Logique

La bibliotheque Logic

La bibliotheque logic est une bibliotheque utilitaire dont le but est de manipuler des expressions logiques. Ces expressions utilisent des variables, les valeurs true (vrai) et false (faux) et des operateurs logiques classiques comme le "et logique" traduit par &and, le "ou logique" traduit par "&or", le "non logique" traduit par "&not", le "ou exclusif logique" traduit par &xor, l'implication logique traduite par &implies, l'equivalence logique traduite par "&iff". Cette bibliotheque doit etre chargee par un with(logic).

```
> with(logic):
```

Une premiere fonction permet de creer au hasard des expressions logiques. C'est la fonction randbool.

```
> randbool([a,b,c]);
```

$$\&or(\&and(a, c, \&not(b)), \&and(c, \&not(b), \&not(a)), \&and(\&not(b), \&not(c), \&not(a)), \&and(c, b, \&not(a)))$$

Son utilite est tres reduite et nous prefererons definir notre propre fonction permettant de creer des expressions logiques au hasard.

(* la fonction rand_expr(variables,op1,op2,n) retourne une expression formelle de profondeur n en les variables utilisant les operateurs unaires de la liste op1 et les operateurs binaires de la liste op2 *)

```
> rand_expr:=proc(variables,op1,op2,n)
                      local nvar,noper,n1,n2,m,rn;
                nvar:=nops(variables);
                noper:=nops(op1)+nops(op2);
                if n=1 then
                    m:=(rand() mod noper)+1;
                    if m<= nops(op1) then

  RETURN(op1[m](variables[(rand() mod nvar)+1]))
                    else
                         RETURN(op2[m-nops(op1)](
                                variables[(rand()
```

```
  mod nvar)+1],
                                        variables[(rand()
  mod nvar)+1]))
                    fi
                 else
                    n1:=(rand() mod (n-1))+1;
  n2:=(rand() mod (n-1))+1;
                    m:=(rand() mod noper)+1;
                    if m<= nops(op1) then

  RETURN(op1[m](rand_expr(variables,op1,op2,n1)))
                    else
                         RETURN(op2[m-nops(op1)]

  (rand_expr(variables,op1,op2,n1),

  rand_expr(variables,op1,op2,n2)))
                    fi
                fi
      end:
```

```
> rand_expr([a,b],[`&not`],[`&and`,`&or`,`&implies`]
  ,3);
```

$$((\&\text{not}(b))\ \&\text{or}\ (b\ \&\text{and}\ b))\ \&\text{or}\ ((b\ \&\text{implies}\ b)\ \&\text{implies}\ (b\ \&\text{and}\ b))$$

(* la fonction rand_logic(vars,n) retourne une expression logique au hasard de profondeur n en les variables de la liste vars *)

```
> rand_logic:= (vars,n) ->
  rand_expr(vars,[`&not`],[`&and`,`&or`,`&iff`,`&xor
  `,`&implies`],n):
> rand_logic([a,b,c],3);
```

$$\&\text{not}((a\ \&\text{iff}\ a)\ \&\text{and}\ (c\ \&\text{xor}\ a))$$

La fonction satisfy resoud des equations logiques: etant donne une expression logique, elle cherche des valeurs des varaibles qui rendent vraie cette expression (on dit encore qu'elle la satisfont)

```
> expr:=rand_logic([a,b,c],5);
  satisfy(expr);
```

$$expr := (((a\ \&\text{implies}\ c)\ \&\text{iff}\ (c\ \&\text{and}\ b))\ \&\text{implies}\ (\&\text{not}(b)))\ \&\text{xor}\ (((c\ \&\text{implies}\ b)\ \&\text{and}\ (\&\text{not}(b)))\ \&\text{iff}\ (b\ \&\text{iff}\ c))$$

$$\{b = true, c = false, a = true\}$$

La fonction tautology teste si une expression est un tautologie,

c'est a dire si elle est vraie pour toute valeurs des variables

```
> tautology((a &implies b) &iff ((&not a) &or b));
```

$$true$$

alors que la fonction bequal teste si deux expressions logiques sont equivalentes (c'est a dire si elles prennent la meme valeur pour toutes les valeurs des variables)

```
> bequal(a &implies b,(&not a) &or b);
```

$$true$$

en fournissant eventuellement un contre-exemple

```
> bequal(a &implies b,(&not b) &or a,ctrex);
```

