

# Rapport de TP

*REPRÉSENTATION DE CONNAISSANCE ET RAISONNEMENT*



**Réalisé par :**

- MESSAOUDENE Lydia
- ZAOUIDI Mohamed

## TP1

### I. Etape 1 :

Dans l'étape 1 nous avons créé un dossier contenant un solver ubcsat

### II. Etape 2:

Après le téléchargement du fichier ubcsat, on a exécuté le fichier : sample.cnf avec la requête `ubcsat -alg saps -i sample.cnf -solve`

```
C:\Windows\System32\cmd.e X + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\messa\Documents\GitHub\TP-RCR\TP 1\ubcsat> ubcsat -alg saps -i sample.cnf -solve
#
# UBCSAT version 1.1.0 (Sea to Sky Release)
#
# http://www.satlib.org/ubcsat
#
# ubcsat -h for help
#
# -alg saps
# -runs 1
# -cutoff 100000
# -timeout 0
# -gtimeout 0
# -noimprove 0
# -target 0
# -wtarget 0
# -seed 552999401
# -solve 1
# -find,-numsol 1
# -findunique 0
# -srestart 0
# -prestart 0
# -drestart 0
#
# -alpha 1.3
# -rho 0.8
# -ps 0.05
# -wp 0.01
# -sapsthresh -0.1
#
# UBCSAT default output:
#   'ubcsat -r out null' to suppress, 'ubcsat -hc' for customization help
#
# Output Columns: |run|found|best|beststep|steps|
#
# run: Run Number
# found: Target Solution Quality Found? (1 => yes)
# best: Best (Lowest) # of False Clauses Found
# beststep: Step of Best (Lowest) # of False Clauses Found
# steps: Total Number of Search Steps
#
#      F  Best      Step      Total
#      Run N Sol'n  of      Search
#      No. D Found  Best  Steps
#
#      1 1      0      67423      67423
#
```

```
C:\Windows\System32\cmd.e  X + v

# run: Run Number
# found: Target Solution Quality Found? (1 => yes)
# best: Best (Lowest) # of False Clauses Found
# beststep: Step of Best (Lowest) # of False Clauses Found
# steps: Total Number of Search Steps
#
#           F   Best      Step      Total
#       Run N Sol'n      of      Search
#       No. D Found      Best      Steps
#
#           1 1      0      67423      67423
#
# Solution found for -target 0

-1 2 3 -4 5 -6 7 -8 9 10
-11 12 -13 14 -15 16 17 -18 19 -20
21 22 23 24 -25 -26 27 28 29 -30
-31 32 -33 34 -35 -36 37 -38 -39 -40
-41 42 -43 44 -45 46 47 48 49 -50
-51 52 53 -54 -55 -56 57 58 -59 -60
-61 62 -63 -64 -65 -66 -67 68 -69 -70
-71 72 -73 -74 75 76 77 -78 79 -80
81 -82 -83 84 85 -86 87 88 89 90
-91 -92 -93 94 95 96 -97 -98 99 100
-101 102 103 104 -105 106 -107 -108 -109 110
111 -112 -113 -114 -115 -116 -117 -118 119 -120
-121 122 123 -124 125 -126 -127 128 129 -130
-131 -132 133 134 -135 -136 137 -138 139 -140
141 -142 143 144 145 146 -147 148 -149 -150
-151 -152 153 -154 -155 156 -157 -158 -159 -160
-161 162 -163 164 165 166 167 168 -169 -170
171 172 173 -174 175 -176 -177 178 -179 180
181 182 183 184 185 186 -187 -188 -189 -190
-191 -192 193 -194 195 -196 197 198 -199 200
-201 -202 -203 204 205 206 207 -208 209 -210
211 -212 213 214 -215 -216 -217 -218 -219 220
221 222 223 -224 -225 -226 227 -228 -229 230
-231 -232 -233 234 235 236 237 -238 -239 240
241 242 -243 -244 245 246 -247 -248 249 -250

Variables = 250
Clauses = 1065
TotalLiterals = 3195
TotalCPUTimeElapsed = 0.031
FlipsPerSecond = 2174943
RunsExecuted = 1
SuccessfulRuns = 1
PercentSuccess = 100.00
Steps_Mean = 67423
Steps_CoeffVariance = 0
```

Voici l'exemple donné dans l'énoncé de ce TP et le resultat :



```
File Edit Selection View Go Run Terminal Help
test1.cnf x
C: > Users > messa > Documents > GitHub > TP-RCR > TP 1 > ubcsat > test1.cnf
1  p cnf 5 9
2  2 -3 0
3  -3 0
4  1 -2 -3 4 0
5  -1 -4 0
6  -1 -2 3 5 0
7  2 -5 0
8  -3 4 -5 0
9  1 2 5 0
10 -3 5 0
11 %
12 0
```

```

C:\Windows\System32\cmd.e  X  +  ~

C:\Users\messa\Documents\GitHub\TP-RCR\TP 1\ubcsat> ubcsat -alg saps -i test1.cnf -solve
#
# UBCSAT version 1.1.0 (Sea to Sky Release)
#
# http://www.satlib.org/ubcsat
#
# ubcsat -h for help
#
# -alg saps
# -runs 1
# -cutoff 100000
# -timeout 0
# -gtimeout 0
# -noimprove 0
# -target 0
# -wtargwt 0
# -seed 553234828
# -solve 1
# -find,-numsol 1
# -findunique 0
# -srestart 0
# -prestart 0
# -drestart 0
#
# -alpha 1.3
# -rho 0.8
# -ps 0.05
# -wp 0.01
# -sapsthresh -0.1
#
# UBCSAT default output:
# 'ubcsat -r out null' to suppress, 'ubcsat -hc' for customization help
#
# Output Columns: |run|found|best|beststep|steps|
#
# run: Run Number
# found: Target Solution Quality Found? (1 => yes)
# best: Best (Lowest) # of False Clauses Found
# beststep: Step of Best (Lowest) # of False Clauses Found
# steps: Total Number of Search Steps
#
#      F   Best      Step      Total
#      Run N Sol'n    of      Search
#      No. D Found    Best    Steps
#
#      1 1      0      3      3
#
# Solution found for -target 0

```

```

# Solution found for -target 0

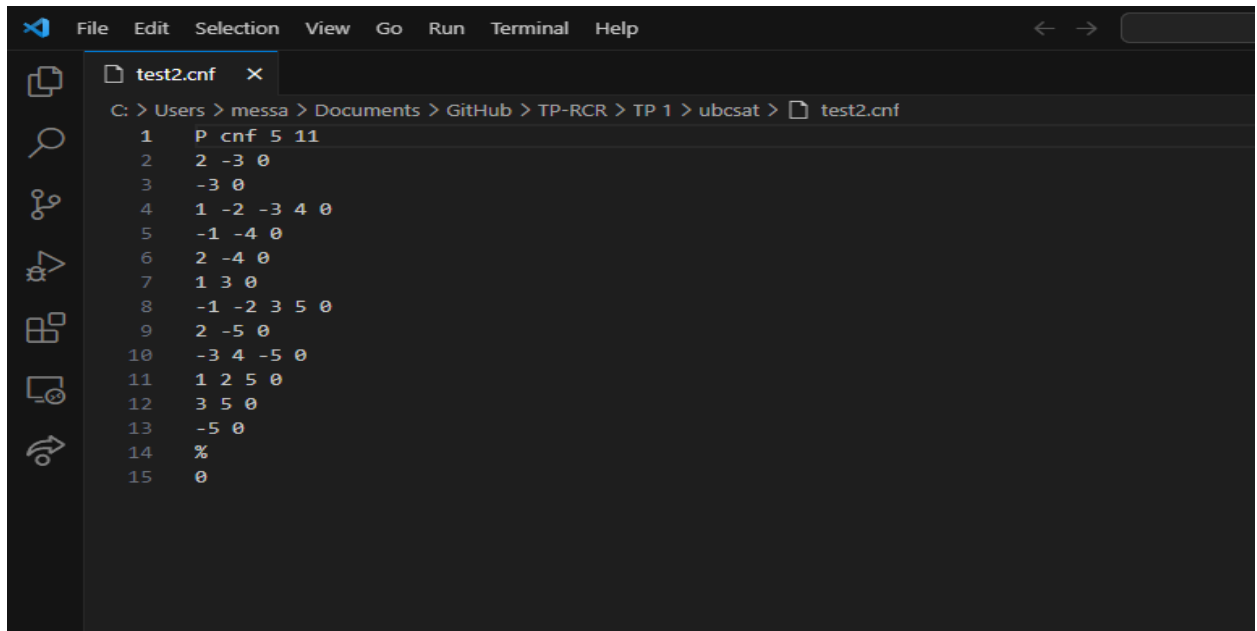
-1 2 -3 4 5

Variables = 5
Clauses = 9
TotalLiterals = 23
TotalCPUtimeElapsed = 0.000
FlipsPerSecond = 1
RunsExecuted = 1
SuccessfulRuns = 1
PercentSuccess = 100.00
Steps_Mean = 3
Steps_CoeffVariance = 0
Steps_Median = 3
CPUtime_Mean = 0
CPUtime_CoeffVariance = 0
CPUtime_Median = 0

```

L'exécution se déroule comme prévu, puisque nous savons que cette base de connaissances est satisfiable.

On fait la même chose avec test2.cnf :

A screenshot of a code editor window with a dark theme. The title bar shows 'test2.cnf' and a close button. The menu bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. The left sidebar contains icons for file explorer, search, source control, and other tools. The main editor area shows the path 'C: > Users > messa > Documents > GitHub > TP-RCR > TP 1 > ubcsat > test2.cnf' and the following SAT formula:

```
1  P cnf 5 11
2  2 -3 0
3  -3 0
4  1 -2 -3 4 0
5  -1 -4 0
6  2 -4 0
7  1 3 0
8  -1 -2 3 5 0
9  2 -5 0
10 -3 4 -5 0
11 1 2 5 0
12 3 5 0
13 -5 0
14 %
15 0
```

```
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\messa\Documents\GitHub\TP-RCR\TP 1\ubcsat> ubcsat -alg saps -i test2.cnf -solve
#
# UBCSAT version 1.1.0 (Sea to Sky Release)
#
# http://www.satlib.org/ubcsat
#
# ubcsat -h for help
#
# -alg saps
# -runs 1
# -cutoff 100000
# -timeout 0
# -gtimeout 0
# -noimprove 0
# -target 0
# -wtarget 0
# -seed 553778082
# -solve 1
# -find,-numsol 1
# -findunique 0
# -srestart 0
# -prestart 0
# -drestart 0
#
# -alpha 1.3
# -rho 0.8
# -ps 0.05
```

```

#
# UBCSAT default output:
#   'ubcsat -r out null' to suppress, 'ubcsat -hc' for customization help
#
#
# Output Columns: |run|found|best|beststep|steps|
#
# run: Run Number
# found: Target Solution Quality Found? (1 => yes)
# best: Best (Lowest) # of False Clauses Found
# beststep: Step of Best (Lowest) # of False Clauses Found
# steps: Total Number of Search Steps
#
#      F  Best      Step      Total
#      Run N Sol'n    of      Search
#      No. D Found    Best    Steps
#
#      1 1      0        3        3
#
# Solution found for -target 0

1 2 -3 -4 5

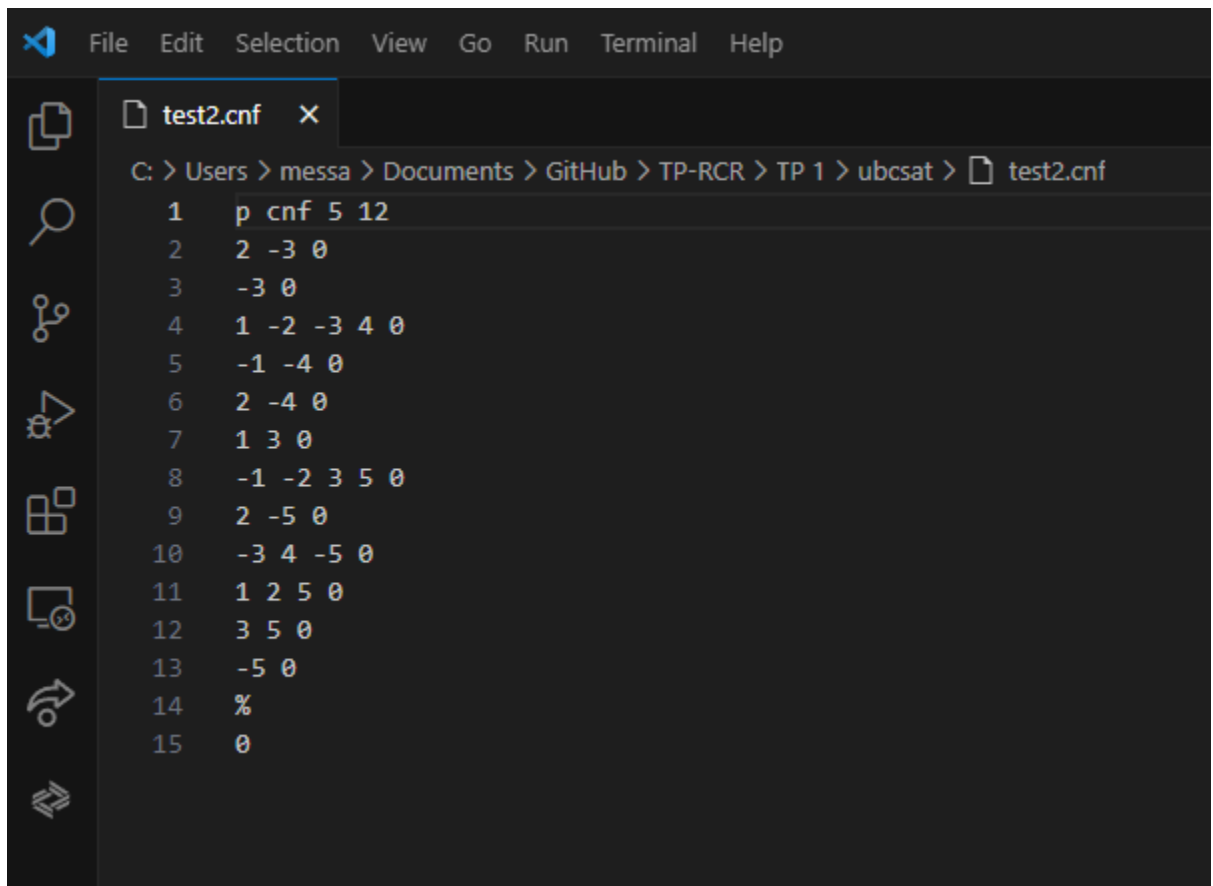
```

```

Variables = 5
Clauses = 11
TotalLiterals = 27
TotalCPUtimeElapsed = 0.001
FlipsPerSecond = 3000
RunsExecuted = 1
SuccessfulRuns = 1
PercentSuccess = 100.00
Steps_Mean = 3
Steps_CoeffVariance = 0
Steps_Median = 3
CPUtime_Mean = 0.000999927520752
CPUtime_CoeffVariance = 0
CPUtime_Median = 0.000999927520752

