

2011-04-29 ADDENDUM TO OO versus FP — a little case study

Maarten Fokkinga
University of Twente, Dept Computer Science
PO Box 217, 7500 AE Enschede, The Netherlands
email: fokkinga@cs.utwente.nl

Version of April 21, 1995

Additional Notes

Here are some hand-written notes that I found on a hard copy of the paper.

1. Essence of OO in this example: temperature is *stored at two* places (yet representing the same thing).
2. OO: compositionality: objects correspond 1-1 to concepts (from the problem analysis); this is “locality”, syntactically apart yet computationally interleaved.
3. FB better: no change needed in centigrade/fahrenheit/therm code (in contrast with OO).
4. Decentralisation/distribution of responsibilities regarding the objects is maintained in FP.
5. FP text is homomorph image of the OO text (hence, all “structure/essence” is preserved in the OO-to-FP transition).
6. also an example of an FP text with nice I/O behaviour, but without preservation of OO aspects (see points 4 and 5 in this list).
7. OO essence here: *reuse* of text; in FP version too.

Notes by Pim van den Broek

From pimvdb@cs.utwente.nl Wed May 3 10:58 MET 1995
To: fokkinga@cs.utwente.nl
Subject: thermometers->thermostaten
Content-Type: text
Content-Length: 496

Maarten,

Thermostaten, i.t.t. thermometers, zet je als gebruiker in een gewenste stand.

Huizen hebben als regel slechts 1 thermostaat.

Wat nu als er meer dan 1 thermostaat is? Bv. behalve in de woonkamer is er ook in de slaapkamer een thermostaat. Er is echter maar 1 CV in huis. Op welke thermostaat moet de CV reageren? Natuurlijk op die die het laatst is bijgesteld. Dus de thermostaten moeten gekoppeld zijn.

Ergo: vervang in de probleemformulering thermometer door thermostaat.

Pim.

Alternative by Pim van den Broek

From pimvdb@cs.utwente.nl Thu May 4 18:10 MET 1995

To: fokkinga@cs.utwente.nl

Content-Type: text

Content-Length: 4027

Hallo Maarten,

Hierbij mijn versie van de gekoppelde thermostaten.

Mijn bedoeling is om eerst thermostaten te definiëren die koppelbaar zijn, en vervolgens een functie te definiëren die twee thermostaten koppelt tot een gekoppelde thermostaat. Deze laatste functie doet alleen de koppeling en mag zich dus niet bemoeien met de inhoud van de berichten van en naar de thermostaten.

De berichten van en naar thermostaten zijn van het type msg:

```
msg ::= Set num | Get | Temp num
```

Een thermostaat is van het type

```
thermostaat == ([[msg]], [[msg]]) -> num -> ([[msg]], [[msg]])
```

Er zijn twee invoerkanalen. Via het eerste kanaal komen de berichten van de vorm Set n en Get binnen, netjes geordend per tijdseenheid. n is hier de temperatuur in de schaal van de thermostaat. Via het tweede kanaal komen alleen berichten van de vorm Set n binnen; hier is n in absolute schaal.

Er zijn ook twee uitvoerkanalen. Via het eerste gaan berichten van de vorm Temp n, met n in de eigen schaal. Via het tweede gaan berichten van de vorm Set n, bestemd voor andere thermostaten, met n in absolute schaal.

Een thermostaat handelt de berichten die binnenkomen in tijdsvolgorde af, waarbij het tweede kanaal voorgaat voor het eerste.

Thermostaten worden gemaakt door de functie `thermostaat_producent`, die gegeven een tweetal conversiefuncties, een thermostaat voor die conversiefuncties oplevert:

```
thermostaat_producent :: (num->num) -> (num->num) -> thermostaat
thermostaat_producent x2k k2x (mss, (Set n : ms') : mss') temp
  = thermostaat_producent x2k k2x (mss,ms':mss') (k2x n)
thermostaat_producent x2k k2x ((Set n : ms):mss, mss') temp
  = (rss,(Set (x2k n) : rs'):rss')
    where (rss,rs':rss') = thermostaat_producent x2k k2x (ms:mss,mss') n
thermostaat_producent x2k k2x ((Get : ms):mss, mss') temp
  = ((Temp temp:rs):rss,rss')
    where (rs:rss,rss') = thermostaat_producent x2k k2x (ms:mss,mss') temp
thermostaat_producent x2k k2x ([]:mss,[]:mss') temp
  = ([]:rss,[]:rss')
    where (rss,rss') = thermostaat_producent x2k k2x (mss,mss') temp
```

