

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique



Université des Sciences et de la Technologie Houari Boumédiène

Faculté d'Electronique et d'Informatique
Département Informatique

Master Systèmes Informatiques intelligents

Module : Représentation des connaissances et raisonnement.

Rapport global de TP

Réalisé par :
BOUROUNA Rania, 181831052716
CHIBANE Ilies, 181831072041

Année universitaire : 2021 / 2022

Table des matières

1	Introduction Générale	2
2	TP 1 : Inférence logique basée sur un solveur SAT	3
3	TP 2 : Logique Modale	15
4	TP 3 : Logique des défauts	22
5	TP 4 : Réseaux sémantiques	42
6	TP 5 : Logique des descriptions	48
7	Conclusion Générale	57

Chapitre 1

Introduction Générale

La représentation des connaissances désigne un ensemble d'outils et de procédés destinés d'une part à représenter et d'autre part à organiser le savoir humain pour l'utiliser et le partager. Il s'agit d'un domaine de l'intelligence artificielle (IA) consacré à la représentation des informations sur le monde sous une forme qu'un système informatique peut utiliser pour résoudre des tâches complexes. Dans cette série de TPs nous allons voir comment peut on faire ceci tout en découvrant les différentes logiques et les types de raisonneurs qui font des déductions dans ces dernières.

Chapitre 2

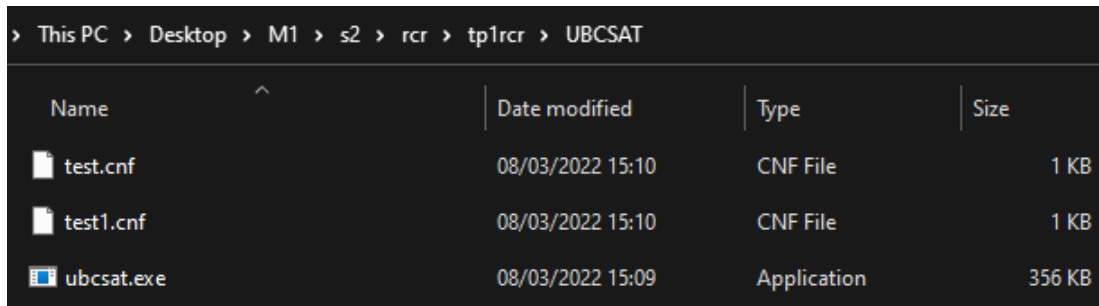
TP 1 : Inférence logique basée sur un solveur SAT

Résumé

Dans ce premier TP, nous allons d'abord utiliser un solveur SAT pour étudier la satisfiabilité de quelques Bases de Connaissance. Nous allons également traduire une BC relative aux connaissances zoologiques, tester sur des Benchmarking et simuler l'inférence d'une BC avec un algorithme.

Étape 1 : Création du répertoire

Dans l'étape 1, on crée un dossier contenant le SAT UBCSAT et on y copie des fichiers sous format CNF



This PC > Desktop > M1 > s2 > rcr > tp1rcr > UBCSAT			
Name	Date modified	Type	Size
test.cnf	08/03/2022 15:10	CNF File	1 KB
test1.cnf	08/03/2022 15:10	CNF File	1 KB
ubcsat.exe	08/03/2022 15:09	Application	356 KB

FIGURE 1 – Le répertoire UBCSAT

Étape 2 : Exécution du solveur SAT

Dans l'étape 2, on exécute le solveur SAT et teste de la satisfiabilité des deux fichiers : test.cnf et test1.cnf

3.1 Le fichier Test.cnf

Pour L'essai numéro 1 Nous avons exécuter la commande suivantes (avec test.cnf le fichier contenant les règles SAT).

```
1 ubcsat -alg saps -i test.cnf -solve
```

Listing 1 – SAT test

Voici le résultat.

```

C:\Users\Ranya BR\Desktop\M1\s2\rcr\tp1rcr\UBCSAT>ubcsat -alg saps -i test.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 484785415
# -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
#

```

FIGURE 2 – Résultats du solveur SAT, partie 1 (test.cnf)

Cet affichage représente les informations sur la version de l'UBCSAT, un site et l'instruction d'aide, ainsi que plusieurs paramètres de notre Solveur.

```

# 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
#

```

FIGURE 3 – Résultats du solveur SAT, partie 2 (test.cnf)

L'affichage de la partie 2 représente le rapport de résultat d'exécution (sortie).

```

#
# Solution found for -target 0
#
# 1 2 -3 -4 5
#

```

FIGURE 4 – Résultats du solveur SAT, partie 3 (solution test.cnf)

Cette partie affiche la sortie de l'UBCSAT. Puisque le solveur a trouvé une solution, nous pouvons dire que le fichier « test.cnf » est satisfiable. La solution trouvée est :

$$a \vee \neg b \vee \neg c \vee \neg d \vee \neg e \quad (1)$$

```

Variables = 5
Clauses = 11
TotalLiterals = 27
TotalCPUtimeElapsed = 0.001
FlipsPerSecond = 7001
RunsExecuted = 1
SuccessfulRuns = 1
PercentSuccess = 100.00
Steps_Mean = 7
Steps_CoeffVariance = 0
Steps_Median = 7
CPUtime_Mean = 0.000999927520752
CPUtime_CoeffVariance = 0
CPUtime_Median = 0.000999927520752

```

FIGURE 5 – Résultats du solveur SAT, partie 4 (rapport test.cnf)

La dernière partie, représente un rapport de statistiques de l'exécution. Parmi les paramètres contenus ici que nous avons vu en cours il y a, le nombre de variables (ici 5) ainsi que le nombre de clauses (ici 11). Il y a aussi le pourcentage de réussite (PercentSuccess) qui est 100% car la base de connaissance est satisfiable et donc exploitable.

3.2 Le fichier Test1.cnf

Pour Le fichier test1.cnf, Nous avons exécuté la commande suivantes (avec test1.cnf le fichier contenant les règles SAT du deuxième exemple donné dans le TP).

```
1 ubcsat -alg saps -i test1.cnf -solve
```

Listing 2 – SAT

Voici le résultats.

```

C:\Users\Ranya BR\Desktop\M1\s2\rcr\tp1rcr\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
# -wtarget 0
# -seed 485827900
# -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

```

FIGURE 6 – Résultats du solveur SAT pour le deuxième exemple (test1.cnf)

```

# 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      2      100000
# No Solution found for -target 0

Variables = 5
Clauses = 13
TotalLiterals = 29
TotalCPUtimeElapsed = 0.009
FlipsPerSecond = 11111328
RunsExecuted = 1
SuccessfulRuns = 0
PercentSuccess = 0.00
Steps_Mean = 100000

```

FIGURE 7 – Résultats du solveur SAT pour le deuxième exemple (test1.cnf, aucune solution trouvée)

Après l'exécution du solveur sur "test2.cnf", le message (No Solution found) s'affiche signifiant que la base n'est pas satisfiable et donc non exploitable. Notez que PercentSuccess ici est 0.00.

Étape 3 :

Ci-dessous, les énoncés vu en cours :

- Les nautilus sont des céphalopodes ;
- Les céphalopodes sont des mollusques ;
- Les mollusques ont généralement une coquille ;
- Les céphalopodes n'en ont généralement pas ;
- Les nautilus en ont une.
- a est un nautilus,
- b est un céphalopode,
- c est un mollusque.

4.1 Partie 1 : Traduction la base de connaissances

Nos non logiques :

Na, Nb, Nc ; Où Na = Nautilus de a

Cea, Ceb, Cec ; Où Cea = Céphalopode de a

Ma, Mb, Mc ; Où Ma = Mollusque de a

Coa, Cob, Coc ; Où Coa = Coquille de a

Même chose en ce qui concerne b et c.