```

Si nous voulons qu'il ne soit pas satisfaisable:



```
File Edit Selection View Go Run Terminal Help

test2.cnf X
C: > Users > messa > Documents > GitHub > TP-RCR > TP 1 > ubcsat > test2.cnf

1  p cnf 5 12
2  2 -3 0
3  -3 0
4  1 -2 -3 4 0
5  -1 -4 0
6  2 -4 0
7  1 3 0
8  -1 -2 3 5 0
9  2 -5 0
10 -3 4 -5 0
11 1 2 5 0
12 3 5 0
13 -5 0
14 %
15 0
```



```

C:\Windows\System32\cmd.e  X  +  v
C:\Users\messa\Documents\GitHub\TP-RCR\TP 1\ubcsat> ubcsat -alg saps -i test2.cnf -solve
#
# UBCSAT version 1.1.0 (Sea to Sky Release)
#
# http://www.satlib.org/ubcsat
#
# ubcsat -h for help
#
# -alg saps
# -runs 1
# -cutoff 100000
# -timeout 0
# -gtimeout 0
# -noimprove 0
# -target 0
# -wtarget 0
# -seed 553815521
# -solve 1
# -find,-numsol 1
# -findunique 0
# -srestart 0
# -prestart 0
# -drestart 0
#
# -alpha 1.3
# -rho 0.8
# -ps 0.05
# -wp 0.01
# -sapsthresh -0.1
#
# UBCSAT default output:
#   'ubcsat -r out null' to suppress, 'ubcsat -hc' for customization help
#
#
# Output Columns: |run|found|best|beststep|steps|
#
# run: Run Number
# found: Target Solution Quality Found? (1 => yes)
# best: Best (Lowest) # of False Clauses Found
# beststep: Step of Best (Lowest) # of False Clauses Found
# steps: Total Number of Search Steps
#
#      F  Best      Step      Total
#      Run N Sol'n    of      Search
#      No. D Found    Best    Steps
#
#      1 0      1      3      100000
# No Solution found for -target 0

Variables = 5

```

```

Variables = 5
Clauses = 12
TotalLiterals = 28
TotalCPUTimeElapsed = 0.010
FlipsPerSecond = 10000010
RunsExecuted = 1
SuccessfulRuns = 0
PercentSuccess = 0.00
Steps_Mean = 100000
Steps_CoeffVariance = 0
Steps_Median = 100000
CPUTime_Mean = 0.00999999046326
CPUTime_CoeffVariance = 0
CPUTime_Median = 0.00999999046326

```

Aucune solution n'a été trouvée, car nous savons que cette base de connaissance n'est pas satisfiable.

### III. Etape 3:

Traduction de la base de connaissances relative aux connaissances zoologiques suivante:

Na : 1, Nb : 2, Nc : 3, Cea : 4, Ceb : 5, Cec : 6, Ma : 7, Mb : 8, Mc : 9, Coa : 11, Cob : 12, Coc : 13.

$\neg Na \vee Cea$  ;  $\neg Nb \vee Ceb$  ;  $\neg Nc \vee Cec$  ;  $\neg Cea \vee Ma$  ;  $\neg Ceb \vee Mb$  ;  $\neg Cec \vee Mc$  ;  $\neg Na \vee Coa$  ;  $\neg Nb \vee Cob$  ;  $\neg Nc \vee Coc$  ;  $\neg Cea \vee Na \vee \neg Coa$  ;  $\neg Ceb \vee Nb \vee \neg Cob$  ;  $\neg Cec \vee Nc \vee \neg Coc$  ;  $\neg Ma \vee Cea \vee Coa$  ;  $\neg Ma \vee \neg Na \vee Coa$  ;  $\neg Mb \vee Ceb \vee Cob$  ;  $\neg Mb \vee \neg Nb \vee Cob$  ;  $\neg Mc \vee Cec \vee Coc$  ;  $\neg Mc \vee \neg Nc \vee Coc$ .

Voilà fichier CNF :

```
File Edit Selection View Go Run Terminal Help
Etape3.cnf x
C: > Users > messa > Documents > GitHub > TP-RCR > TP 1 > ubcsat > Etape3.cnf
1 p cnf 12 21
2 1 0
3 5 0
4 9 0
5 -1 4 0
6 -2 5 0
7 -3 6 0
8 -4 7 0
9 -5 8 0
10 -6 9 0
11 -1 10 0
12 -2 11 0
13 -3 12 0
14 -4 1 -10 0
15 -5 2 -11 0
16 -6 3 -12 0
17 -7 4 10 0
18 -7 -1 10 0
19 -8 5 11 0
20 -8 -2 11 0
21 -9 6 12 0
22 -9 -3 12 0
23 %
24 0
25
26
27
```

```
C:\Windows\System32\cmd.e x + v
C:\Users\messa\Documents\GitHub\TP-RCR\TP 1\ubcsat> ubcsat -alg saps -i etape3.cnf -solve
#
# UBCSAT version 1.1.0 (Sea to Sky Release)
#
# http://www.satlib.org/ubcsat
#
# ubcsat -h for help
#
# -alg saps
# -runs 1
# -cutoff 100000
# -timeout 0
# -gtimeout 0
# -noimprove 0
# -target 0
# -wtarg 0
# -seed 554559361
# -solve 1
# -find,-numsol 1
# -findunique 0
# -srestart 0
# -prestart 0
# -drestart 0
#
# -alpha 1.3
# -rho 0.8
# -ps 0.05
# -wp 0.01
# -sapsthresh -0.1
#
# UBCSAT default output:
# 'ubcsat -r out null' to suppress, 'ubcsat -hc' for customization help
#
#
# Output Columns: |run|found|best|beststep|steps|
#
# run: Run Number
# found: Target Solution Quality Found? (1 => yes)
# best: Best (Lowest) # of False Clauses Found
# beststep: Step of Best (Lowest) # of False Clauses Found
# steps: Total Number of Search Steps
#
# F Best Step Total
# Run N Sol'n of Search
# No. D Found Best Steps
#
# 1 1 0 7 7
```

```

# Output Columns: |run|found|best|beststep|steps|
#
# run: Run Number
# found: Target Solution Quality Found? (1 => yes)
# best: Best (Lowest) # of False Clauses Found
# beststep: Step of Best (Lowest) # of False Clauses Found
# steps: Total Number of Search Steps
#
#      F   Best      Step      Total
#      Run N Sol'n    of      Search
#      No. D Found    Best    Steps
#
#      1 1      0      7      7
#
# Solution found for -target 0
#
1 2 -3 4 5 6 7 8 9 10
11 -12

Variables = 12
Clauses = 21
TotalLiterals = 48
TotalCPUtimeElapsed = 0.000
FlipsPerSecond = 1
RunsExecuted = 1
SuccessfulRuns = 1
PercentSuccess = 100.00
Steps_Mean = 7
Steps_CoeffVariance = 0
Steps_Median = 7
CPUtime_Mean = 0
CPUtime_CoeffVariance = 0
CPUtime_Median = 0

```

#### IV. ETAPE 4

Après avoir récupéré la base de connaissances, la première étape consiste à vérifier sa satisfiabilité. Si la base de connaissances n'est pas satisfiable, une exception est levée. En revanche, si la base de connaissances est satisfiable, nous procédons au calcul de la négation de la littérale, puis nous inférons dans la base de connaissances en fonction de sa satisfiabilité.

Lors de l'ajout de la nouvelle littérale, il est essentiel de prendre en compte le nombre de clauses indiqué dans la première ligne du fichier CNF. De plus, il est nécessaire de vérifier si la nouvelle formule contient un nombre supérieur à celui des formules déjà présentes.

Ci-dessous se trouve le code source en Python correspondant à ces opérations:

#### 1. Negation Litteral Function (`negation\_litteral`):

- La fonction prend une littérale en entrée et la divise en parties.
- Elle convertit chaque partie en entier, en ignorant les zéros.
- Ensuite, elle génère la négation de la littérale en ajoutant "0" à la fin de chaque littérale.
- La fonction retourne la négation de la littérale, sa taille et le nombre maximum présent dans la littérale.

```
def negation_litteral(litterale):  
    #separe le litterale en plusieurs partie  
    litterale = litterale.split(" ")  
    #on convertit en entier en enlevant le 0  
    litterale = [int(i) for i in litterale if i != "0"]  
    #la négation :  
    litterale = [-i for i in litterale]  
    negation_litterale = ""  
    for i in litterale:  
        negation_litterale += "{} 0\n".format(i)  
    return negation_litterale, len(litterale), max([abs(i) for i in litterale])
```

#### 2. Insertion Litterale Function (`insertion\_litterale`):

- La fonction prend le chemin du fichier CNF (`cnf\_path\_file`) et une littérale en entrée.
- Elle utilise la fonction `negation\_litteral` pour obtenir la négation de la littérale.
- Elle lit le fichier CNF, effectue un traitement pour éliminer les commentaires, les sauts de ligne, etc.
- Elle ajuste le nombre de clauses et de variables dans la première ligne du fichier CNF en fonction de la négation de la littérale.
- Elle ajoute la négation de la littérale à la base de connaissances.
- Elle crée un fichier temporaire contenant la base de connaissances mise à jour.

```

def insertion_litterale(cnf_path_file, litterale):
    negation_litterale, nb_lignes, var_max = negation_litteral(litterale)
    #initialisation de la base de connaissance
    with open(cnf_path_file, 'r') as cnf_file:
        BC = cnf_file.read()
    #nous traitons le fichier cnf en enlevant tout ce qui est commentaire,
    # sauts de lignes, ou meme espacement
    BC = BC.split("\n")
    i = 0
    while (BC[i] == "" or BC[i][0] != "p"):
        i+=1
    BC[i].replace("\t", " ")

    #réajouter les clauses apres le traitement !
    f1, f2, var_nb, nb_clause = [f for f in BC[i].split(" ") if f!=""]
    nb_clause, var_nb = (int(nb_clause), int(var_nb))
    nb_clause += nb_lignes
    #test de nombre des clauses dans le fichier
    if var_max > var_nb:
        var_nb = var_max

    BC[i] = " ".join([f1, f2, str(var_nb), str(nb_clause)])
    BC = "\n".join(BC)
    if BC[-1] != '\n':
        BC += '\n'
    #ajout de la negation du litterale
    BC += negation_litterale
    temp_BC = os.path.basename(cnf_path_file)[:4] + "_temporary.cnf"
    with open(temp_BC, 'w') as temporary_cnf:
        temporary_cnf.write(BC)

```

### 3. Solver Function (`solver`):

- La fonction prend le chemin du fichier CNF en entrée.

```

def solver(cnf_path_file):
    return subprocess.call("ubcsat "+ "-alg " + "saps " + "-i " + f"{cnf_path_file} " + "-solve", shell=True)

```

### 4. Satisfiable Function (`satisfiable`):

- La fonction utilise le solveur pour tester la satisfiabilité du fichier CNF.
- Elle retourne `True` si le fichier est satisfiable, sinon `False`.

```

def satisfiable(cnf_path_file):
    solvable=solver(cnf_path_file)
    if(solvable==0):
        return True
    return False

```

## 5. Algorithme Function (`algorithme`):

- La fonction prend le chemin du fichier CNF (`cnf\_path\_file`) et une littérale en entrée.
- Elle teste d'abord la satisfiabilité du fichier CNF.
- Si le fichier est satisfiable, elle insère la négation de la littérale dans la base de connaissances.
- Elle crée un fichier temporaire et teste à nouveau la satisfiabilité.
- Elle affiche un message indiquant si la base de connaissances infère ou non la littérale.

