Modelat usant Haskell

Albert Rubio

Especialitat de Computació
Grau en Enginyeria Informàtica

FIB

Pla de la sessió

- Contractes financers
- Què cal fer.
- Tipus de dades
- Combinadors primitius
- Contracte depenent del moment
- Obligació cupó-zero
- Implementació dels tipus dades
- Implementació de primitives
- Exemples i exercicis

Contractes financers

Hi ha molts tipus de contractes:

- Obligació (o bó) cupó-zero (zero-coupon bond):
  Una persona que adquireix un contracte zcb rebrà x Euros (o USD, o ...) en un determinat moment.
  Example: rebrà 100 USD el 20 July 2011
  En lloc de pagar interessos es venen a un preu més baix del seu valor nominal.
- Permuta (swap): intercanvi futur de diners o bens.
- Futurs (Futures): contracte per negociar en una data futura un actiu financer (indicant el preu).
- Opcions: contracte que dóna dret a vendre o a comprar un actiu a un preu determinat en un cert moment o període.

Què cal fer

- Identificar els tipus de dates (p.e. Contracte, Data, etc.)
- Definir un conjunt de funcions primitives, *Combinadors*, que expressin diferents funcionalitats.
- Crear nous combinadors usant les primitives.
- Descriure els diferents tipus de contractes usant els tipus de dades i els combinadors.

Tipus de dades

Crearem contractes usant els següents tipus de dades:

- Contracte (`Contract`)
- Observable (`Obs`): és un valor objectivament observable.
  - Exemple: 24 és la temperatura en Centígrads observada a Barcelona el 20 de Juliol de 2011 a les 3pm GMT+1. Això és un exemple d'un numèric observable el valor del qual és 24.
  - Exemple: la data d'avui es posterior a l'1 de Gener de 2012. Això és un example d'un Boolea observable el valor del qual és Cert.
- Moneda (`Currency`)
- Data (`Date`)

Combinadors primitius

```
zero :: Contract
-- Un contracte sense cap valor

one :: Currency -> Contract
-- Un contracte amb valor d'1 EUR o 1 USD...

and :: Contract -> Contract -> Contract
-- Unió de dos sub-contractes. Si adquireixes
-- (and c1 c2) obtens ambdos sub-contractes

or :: Contract -> Contract -> Contract
-- Si adquireixes (or c1 c2) llavors has
-- d'escollir un dels dos sub-contracts
```

Combinadors primitius

```
cond :: Obs Bool -> Contract -> Contract -> Contract
-- Amb (cond b c1 c2) adquireixes c1 si b és cert
-- i c2 en altre cas.

scale :: Obs Double -> Contract -> Contract
-- Amb (scale o c) adquireixes un contracte que té el
-- valor de c multiplicat pel valor d'o

when :: Obs Bool -> Contract -> Contract
-- Amb (when o c) adquireixes un contracte que té valor
-- zero excepte quan o és cert
```

Després els implementarem.

Combinadors primitius

Combinadors sobre observables

```
konst :: a -> Obs a
-- (konst x) és un observable amb valor x

date :: Obs Date
-- és una constant de tipus Date observable

lift2 :: (a -> b -> c) -> Obs a -> Obs b -> Obs c
-- (lift2 f o1 o2) és el observable resultant d'applicar
-- la funció f als observables o1 i o2

-- Exemple: lift2 (==) data (konst "21 Juliol 2011")
-- retorna (Obs True) si data és "21 Juliol 2011" i
-- (Obs False) en altre cas
```

Contracte depenent del moment

Crearem un contracte que assoleix el seu valor només quan la data és una donada.
Necessitarem una funció `at`.

```
at :: Date -> Obs Bool
```

que retorna (`Obs True`) si la data actual coincideix amb la donada.

```
at t = lift2 (==) date (konst t)
```

Si `t` és una data i `c` un contracte

```
c1 = when (at t) c)
```

llavors `c1` és un contracte que assoleix el seu valor només

en la data `t`

Com crear contractes

- El contracte anterior s'ha creat usant "at" i la primitiva "when".
- "at" s'ha creat usant les primitives "lift2", "date" i "konst".
- El contracte s'ha creat sense saber la implementació del tipus de dades `Contract` i `Obs`, els combinadors `when` i `konst`, etc.