En ignorant les connaissances utilisant le mot « généralement » (vu en cours) :

1 - Les nautilus sont des céphalopodes.

$$(Na \supset Cea); (Nb \supset Ceb); (Nc \supset Cec);$$

2 - Les céphalopodes sont des mollusques.

$$(Cea \supset Ma); (Ceb \supset Mb); (Cec \supset Mc);$$

4- Les mollusques ont une coquille. on a enlevé le mot généralement

$$(Ma \supset Coa); (Mb \supset Cob); (Mc \supset Coc);$$

5- Les nautilus en ont une.(coquille)

$$(Na \supset Coa); (Nb \supset Cob); (Nc \supset Coc)$$

6- Les céphalopodes n'en ont pas.(coquille) on a enlevé le mot généralement

$$(Cea \supset \neg Coa); (Ceb \supset \neg Cob); (Cec \supset \neg Coc)$$

7- a est un nautilus, b est un céphalopode, c est un mollusque.

$$Na; Ceb; Mc$$

Si on ignore totalement le mot « généralement », notre système sera incohérent. Donc on procède à la transformation de ces clauses en CNF en transformant l'implication en une disjonction :

Solution : Application de la règle : $a \supset b \equiv \neg a \vee b$

7- a est un nautilus, b est un céphalopode, c est un mollusque.

$$Na$$

$$Ceb$$

$$Mc$$

1 - Les nautilus sont des céphalopodes.

$$(\neg Na \vee Cea); (\neg Nb \vee Ceb); (\neg Nc \vee Cec)$$

2 - Les céphalopodes sont des mollusques.

$$(\neg Cea \vee Ma); (\neg Ceb \vee Mb); (\neg Cec \vee Mc)$$

5- Les nautilus en ont une.(coquille)

$$(\neg Na \vee Coa); (\neg Nb \vee Cob); (\neg Nc \vee Coc)$$

4- Les mollusques ont une coquille. (on a enlevé le mot généralement)

$$(\neg Ma \vee Coa); (\neg Mb \vee Cob); (\neg Mc \vee Coc)$$

6- Les céphalopodes n'en ont pas.(coquille) on a enlevé le mot généralement

$$(\neg Cea \vee \neg Coa); (\neg Ceb \vee \neg Cob); (\neg Cec \vee \neg Coc)$$

On remplace les clauses qui comportaient le mot « généralement » avec une autre interprétation (les clauses 4 et 6) en utilisant les autres clauses :

$$(\neg Na \vee Cea); (\neg Nb \vee Ceb); (\neg Nc \vee Cec)$$

$$(\neg Cea \vee Ma); (\neg Ceb \vee Mb); (\neg Cec \vee Mc)$$

$$(\neg Na \vee Coa); (\neg Nb \vee Cob); (\neg Nc \vee Coc)$$

$$(\neg Ma \vee Cea \vee Coa); (\neg Ma \vee \neg Na \vee Coa)$$

$$(\neg Mb \vee Ceb \vee Cob); (\neg Mb \vee \neg Nb \vee Cob)$$

$$(\neg Mc \vee Cec \vee Coc); (\neg Mc \vee \neg Nc \vee Coc)$$

$$(\neg Cea \vee Na \vee \neg Coa)$$

$$(\neg Ceb \vee Nb \vee \neg Cob)$$

$$(\neg Cec \vee Nc \vee \neg Coc)$$

$$Na; Ceb; Mc$$

Nous avons 12 variables et 21 clauses au total. Représentation dans le fichier :

1 = Na; 2 = Nb; 3 = Nc;

4 = Cea; 5 = Ceb; 6 = Cec;

7 = Coa; 8 = Cob; 9 = Coc;

10 = Ma; 11 = Mb; 12 = Mc;

```

solu.cnf - Notepad
File Edit View

p cnf 12 21
-1 4 0
-2 5 0
-3 6 0
-4 10 0
-5 11 0
-6 12 0
-10 4 7 0
-10 -1 7 0
-11 5 8 0
-11 -2 8 0
-12 6 9 0
-12 -3 9 0
-4 1 -7 0
-5 2 -8 0
-6 3 -9 0
-1 7 0
-2 8 0
-3 9 0
1 0
5 0
12 0

```

FIGURE 8 – Contenu du fichier cnf

NB : nous avons 12 variables et 21 clauses. Si nous n'avons pas rajouté les autres clauses en réinterprétant les clauses avec «généralement», la BC n'aurait pas été satisfiable avec les 18 clauses initiales et donc non exploitable.

```

#          F   Best      Step      Total
#          Run N Sol'n    of        Search
#          No. D Found    Best      Steps
#
#          1 1      0      12        12
#
# Solution found for -target 0
1 -2 -3 4 5 6 7 -8 -9 10
11 12

Variables = 12
Clauses = 21
TotalLiterals = 48
TotalCPUtimeElapsed = 0.002
FlipsPerSecond = 6000
RunsExecuted = 1
SuccessfulRuns = 1
PercentSuccess = 100.00
Steps_Mean = 12
Steps_CoeffVariance = 0
Steps_Median = 12
CPUtime_Mean = 0.0019998550415
CPUtime_CoeffVariance = 0
CPUtime_Median = 0.0019998550415

C:\Users\Ranya BR\Desktop\M1\s2\rcr\tp1rcr\UBCSAT>

```

FIGURE 9 – Résultats du solveur SAT pour le fichier

Solution : $Na \wedge \neg Nb \wedge \neg Nc \wedge Cea \wedge Ceb \wedge CeC \wedge Coa \wedge \neg Cob \wedge \neg CoC \wedge Ma \wedge Mb \wedge Mc$

4.1.a Teste de Benchmarking

On télécharge deux fichiers : un satisfiable et un non-satisfiable, et nous le testons.

```

C:\Users\Ranya BR\Desktop\M1\s2\rcr\tp1rcr\UBCSAT>ubcsat -alg saps -i uuf75-01.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 496330784
# -solve 1
# -find,-numsol 1
# -findunique 0
# -srestart 0
# -prestart 0
# -drestart 0

```

FIGURE 10 – Résultats du solveur SAT pour le premier fichier, partie 1

```

1 0 1 895 100000
# No Solution found for -target 0

Variables = 75
Clauses = 325
TotalLiterals = 975
TotalCPUtimeElapsed = 0.039
FlipsPerSecond = 2564100
RunsExecuted = 1
SuccessfulRuns = 0
PercentSuccess = 0.00
Steps_Mean = 100000
Steps_CoeffVariance = 0

```

FIGURE 11 – Résultats du solveur SAT pour le premier fichier, partie 2

La base est donc non salifiable.

```

C:\Users\Ranya BR\Desktop\M1\s2\rcr\tp1rcr\UBCSAT>ubcsat -alg saps -i uf75-01.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 496807074
# -solve 1
# -find,-numsol 1
# -findunique 0
# -srestart 0
# -prestart 0
# -drestart 0

```

FIGURE 12 – Résultats du solveur SAT pour le deuxième fichier, partie 1

```
#
# 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      144      144
#
# 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
```

FIGURE 13 – Résultats du solveur SAT pour le deuxième fichier, partie 2

La base est satisfiable.

Étape 4 : simulation de l'inférence d'une base de connaissances

En utilisant le le raisonnement par l'absurde, nous allons tester si une BC infère un but donné en optant pour le solveur UBSAT pour le teste de satisfiabilité d'une base.

5.1 Déroulement

- On prend le fichier de la BC choisie.
- On ouvre un fichier temporaire (pour copier la BC) et on y met les information de la BC (nombre de variable et nombre de clauses +1).
- Nous avons incrémenté le nombre de clauses pour rajouter le non_but dans le fichier à traiter (avec un "0" à la fin comme toutes les clauses).
- On execute les Solveur SAT et on affiche le résultat.