## 6. Main Section (`main`):

- Le script s'exécute uniquement si c'est le programme principal.
- Il change le répertoire de travail vers le répertoire contenant le solveur UBSCAT.
- Il utilise la fonction `algorithme` avec le fichier CNF "sample.cnf" et la littérale "-4 6 2 0".

```
def algorithme(cnf_path_file,litterale):
    #test de la satisfiabilite
    print("11111111111111111111111111111111")
    if satisfiable(cnf_path_file):
        print("12222222222222222222222222222222",cnf_path_file)
        #insertion du litterale
        insertion_litterale(cnf_path_file, litterale)
        print("33333333333333333333333333333333",cnf_path_file)
        temp_BC = os.path.basename(cnf_path_file)[:4] + "_temporary.cnf"

        #test de la satisfiabilite
        if satisfiable(temp_BC):
            print("la base de connaissance {} infère le litterale {}".format(os.path.basename(cnf_path_file), litterale))
        else:
            print("74444444444444444444444444444444",cnf_path_file)
            print("la base de connaissance {} n'infère pas le litterale {}".format(os.path.basename(cnf_path_file), litterale))

        os.remove(temp_BC)
    else:
        raise ValueError("veuillez introduire une Base de Connaissance satisfiable svp !")

if __name__ == "__main__":
    os.chdir('../ubcsat')
    os.system("start cmd /K py <ubcsat> <variant string> -p <ubcsat>")
    algorithme("sample.cnf","-4 6 2 0")
```

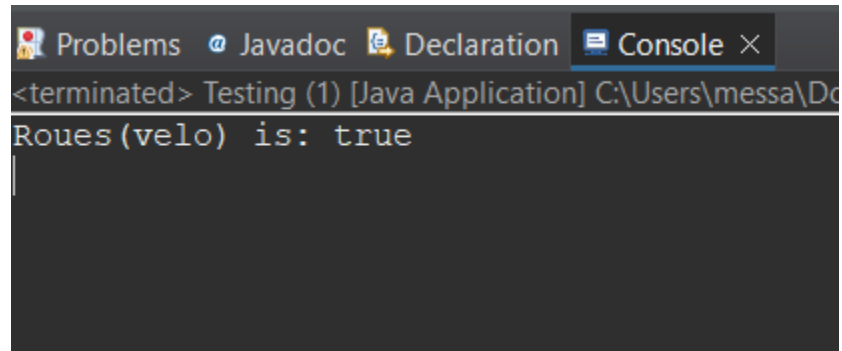
## Partie 2 :

1. Définition des symboles : La classe crée des symboles logiques comme "véhicule" et "Roues" en utilisant la classe `Predicate`, et une constante "velo" en utilisant la classe `Constant`.
2. Initialisation de la signature et de l'ensemble de croyances : Les objets `FolSignature` et `FolBeliefSet` sont créés pour représenter la signature logique et l'ensemble de croyances, respectivement.
3. Analyseur de formules : Un objet `FolParser` est créé pour analyser les formules logiques.
4. Configuration de la signature : Les symboles logiques sont ajoutés à la signature.
5. Ajout de formules logiques : Deux formules logiques sont ajoutées à l'ensemble de croyances. La première formule exprime que tout X étant un véhicule implique que X a des roues. La deuxième formule affirme que "velo" a des roues.
7. Configuration du raisonneur : Le raisonneur par défaut est configuré en utilisant la classe `SimpleFolReasoner`.
8. Interrogation du raisonneur : Le raisonneur est utilisé pour répondre à la question "Velo a-t-il des roues ?". Le résultat de la requête est affiché à la console.

```
9 public class Testing {
10     public static void main(String[] args) throws IOException {
11
12         FolSignature signature = new FolSignature();
13         FolBeliefSet BC = new FolBeliefSet();
14         FolParser parser = new FolParser();
15
16         Predicate véhicule = new Predicate("véhicule", 1);
17         signature.add(véhicule);
18
19         Predicate Roues = new Predicate("Roues", 1);
20         signature.add(Roues);
21
22         Constant velo = new Constant("velo");
23         signature.add(velo);
24
25         BC.setSignature(signature);
26         parser.setSignature(signature);
27
28         BC.add(parser.parseFormula("forall X: (véhicule(X) => Roues(X)"));
29         BC.add(parser.parseFormula("Roues(velo)"));
30
31         FolReasoner.setDefaultReasoner(new SimpleFolReasoner());
32         FolReasoner prover = FolReasoner.getDefaultReasoner();
33         System.out.println("Roues(velo) is: " + prover.query(BC, (FolFormula) parser.parseFormula("Roues(velo)")));
34     }
35 }
```



Résultat:



The screenshot shows an IDE's console window with a dark background. The title bar at the top contains four tabs: 'Problems' with a bug icon, 'Javadoc' with a '@' icon, 'Declaration' with a magnifying glass icon, and 'Console' with a speech bubble icon and a close button. The console text shows a test execution status: '<terminated> Testing (1) [Java Application] C:\Users\messa\Do'. Below this, the output of a test is displayed: 'Roues(velo) is: true'. A vertical cursor is visible on the left side of the console area.

```
<terminated> Testing (1) [Java Application] C:\Users\messa\Do
Roues(velo) is: true
```

## **TP2**

Au cours du TP2, nous examinerons la validité des formules des exercices 2 et 4 abordés lors de la séance dirigée (TD). Nous utiliserons deux outils distincts à cette fin : le 'Modal Logic Playground' et la bibliothèque 'Tweety' en JAVA.

### 1. Modal Logic Playground

Cet outil ne permet pas d'introduire un nombre variable différent de variables dans chaque monde. Ainsi, lorsque seulement une variable est présente dans un monde, les autres variables sont considérées comme négatives. Il est important de noter que les résultats de cette approche peuvent par conséquent ne pas être entièrement cohérents.

#### **Exemple:**

simple computer vision scenario with objects and their properties. In this scenario, we have two types of objects, "Car" and "Pedestrian," and each object can have different colors, such as "Red," "Blue," or "Green." Additionally, we want to express some observations using modal logic.

#### **Scénario:**

- Objects: Car, Pedestrian
- Properties: Color (Red, Blue, Green)

#### **Belief Sets:**

##### 1. World w1:

- Objects: Car, Pedestrian
- Observations: Car is Red, Pedestrian is Blue

##### 2. World w2:

- Objects: Car
- Observations: Car is Red

3. World w3:

- Objects: Car, Pedestrian
- Observations: Car is Red, Pedestrian is Blue

4. World w4:

- Objects: Pedestrian
- Observations: Pedestrian is Blue

5. World w5:

- Objects: Car
- Observations: Car is Red

Modal Logic Formulas:

1.  $(p \text{ and } q)$  - It is observed that there is a Red Car and a Blue Pedestrian.
2.  $(\Box p)$  - In all accessible worlds, a Red Car is observed.
3.  $\Box(p \Rightarrow q)$  - In all accessible worlds, if a Red Car is observed, then a Blue Pedestrian is also observed.
4.  $\Box(q \text{ and } \Diamond \neg p)$  - In all accessible worlds, a Blue Pedestrian is observed, and there exists a world where a Red Car is not observed.

```

eclipse-workspace - mytweetyapp/src/main/java/mytweetyapp/Testing.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
  mytweetyapp [TP-RCR main]
    src/main/java
      mytweetyapp
        Testing.java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
    src
      main
      target
    pom.xml
    mytweetyapp2

Testing.java
1 package mytweetyapp;
2
3 import java.io.IOException;
4 import java.util.ArrayList;
5
6 import org.tweetyproject.commons.ParserException;
7 import org.tweetyproject.logics.ml.reasoner.SimpleMLReasoner;
8 import org.tweetyproject.logics.ml.syntax.MLBeliefSet;
9 import org.tweetyproject.logics.commons.syntax.Predicate;
10 import org.tweetyproject.logics.fol.syntax.FolFormula;
11 import org.tweetyproject.logics.fol.syntax.FolSignature;
12 import org.tweetyproject.logics.ml.parser.MLParser;
13
14 public class Testing {
15
16     public static void main(String[] args) throws ParserException, IOException {
17
18         // Suppose we have a computer vision scenario with objects (p and q) and their types.
19         MLParser parser = new MLParser();
20         FolSignature sig = new FolSignature();
21         sig.add(new Predicate("type", 1));
22         sig.add(new Predicate("p", 0));
23         sig.add(new Predicate("q", 0));
24         parser.setSignature(sig);
25
26         MLBeliefSet w1 = parser.parseBeliefBase("type(p) \n type(q) \n" + "(p) \n" + "(q) \n");
27         MLBeliefSet w2 = parser.parseBeliefBase("type(p) \n type(q) \n" + "(p) \n");
28         MLBeliefSet w3 = parser.parseBeliefBase("type(p) \n type(q) \n" + "(p) \n" + "(q) \n");
29         MLBeliefSet w4 = parser.parseBeliefBase("type(p) \n type(q) \n" + "(q) \n");
30         MLBeliefSet w5 = parser.parseBeliefBase("type(p) \n type(q) \n" + "(p) \n");

```

```

31         MLBeliefSet w5 = parser.parseBeliefBase("type(p) \n type(q) \n" + "(p) \n");
32
33         // Modal logic formulas related to computer vision
34         SimpleMLReasoner reasoner = new SimpleMLReasoner();
35         FolFormula f1 = (FolFormula) parser.parseFormula("p && q");
36         FolFormula f2 = (FolFormula) parser.parseFormula("(p)");
37         FolFormula f3 = (FolFormula) parser.parseFormula("(p => q)");
38         FolFormula f4 = (FolFormula) parser.parseFormula("(q <=> (!p))");
39
40         // Relations between worlds
41         MLBeliefSet[] Relation1 = { w2, w3, w4 };
42         MLBeliefSet[] Relation2 = { w5 };
43         MLBeliefSet[] Relation3 = { w4 };
44         MLBeliefSet[] Relation5 = { w4 };
45
46         // Example application of modal logic in computer vision scenario
47         System.out.println("=====Computer Vision Example with Modal Logic=====");
48
49         boolean correct = false;
50
51         // Check if the formula is true in any accessible world from w1 (Relation1)
52         for (MLBeliefSet w : Relation1) {
53             if (reasoner.query(w, f1)) {
54                 correct = true;
55             }
56         }
57         System.out.println("In the computer vision scenario, is it possible to observe both p and q in accessible world from w1? : " + correct);

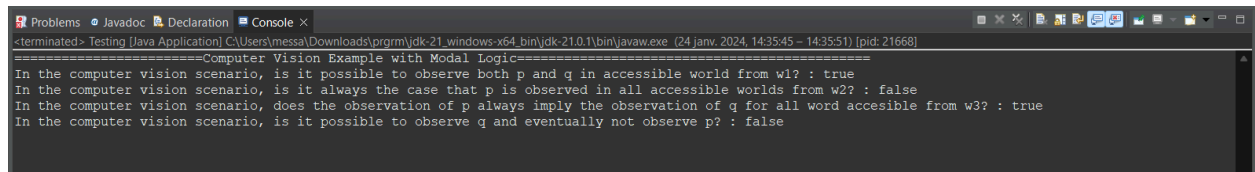
```

```

46         System.out.println("=====Computer Vision Example with Modal Logic=====");
47
48         boolean correct = false;
49
50         // Check if the formula is true in any accessible world from w1 (Relation1)
51         for (MLBeliefSet w : Relation1) {
52             if (reasoner.query(w, f1)) {
53                 correct = true;
54             }
55         }
56         System.out.println("In the computer vision scenario, is it possible to observe both p and q in accessible world from w1? : " + correct);
57
58         // Check if the formula is true in all accessible worlds from w2 (Relation2)
59         correct = false;
60         for (MLBeliefSet w : Relation2) {
61             if (reasoner.query(w, f2)) {
62                 correct = true;
63             }
64         }
65         System.out.println("In the computer vision scenario, is it always the case that p is observed in all accessible worlds from w2? : " + !correct);
66
67         // Check a specific relation (w3 to w4) for the formula (p => q)
68         System.out.println("In the computer vision scenario, does the observation of p always imply the observation of q for (w3 to w4)? : "
69             + reasoner.query(w4, f3));
70
71         // Check a specific relation (w4 to w4) for the formula (q && <=> (!p))
72         System.out.println("In the computer vision scenario, is it possible to observe q and eventually not observe p? : "
73             + reasoner.query(w4, f4));
74     }

```

Résultat :



```
<terminated> Testing [Java Application] C:\Users\ymessa\Downloads\prgrm\jdk-21_windows-x64_bin\jdk-21.0.1\bin\javaw.exe (24 janv. 2024, 14:35:45 - 14:35:51) [pid: 21668]
=====Computer Vision Example with Modal Logic=====
In the computer vision scenario, is it possible to observe both p and q in accessible world from w1? : true
In the computer vision scenario, is it always the case that p is observed in all accessible worlds from w2? : false
In the computer vision scenario, does the observation of p always imply the observation of q for all word accesible from w3? : true
In the computer vision scenario, is it possible to observe q and eventually not observe p? : false
```

## **TP3**

### **Outil : DefaultLogicModelCheck**

L'outil DefaultLogicModelCheck est accessible dans le répertoire :(<https://github.com/edm92/defaultlogic>).