$$false$$

```
> ctrex;
```

$$\{a = true, b = false\}$$

On sait que l'on peut effectuer les calculs logiques en se placant dansl'anneau quotient Z/2Z, en remplacant true par 1, 0 par false, et les operateurs logiques par des operations algebriques a base d'addition et de multiplications. La fonction convert(...,MOD2) se charge de cette conversion

```
> convert(a &or b, MOD2,expanded);
```

$$b + a + a\,b$$

```
> convert(a &and b, MOD2,expanded);
```

$$a\,b$$

```
> convert(a &implies b, MOD2,expanded);
```

$$1 + a + a\,b$$

```
> convert(a &iff b, MOD2,expanded);
```

$$a + b + 1$$

```
> convert(a &xor b, MOD2,expanded);
```

$$a + b$$

La fonction bsimp se charge de simplifier une expression en une somme (ou disjonction, ou "ou logique") de conjonctions ("et logique") de variables ou negations de variables, fournissant aisement les solutions de l'expression logique

```
> expr:=`&or`(`&xor`(`&iff`(`&iff`(c,b),`&not`(a)),`
  &iff`(c,c)),`&not`(`&and`(`&and`(b,b),`&implies`(a
  ,a)))):
  bsimp(expr);
```

$$\&or(\&not(b), a\ \&and\ c, (\&not(c))\ \&and\ (\&not(a)))$$

autrement dit l'expression est vraie si et seulement si soit b est

faux, soit a et c sont vrais, soit a et c sont faux.
Quant a la fonction canon, avec ses diverses options, elle permet de reduire une expression logique en sa forme normale disjonctive (disjonction de conjonctions portant sur toutes les variables) ou forme normale conjonctive (conjonction de disjonctions portant sur toutes les variables) ou encore modulo 2

```
> canon(expr,[a,b,c],DNF);
```

$$\&or(\&and(a, c, \&not(b)), \&and(\&not(c), \&not(b), a), \&and(\&not(a), c, \&not(b)), \&and(\&not(a), \&not(c), \&not(b)), \&and(\&not(a), \&not(c), b), \&and(c, b, a))$$

```
> canon(expr,[a,b,c],CNF);
```

$$(\&or(a, \&not(c), \&not(b))) \&and (\&or(\&not(a), \&not(b), c))$$

```
> canon(expr,[a,b,c],MOD2);
```

$$1 + b\,c + a\,b$$

Resolution de problemes de logique

Les exercices suivants sont extraits de l'ouvrage hautement recomandable "Quel est le titre de ce livre" par Raymond Smullyan aux editions Dunod.

Dans les dossiers de l'Inspecteur Craig

L'Inspecteur Leslie Craig de Scotland Yard nous a aimablement permis de consulter les dossiers des affaires les plus marquantes auxquelles il ait ete confronte, cela pour le plus grand interet de ceux qui voient dans la detection des criminels une application de la Logique.

Nous debuterons par une affaire tres simple. Une enorme quantite de marchandise avait ete derobee dans un magasin. Le voleur (ou les voleurs) s'etait enfui en voiture. Trois malfaiteurs notoires A, B, C furent convoques a Scotland Yard pour y etre interroges. On put etablir les faits suivants de facon certaine:

(1) Nul autre que A, B, C n'avait pu participer au cambriolage.
(2) C ne faisait jamais un coup sans la complicite de A.

(3) B ne savait pas conduire. A etait-il innocent ou coupable?

```
> regle70:=(a &or b &or c) &and (c &implies a) &and
  (a &or c):
> bsimp(regle70);
```

$$a$$

Voici un autre cas tres simple de vol: Quand A, B et C furent conduits a Scotland Yard pour y etre interroges, les faits suivants furent etablis de facon certaine:

(1) Nul autre que A, B, C ne pouvait etre implique dans l'affaire.

(2) A ne travaillait jamais sans au moins un complice.
(3) C etait innocent.