5.2 Tests

```
PS C:\Users\Ranya BR\Desktop\MI\s2\rcr\tp1rcr\UBCSAT> gcc main.c -o main.exe -stdc++
PS C:\Users\Ranya BR\Desktop\MI\s2\rcr\tp1rcr\UBCSAT> ./main
Entrez le nom de votre Fichier BC (sans extension) :
Zootest
Liste des variables de la BC (litteraux):
1: Na;      2: Nb;      3: Nc;
4: Cea;     5: Ceb;     6: Cec;
7: Coa;     8: Cob;     9: Coc;
10: Ma;     11: Mb;     12: Mc;

Donnez le nombre de litteraux :
1
Entrez le literal 1 :
1

BC U {Non but} est non satisfiable. La base n'infere pas le but.
-1 ne peut pas etre atteint
PS C:\Users\Ranya BR\Desktop\MI\s2\rcr\tp1rcr\UBCSAT> []
```

FIGURE 14 – Résultats du programme, cas d'echec

```

PS C:\Users\Ranya BR\Desktop\M1\s2\rcr\tp1rcr\UBCSAT> ./main
Entrez le nom de votre Fichier BC (sans extension) :
solu
Liste des variables de la BC (littereaux):
1: Na;      2: Nb;      3: Nc;
4: Cea;     5: Ceb;     6: Cec;
7: Coa;     8: Cob;     9: Coc;
10: Ma;     11: Mb;     12: Mc;

Donnez le nombre de littereaux :
1
Entrez le literal 1 :
3

BC U {Non but} est satisfiable. La base infere le but.
Solution :
1 -2 3 4 5 6 7 -8 9 10
11 12

```

FIGURE 15 – Résultats du programme, cas de réussite

5.3 Code Source

```

1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5
6 int main(){
7     FILE * fich_base = NULL;
8     FILE * fich_temp = NULL;
9     int nbr_propositions, nbr_variables, nbr_clauses, i, but[20], non_but[20];
10    char nom_base[20], c;
11
12
13    printf("Entrez le nom de votre Fichier BC (sans extension) :\n");
14    gets(nom_base);
15    strcat(nom_base, ".cnf");
16    fich_base = fopen(nom_base, "r+");
17
18    if(fich_base == NULL)
19        printf("Impossible d'ouvrir BC...\n");
20    else{
21
22        fich_temp = fopen("Temp.cnf", "rw+");
23
24        if(fich_temp == NULL)
25            printf("Impossible de transferer BC \n");
26        else{
27            fscanf(fich_base, "p cnf %d %d ", &nbr_variables, &nbr_clauses); // on prend la premi re
28            ligne de la BC
29            nbr_clauses += 1 ; // pour rajouter le non_but
30            fprintf (fich_temp, "p cnf %d %d ""\n", nbr_variables, nbr_clauses); // on met les infos
31            dans le fichier qu'on va traiter
32        }
33        c = fgetc(fich_base);
34        while(c!= EOF){
35            c = fgetc(fich_base);
36            if(c!= EOF) fputc(c, fich_temp);
37        } // on a recopier le reste de la BC dans le fichier qu'on va traiter
38
39        printf("Liste des variables de la BC (littereaux):\n "); //afficher les codes pour l'utilisateur
40        printf("1: Na;\t\t 2: Nb;\t\t 3: Nc;\t \n 4: Cea;\t 5: Ceb;\t 6: Cec;\t \n 7: Coa;\t 8: Cob;\t
41        9: Coc;\t \n 10: Ma;\t 11: Mb;\t 12: Mc;\t\n\n");
42        printf("Donnez le nombre de littereaux : \n"); //prendre les buts inferer
43        scanf("%d", &nbr_propositions);
44
45        for(i = 1; i<nbr_propositions+1; i++){
46
47            printf("Entrez le literal %d : \n", i);
48            scanf("%d", &but[i]);
49            if(but[i]>-13 && but[i]<13)

```

```

48     non_but[i] = but[i]*(-1); //si les codes sot corrects on prend la negation du but (
absurde)
49     else
50         puts("Erreur, Vous avez entrer un code invalide");
51 }
52
53
54 fprintf(fich_temp, "\n"); //completion du fichier traiter
55 for(i = 1; i<nbr_propositions+1; i++)fprintf(fich_temp,"%d ",non_but[i]); //Ajout des negations
au fichier
56 fprintf(fich_temp,"0"); //Ajout des 0 pour marquer la fin
57
58 system("ubcsat -alg saps -i Temp.cnf -solve > results.txt"); //execution solveur
59 }
60 fclose(fich_temp);
61
62
63 int termine = 0; //signaler fin affichage
64 FILE *fich = fopen("results.txt","r+");
65 if(fich_base == NULL) printf("Impossible d'accéder aux resultats...\n");
66 else{
67     char texte[1000];
68     while(fgets(texte, 1000, fich) && !termine){
69         if(strstr(texte, "# Solution found for -target 0")){
70
71             printf("\n BC U {Non but} est satisfiable. La base infere le but. \n Solution : \n");
72
73             fscanf(fich, "\n");
74             while(!strstr(fgets(texte, 1000, fich), "Variables"))
75                 printf("%s", texte);
76             termine = 1;
77         }
78         if(termine == 0){ //cas de non satisfiablit
79             printf("\n BC U {Non but} est non satisfiable. La base n'infere pas le but.\n");
80             int j;
81             for(j = 1; j<nbr_propositions+1; j++){
82                 printf("%d ",(-1)*but[j]);
83             }
84             if(j>2)
85                 printf("ne peuvent pas etre atteints");
86             else
87                 printf("ne peut pas etre atteint\n");
88         }
89     }
90
91     fclose(fich_temp);
92 }

```

Chapitre 3

TP 2 : Logique Modale

Résumé

Dans ce deuxième TP, nous allons simplement vérifier la véracité de certaines formules en utilisant plusieurs outils :

- Modal Logic Playground.
- La librairie Java tweety.

Enoncé :

TP :

Vérifiez la véracité des formules en utilisant :

- l'outil Modal Logic Playground <https://rkirsling.github.io/modallogic/>
- La librairie Java tweety dédiée aux modes logiques dans le domaine de la représentation des connaissances (Logique Propositionnelle, Logique des prédicats, Logique modale, Logique des défauts et Logique de description)
<https://tweetyproject.org/>

FIGURE 1 – Enoncé du Tp

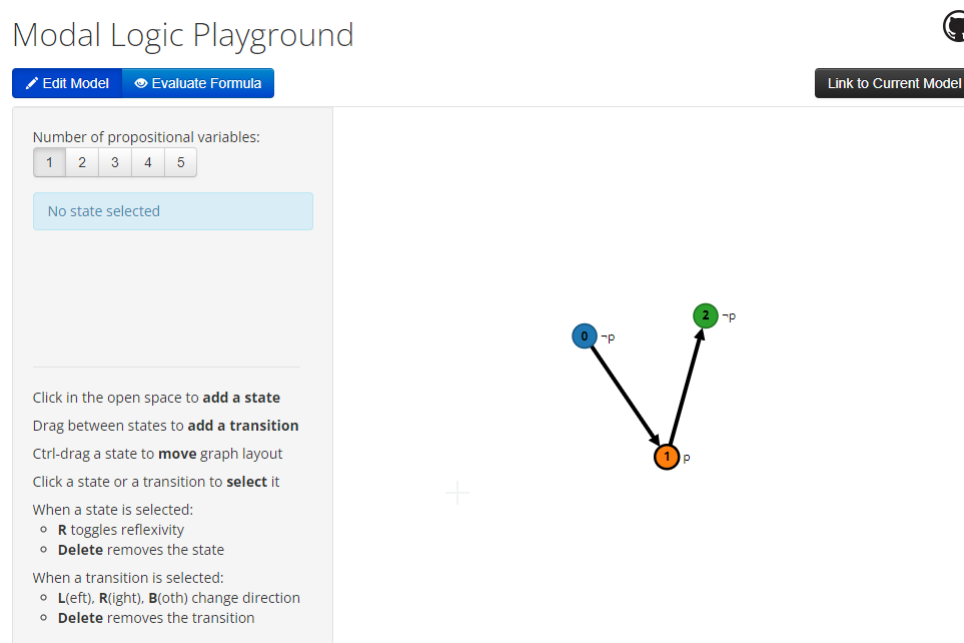


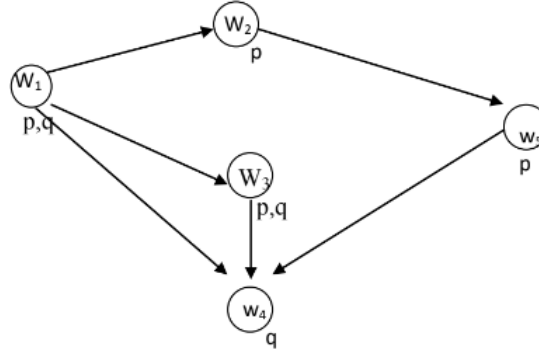
FIGURE 2 – L'interface de Modal Logic Playground

Modal Logic Playground : Exercice 2 du TD

Voici l'exercice 2 :

Exercice 2:

- 1- Spécifier les assertions vraies dans le modèle suivant avec la spécificité que $M, x \models \neg B$ ssi non $(M, x \models B)$.