La logique des défauts représente les connaissances sous forme de règles comportant des exceptions ou des conditions spéciales pouvant être activées ou désactivées en fonction de la situation. Cela permet de modéliser des situations où une règle peut être vraie dans la plupart des cas, mais présenter des exceptions.

Dans le cadre de ce TP, nous explorons la logique des défauts en mettant en œuvre le premier exercice de la série du TD. Nous utilisons l'outil DefaultLogicModelCheck, un instrument puissant employé dans la logique formelle pour le raisonnement par défaut. Cet outil offre un cadre pour la représentation des connaissances et le raisonnement dans des contextes où l'information est incomplète ou incertaine.

```
Appjava TDEX.java x
1 package be.fnord.DefaultLogic;
2
3 import a.e;
4 import be.fnord.util.logic.DefaultReasoner;
5 import be.fnord.util.logic.WFF;
6 import be.fnord.util.logic.defaultLogic.DefaultRule;
7 import be.fnord.util.logic.defaultLogic.RuleSet;
8 import be.fnord.util.logic.defaultLogic.WorldSet;
9 import java.util.HashSet;
10
11
12 public class TDEX {
13     public static void EXERCISE1() {
14
15         // création de l'ensemble des défauts
16         RuleSet rules = new RuleSet();
17
18         // création du défaut d1
19         DefaultRule d1 = new DefaultRule();
20         d1.setPrerequisite("A");
21         d1.setJustificatoir("B");
22         d1.setConsequence("C");
23
24         // ajout de d1 dans l'ensemble des défauts
25         rules.addRule(d1);
26
27         // même chose pour d2
28         DefaultRule d2 = new DefaultRule();
29         d2.setPrerequisite("A");
30         d2.setJustificatoir(e.NOT + "C");
31         d2.setConsequence("D");
32
33         rules.addRule(d2);
34
35         // définition d'un monde w1
36         WorldSet w1 = new WorldSet();
37         w1.addFormula(e.NOT + "A");
38     }
}
```

```

38
39 // déf w2
40 WorldSet w2 = new WorldSet();
41 w2.addFormula("A");
42 w2.addFormula(e.NOT + "B");
43
44 // déf w3
45 WorldSet w3 = new WorldSet();
46 w3.addFormula("A");
47 w3.addFormula("(" + e.NOT + "C" + e.OR + e.NOT + "D");
48
49 // déf w4
50 WorldSet w4 = new WorldSet();
51 w4.addFormula("A");
52 w4.addFormula("(" + e.NOT + "B" + e.AND + "C");
53
54 // w1
55 try {
56     a.e.println("\nMonde 1:");
57     DefaultReasoner r = new DefaultReasoner(w1, rules); // raisonner
58     HashSet<String> scenarios = r.getPossibleScenarios(); // extraction des extensions
59     a.e.println("w1: [" + w1.toString() + "]\nDefaults : [" + rules.toString() + "]);
60     if (scenarios.isEmpty()) a.e.println("ces défauts ne génèrent pas d'extension ");
61     for (String c : scenarios) {
62         a.e.println("E: Th(W U {" + c + "})");
63         WFF world_and_ext = new WFF("(" + w1.getWorld() + " ) & (" + c + "));
64         a.e.println(world_and_ext.getClosure());
65         a.e.decIndent();
66     }
67     a.e.println("");
68 } catch (Exception e) {
69     System.out.println(e);
70

```

```

71
72 // w2
73 try {
74     a.e.println(" Monde 2:");
75     DefaultReasoner r = new DefaultReasoner(w2, rules);
76     HashSet<String> scenarios = r.getPossibleScenarios();
77     a.e.println("w2: [" + w2.toString() + "]\nDefaults : [" + rules.toString() + "]);
78     for (String c : scenarios) {
79         a.e.println("E: Th(W U {" + c + "})");
80         WFF world_and_ext = new WFF("(" + w2.getWorld() + " ) & (" + c + "));
81         a.e.println(world_and_ext.getClosure());
82         a.e.decIndent();
83     }
84     a.e.println("");
85 } catch (Exception e) {
86     System.out.println(e);
87 }
88
89 // w3
90 try {
91     a.e.println(" Monde 3:");
92     DefaultReasoner r = new DefaultReasoner(w3, rules);
93     HashSet<String> scenarios = r.getPossibleScenarios();
94     a.e.println("w3: [" + w3.toString() + "]\nDefaults : [" + rules.toString() + "]);
95     for (String c : scenarios) {
96         a.e.println("E: Th(W U {" + c + "})");
97         WFF world_and_ext = new WFF("(" + w3.getWorld() + " ) & (" + c + "));
98         a.e.println(world_and_ext.getClosure());
99         a.e.decIndent();
100     }
101     a.e.println("");
102 } catch (Exception e) {
103     System.out.println(e);
104 }
105

```



```

15
16 // W4
17 try {
18     a.e.println(" Monde 4:");
19     DefaultReasoner r = new DefaultReasoner(w4, rules);
20     HashSet<String> scenarios = r.getPossibleScenarios();
21     a.e.println("w3: [" + w4.toString() + " ]\nDefaults : [" + rules.toString() + "]");
22     for (String c : scenarios) {
23         a.e.println("E: Th(W U {" + c + "})");
24         WFF world_and_ext = new WFF("(" + w4.getWorld() + " ) & (" + c + "));
25         a.e.println(world_and_ext.getClosure());
26         a.e.decIndent();
27     }
28     a.e.println("");
29 } catch (Exception e) {
30     System.out.println(e);
31 }
32 }
33
34 }
35

```

et nous appelons la fonction Exercice 1

```

App.java × TDEX.java
1 package be.fnord.DefaultLogic;
2
3 /**
4  * Starting point
5  * Loads the examples
6  *
7  */
8 public class App
9 {
10     public static void main( String[] args )
11     {
12         a.e.println("Demonstrating reasoners:");
13
14         //AbductiveExample.main(args);
15         TDEX.EXERCISE1();
16     }
17 }
18

```

et voila le resultat

```

Problems Javadoc Declaration Console ×
<terminated> App [Java Application] C:\Users\messa\Downloads\prgrm\jdk-21_windows-x64_bin\jdk-21.0.1\bin\javaw.exe (26 déc. 2023, 14:46:26 – 14:46:27) [pid: 5168]
Demonstrating reasoners:

Monde 1:
w1: [~A]
Defaults : [[(A):(B) ==> (C)] , [(A):(~C) ==> (D)]]
ces défauts ne génèrent pas d'extension

Monde 2:
Trying eeee & A & ~B
Trying eeee & A & ~B
w2: [A & ~B]
Defaults : [[(A):(B) ==> (C)] , [(A):(~C) ==> (D)]]
E: Th(W U {D})
D & ~B & eeee & A

Monde 3:
Trying eeee & A & (~C|~D)
Trying eeee & A & (~C|~D)
Trying eeee & A & (~C|~D)
Trying eeee & A & (~C|~D)
w3: [A & (~C|~D)]
Defaults : [[(A):(B) ==> (C)] , [(A):(~C) ==> (D)]]
E: Th(W U {C})
C & ~D & eeee & (~D | ~C) & A

Monde 4:
Trying eeee & A & (~B&C)
Trying eeee & A & (~B&C)
error
java.lang.NullPointerException: Cannot invoke "orbital.logic.sign.Signature.iterator()" because "this.sigma" is null

```

## Explication :

— W1 : Le raisonneur ne trouve pas d'extensions car : les défauts d1 et d2 ne sont pas utilisables car leurs prérequis  $A \in \Gamma_1(E)$ . D'où, par clôture déductive et minimalité, les deux défauts ne sont pas générateurs d'extension.

— W2 : Le raisonneur arrive à trouver une extension  $TH(\{A, \neg B, D\})$  car : les deux défauts de la théorie d1 et d2 sont utilisables car leur prérequis  $A \in \Gamma_{\Delta 2}(E)$  (vu que  $A \in W$  et  $W \subset \Gamma_{\Delta 2}(E)$ ). L'applicabilité de d1 rend d2 non applicable. Or d1 est non applicable vu que la négation de sa justification  $\neg B \in \Gamma_{\Delta 2}(E)$  (vu que justification  $\neg B \in W_2$  et  $W_2 \subset \Gamma_{\Delta 2}(E)$ ) et que si E est une extension  $E = \Gamma_{\Delta 2}(E)$ . D'où,  $\neg B \in E$ . Pour d2, il est applicable. D'où, par clôture déductive et minimalité, cette théorie admet une seule extension :  $E = \Gamma_{\Delta 2}(E) = TH(\{A, \neg B, D\})$ . Seul d2 est générateur d'extension.

— W3 : Le raisonneur arrive à trouver une seule extension  $E = \Gamma_{\Delta 3}(E) = TH(A, \neg C \vee \neg D, C)$

— W4 : le raisonneur affiche une erreur car la négation des justifications des défauts d1 et d2  $\neg B$  et  $\neg\neg C$  appartiennent à E

## **TP4**

La logique de description vise à représenter et à raisonner sur des connaissances complexes. Elle offre un cadre formel permettant de décrire des concepts, des relations et des contraintes dans un domaine spécifique. Elle repose sur une structure de langage qui permet de définir des classes, des propriétés et des relations entre ces classes. Elle offre des mécanismes pour spécifier des axiomes, des restrictions et des règles d'inférence, ce qui permet de déduire de nouvelles informations à partir des connaissances existantes. Ainsi, elle permet de représenter de manière précise et formelle des connaissances sur un domaine spécifique et de raisonner logiquement sur ces connaissances. Dans ce TP, pour le raisonneur on a opté pour Pellet et pour l'outil l'app WebProtégé.

### **Notre représentation**

Dans le domaine de la vision par ordinateur, un outil similaire à Web Protégé pourrait être utilisé pour modéliser des connaissances basées sur des ontologies. Prenons un exemple hypothétique avec une ontologie décrivant des concepts liés à la détection d'objets dans des images.

#### **TBOX (Terminological Box) :**

- Concept : Object
- Sous-concepts : Person, Car, Animal, Building
- Attributs : Color, Size, Shape