B etait-il innocent ou coupable?

```
> regle71:=(a &or b &or c) &and (a &implies (b &or
  c)) &and (&not c):
> bsimp(regle71);
```

$$b \,\&\text{and}\, (\&\text{not}(c))$$

```
>
```

L'affaire des jumeaux indiscernables

Voici le cas plus interessant, d'un cambriolage commis a Londres pour lequel trois malfaiteurs notoires A, B et C furent interpeles et entendus a Scotland Yard. Detail tres important: A et C etaient des vrais jumeaux et ils se ressemblaient tant, qu'a peu pres personne ne pouvait les distinguer. Ces trois malfaiteurs avaient un dossier tres charge et leurs habitudes etaient bien connues. En particulier, on savait que les jumeaux n'etaient pas temeraires, au point qu'aucun d'eux n'aurait ose entreprendre un coup sans complice. Au contraire B aimait agir seul et il ne s'encombrait jamais de complice. D'autre part, plusieurs temoins avaient affirme qu'a l'heure du vol, ils avaient vu l'un des jumeaux dans un bar de Douvres, mais sans pouvoir dire lequel c'etait.

En supposant, cette fois encore, que nul autre que A, B ou C ne pouvait etre implique dans ce vol, qui etait le coupable, qui etait innocent ?

```
> regle72:=(a &or b &or c) &and (a &implies (b &or
  c)) &and (c &implies (a &or b)) &and (b &implies
  (&not a &and &not c)) &and (&not a &or &not c):
> bsimp(regle72);
```

$$\&and(b, \&not(c), \&not(a))$$

" Que pensez-vous de ces trois faits ", demanda un jour l'Inspecteur Craig au Sergent McPherson.

(1) Si A est coupable et B est innocent, alors C est coupable.
(2) C n'agit jamais seul.
(3) A n'opere jamais avec C.
(4) Nul autre que A, B ou C ne peut etre implique dans l'affaire et l'un au moins des trois est coupable.

Le Sergent se gratta la tete et dut avouer: " Pas grand chose, j'en ai peur. Mais de votre cote, pouvez- vous en deduire qui est innocent ou coupable ? "

" Non ", repondit Craig " mais j'en sais assez pour inculper l'un des trois suspects de facon certaine. "

```
> regle73:=((a &and &not b)&implies c)&and (c
  &implies (a &or b)) &and (a&implies &not c) &and
  (a &or b &or c):
> bsimp(regle73);
```

$$(b \ \&and \ (\&not(c))) \ \&or \ (b \ \&and \ (\&not(a)))$$

L'ile des Purs et des Pires

Voici un grand assortiment d'enigmes qui concernent une ile ayant deux especes d'habitants: les Purs qui disent toujours la verite et les Pires qui mentent toujours. Chaque habitant de l'ile est soit un Pur soit un Pire. Je commencerai par un probleme bien connu, puis j'enchainerai avec toute une serie de problemes de mon invention.

Selon ce vieux probleme, trois habitants de l'ile -A, B et C- sont ensemble dans un jardin. Un etranger vient a passer qui demande a A, " Etes-vous un Pur ou un Pire? " A repond, mais en bredouillant tant que l'etranger ne comprend pas sa reponse. Alors l'etranger demande a B " Qu'est-ce qu'il a dit ? " B repond " Il a dit qu'il est un Pire ". Mais a ce moment C intervient et dit " Ne croyez pas B, il ment! ".

Determinez si B et C sont des Purs ou des Pires.

```
> ditque:=(a,p) -> a &iff p;
> regle25:=ditque(c,&not b) &and
  ditque(b,ditque(a,&not a)):
> bsimp(regle25);
```

$$c \;\&and\; (\&not(b))$$

Dans ce probleme il n'y a plus que deux personnes A et B et chacune d'elle est soit un Pur, soit un Pire. A affirme " Au moins l'un de nous deux est un Pire. "
Que sont A et B ?

```
> regle27:=ditque(a,&not a &or &not b):
> bsimp(regle27);
```

$$a \;\&and\; (\&not(b))$$

Supposons que A dise: " Je suis un Pire ou B est un Pur. " Que sont A et B ?

```
> regle28:=ditque(a,&not a &or b):
> bsimp(regle28);
```

$$b \;\&and\; a$$

Nous retrouvons nos trois personnages A, B et C et chacun d'eux est soit un Pur, soit un Pire. A et B font les declarations suivantes:
A: Nous sommes tous des Pires.
B: Un et un seul d'entre-nous est un Pur.
Que sont A, B, C ?

```
> regle30:=ditque(a,&not a &and &not b &and &not c)
  &and ditque (b,(a &or b &or c) &and &not (a &and
  b) &and &not (b &and c) &and &not (a &and c)):
> bsimp(regle30);
```

&and(*b*, &not(*c*), &not(*a*))

Les Purs, les Pires et les Versatiles

Des problemes tout aussi fascinants mettent en jeux une troisieme espece de personnages, en plus des Purs qui disent toujours la verite, et des Pires qui mentent toujours. Ce sont les Versatiles, qui disent de temps en temps la verite ou qui mentent parfois, au gre de leur fantaisie. Voici quelques enigmes de mon invention concernant les Purs, les Pires et les Versatiles.