- a- $M, w_1 \models \Diamond(p \wedge q)$
- b- $M, w_2 \models \neg \Box p$
- c- $M, w_3 \models \Box(p \supset q)$
- d- $M, w_4 \models \Box(q \wedge \Diamond \neg p)$
- e- $M, w_5 \models \Box(q \wedge \Diamond \neg p)$

FIGURE 3 – Enoncé de l'exercice 2

Nous allons d'abord commencer par dessiner le modèle de l'exercice

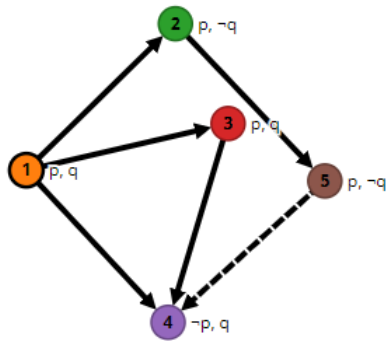


FIGURE 4 – Modèle de l'exercice 2

Nous allons commencer par la formule suivante :

- $M, w_1 \models \Diamond(p \wedge q)$

Enter a formula:

Evaluate

True:
 w_1

False:
 w_2, w_3, w_4, w_5

When entering a formula:

- use $\neg A$ for $\neg A$
- use $\Box A$ for $\Box A$
- use $\Diamond A$ for $\Diamond A$
- use $(A \& B)$ for $(A \wedge B)$
- use $(A | B)$ for $(A \vee B)$
- use $(A \rightarrow B)$ for $(A \rightarrow B)$
- use $(A \leftrightarrow B)$ for $(A \leftrightarrow B)$

Current formula:
 $\Diamond(p \wedge q)$

FIGURE 5 – Formule 1 de l'exercice 2

Explication :

La formule est vraie car $(p \wedge q)$ est vrai en w_3 (accessible à partir de w_1 et $w_1 R w_3$). Ceci correspond bien à la réponse du TD.

La formule :

- $M, w_2 \models \neg \Box p$

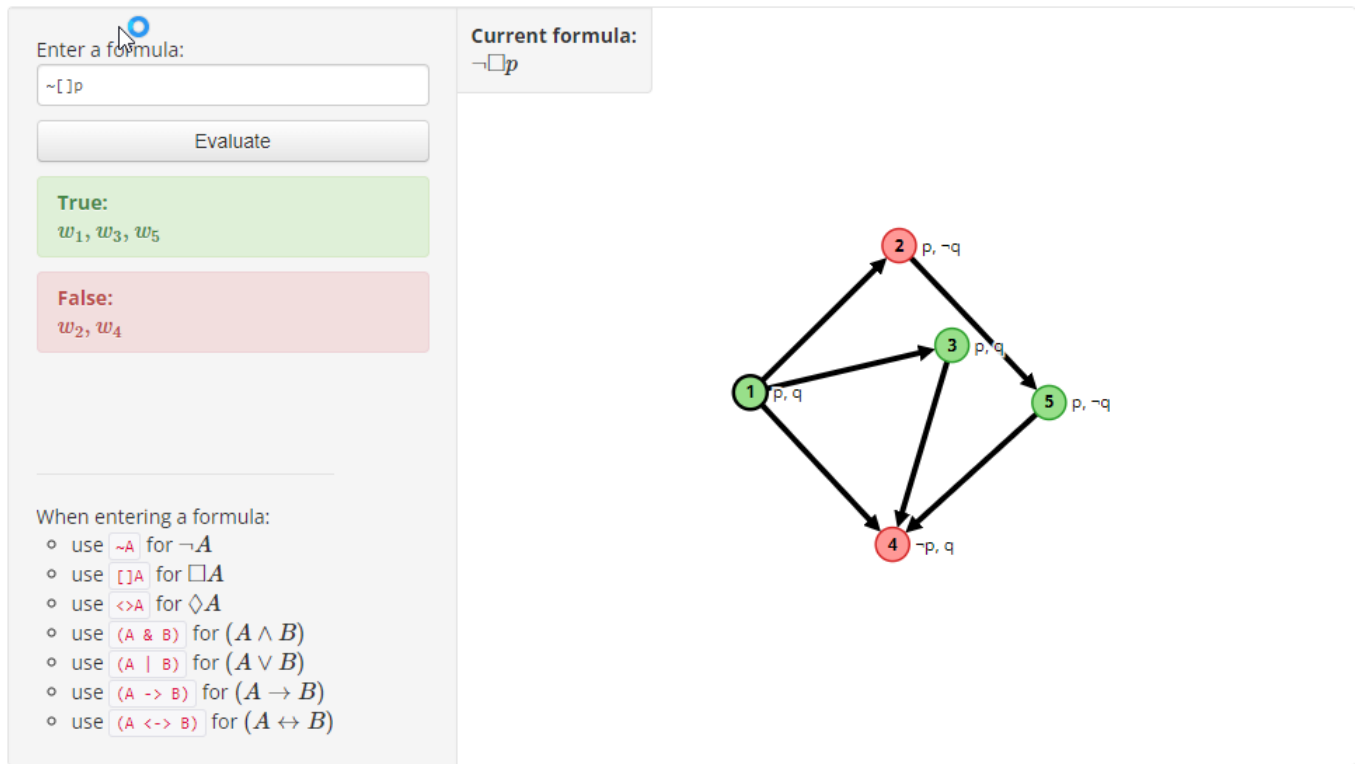


FIGURE 6 – Formule 2 de l'exercice 2

Explication :

La formule est fausse en w_2 car le seul monde accessible à partir de ce dernier est w_5 dans lequel p est vraie. Ceci correspond bien aux résultats obtenus au TD. La Formule :

- $M, w_3 \models \Box(p \supset q)$

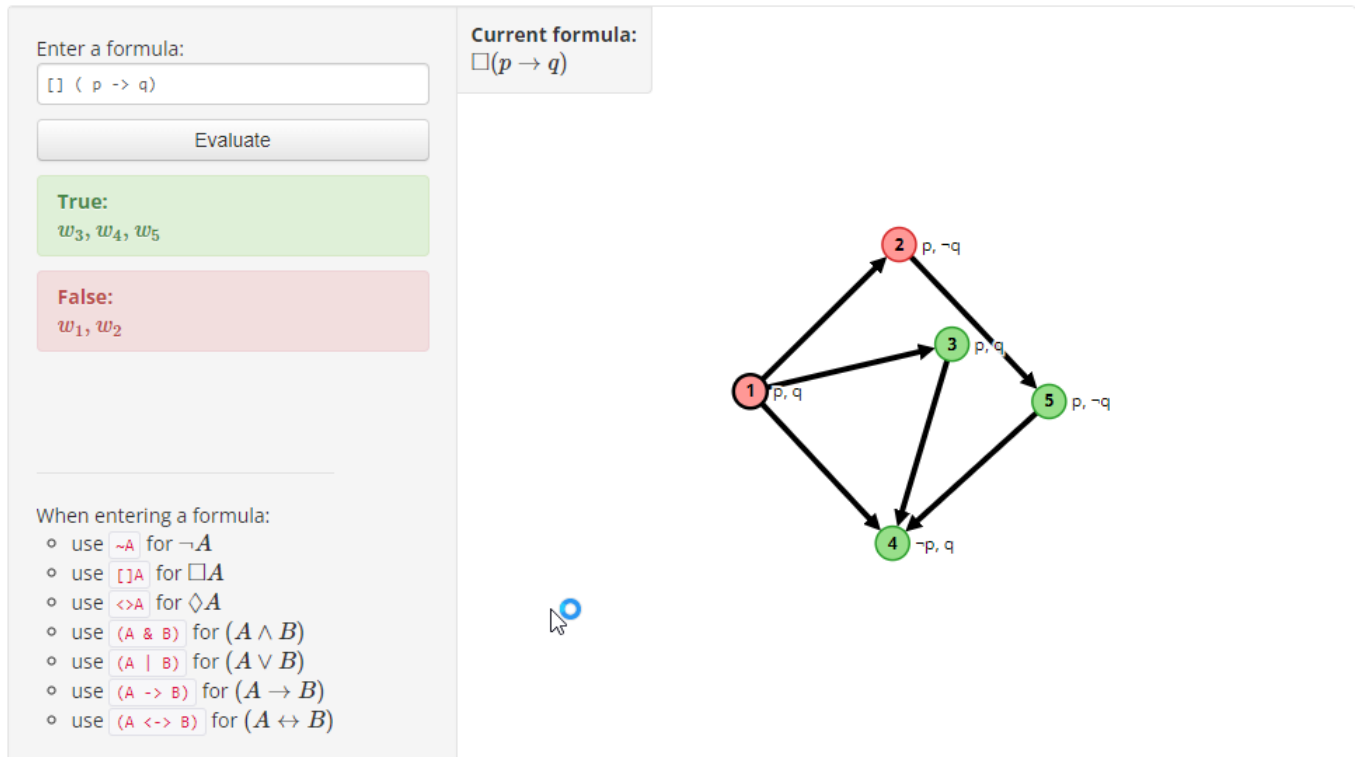


FIGURE 7 – La formule 3 de l'exercice 2

Explication :

w_3 accède seulement à w_4 où l'expression $p \supset q$ est vraie. la formule est donc vraie. Ceci correspond bien aux résultats obtenus au TD.