#### **ABOX (Assertion Box) :**

\* Individu : Person1

- Type : Person
- Attribut : Color = Blue, Size = Medium

\* Individu : Car1

- Type : Car
- Attribut : Color = Red, Size = Large

\* Individu : Dog1

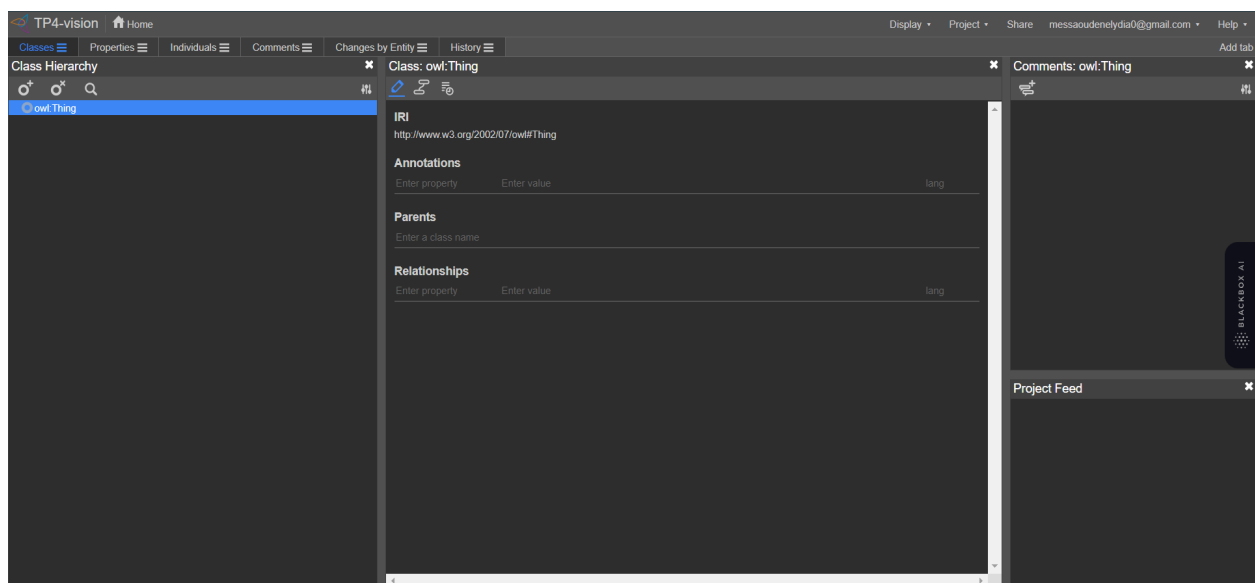
- Type : Animal

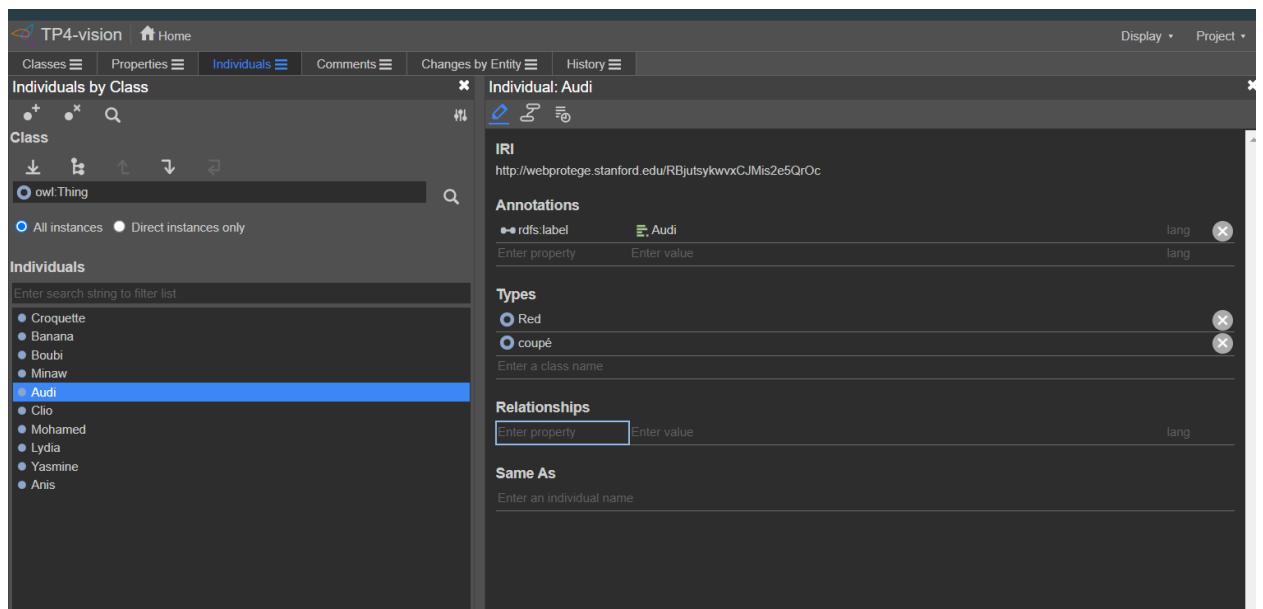
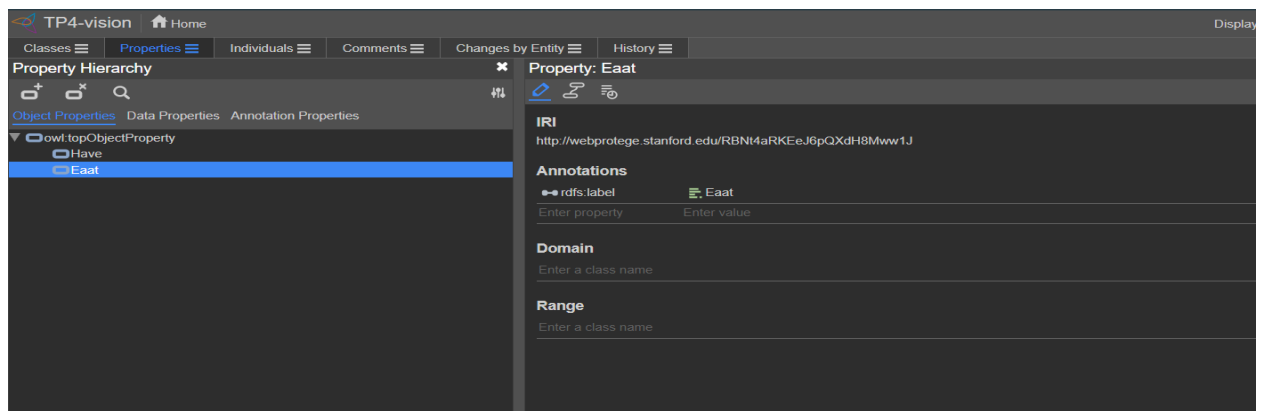
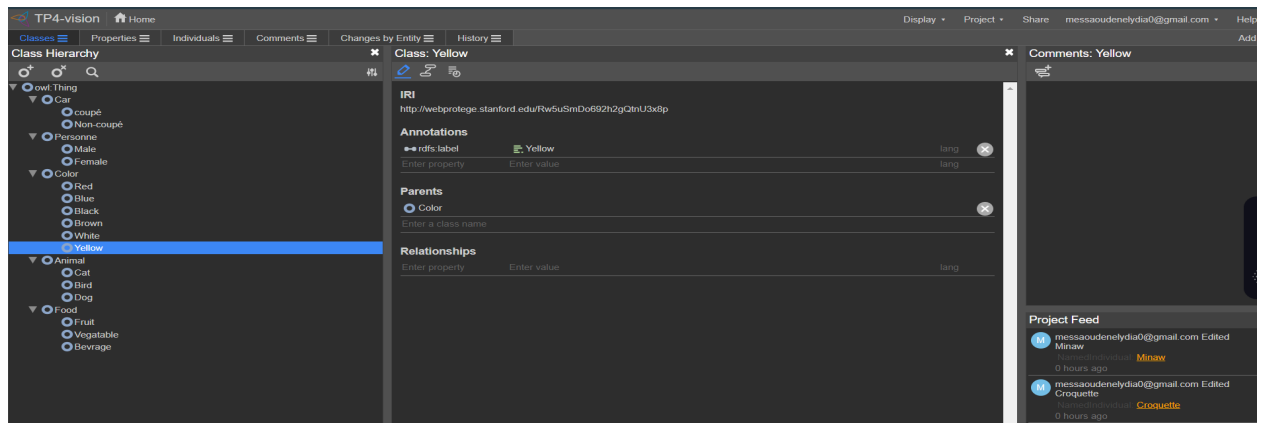
- Attribut : Color = Brown, Size = Small

Dans cet exemple, l'ontologie modélise des concepts tels que "Person", "Car" et "Animal", avec des attributs comme la couleur, la taille et la forme. L'ABOX contient des individus spécifiques avec des caractéristiques définies, comme une personne avec une couleur bleue et une taille moyenne, une voiture avec une couleur rouge et une grande taille, et un chien avec une couleur brune et une petite taille.

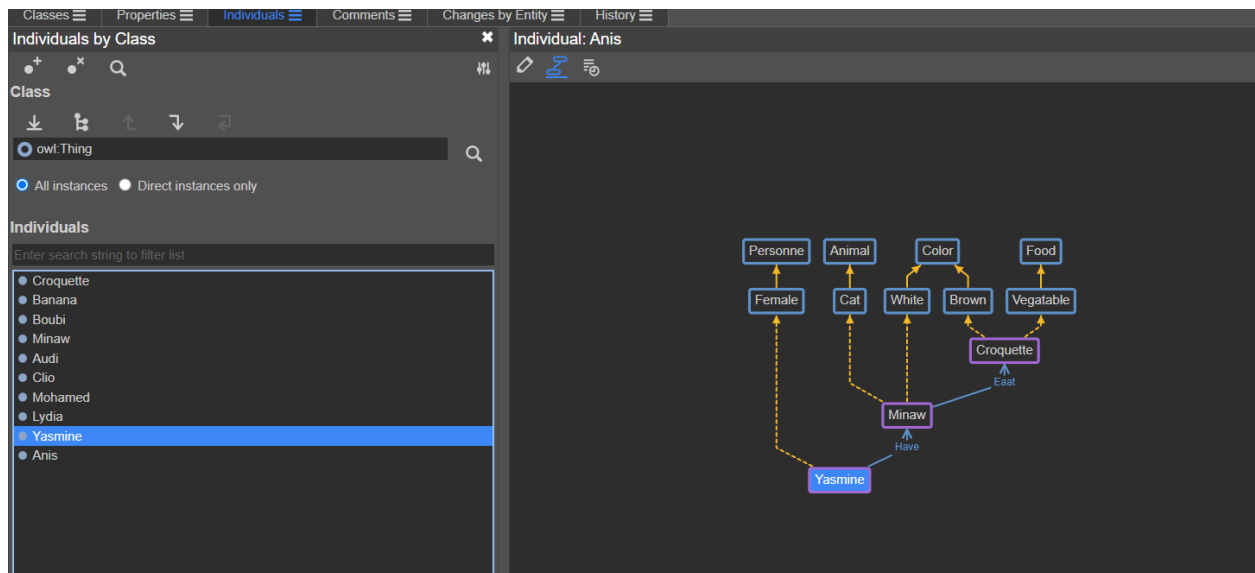
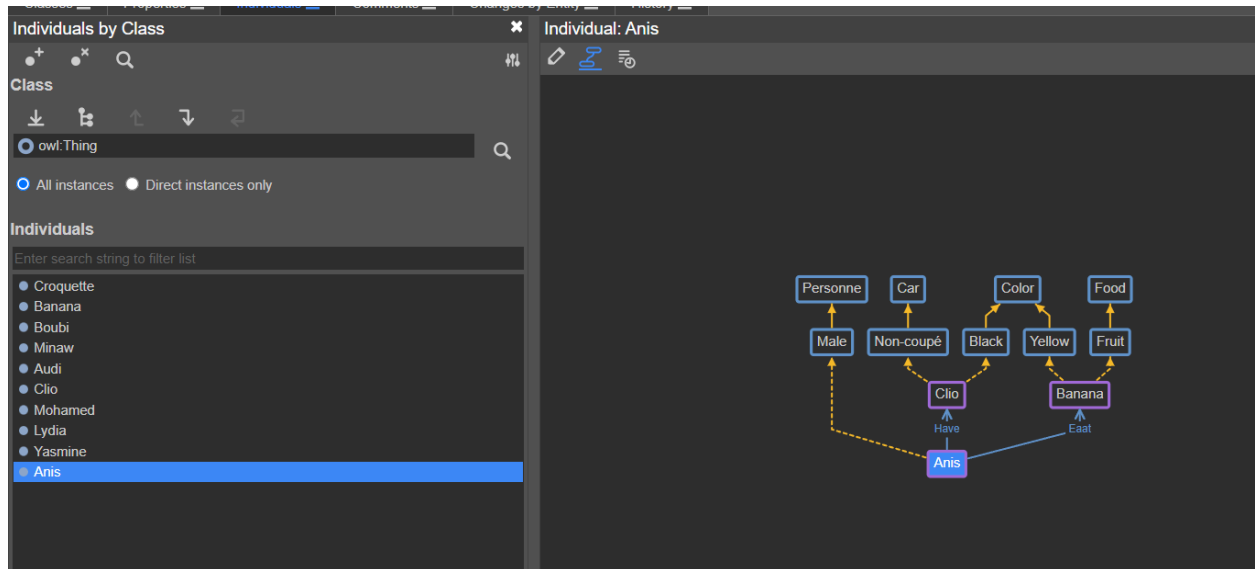
Cet exemple illustre comment un outil comme Web Protégé pourrait être utilisé pour définir une ontologie dans le domaine de la vision par ordinateur, en permettant aux utilisateurs de spécifier des concepts, des sous-concepts et des relations, tout en ajoutant des faits concrets sur des individus et leurs caractéristiques.

