Université de Caen Normandie

L2 Informatique

Autres Paradigmes

Membre du groupe :
Enzo Durand
Thomas Gignoux

Enseignant :
Patrice Boizumault

26 avril 2020

Table des matières

0.1 Question 1

```
visuFormule :: Formule -> String
visuFormule (Var p)   = p
visuFormule (Non f)   = "~" ++ visuFormule f
visuFormule (Et g d)  = "(" ++ (visuFormule g) ++ " & " ++ (visuFormule d) ++ ")"
visuFormule (Ou g d)  = "(" ++ (visuFormule g) ++ " v " ++ (visuFormule d) ++ ")"
visuFormule (Imp g d) = "(" ++ (visuFormule g) ++ " => " ++ (visuFormule d) ++ ")"
visuFormule (Equi g d) = "(" ++ (visuFormule g) ++ " <=> " ++ (visuFormule d) ++ ")"
```

0.2 Question 2

```
- (Imp g d) et (Ou (Non g) d) sont egaux, En effet on a :
  1) (vrai => vrai = vrai) et (~ vrai v vrai = vrai)
  2) (faux => faux = vrai) et (~ faux v faux = vrai)
  3) (faux => vrai = vrai) et (~ faux v vrai = vrai)
  4) (vrai => faux = faux) et (~ vrai v faux = faux)
  On a donc : (Imp g d) <=> (Ou (Non g) d).

- (Equi g d) <=> (Imp g d) & (Imp d g)
  (Imp g d) <=> (Ou (Non g) d)
  (Imp d g) <=> (Ou (Non d) g)
  On a donc : (Equi g d) <=> ((Ou (Non g) d) & (Ou (Non d) g))
```

0.3 Question 3

```
elimine :: Formule -> Formule
elimine (Var p) = (Var p)
elimine (Non f) = (Non f)
elimine (Et g d) = (Et g d)
elimine (Ou g d) = (Ou g d)
elimine (Imp g d) = (Ou (Non g) d)
elimine (Equi g d) = (Et (Ou (Non g) d) (Ou (Non d) g))
```

0.4 Question 4

```
La double négation s'annule :
(~ (~ f)) <=> f
```

0.5 Question 5

```
Les deux lois de Morgan sont :
~ (g v d) <=> (~ g & ~ d)
~ (g & d) <=> (~ g v ~ d)
```

0.6 Question 6

```
La fonction disNon supprime les doubles négations et applique les 2 lois De Morgan

disNon (Var p)   = (Var p)
disNon (Non f)   = f
disNon (Et g d)  = (Ou (Non g) (Non d))
disNon (Ou g d)  = (Et (Non g) (Non d))
```

0.7 Question 7

```
developper :: Formule -> Formule -> Formule
developper (Var p) f  = (Et (Ou (Var p) (recupererA f)) (Ou (Var p) (recupererB f)))
developper (Et g d) f = (concEt (developper g f) (developper d f))

recupererA :: Formule -> Formule
recupererA (Var p)  = (Var p)
recupererA (Et g d) = (recupererA g)

recupererB :: Formule -> Formule
recupererB (Var p)  = (Var p)
recupererB (Et g d) = (recupererB d)
```

0.8 Question 8

```
formeClausale :: Formule -> Formule
formeClausale f = (normalise (ameneNon (elimine f)))
```

0.9 Question 9

```
etToListe :: Formule -> FormuleBis
etToListe (Et g d) = (ouToListe g) : etToListe d
etToListe f        = [ouToListe f]

ouToListe :: Formule -> Clause
ouToListe (Ou g d) = g : ouToListe d
ouToListe  f       = [f]
```

0.10 Question 10

```
neg :: Formule -> Formule
neg (Var p)       = (Non (Var p))
neg (Non (Var p)) = (Var p)
```

0.11 Question 11

```
sontLiees :: Clause -> Clause -> Bool
sontLiees [] _ = False
sontLiees (x:xs) ys = (neg x) `elem` ys || (sontLiees xs ys)
```

0.12 Question 12

```
resolvante :: Clause -> Clause -> Clause
resolvante [] _ = []
resolvante (x:xs) ys
     | sontLiees (x:xs) ys == False = []
     | x `elem` ys                  = [] ++ resolvante xs ys
     | (neg x) `elem` ys            = xs ++ supprimer (neg x) ys
     | otherwise                    = [x] ++ (resolvante xs ys)
          where supprimer a [] = []
                supprimer a (y:ys)
                    | a == y    = [] ++ supprimer a ys
                    | otherwise = [y] ++ supprimer a ys
```