Les formules :

- $M, w_4 \models \Box(q \wedge \Diamond \neg p)$
- $M, w_5 \models \Box(q \wedge \Diamond \neg p)$

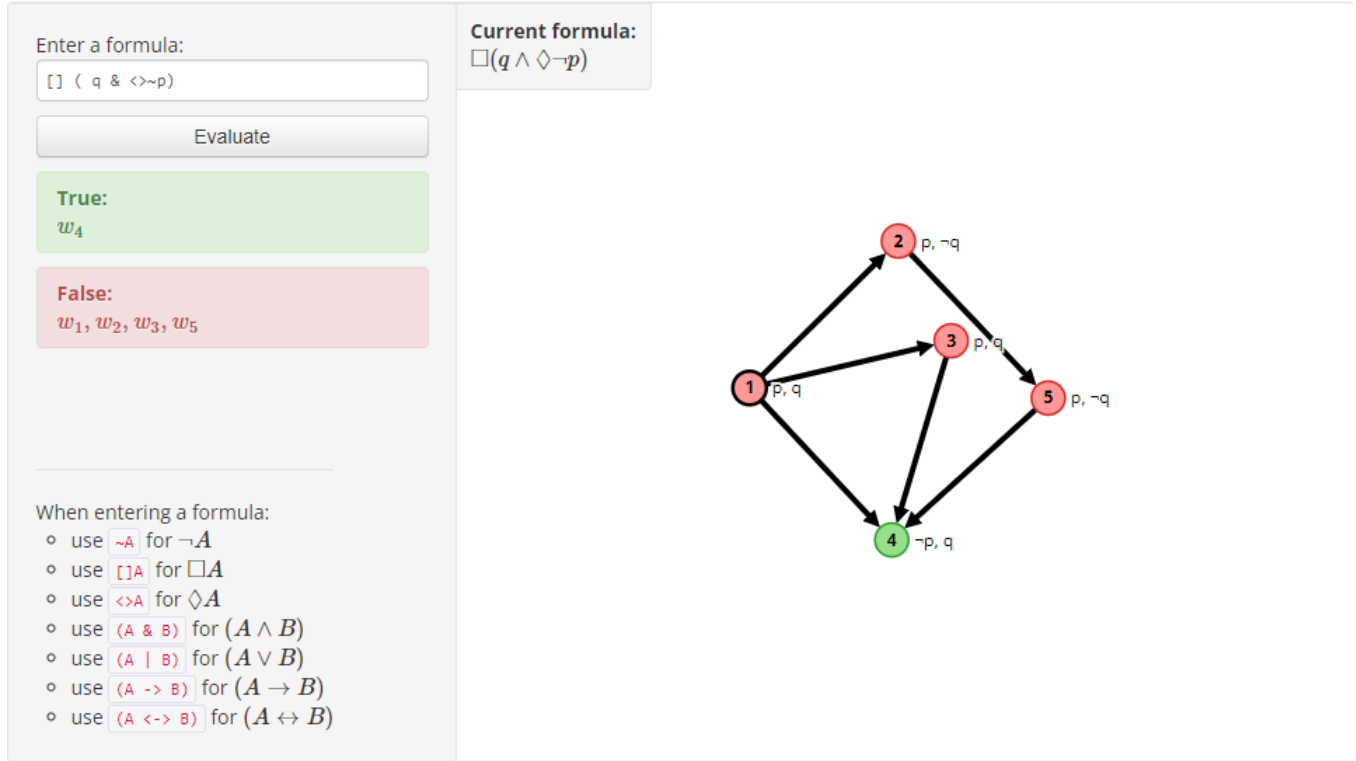


FIGURE 8 – Les Formules 4 et 5 de l'exercice 2

Explication :

La formule est vraie en seulement w_4 car il n'existe aucun monde accessible depuis w_4 . Ceci correspond bien aux résultats obtenus au TD.

NB : nous allons nous contenter des formules de l'exercice 2 car le type de logique modale est classique et représentable sur les outils proposés.

La librairie Java tweety : Exercice 2 du TD

Ci-dessous, les formules vu en TD :

- 1- $M, w_1 \models \Diamond(p \wedge q)$
- 2- $M, w_2 \models \neg \Box p$
- 3- $M, w_3 \models \Box(p \supset q)$
- 4- $M, w_4 \models \Box(q \wedge \Diamond \neg p)$
- 5- $M, w_5 \models \Box(q \wedge \Diamond \neg p)$.

4.1 Explication

Pour utiliser Java tweety Modal Logic library, il faut qu'on dispose d'un modèle logique avec une base à partir de laquelle nous pouvons déduire si les formules sont vraies où fausses.

Puisque nous n'avons pas un exercice structuré ainsi en TD, nous allons utiliser les formules déduites vraies dans

l'exercice 2 et étudier la véracité des autres formules avec des raisonneurs.
 Nous avons donc :
 BliefSet = { Formules 1 3 4 }

4.2 Code Source

```

1 package maven.rcrtp2;
2
3 import java.io.IOException;
4
5 import org.tweetyproject.commons.ParserException;
6 import org.tweetyproject.logics.commons.syntax.Predicate;
7 import org.tweetyproject.logics.commons.syntax.RelationalFormula;
8 import org.tweetyproject.logics.fol.syntax.FolFormula;
9 import org.tweetyproject.logics.fol.syntax.FolSignature;
10 import org.tweetyproject.logics.ml.parser.MlParser;
11 import org.tweetyproject.logics.ml.reasoner.AbstractMlReasoner;
12 import org.tweetyproject.logics.ml.reasoner.MleanCoPReasoner;
13 import org.tweetyproject.logics.ml.reasoner.SPASSMlReasoner;
14 import org.tweetyproject.logics.ml.reasoner.SimpleMlReasoner;
15 import org.tweetyproject.logics.ml.syntax.MlBeliefSet;
16
17
18 public class MlExample2 {
19     public static void main(String[] args) throws ParserException, IOException {
20         MlBeliefSet bs = new MlBeliefSet();
21         MlParser parser = new MlParser();
22         FolSignature sig = new FolSignature();
23         sig.add(new Predicate("p", 0));
24         sig.add(new Predicate("q", 0));
25
26         parser.setSignature(sig);
27         bs.add((RelationalFormula) parser.parseFormula("<>(p && q)"));
28         bs.add((RelationalFormula) parser.parseFormula("[ ](!p) || q)"));
29         bs.add((RelationalFormula) parser.parseFormula("[ ](q && <>(!q))"));
30
31         System.out.println("Modal knowledge base: " + bs);
32         SimpleMlReasoner reasoner = new SimpleMlReasoner();
33         System.out.println("With Simple reasoner :\n ");
34         System.out.println("[ ](!p) " + reasoner.query(bs, (FolFormula) parser.parseFormula("[ ](!p)"
35 ))+"\n");
36         System.out.println("<>(p && q) " + reasoner.query(bs, (FolFormula) parser.parseFormula("<>(
37 p && q)"))+"\n");
38     }
39 }

```

```

Modal knowledge base: { [ ](!p||q), <>(p&&q), [ ](q&&<>(!q)) }
[ ](!p)           true

<>(p && q)         true

```

FIGURE 9 – Résultat de l'exercice

Chapitre 4

TP 3 : Logique des défauts

Résumé

Dans ce TP, nous allons implémenter quelques exercices de la série de TD en utilisant la toolbox «defaultlogic» conçue en java par Evan Morrison.