[https://webprotege.stanford.edu/#projects/636ce114-5485-460a-955e-113cce1f389d/edit/Individuals?selection=NamedIndividual\(%3Chttp://webprotege.stanford.edu/R7YtDdLfZOOtcu7HCb22YZU%3E\)](https://webprotege.stanford.edu/#projects/636ce114-5485-460a-955e-113cce1f389d/edit/Individuals?selection=NamedIndividual(%3Chttp://webprotege.stanford.edu/R7YtDdLfZOOtcu7HCb22YZU%3E))





Après avoir initialisé toute les relation



TP4-vision Home

Classes Properties Individuals Comments Changes by Entity History

Individuals by Class

Class

owl:Thing

All instances Direct instances only

Individuals

Enter search string to filter list

- Croquette
- Banana
- Boubi
- Minaw
- Audi
- Clio
- Mohamed
- Lydia
- Yasmine
- Anis

0 instances

Page 1 of 1 Prev Next

Individual: Anis

Comments: Lydia

Project Feed

- messaoudenelydia0@gmail.com Edited Minaw 2 minutes ago
- messaoudenelydia0@gmail.com Edited Croquette 2 minutes ago
- messaoudenelydia0@gmail.com Edited Croquette 2 minutes ago

TP4-vision Home

Classes Properties Individuals Comments Changes by Entity History

Individuals by Class

Class

owl:Thing

All instances Direct instances only

Individuals

Enter search string to filter list

- Croquette
- Banana
- Boubi
- Minaw
- Audi
- Clio
- Mohamed
- Lydia
- Yasmine
- Anis

Individual: Anis

Comments: Mohamed

Project Feed

- messaoudenelydia0@gmail.com Edited Minaw 2 minutes ago
- messaoudenelydia0@gmail.com Edited Croquette 2 minutes ago

## TP5

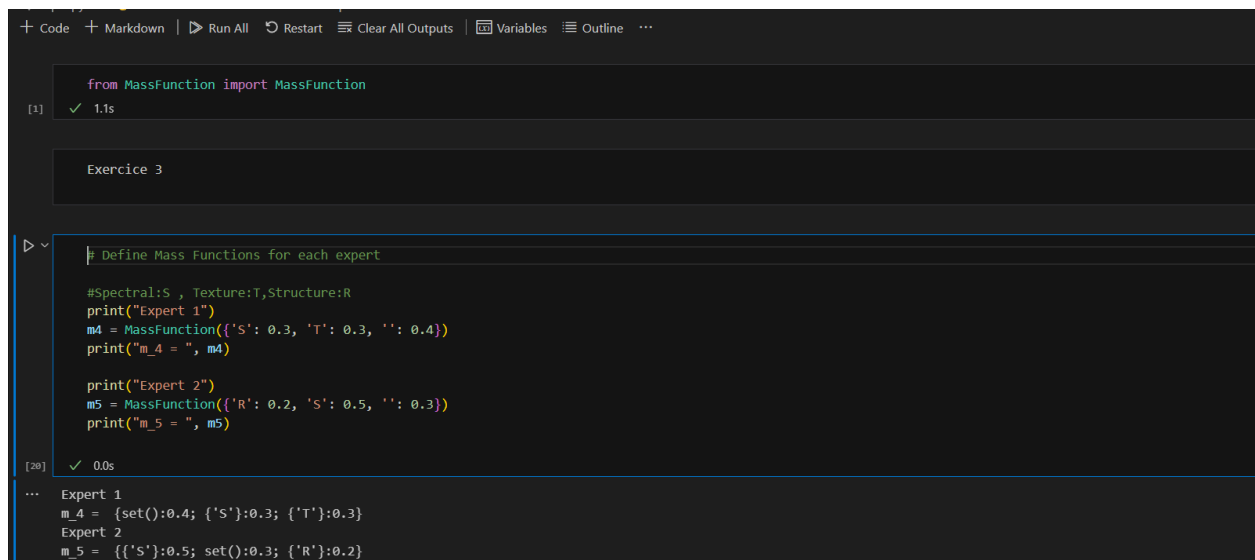
Dans ce premier TP portant sur la logique des fonctions de croyances, nous avons employé la bibliothèque Python pyds. Cet outil permet d'effectuer des calculs dans le cadre de la théorie de Dempster-Shafer. La bibliothèque pyds était initialement disponible sur le dépôt GitHub suivant : [pyds de python](#). Cependant, dans la version actuelle, la classe MassFunction n'était plus incluse. Pour remédier à cela, nous avons importé la classe depuis un autre dépôt existant, qui conservait la version antérieure, et avons ainsi pu résoudre les exercices 4 et 5 de la série de TD.

### Exercice 3:

Deux experts discutent à propos des performances des informations qui peuvent être extraites à partir d'une image haute-résolution dans le cadre de la classification des images. Le premier atteste l'information spectrale ou l'information de texture sont importantes à 30%. Le second affirme que l'information de structure est efficace à 20%, l'information spatiale l'est à 50%.

- 4- Modélisez ce problème en utilisant la théorie de Dempster-Shafer.
- 5- Comment tenir compte des deux sources de connaissances. Explicitez
- 6- Calculez les degrés de croyances de l'ensemble des éléments.

voilà le code avec les résultats :



```
+ Code + Markdown | ▶ Run All ⏮ Restart ⌵ Clear All Outputs | 📄 Variables 📄 Outline ...

[1] ✓ 1.1s

Exercise 3

▶ ▾ Define Mass Functions for each expert

#Spectral:S , Texture:T,Structure:R
print("Expert 1")
m4 = MassFunction({'S': 0.3, 'T': 0.3, '': 0.4})
print("m_4 = ", m4)

print("Expert 2")
m5 = MassFunction({'R': 0.2, 'S': 0.5, '': 0.3})
print("m_5 = ", m5)

[20] ✓ 0.0s

... Expert 1
m_4 = {set():0.4; {'S':0.3; {'T':0.3}
Expert 2
m_5 = {'S':0.5; set():0.3; {'R':0.2}
```



```
combined_mass_images = m4 & m5
print("\nCombined Mass Function for Image Classification:")
print("Combined Mass Function =", combined_mass_images)
```

[21] ✓ 0.0s

...

```
Combined Mass Function for Image Classification:
Combined Mass Function = {{'S'}:1.0}
```

```
#affichage de croyance et plausibilité pour chaque expert
print("Croyance et possibilité pour l'expert 1")
```

```
print("bel_1(S) = ", m4.bel({'S'}))
print("pl_1(S) = ", m4.pl({'S'}))
print("Dis1(S) = ", 1- m4.pl({'S'}))
```

```
print("bel_1(T) = ", m4.bel({'T'}))
print("pl_1(T) = ", m4.pl({'T'}))
print("Dis1(T) = ", 1- m4.pl({'T'}))
```

```
print("bel_1(S) = ", m4.bel({'R'}))
print("pl_1(S) = ", m4.pl({'R'}))
print("Dis1(S) = ", 1- m4.pl({'R'}))
```

[23] ✓ 0.0s

... Croyance et possibilité pour l'expert 1

```
bel_1(S) = 0.3
pl_1(S) = 0.3
Dis1(S) = 0.7
bel_1(T) = 0.3
pl_1(T) = 0.3
Dis1(T) = 0.7
bel_1(S) = 0.0
pl_1(S) = 0.0
Dis1(S) = 1.0
```

```
#affichage de croyance et plausibilité pour chaque expert
print("Croyance et possibilité pour l'expert 2")

print("bel_1(S) = ", m5.bel({'S'}))
print("pl_1(S) = ", m5.pl({'S'}))
print("Dis1(S) = ", 1- m5.pl({'S'}))

print("bel_1(T) = ", m5.bel({'T'}))
print("pl_1(T) = ", m5.pl({'T'}))
print("Dis1(T) = ", 1- m5.pl({'T'}))

print("bel_1(S) = ", m5.bel({'R'}))
print("pl_1(S) = ", m5.pl({'R'}))
print("Dis1(S) = ", 1- m5.pl({'R'}))
```

[24] ✓ 0.0s

```
... Croyance et possibilité pour l'expert 2
bel_1(S) = 0.5
pl_1(S) = 0.5
Dis1(S) = 0.5
bel_1(T) = 0.0
pl_1(T) = 0.0
Dis1(T) = 1.0
bel_1(S) = 0.2
pl_1(S) = 0.2
Dis1(S) = 0.8
```

#### **Exercice 4:**

Trois experts discutent à propos du bruit présent sur une image. Le premier atteste qu'il est causé lors du processus d'acquisition à 38% et lors de la transmission à 55%. Le second expert affirme qu'il est dû lors du processus de stockage à 88%. Le troisième expert atteste que les différentes hypothèses sont équiprobables.

1. Modélisez ces connaissances à l'aide de la théorie de Dempster & Shafer. Quelles sont les spécificités des différentes distributions de masse.
2. Calculez les degrés de croyance et de plausibilité associés aux différents éléments. Que peut-on conclure ?
3. Calculez les degrés de doute associés aux différents éléments.
4. Que représentent les degrés de croyance et de plausibilité associés aux trois distributions ?
5. Combinez les trois sources d'expertise en explicitant chaque étape. Que pouvez-vous conclure ?

```

# affichage des masses pour chaque expert

print("Expert 1")
m1 = MassFunction({'A':0.38,'T':0.55,'':0.07})
print("m_1 = ", m1)

print("Expert 2")
m2 = MassFunction({'S':0.88,'':0.12})
print("m_2 = ", m2)

print("Expert 3")
m3 = MassFunction({'A': 0.33, 'T': 0.33, 'S': 0.33})
print("m_3 = ", m3)

```

[8] ✓ 0.0s

```

... Expert 1
m_1 = {'T':0.55; 'A':0.38; set():0.07}
Expert 2
m_2 = {'S':0.88; set():0.12}
Expert 3
m_3 = {'A':0.33; 'T':0.33; 'S':0.33}

```

```

#affichage de croyance et plausibilité pour chaque expert
print("Croyance et possibilité pour l'expert 1")

print("bel_1(A) = ", m1.bel({'A'}))
print("pl_1(A) = ", m1.pl({'A'}))
print("Dis1(A) = ", 1- m1.pl({'A'}))

print("bel_1(T) = ", m1.bel({'T'}))
print("pl_1(T) = ", m1.pl({'T'}))
print("Dis1(T) = ", 1- m1.pl({'T'}))

print("bel_1(S) = ", m1.bel({'S'}))
print("pl_1(S) = ", m1.pl({'S'}))
print("Dis1(S) = ", 1- m1.pl({'S'}))

```

[12] ✓ 0.0s

```

... Croyance et possibilité pour l'expert 1
bel_1(A) = 0.38
pl_1(A) = 0.38
Dis1(A) = 0.62
bel_1(T) = 0.55
pl_1(T) = 0.55
Dis1(T) = 0.44999999999999996
bel_1(S) = 0.0
pl_1(S) = 0.0
Dis1(S) = 1.0

```

```
#affichage de croyance et plausibilité pour chaque expert
print("Croyance et possibilité pour l'expert 2")

print("bel_1(A) = ", m2.bel({'A'}))
print("pl_1(A) = ", m2.pl({'A'}))
print("Dis1(A) = ", 1- m2.pl({'A'}))

print("bel_1(T) = ", m2.bel({'T'}))
print("pl_1(T) = ", m2.pl({'T'}))
print("Dis1(T) = ", 1- m2.pl({'T'}))

print("bel_1(S) = ", m2.bel({'S'}))
print("pl_1(S) = ", m2.pl({'S'}))
print("Dis1(S) = ", 1- m2.pl({'S'}))
```

[13] ✓ 0.0s

... Croyance et possibilité pour l'expert 1

```
bel_1(A) = 0.0
pl_1(A) = 0.0
Dis1(A) = 1.0
bel_1(T) = 0.0
pl_1(T) = 0.0
Dis1(T) = 1.0
bel_1(S) = 0.88
pl_1(S) = 0.88
Dis1(S) = 0.12
```

```
#affichage de croyance et plausibilité pour chaque expert
print("Croyance et possibilité pour l'expert 3")

print("bel_1(A) = ", m3.bel({'A'}))
print("pl_1(A) = ", m3.pl({'A'}))
print("Dis1(A) = ", 1- m3.pl({'A'}))

print("bel_1(T) = ", m3.bel({'T'}))
print("pl_1(T) = ", m3.pl({'T'}))
print("Dis1(T) = ", 1- m3.pl({'T'}))

print("bel_1(S) = ", m3.bel({'S'}))
print("pl_1(S) = ", m3.pl({'S'}))
print("Dis1(S) = ", 1- m3.pl({'S'}))
```

[14] ✓ 0.0s

... Croyance et possibilité pour l'expert 1

```
bel_1(A) = 0.33
pl_1(A) = 0.33
Dis1(A) = 0.6699999999999999
bel_1(T) = 0.33
pl_1(T) = 0.33
Dis1(T) = 0.6699999999999999
bel_1(S) = 0.33
pl_1(S) = 0.33
Dis1(S) = 0.6699999999999999
```

```
#combinaisons des masses par fusion de Dempster-Shafer
print("Dempster-Shafer Combinaison rule")
print("Dempster-Shafer Combinaison rule for m_1 and m_2 = ", m1 & m2)
print("Dempster-Shafer Combinaison rule for m_2 and m_3 = ", m2 & m3)
print("Dempster-Shafer Combinaison rule for m_1 and m_3 = ", m1 & m3)

print("Dempster-Shafer Combinaison rule for m_1, m_2 and m_3 = ", m1.combine_conjunctive(m2, m3))
print("Dempster-Shafer Combinaison rule for m_2, m_1 and m_3 = ", m2.combine_conjunctive(m1, m3))
print("Dempster-Shafer Combinaison rule for m_3, m_1 and m_2 = ", m3.combine_conjunctive(m1, m2))
```