Obligació cupó-zero

Zero-Coupon Discount Bond:

```
zcb :: Date -> Double -> Currency -> Contract
```

Example: `(zcb "July 22, 2011" 100 USD)`

```
zcb t x k =
    when (at t) (scale (konst x) (one k))
```

Implementació dels tipus dades

```
data Contract = Contract Currency Double
              | SubContract [Contract]
              deriving (Show)

data Obs a = Obs a
-- (Obs Bool) o (Obs Double) ...

data Currency = EUR
              | USD
              | GBP
              deriving (Show, Eq)

data Date = Date String
            deriving (Show, Eq)
```

Implementació de primitives

```
one :: Currency -> Contract
one c = Contract c 1

scale :: Obs Double -> Contract -> Contract
scale (Obs v) (Contract m q) = Contract m (q * v)

when :: Obs Bool -> Contract -> Contract
when (Obs True) c1 = c1
when (Obs False) c1 = zero

zero :: Contract
zero =  SubContract []
```

Exercici

Implementeu

1. `cond::Obs Bool -> Contract -> Contract -> Contract`
   És un if-then-else.
2. `anytime :: Obs Bool -> Contract -> Contract`
   Tens els contracte en qualsevol moment que el Bool és True.
3. `until :: Obs Bool -> Contract -> Contract`
   Mentre el Bool és False
4. `and :: Contract -> Contract -> Contract`
   Adquireixes els dos.
5. `or :: Contract -> Contract -> Contract`
   És un or exclusiu.
6. `give :: Contract -> Contract`
   Inverteix les obligacions

Altres Primitives

```
konst :: a -> Obs a
konst x  =  Obs x


lift :: (a -> b) -> Obs a -> Obs b
lift f (Obs o) = f o


lift2 :: (a -> b -> c) -> Obs a -> Obs b -> Obs c
lift2 f (Obs o1) (Obs o2) = f o1 o2
```

Altres Primitives

Podem sobrecarregar els operadors numèrics

```
instance Num a => Num (Obs a) where
  fromInteger i = konst (fromInteger i)
  (+) = lift2 (+)
  (-) = lift2 (-)
  (*) = lift2 (*)
  abs = lift abs
  signum = lift signum
```

Però no els relacionals

```
(%<),(%<=)::Ord a => Obs a -> Obs a -> Obs Bool
(%=),(%>=),(%>)::Ord a => Obs a -> Obs a -> Obs Bool

(%<) = lift2 (<)
....
```

Més exemples

Sigui

```
plujaJuliolSitges :: Obs Double
```

una constant que indica la pluja al Juliol a Sitges en un any determinat en litres. Definiu

```
cPluja :: Contract
cPulja = ...
```

com un contracte pel que rebrem 1.000 euros per la quantitat de pluja que superi els 7 litres.

Més exemples

Sigui

```
plujaJuliolSitges :: Obs Double
```

una constant que indica la pluja al Juliol a Sitges en un any determinat en litres. Definiu

```
cPluja :: Contract
cPulja = ...
```

com un contracte pelque rebrem 1.000 euros per la quantitat de pluja que superi els 7 litres.

```
cPulja = scale ((plujaJuliolSitges - 7) * 1000)
               ( one EUR )
```

Més exercicis

Sigui

```
european::Date -> Contract -> Contract
```

un combinador que donada una data i un contracte ens permet escollir entre el contracte o res en la data indicada

Sigui

```
american::Date -> Date -> Contract -> Contract
```

un combinador que donades dues dates i un contracte ens permet escollir entre el contracte o res entre les dues dates indicada Cal definir

```
between::Date -> Date -> Obs Bool
```

Model alternatiu

Alternativa per modelar contractes:

```
data Contract =
    Zero
  | One  Currency
  | Give Contract
  | And  Contract Contract
  | Or   Contract Contract
  | Cond    (Obs Bool)   Contract Contract
  | Scale   (Obs Double) Contract
  | When    (Obs Bool)   Contract
  | Anytime (Obs Bool)   Contract
  | Until   (Obs Bool)   Contract
  deriving Show
```