Toolbox 1 : DefaultLogic

2.1 Explication des lignes de code

Comme nous l'avons vu en cours, TD et TP, un défaut est constitué de trois parties :

- Un prérequis (prerequisite).
- Une Justification(Justificatoin).
- Une conséquence(Consequence).

```
DefaultRule d1 = new DefaultRule(); //création d'un défaut
```

FIGURE 1 – Création d'un défaut

```
/****** Définition dun défaut *****/
d1.setPrerequisite("A");
d1.setJustificatoin("B");
d1.setConsequence("C");
```

FIGURE 2 – Définition d'un défaut

```
/****** Définition dun monde w3 *****/
WorldSet w3= new WorldSet();
w3.addFormula("A");
w3.addFormula("(" + e.NOT + "C" + e.OR + e.NOT + "D");
```

FIGURE 3 – Création et définition d'un monde

2.2 Code Source pour l'exercice 1

```
1 package be.fnord.DefaultLogic;
2 import a.e;
3 import be.fnord.util.logic.DefaultReasoner;
4 import be.fnord.util.logic.WFF;
5 import be.fnord.util.logic.defaultLogic.DefaultRule;
6 import be.fnord.util.logic.defaultLogic.RuleSet;
7 import be.fnord.util.logic.defaultLogic.WorldSet;
8
9 import java.util.HashSet;
10
11 public class tp3 {
12
13     public static void exo1(){
14
15         RuleSet rules = new RuleSet(); //pour mettre les d fauts
16
17
18         DefaultRule d1 = new DefaultRule(); //cr ation d'un d faut d1
19         /****** D finition dun d faut d1 *****/
20         d1.setPrerequisite("A");
```



```

21     d1.setJustificatoin("B");
22     d1.setConsequence("C");
23     rules.addRule(d1);
24
25
26     DefaultRule d2 = new DefaultRule(); //cr ation d'un d faut d2
27     /***** D finition dun d faut d2 *****/
28     d2.setPrerequisite("A");
29     d2.setJustificatoin(e.NOT+"C");
30     d2.setConsequence("D");
31     rules.addRule(d2);
32
33
34
35     /***** D finition dun monde w1 *****/
36     WorldSet w1= new WorldSet();
37     w1.addFormula(e.NOT+"A");
38
39
40
41     /***** D finition dun monde w2 *****/
42     WorldSet w2= new WorldSet();
43     w2.addFormula("A");
44     w2.addFormula(e.NOT+"B");
45
46
47     /***** D finition dun monde w3 *****/
48     WorldSet w3= new WorldSet();
49     w3.addFormula("A");
50     w3.addFormula("(" + e.NOT+"C"+ e.OR+e.NOT+"D)");
51
52
53     /***** D finition dun monde w4 *****/
54     WorldSet w4= new WorldSet();
55     w4.addFormula("A");
56     w4.addFormula("(" + e.NOT+"B"+ e.AND+"C)");
57
58
59
60
61     /*****execution W1 *****/
62     try {
63         a.e.println("/*****execution World 1 *****/\n\n\n");
64         DefaultReasoner r = new DefaultReasoner(w1, rules); //cr ation du raisonneur
65
66
67         HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
68         a.e.println("W1 : \n\t { " + w1.toString()
69             + " }\n D: \n\t { " + rules.toString() + " }");
70         a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
extension");
71         for (String c : scenarios) {
72             a.e.println("\t E: Th(W U ( " + c + " )");
73             // Added closure operator
74             a.e.incIndent();
75             WFF world_and_ext = new WFF("(" + w1.getWorld() + " ) & ( "
76                 + c + " )");
77             a.e.println(" = " + world_and_ext.getClosure());
78             a.e.decIndent();
79         }
80         a.e.println("");
81     } catch (Exception e){
82
83     }
84     /*****execution W2 *****/
85     try {
86         a.e.println("/*****execution World 2 *****/\n\n\n");
87         DefaultReasoner r = new DefaultReasoner(w2, rules);
88         HashSet<String> scenarios = r.getPossibleScenarios();
89         a.e.println("W1 : \n\t { " + w2.toString()
90             + " }\n D: \n\t { " + rules.toString() + " }");

```

```

91         a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
extension");
92         for (String c : scenarios) {
93             a.e.println("\t E: Th(W U (" + c + "));");
94             // Added closure operator
95             a.e.incIndent();
96             WFF world_and_ext = new WFF("(( " + w2.getWorld() + " ) & ( "
97                 + c + "));");
98             a.e.println("= " + world_and_ext.getClosure());
99             a.e.decIndent();
100         }
101         a.e.println("");
102     }catch(Exception e){
103     }
104
105     /*****execution W3 *****/
106     try {
107         a.e.println("/*****execution World 3 *****/\n\n\n");
108         DefaultReasoner r = new DefaultReasoner(w3, rules);
109         HashSet<String> scenarios = r.getPossibleScenarios();
110         a.e.println("W1 : \n\t { " + w3.toString()
111             + " }\n D: \n\t { " + rules.toString() + " }");
112         a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
extension");
113         for (String c : scenarios) {
114             a.e.println("\t E: Th(W U (" + c + "));");
115             // Added closure operator
116             a.e.incIndent();
117             WFF world_and_ext = new WFF("(( " + w3.getWorld() + " ) & ( "
118                 + c + "));");
119             a.e.println("= " + world_and_ext.getClosure());
120             a.e.decIndent();
121         }
122         a.e.println("");
123     }catch(Exception e){
124     }
125
126     /*****execution W4*****/
127     try {
128         a.e.println("/*****execution World 4 *****/\n\n\n");
129         DefaultReasoner r = new DefaultReasoner(w4, rules);
130
131         HashSet<String> scenarios = r.getPossibleScenarios();
132         a.e.println("W1 : \n\t { " + w4.toString()
133             + " }\n D: \n\t { " + rules.toString() + " }");
134         a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
extension");
135         for (String c : scenarios) {
136             a.e.println("\t E: Th(W U (" + c + "));");
137             // Added closure operator
138             a.e.incIndent();
139             WFF world_and_ext = new WFF("(( " + w4.getWorld() + " ) & ( "
140                 + c + "));");
141             a.e.println("= " + world_and_ext.getClosure());
142             a.e.decIndent();
143         }
144         a.e.println("");
145     }catch(Exception e){
146     }
147
148     }
149
150 }
151
152
153 public static void main(String[] args) {
154
155
156     exo1();
157 }
158 }

```

NB : nous avons fait l'exécution en suivant l'exemple fourni avec la librairie defaultlogic

2.3 Résultat de l'exercice 1

$D1 = \{ A : B/C \}$
 $D2 = \{ A / \neg C/D \}$

- $w_1 = \{ \neg A \}$

```
/******execution World 1 *****/

W1 :
    { ~A }
D:
    { [(A):(B) ==> (C)] , [(A):(~C) ==> (D)] }
Par clôture déductive et minimalité, cette théorie admet une seule extension
/******execution World 1 *****/
```

FIGURE 4 – Résultats pour World 1

Pas d'extensions, les défauts ne sont pas générateurs d'extension.

- $w_2 = \{ A, \neg B \}$

```
/******execution World 2 *****/

Trying (eeee) & A & ~B
Trying (eeee) & A & ~B
W1 :
    { A & ~B }
D:
    { [(A):(B) ==> (C)] , [(A):(~C) ==> (D)] }
Par clôture déductive et minimalité, cette théorie admet une seule extension
E: Th(W U (D))
= D & ~B & eeee & A
```

FIGURE 5 – Résultats pour World 2

- $w_3 = \{ A, \neg C \vee \neg D \}$

```

/*****execution World 3 *****/

Trying (eeee) & A & (~C|~D)
Trying (eeee) & A & (~C|~D)
Trying (eeee) & A & (~C|~D)
Trying (eeee) & A & (~C|~D)
W1 :
    { A & (~C|~D) }
D:
    {[ (A):(B) ==> (C) ] , [ (A):(~C) ==> (D) ] }
Par clôture déductive et minimalité, cette théorie admet une seule extension
E: Th(W U (C))
   = C & ~D & eeee & (~D | ~C) & A

```

FIGURE 6 – Résultats pour World 3

- $w_4 = \{ A, \neg B \wedge C \}$

```

/*****execution World 4 *****/

Trying (eeee) & A & (~B&C)
Trying (eeee) & A & (~B&C)
error

```

FIGURE 7 – Résultats pour World 4

Le raisonneur nous a affiché une erreur car il a rencontré un défaut utilisable mais pas applicable.