Voici une enigme peu ordinaire: Deux personnes A et B (chacune est soit un Pur, un Pire ou un Versatile) font les declarations suivantes:
A: B est un Pur.
B: A n'est pas un Pur.
Prouvez qu'au moins l'une d'elles dit la verite sans pour autant etre un Pur.

```
> pur:=x->x.`_Pur`: pire:=x->x.`_Pire`:
  versatile:=x->&not pur(x) &and &not pire(x):
  exclu:=x-> &not(pur(x) &and pire(x)):
  var1:=x->(pur(x),pire(x)):
  lesvar:=l->map(var1,l):
  litanie:=l->&and(op(map(exclu,l))):
  ditque:=(x,aff)->(pur(x) &implies aff) &and
  (pire(x)&implies &not aff):
> exp39:=litanie([a,b]) &and ditque(a,pur(b)) &and
  ditque(b,&not pur(a)):
> bsimp(exp39);
```

(&and(&not(*a_Pur*), &not(*a_Pire*), &not(*b_Pire*))) &or
(&and(&not(*a_Pur*), &not(*b_Pur*), &not(*b_Pire*)))

Voici maintenant ce que disent A et B. A: B est un Pur. B: A est un Pire. Prouvez qu'une de ces personnes dit la verite sans etre un Pur, ou qu'une des deux ment sans etre un Pire.

```
> exp40:=litanie([a,b]) &and ditque(a,pur(b)) &and
  ditque(b,pire(a)):
> bsimp(exp40);
```

(&and(&not(*a_Pur*), &not(*a_Pire*), &not(*b_Pur*))) &or
(&and(&not(*a_Pur*), &not(*b_Pur*), &not(*b_Pire*)))

Sur l'ile des Purs, des Pires et des Versatiles, les Pires forment la

classe inferieure, les Versatiles la classe moyenne et les Purs la classe superieure.

Deux personnes A et B (qui sont des Purs, des Pires ou des Versatiles) font les declarations suivantes:
A: Ma classe est inferieure a celle de B.
B: Ce n'est pas vrai!

Peut-on determiner les classes de A et de B? Peut-on dire de chacune des declarations precedentes qu'elles est vraie ou fausse ?

```
> classeinf:=(x,y) -> distrib((pire(x) &implies
  (versatile(y) &or pur(y))) &and (versatile(x)
  &implies pur(y)) &and (&not pur(x))):
> exp41:=litanie([a,b]) &and
  ditque(a,bsimp(classeinf(a,b))) &and ditque(b,&not
  bsimp(classeinf(a,b))):
> bsimp(exp41);
```

&and(&not(*a_Pur*), &not(*a_Pire*), &not(*b_Pur*), &not(*b_Pire*))

On a trois personnes A, B et C. Une d'elles est un Pur, une autre un Pire et enfm la troisieme un Versatile. A et B font les declarations suivantes:

A: La classe de B est superieure a celle de C.
B: La classe de C est superieure a celle de A.
On demande alors a C: " Qui fait partie de la classe la plus elevee, A ou B ? ". Que repond - il ?

```
> unique:=bsimp(&or(op(map(l->(pur(l[1]) &and
  pire(l[2]) &and
  versatile(l[3])),combinat[permute]([a,b,c])))):
  exp42:=litanie([a,b,c]) &and
  bsimp(ditque(a,bsimp(classeinf(c,b)))) &and
  bsimp(ditque(b,bsimp(classeinf(a,c)))) &and
  unique:
> bsimp(exp42);
```

(&and(&not(*a_Pire*), &not(*b_Pur*), &not(*b_Pire*), *c_Pire*, *a_Pur*, &not(*c_Pur*))) &or (&and(*a_Pire*, &not(*a_Pur*), &not(*b_Pur*), &not(*b_Pire*), *c_Pur*, &not(*c_Pire*)))