Dus een `celsius_thermostaat` is gegeven door:

```
celsius_thermostaat :: thermostaat
celsius_thermostaat = thermostaat_producent c2k k2c
                      where c2k x = x+273
                            k2c x = x-273
```

en een `fahrenheit_thermostaat` door:

```
fahrenheit_thermostaat :: thermostaat
fahrenheit_thermostaat = thermostaat_producent f2k k2f
                        where f2k x = (x-32)*5 div 9+273
                              k2f x = (x-273)*9 div 5+32
```

De berichten die van een gekoppelde thermostaat zijn van het type:

```
msg' ::= L msg | R msg
```

Een gekoppelde thermostaat heeft type:

```
gekoppelde_thermostaat == [[msg']] -> [[msg']]
```

De functie die twee thermostaten koppelt is nu heel simpel:

```
koppel :: thermostaat -> thermostaat -> gekoppelde_thermostaat
koppel th1 th2 in = merge out1 out3
```

```

where
  (in1,in3) = split in
  (out1,out2) = th1 (in1,[],:out4) undef
  (out3,out4) = th2 (in3,[],:out2) undef

```

Merk op dat de berichten tussen de beide thermostaten arriveren met een vertraging met 1 tijdseenheid.

De functies `murge` en `split` zijn ook simpel:

```

murge :: [[msg]] -> [[msg]] -> [[msg]]
murge (ms1:mss1) (ms2:mss2) = (map L ms1 ++ map R ms2) : murge mss1 mss2

```

```

split :: [[msg']] -> ([[msg]], [[msg]])
split [] = ([], [])
split ([]:mss) = ([]:mss1, []:mss2)
                where (mss1,mss2) = split mss
split ((L m:ms):mss) = ((m:msL):mssL, mssR)
                    where (msL:mssL, mssR) = split(ms:mss)
split ((R m:ms):mss) = (mssL, (m:msR):mssR)
                    where (mssL, msR:mssR) = split(ms:mss)

```

Een gekoppelde thermostaat waarvan de linker thermostaat een celsius_thermostaat en de rechter een fahrenheit_thermostaat is is:

```

c_f_koppeling :: gekoppelde_thermostaat
c_f_koppeling = koppel celsius_thermostaat fahrenheit_thermostaat

```

Bedenk bij het testen dat zowel invoer als uitvoer onbegrensd zijn.

Ik ben benieuwd wat je hiervan vindt.

Vriendelijke groeten,

Pim.

===== commentaar door MMF =====

> Ik ben benieuwd wat je hiervan vindt.

- 0) Aardige oplossing, gebaseerd op andere uitgangspunten dan de mijne. Eigenlijk zou je de Imperatieve OO oplossing bij moeten zetten, en de parallel tussen die OO en jouw FP oplossing laten blijken.
- 1) Een gekoppelde_thermostaat is geen thermostaat; jammer.
- 2) De modelbouwer heeft een zeer vooruitziende blik moeten hebben om thermostaten als koppelbare thermostaten te definiëren. Liever had ik gezien dan je eerst een notie van thermostaat zou definiëren (en daarmee wat programma's schrijft), en dan later de

- aanpassing tot koppelbare en gekoppelde thermostaten doet.
- 3) murge en split (en de koppeling zelf) zijn vergelijkbaar met (maar inderdaad iets eenvoudiger dan) die van mij.
 - 4) Jammer dat de notie van tijd zo'n grote rol speelt in de overwegingen en uitleg; jammer ook dat je type `[[msg]]` nodig hebt, waar naief een type `[msg]` zou moeten staan.

(Ik heb geen testen uitgevoerd.)

Maarten