NB : Nous remarquons que le raisonneur affiche toujours (eeee), ceci n'est qu'un simple bug.

2.4 Exercice 2 :

Exercice 2 :

Considérons la théorie $\Delta = \langle W, D \rangle$ telle que $W = \{A\}$ et $D = \{A : \neg B/B\}$. Montrez que cette théorie n'admet pas d'extension.

FIGURE 8 – Enoncé de l'exercice 2

Nous avons donc :

$W = \{A\}$ et $D = \{A : \neg B/B\}$.

2.5 Code Source pour l'exercice 2

```

1 package be.fnord.DefaultLogic;
2
3
4
5
6 import a.e;
7 import be.fnord.util.logic.DefaultReasoner;
8 import be.fnord.util.logic.WFF;
9 import be.fnord.util.logic.defaultLogic.DefaultRule;
10 import be.fnord.util.logic.defaultLogic.RuleSet;
11 import be.fnord.util.logic.defaultLogic.WorldSet;
12
13 import java.util.HashSet;
14
15 public class tp3 {
16
17
18     public static void exo2() {
19         RuleSet rules = new RuleSet(); //pour mettre les d fauts
20         DefaultRule d = new DefaultRule(); //cr ation d'un d faut d1
21         d.setPrerequisite("A");
22         d.setJustificatoin(e.NOT+"B");
23         d.setConsequence("B");
24         rules.addRule(d);
25
26         WorldSet w= new WorldSet();
27         w.addFormula("A");
28
29
30         /*****execution *****/
31         try {
32             a.e.println("/*****execution World *****/\n\n\n");
33             DefaultReasoner r = new DefaultReasoner(w, rules); //cr ation du raisonneur
34
35
36             HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
37             a.e.println("W1 : \n\t { " + w.toString()
38                 + " }\n D: \n\t { " + rules.toString() + " }");
39             a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
extension");
40             for (String c : scenarios) {
41                 a.e.println("\t E: Th(W U ( " + c + " ))");
42                 // Added closure operator
43                 a.e.incIndent();
44                 WFF world_and_ext = new WFF("( ( " + w.getWorld() + " ) & ( "
45                     + c + " ) )");
46                 a.e.println("= " + world_and_ext.getClosure());
47                 a.e.decIndent();
48             }
49             a.e.println("");
50         } catch (Exception e) {
51
52         }
53     }
54
55 }
56
57 public static void main(String[] args) {
58
59
60     exo2();
61 }
62 }

```

```

/*****execution World *****/

W1 :
    { A }
D:
    [[(A):(¬B) ==> (B)] ]
Par clôture déductive et minimalité, cette théorie admet une seule extension

```

FIGURE 9 – Résultat de l'exercice 2

2.6 Exercice 3 :

Exercice 3 : (non monotonie du raisonnement par défaut)

Quelles sont les extensions des théories $\Delta = \langle W, D \rangle$ et $\Delta' = \langle W', D \rangle$ telles que ;

$W = \{A, B\}$,

$W' = \{A, B, C\}$ et

$D = \{A \wedge B : \neg C / \neg C\}$.

FIGURE 10 – Enoncé de l'exercice 3

Nous avons donc :

$W = \{A, B\}$ et $D = \{A \wedge B : \neg C / \neg C\}$.

$W' = \{A, B, C\}$ et $D = \{A \wedge B : \neg C / \neg C\}$.

2.7 Code Source pour l'exercice 3

```

1 package be.fnord.DefaultLogic;
2
3
4
5
6 import a.e;
7 import be.fnord.util.logic.DefaultReasoner;
8 import be.fnord.util.logic.WFF;
9 import be.fnord.util.logic.defaultLogic.DefaultRule;
10 import be.fnord.util.logic.defaultLogic.RuleSet;
11 import be.fnord.util.logic.defaultLogic.WorldSet;
12
13 import java.util.HashSet;
14
15 public class tp3 {
16
17     public static void exo3() {
18
19         RuleSet rules = new RuleSet(); //pour mettre les d fauts
20         DefaultRule d = new DefaultRule(); //cr ation d'un d faut d1
21         d.setPrerequisite("A"+e.AND+"B");
22         d.setJustificatoIn(e.NOT+"C");
23         d.setConsequence(e.NOT+"C");
24         rules.addRule(d);
25
26         WorldSet w1= new WorldSet();
27         w1.addFormula("A");

```

```

28 w1.addFormula("B");
29
30 WorldSet w2= new WorldSet();
31 w2.addFormula("A");
32 w2.addFormula("B");
33 w2.addFormula("C");
34
35 /*****execution world 1*****/
36 try {
37     a.e.println("/****execution World 1 *****/\n\n");
38     DefaultReasoner r = new DefaultReasoner(w1, rules); //cr ation du raisonneur
39
40
41     HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
42     a.e.println("W1 : \n\t { " + w1.toString()
43         + " }\n D: \n\t { " + rules.toString() + " }");
44     a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
extension");
45     for (String c : scenarios) {
46         a.e.println("\t E: Th(W U ( " + c + " ))");
47         // Added closure operator
48         a.e.incIndent();
49         WFF world_and_ext = new WFF("(( " + w1.getWorld() + " ) & ( "
50             + c + " ))");
51         a.e.println("= " + world_and_ext.getClosure());
52         a.e.decIndent();
53     }
54     a.e.println("");
55 }catch(Exception e){
56
57 }
58
59 /*****execution world 2 *****/
60 try {
61     a.e.println("/****execution World 2*****/\n\n");
62     DefaultReasoner r = new DefaultReasoner(w2, rules); //cr ation du raisonneur
63
64
65     HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
66     a.e.println("W1 : \n\t { " + w2.toString()
67         + " }\n D: \n\t { " + rules.toString() + " }");
68     a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
extension");
69     for (String c : scenarios) {
70         a.e.println("\t E: Th(W U ( " + c + " ))");
71         // Added closure operator
72         a.e.incIndent();
73         WFF world_and_ext = new WFF("(( " + w2.getWorld() + " ) & ( "
74             + c + " ))");
75         a.e.println("= " + world_and_ext.getClosure());
76         a.e.decIndent();
77     }
78     a.e.println("");
79 }catch(Exception e){
80
81 }
82
83 }
84
85 }
86 public static void main(String[] args) {
87     exo3();
88 }
89 }

```

```

/*****execution World 1 *****/

```

```

Trying (eeee) & A & B

```

```

W1 :

```

```

    { A & B }

```

```

D:

```

```

    {[ (A&B):(¬C) ==> (¬C) ] }

```

```

Par clôture déductive et minimalité, cette théorie admet une seule extension

```

```

E: Th(W U (¬C))

```

```

= B & ¬C & eeee & A

```

FIGURE 11 – Résultat de l'exercice 3 world 1

```

/*****execution World 2*****/

```

```

W1 :

```

```

    { A & B & C }

```

```

D:

```

```

    {[ (A&B):(¬C) ==> (¬C) ] }

```

```

Par clôture déductive et minimalité, cette théorie admet une seule extension

```

FIGURE 12 – Résultat de l'exercice 3 world 2

2.8 Exercice 4 :

Exercice 4 :

Considérons les connaissances suivantes :

- Les chrétiens libanais sont des chrétiens.
- En général, les chrétiens libanais sont des Maronites.
- Les Melkites sont des chrétiens libanais qui ne sont pas Maronites.
- En général, les chrétiens libanais ne sont pas des Arabes.
- Les Melkites sont des Arabes.
- En général, les libanais parlent le Français.
- Les Melkites ne parlent pas le Français.

1- Formalisez ces connaissances en utilisant la logique des défauts.

2- Si Mohamed est un Melkite et Georges est un Maronite Arabe, que pouvez-vous conclure?