15] ✓ 0.0s

```
.. Dempster-Shafer Combinaison rule
Dempster-Shafer Combinaison rule for m_1 and m_2 = {}
Dempster-Shafer Combinaison rule for m_2 and m_3 = {{'S':1.0}}
Dempster-Shafer Combinaison rule for m_1 and m_3 = {{'T':0.5913978494623655; {'A':0.4086021505376344}}
Dempster-Shafer Combinaison rule for m_1, m_2 and m_3 = {}
Dempster-Shafer Combinaison rule for m_2, m_1 and m_3 = {}
Dempster-Shafer Combinaison rule for m_3, m_1 and m_2 = {{'T':0.5913978494623655; {'A':0.4086021505376344}}
```

## Exemple de vision

Each expert provides their opinion about the color of an object in an image. The Dempster-Shafer combination rule is then used to combine these opinions, and a decision is made based on the combined mass. Adjust the colors and their associated masses according to your specific computer vision problem.

```
# Masses provided by three experts regarding the color of an object
print("Expert 1")
m1 = MassFunction({'r': 0.6, 'g': 0.3, 'b': 0.1})
print("m_1 = ", m1)

print("Expert 2")
m2 = MassFunction({'r': 0.2, 'g': 0.5, 'b': 0.3})
print("m_2 = ", m2)

print("Expert 3")
m3 = MassFunction({'r': 0.4, 'g': 0.2, 'b': 0.4})
print("m_3 = ", m3)
```

17] ✓ 0.0s

```
... Expert 1
m_1 = {{'r':0.6; {'g':0.3; {'b':0.1}}
Expert 2
m_2 = {{'g':0.5; {'b':0.3; {'r':0.2}}
Expert 3
m_3 = {{'r':0.4; {'b':0.4; {'g':0.2}}
```

```

# Belief, plausibility, and disbelief for each expert
colors = ['red', 'green', 'blue']

for expert, m in zip(['Expert 1', 'Expert 2', 'Expert 3'], [m1, m2, m3]):
    print(f"\nCroyance et possibilité pour {expert}")
    for c in colors:
        print(f"bel_{expert}({c}) = {m.bel({c})}")
        print(f"pl_{expert}({c}) = {m.pl({c})}")
        print(f"Dis{expert}({c}) = {1 - m.pl({c})}")

[18] ✓ 0.0s
...
Croyance et possibilité pour Expert 1
bel_Expert 1(red) = 0.0
pl_Expert 1(red) = 0.0
DisExpert 1(red) = 1.0
bel_Expert 1(green) = 0.0
pl_Expert 1(green) = 0.0
DisExpert 1(green) = 1.0
bel_Expert 1(blue) = 0.0
pl_Expert 1(blue) = 0.0
DisExpert 1(blue) = 1.0

Croyance et possibilité pour Expert 2
bel_Expert 2(red) = 0.0
pl_Expert 2(red) = 0.0
DisExpert 2(red) = 1.0
bel_Expert 2(green) = 0.0
pl_Expert 2(green) = 0.0
DisExpert 2(green) = 1.0
bel_Expert 2(blue) = 0.0
pl_Expert 2(blue) = 0.0
DisExpert 2(blue) = 1.0

Croyance et possibilité pour Expert 3
bel_Expert 3(red) = 0.0
pl_Expert 3(red) = 0.0
DisExpert 3(red) = 1.0
bel_Expert 3(green) = 0.0
pl_Expert 3(green) = 0.0
DisExpert 3(green) = 1.0
bel_Expert 3(blue) = 0.0
pl_Expert 3(blue) = 0.0
DisExpert 3(blue) = 1.0

```

```

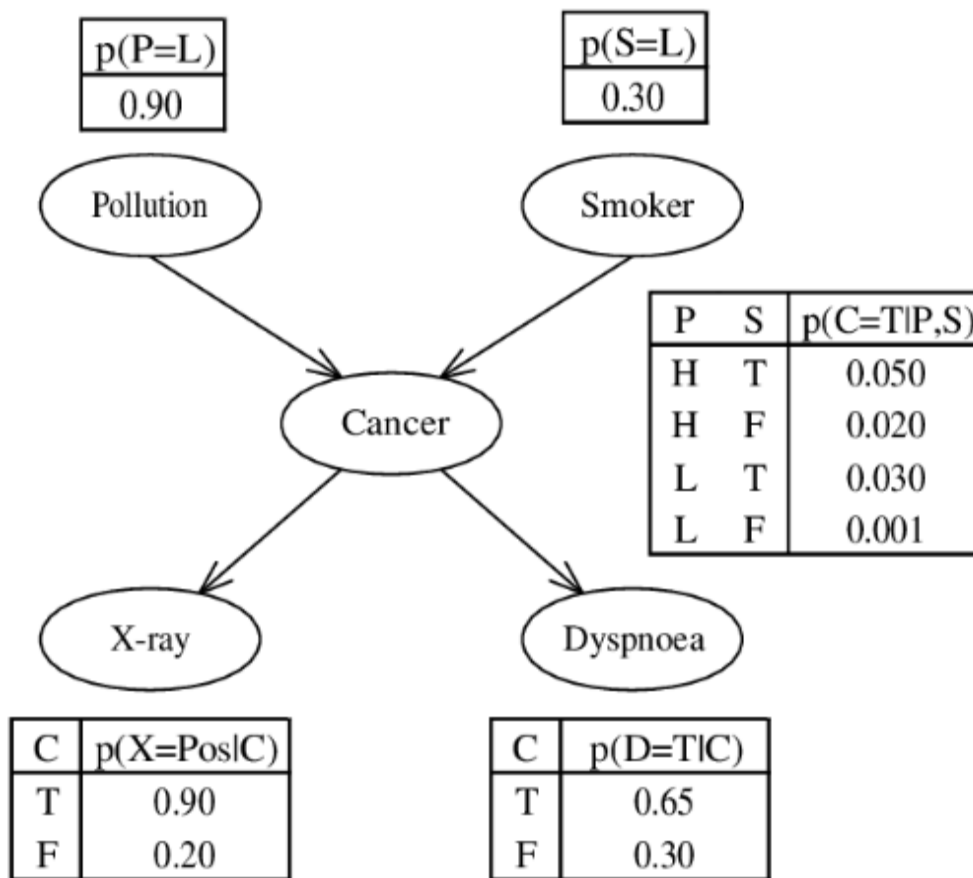
#combinaisons des masses par fusion de Dempster-Shafer
print("Dempster-Shafer Combinaison rule")
print("Dempster-Shafer Combinaison rule for m_1 and m_2 = ", m1 & m2)
print("Dempster-Shafer Combinaison rule for m_2 and m_3 = ", m2 & m3)
print("Dempster-Shafer Combinaison rule for m_1 and m_3 = ", m1 & m3)

print("Dempster-Shafer Combinaison rule for m_1, m_2 and m_3 = ", m1.combine_conjunctive(m2, m3))
print("Dempster-Shafer Combinaison rule for m_2, m_1 and m_3 = ", m2.combine_conjunctive(m1, m3))
print("Dempster-Shafer Combinaison rule for m_3, m_1 and m_2 = ", m3.combine_conjunctive(m1, m2))

[19] ✓ 0.0s
...
Dempster-Shafer Combinaison rule
Dempster-Shafer Combinaison rule for m_1 and m_2 = {'g':0.5; 'r':0.4; 'b':0.1}
Dempster-Shafer Combinaison rule for m_2 and m_3 = {'b':0.3999999999999999; 'g':0.3333333333333333; 'r':0.2666666666666666}
Dempster-Shafer Combinaison rule for m_1 and m_3 = {'r':0.7058823529411765; 'g':0.17647058823529413; 'b':0.11764705882352945}
Dempster-Shafer Combinaison rule for m_1, m_2 and m_3 = {'g':0.5; 'r':0.4; 'b':0.1}
Dempster-Shafer Combinaison rule for m_2, m_1 and m_3 = {'g':0.5; 'r':0.4; 'b':0.1}
Dempster-Shafer Combinaison rule for m_3, m_1 and m_2 = {'r':0.7058823529411765; 'g':0.17647058823529413; 'b':0.11764705882352945}

```

## TP6



construct a Bayesian network considering medical data.

```
from pgmpy.models import BayesianNetwork

cancer_model = BayesianNetwork([
    ('Pollution', 'Cancer'),
    ('Smoker', 'Cancer'),
    ('Cancer', 'X-ray'),
    ('Cancer', 'Dyspnoea')])
```

✓ 0.0s

```

print(cancer_model)
[15] ✓ 0.0s
... BayesianModel with 5 nodes and 4 edges

cancer_model.nodes()
[16] ✓ 0.0s
... NodeView(['Pollution', 'Cancer', 'Smoker', 'Xray', 'Dyspnoea'])

cancer_model.edges()
[17] ✓ 0.0s
... OutEdgeView([(('Pollution', 'Cancer'), ('Cancer', 'Xray'), ('Cancer', 'Dyspnoea'), ('Smoker', 'Cancer'))])

cancer_model.get_cpds()
[18] ✓ 0.0s
... []

```

construct a Bayesian

```

from pgmpy.models import BayesianNetwork
cancer_model = BayesianNetwork([('Pollution', 'Cancer'), ('Smoker', 'Cancer'), ('Cancer', 'Xray'), ('Cancer', 'Dyspnoea')])

```

## 7.1.2 Creation of Conditional Probability Table

```

# Définition des relations.
from pgmpy.factors.discrete import TabularCPD

cpd_poll = TabularCPD(variable='Pollution', variable_card=2,
                      values=[[0.9], [0.1]])
cpd_smoke = TabularCPD(variable='Smoker', variable_card=2,
                      values=[[0.3], [0.7]])
cpd_cancer = TabularCPD(variable='Cancer', variable_card=2,
                      values=[[0.03, 0.05, 0.001, 0.02],
                              [0.97, 0.95, 0.999, 0.98]],
                      evidence=['Smoker', 'Pollution'],
                      evidence_card=[2, 2])
cpd_xray = TabularCPD(variable='Xray', variable_card=2,
                      values=[[0.9, 0.2], [0.1, 0.8]],
                      evidence=['Cancer'], evidence_card=[2])
cpd_dysp = TabularCPD(variable='Dyspnoea', variable_card=2,
                      values=[[0.65, 0.3], [0.35, 0.7]],
                      evidence=['Cancer'], evidence_card=[2])

```

[19] ✓ 0.0s



```

# Associating the parameters with the model structure.
cancer_model.add_cpds(cpd_poll, cpd_smoke, cpd_cancer, cpd_xray, cpd_dysp)

# Checking if the cpds are valid for the model.
cancer_model.check_model()
✓ 0.0s Python

True

#trouver les chemins des noeuds pour déterminer qu'est ce qui donne des infos sur un noeud
print(cancer_model.active_trail_nodes('Pollution'))
print(cancer_model.active_trail_nodes('Smoker'))
✓ 0.0s Python

{'Pollution': {'Xray', 'Pollution', 'Dyspnoea', 'Cancer'}}
{'Smoker': {'Xray', 'Dyspnoea', 'Cancer', 'Smoker'}}

cancer_model.get_cpds()
✓ 0.0s Python

[<TabularCPD representing P(Pollution:2) at 0x29f7fcc8bd0>,
<TabularCPD representing P(Smoker:2) at 0x29f7f671150>,
<TabularCPD representing P(Cancer:2 | Smoker:2, Pollution:2) at 0x29f7fccbb90>,
<TabularCPD representing P(Xray:2 | Cancer:2) at 0x29f7fccbd50>,
<TabularCPD representing P(Dyspnoea:2 | Cancer:2) at 0x29f7fcc91d0>]

```

```

print(cancer_model.get_cpds('Pollution'))
[23] ✓ 0.0s

...
+-----+-----+
| Pollution(0) | 0.9 |
+-----+-----+
| Pollution(1) | 0.1 |
+-----+-----+

print(cancer_model.get_cpds('Smoker'))
[24] ✓ 0.0s

...
+-----+-----+
| Smoker(0) | 0.3 |
+-----+-----+
| Smoker(1) | 0.7 |
+-----+-----+

print(cancer_model.get_cpds('Xray'))
[25] ✓ 0.0s

...
+-----+-----+-----+
| Cancer | Cancer(0) | Cancer(1) |
+-----+-----+-----+
| Xray(0) | 0.9      | 0.2      |
+-----+-----+-----+
| Xray(1) | 0.1      | 0.8      |
+-----+-----+-----+

```

```

print(cancer_model.get_cpds('Dyspnoea'))
✓ 0.0s
+-----+-----+-----+
| Cancer      | Cancer(0) | Cancer(1) |
+-----+-----+-----+
| Dyspnoea(0) | 0.65      | 0.3       |
+-----+-----+-----+
| Dyspnoea(1) | 0.35      | 0.7       |
+-----+-----+-----+

print(cancer_model.get_cpds('Cancer'))
✓ 0.0s
+-----+-----+-----+-----+-----+
| Smoker      | Smoker(0) | Smoker(0) | Smoker(1) | Smoker(1) |
+-----+-----+-----+-----+-----+
| Pollution   | Pollution(0) | Pollution(1) | Pollution(0) | Pollution(1) |
+-----+-----+-----+-----+-----+
| Cancer(0)   | 0.03      | 0.05      | 0.001     | 0.02      |
+-----+-----+-----+-----+-----+
| Cancer(1)   | 0.97      | 0.95      | 0.999     | 0.98      |
+-----+-----+-----+-----+-----+

```

## 7.1.4 Determining the Local independencies

```

cancer_model.local_independencies('Xray')
[28] ✓ 0.0s
... (Xray ⊥ Pollution, Dyspnoea, Smoker | Cancer)

cancer_model.local_independencies('Pollution')
[29] ✓ 0.0s
... (Pollution ⊥ Smoker)

cancer_model.local_independencies('Smoker')
[30] ✓ 0.0s
... (Smoker ⊥ Pollution)

cancer_model.local_independencies('Dyspnoea')
[31] ✓ 0.0s
... (Dyspnoea ⊥ Xray, Pollution, Smoker | Cancer)

```

## TP7

Étapes de la conception d'un contrôleur flou : nous allons faire l'exercice sous forme vision

La conception d'un contrôleur flou consiste en 5 étapes distinctes qui sont :

- Définition des E/S du contrôleur.

