
      [(16*60,  4*60, "La Luna Pizzaria"), (* 16:00- 4:00 *)
       ( 8*60, 22*60, "Netto"),            (*  8:00-22:00 *)
       ( 0, 23*60+59, "Steno Apotek")]     (*  0:00-23:59 *)
```

Skriv en funktion `openStores xs t` der givet en liste af åbningstider `xs`, og et tidspunkt `t` returnerer alle de butikker der har åbent. Typesignatur:

```
openStores : (int * int * 'a) list -> int -> 'a list
```

Undtagelsen `Domain` rejses hvis t eller et af tidspunkterne i xs er ugyldigt.

Husk at tage højde for at tidspunkter kan krydse midnat (e.g. kl. 16:00 til 4:30) og at skrive udførlig afprøvning af funktionen.

7 Ugens nød

I denne uge er *Ugens nød* en kreativ udfordring. Vinderen af Ugens nød er den studerende der det program der producerer det *flotteste* billede i dimensionerne 512x512 pixels. I må maksimalt skrive 20 linjer SML-kode, af maks 80 tegn per linje. Blanke linjer og linjer kun med kommentarer fraregnes. Vi vil også se på kvaliteten af selve koden: valg af navne, kommentarer og brug af højereordensfunktioner frem for eksplicit rekursion.

Dommerkomitéen består alle undervisere og instruktorer på kurset.

Afleveringsfristen er den samme som for den individuelle opgave. I skal både aflevere BMP-filen og SML-filen. Ved fredagsforelæsningen i uge 4 kåres den bedste løsning, og vinderen får en lille præmie.