```haskell
-- Truth tables in a principled manner
-- Fritz Ruehr, WU CS 465, Spring 2015

module TTable where
import FinFun
import ASCII


-- propositional terms, their fold and scan

data Prop a b c = Lit a | Var b | Bop c (Prop a b c) (Prop a b c)

fold f g h (Lit a)     = f a
fold f g h (Var b)     = g b
fold f g h (Bop c l r) = h c (fold f g h l) (fold f g h r)

scan f g h = fold (Lit . f) (Var . g) (\c l r -> Bop (h c (root l) (root r)) l r)

root = fold id id (const . const)


-- propositional operators and variables

data POpr = Dis | Con | Imp | Bic

foldo f g h j Dis = f
foldo f g h j Con = g
foldo f g h j Imp = h
foldo f g h j Bic = j

semo = foldo (||) (&&) (<=) (==)
syno = foldo "v"  "^"  ">"  "="

data PVar = P | Q | R  deriving (Eq, Enum, Bounded, Show)

vars = full :: [PVar]
ints = full :: [PVar -> Bool]


-- truth table functions

body p = [flat (scan id i semo p) | i <- ints]

flat p = fold prb prb  (par " " " " . prb)  p

prt  p = fold prb show (par "(" ")" . syno) p

ttab p = unlines (tab [vhdr, prt p] [vbod, body p])
         where vhdr =  unwords (map show      vars)
               vbod = [unwords (map (prb . i) vars) | i <- ints]


-- utilities and testing

par l r c x y = l ++ unwords [x,c,y] ++ r

prb b = if b then "T" else "F"

[p, q, r]                  = map Var [P, Q, R]
[(|:), (&:), (>:), (=:)] = map Bop [Dis, Con, Imp, Bic]

samps = [ ((p |: q) >: r) =: ((p >: r) &: (q >: r)) ,
          ((p |: q) &: r) =: ((p &: r) |: (q &: r)) ,
          (p >: (q &: r)) =: ((p >: q) &: (p >: r)) ,
          ((p &: q) >: r) =: (p >: (q >: r))          ]

main = putStrLn (sep "\n\n\n" (map ttab samps))
```

```
P Q R | (((P v Q) > R) = ((P > R) ^ (Q > R)))
------+--------------------------------------
F F F |    F F F  T F  T   F T F  T  F T F
T F F |    T T F  F F  T   T F F  F  F T F
F T F |    F T T  F F  T   F T F  F  T F F
T T F |    T T T  F F  T   T F F  F  T F F
F F T |    F F F  T T  T   F T T  T  F T T
T F T |    T T F  T T  T   T T T  T  F T T
F T T |    F T T  T T  T   F T T  T  T T T
T T T |    T T T  T T  T   T T T  T  T T T
```

```
P Q R | (((P v Q) ^ R) = ((P ^ R) v (Q ^ R)))
------+--------------------------------------
F F F |    F F F  F F  T   F F F  F  F F F
T F F |    T T F  F F  T   T F F  F  F F F
F T F |    F T T  F F  T   F F F  F  T F F
T T F |    T T T  F F  T   T F F  F  T F F
F F T |    F F F  F T  T   F F T  F  F F T
T F T |    T T F  T T  T   T T T  T  F F T
F T T |    F T T  T T  T   F F T  T  T T T
T T T |    T T T  T T  T   T T T  T  T T T
```

```
P Q R | ((P > (Q ^ R)) = ((P > Q) ^ (P > R)))
------+--------------------------------------
F F F |   F T  F F F   T   F T F  T  F T F
T F F |   T F  F F F   T   T F F  F  T F F
F T F |   F T  T F F   T   F T T  T  F T F
T T F |   T F  T F F   T   T T T  F  T F F
F F T |   F T  F F T   T   F T F  T  F T T
T F T |   T F  F F T   T   T F F  F  T T T
F T T |   F T  T T T   T   F T T  T  F T T
T T T |   T T  T T T   T   T T T  T  T T T
```

```
P Q R | (((P ^ Q) > R) = (P > (Q > R)))
------+--------------------------------
F F F |    F F F  T F  T  F T  F T F
T F F |    T F F  T F  T  T T  F T F
F T F |    F F T  T F  T  F T  T F F
T T F |    T T T  F F  T  T F  T F F
F F T |    F F F  T T  T  F T  F T T
T F T |    T F F  T T  T  T T  F T T
F T T |    F F T  T T  T  F T  T T T
T T T |    T T T  T T  T  T T  T T T
```