### 1) Définition des E/S du controlleur

```
# initialisation de notre premier controleur d'entrée
Resolution = Domain('Resolution', 0, 100)

# initialisation de notre deuxieme controleur d'entrée
Luminosite = Domain('Luminosite', 9, 70)

# initialisation de notre troisième controleur d'entrée
QualiteImage = Domain('QualiteImage', 5, 50)

#Sortie
ContrasteImage = Domain('ContrasteImage', 0, 100)
```

✓ 0.0s

Pyth

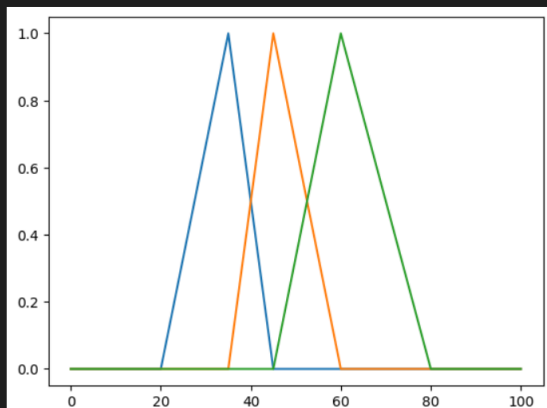
- Subdivision des E/S en sous ensembles flou (Fuzzification).

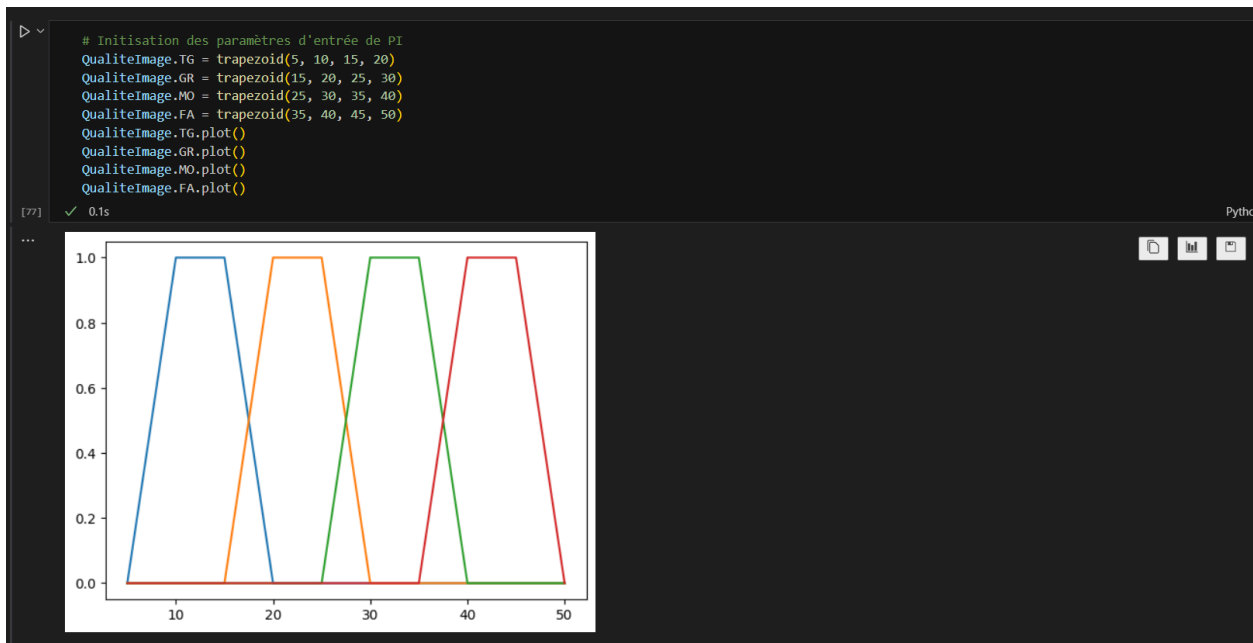
### 2) Fuzification

```
# Initiation des paramètres d'entrée de Resolution
Resolution.AV = trapezoid(20, 35, 35, 45)
Resolution.AC = trapezoid(35, 45, 45, 60)
Resolution.IN = trapezoid(45, 60, 60, 80)
Resolution.AV.plot()
Resolution.AC.plot()
Resolution.IN.plot()
```

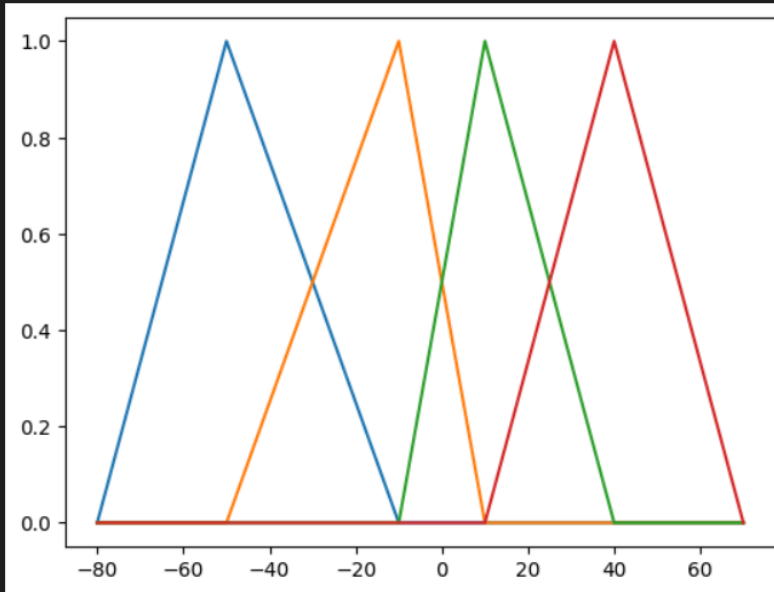
✓ 0.1s

Pyth





```
# Initiation des paramètres de sortie de RC
RC.TF = trapezoid(-80, -50, -50, -10)
RC.FO = trapezoid(-50, -10, -10, 10)
RC.MO = trapezoid(-10, 10, 10, 40)
RC.FA = trapezoid(10, 40, 40, 70)
RC.TF.plot()
RC.FO.plot()
RC.MO.plot()
RC.FA.plot()
```



- Définition de la base de règle floue.

```
# initialisation des regles de notre systeme
rules = Rule([
    (Resolution.AV, Luminosite.DN, QualiteImage.TG): ContrasteImage.FA, # R1
    (Resolution.AV, Luminosite.DN, QualiteImage.GR): ContrasteImage.MO, # R2
    (Resolution.AV, Luminosite.DN, QualiteImage.MO): ContrasteImage.FO, # R3
    (Resolution.AV, Luminosite.DN, QualiteImage.FA): ContrasteImage.TF, # R4
    (Resolution.AV, Luminosite.HN, QualiteImage.TG): ContrasteImage.FA, # R5
    (Resolution.AV, Luminosite.HN, QualiteImage.GR): ContrasteImage.MO, # R6
    (Resolution.AV, Luminosite.HN, QualiteImage.MO): ContrasteImage.FO, # R7
    (Resolution.AV, Luminosite.HN, QualiteImage.FA): ContrasteImage.TF, # R8
    (Resolution.AC, Luminosite.DN, QualiteImage.TG): ContrasteImage.FA, # R9
    (Resolution.AC, Luminosite.DN, QualiteImage.GR): ContrasteImage.MO, # R10
    (Resolution.AC, Luminosite.DN, QualiteImage.MO): ContrasteImage.FO, # R11
    (Resolution.AC, Luminosite.DN, QualiteImage.FA): ContrasteImage.TF, # R12
    (Resolution.AC, Luminosite.HN, QualiteImage.TG): ContrasteImage.FA, # R13
    (Resolution.AC, Luminosite.HN, QualiteImage.GR): ContrasteImage.MO, # R14
    (Resolution.AC, Luminosite.HN, QualiteImage.MO): ContrasteImage.FO, # R15
    (Resolution.AC, Luminosite.HN, QualiteImage.FA): ContrasteImage.TF, # R16
    (Resolution.IN, Luminosite.DN, QualiteImage.TG): ContrasteImage.FA, # R17
    (Resolution.IN, Luminosite.DN, QualiteImage.GR): ContrasteImage.MO, # R18
    (Resolution.IN, Luminosite.DN, QualiteImage.MO): ContrasteImage.FO, # R19
    (Resolution.IN, Luminosite.DN, QualiteImage.FA): ContrasteImage.TF, # R20
    (Resolution.IN, Luminosite.HN, QualiteImage.TG): ContrasteImage.FA, # R21
    (Resolution.IN, Luminosite.HN, QualiteImage.GR): ContrasteImage.MO, # R22
    (Resolution.IN, Luminosite.HN, QualiteImage.MO): ContrasteImage.FO, # R23
    (Resolution.IN, Luminosite.HN, QualiteImage.FA): ContrasteImage.TF # R24
])
```

- Application de la méthode d'inférence.

```
# On récupère les pourcentages d'appartenances de nos controleurs d'entrée selon leur valeur
tc_output = list(Resolution(52).values())
tc_output = [float(x) for x in tc_output]

nc_output = list(Luminosite(42).values())
nc_output = [float(x) for x in nc_output]

pi_output = list(QualiteImage(17).values())
pi_output = [float(x) for x in pi_output]

# On applique les règles d'inférence de mandanie
rc_tf = max(
    min(tc_output[2], nc_output[1], pi_output[2]),
    min(tc_output[0], nc_output[1], pi_output[3]),
    min(tc_output[2], nc_output[1], pi_output[3]),
    min(tc_output[1], nc_output[1], pi_output[3])
)

rc_fo = max(
    min(tc_output[0], nc_output[0], pi_output[2]),
    min(tc_output[1], nc_output[0], pi_output[2]),
    min(tc_output[2], nc_output[0], pi_output[2]),
    min(tc_output[1], nc_output[1], pi_output[1]),
    min(tc_output[2], nc_output[1], pi_output[1]),
    min(tc_output[1], nc_output[1], pi_output[2]),
    min(tc_output[0], nc_output[1], pi_output[2])
)

rc_mo = max(
    min(tc_output[0], nc_output[1], pi_output[0]),
    min(tc_output[1], nc_output[1], pi_output[0]),
    min(tc_output[2], nc_output[1], pi_output[0]),
    min(tc_output[0], nc_output[1], pi_output[1]),
    min(tc_output[1], nc_output[0], pi_output[1]),
    min(tc_output[2], nc_output[0], pi_output[1]),
    min(tc_output[0], nc_output[0], pi_output[2]),
    min(tc_output[1], nc_output[0], pi_output[2]),
    min(tc_output[2], nc_output[0], pi_output[2])
)

rc_fa = max(
    min(tc_output[0], nc_output[0], pi_output[0]),
    min(tc_output[1], nc_output[0], pi_output[0]),
    min(tc_output[2], nc_output[0], pi_output[0]),
    min(tc_output[0], nc_output[0], pi_output[1])
)

all_values = [rc_tf, rc_fo, rc_mo, rc_fa]

print(f"ContrasteImage tres fort {rc_tf}\nContrasteImage {rc_fo}\nContrasteImage moyen {rc_mo}\nContrasteImage faible {rc_fa}")
```

0) ✓ 0.0s

ContrasteImage tres fort 0.0  
ContrasteImage 0.13333333333333333  
ContrasteImage moyen 0.4  
ContrasteImage faible 0.5333333333333333

- Application de la défuzzification.