FIGURE 13 – Enoncé de l'exercice 4

Nous avons déjà formalisé le problème en TD ainsi :

$W = \{ (\forall x) (CHRETIENS-LIBANAIS(x) \supset LIBANAIS(x)); (\forall x) (MELKITE(x) \supset ARABE(x)); (\forall x) (MELKITE(x) \supset \neg PARLE(x, FRANCAIS)); (\forall x) ((MELKITE(x) \supset CHRETIENS-LIBANAIS(x) \wedge \neg MARONITE(x)); \}$

$D = \{ CHRETIENS-LIBANAIS(x) : MARONITE(x) / MARONITE(x); CHRETIENS-LIBANAIS(x) : \neg ARABE(x) / \neg ARABE(x); LIBANAIS(x) : PARLE(x, FRANCAIS) / PARLE(x, FRANCAIS) \}.$

Question : Si Mohamed est un Melkite et Georges est un Maronite Arabe, que pouvez-vous conclure ?

Nous n'avons rien pu conclure dans cet exercice.

2.9 Exercice 6 :

Exercice 6:

La théorie des défauts prioritisée $\Delta = \langle W, D, < \rangle$ étend la théorie des défauts à l'aide d'un ordre $<$ sur les règles de défaut. Un défaut d devra être préféré à un défaut d' quand l'ordre $d < d'$ apparaît.

Considérons la théorie avec défauts prioritisée $\Delta = \langle W, D, < \rangle$ suivante :

$W = \emptyset$

$D = \{a:b/b; \neg a/\neg a; a/a\}$ et

$<: \{d_1 < d_3 < d_2\}$.

- 1- Quelles sont les extensions classiques de cette théorie.
- 2- Quelle est l'extension préférée?

FIGURE 14 – Enoncé de l'exercice 6

Nous avons donc :

$W = \{\emptyset\}$ et $D = \{a : b/b; \neg a/\neg a; a/a\}$.

2.10 Code Source pour l'exercice 6

```
1 package be.fnord.DefaultLogic;
2
3
4
5
6 import a.e;
7 import be.fnord.util.logic.DefaultReasoner;
8 import be.fnord.util.logic.WFF;
9 import be.fnord.util.logic.defaultLogic.DefaultRule;
10 import be.fnord.util.logic.defaultLogic.RuleSet;
11 import be.fnord.util.logic.defaultLogic.WorldSet;
12
13 import java.util.HashSet;
14 import java.util.LinkedList;
15
16 public class tp3 {
17
18     public static void exo6() {
19
20
21         RuleSet rules = new RuleSet(); //pour mettre les d fauts
22
23
24         DefaultRule d1 = new DefaultRule(); //cr ation d'un d faut d1
25         d1.setPrerequisite("a");
26         d1.setJustificatoin("b");
27         d1.setConsequence("b");
28         rules.addRule(d1);
29
30
31         DefaultRule d2 = new DefaultRule(); //cr ation d'un d faut d1
32         d2.setPrerequisite(e.EMPTY_FORMULA);
33         d2.setJustificatoin(e.NOT+"a");
34         d2.setConsequence(e.NOT+"a");
35         rules.addRule(d2);
36
```

```

37 DefaultRule d3 = new DefaultRule(); //cr ation d'un d faut d1
38 d3.setPrerequisite(e.EMPTY_FORMULA);
39 d3.setJustificatoin("a");
40 d3.setConsequence("a");
41 rules.addRule(d3);
42
43 WorldSet w= new WorldSet();
44 w.addFormula(e.EMPTY_FORMULA);
45
46
47
48
49 /*****execution world *****/
50 try {
51     a.e.println("/*****execution for empty World *****/\n\n\n");
52     DefaultReasoner r = new DefaultReasoner(w, rules); //cr ation du raisonneur
53
54
55     HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
56     a.e.println("W1 : \n\t { " + w.toString()
57         + " }\n D: \n\t { " + rules.toString() + " }");
58     a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
extension");
59     for (String c : scenarios) {
60         a.e.println("\t E: Th(W U ( " + c + " ))");
61         // Added closure operator
62         a.e.incIndent();
63         WFF world_and_ext = new WFF("(" + w.getWorld() + " ) & ( "
64             + c + " )");
65         a.e.println("\t = " + world_and_ext.getClosure());
66         a.e.decIndent();
67     }
68     a.e.println("");
69 }catch(Exception e){
70
71 }
72
73 }
74
75
76 }
77 public static void main(String[] args) {
78
79     exo6();
80 }
81 }

```

```

/*****execution for empty World *****/

Trying (eeee) & (eeee)
Trying (eeee) & (eeee)
Trying (eeee) & (eeee)
Trying (eeee) & (eeee)
Trying (eeee) & (eeee)
Trying (eeee) & (eeee)
W1 :
    { }
D:
    {[ (a):(b) ==> (b) ] , [ ([]):(¬a) ==> (¬a) ] , [ ([]):(a) ==> (a) ] }
Par clôture déductive et minimalité, cette théorie admet une seule extension
E: Th(W U (¬a))
= eeee & ¬a
E: Th(W U (b & a))
= b & eeee & a

```

FIGURE 15 – Résultat de l'exercice 6 empty world

2.11 Exercice 7 :

Exercice 7 :

Soit la théorie des défauts prioritisée $\Delta = \langle W, D, < \rangle$ suivante :

$W = \{ p \supset q \wedge r; r \supset \neg s \}$

$D = \{ p/p; r: \neg q/\neg q; s: t/t; p:v/v; q: \neg v/\neg v; v:u/u \}$ et

$<: \{ d_1 < d_2 < d_3 < d_4 < d_5 < d_6 \}$.

- 1- Quelles sont les extensions de cette théorie ?
- 2- Quelle est l'extension préférée ? Justifiez.

FIGURE 16 – Enoncé de l'exercice 7

Nous avons donc :

$W = \{ p \supset q \wedge r; r \supset \neg s \}$ et $D = \{ p:p; r: \neg q/\neg q; s: t/t; p:v/v; q: \neg v/\neg v; v:u/u \}$.

2.12 Code Source pour l'exercice 7

```

1 package be.fnord.DefaultLogic;
2
3
4
5
6 import a.e;
7 import be.fnord.util.logic.DefaultReasoner;
8 import be.fnord.util.logic.WFF;
9 import be.fnord.util.logic.defaultLogic.DefaultRule;
10 import be.fnord.util.logic.defaultLogic.RuleSet;
11 import be.fnord.util.logic.defaultLogic.WorldSet;
12

```

```

13 import java.util.HashSet;
14 import java.util.LinkedList;
15
16 public class tp3 {
17
18     public static void exo7() {
19 RuleSet rules = new RuleSet(); //pour mettre les d fauts
20
21
22     DefaultRule d1 = new DefaultRule(); //cr ation d'un d faut d1
23 d1.setPrerequisite(e.EMPTY_FORMULA);
24 d1.setJustificatoin("p");
25 d1.setConsequence("p");
26 rules.addRule(d1);
27
28
29     DefaultRule d2 = new DefaultRule(); //cr ation d'un d faut d1
30 d2.setPrerequisite("r");
31 d2.setJustificatoin(e.NOT+"q");
32 d2.setConsequence(e.NOT+"q");
33 rules.addRule(d2);
34
35     DefaultRule d3 = new DefaultRule(); //cr ation d'un d faut d1
36 d3.setPrerequisite("s");
37 d3.setJustificatoin("t");
38 d3.setConsequence("t");
39 rules.addRule(d3);
40
41
42     DefaultRule d4 = new DefaultRule(); //cr ation d'un d faut d1
43 d4.setPrerequisite("p");
44 d4.setJustificatoin("v");
45 d4.setConsequence("v");
46 rules.addRule(d4);
47
48     DefaultRule d5 = new DefaultRule(); //cr ation d'un d faut d1
49 d5.setPrerequisite("q");
50 d5.setJustificatoin(e.NOT+"v");
51 d5.setConsequence(e.NOT+"v");
52 rules.addRule(d5);
53
54     DefaultRule d6 = new DefaultRule(); //cr ation d'un d faut d1
55 d6.setPrerequisite("v");
56 d6.setJustificatoin("u");
57 d6.setConsequence("u");
58 rules.addRule(d6);
59
60     WorldSet w= new WorldSet();
61 w.addFormula("(p -> (q "+e.AND+"r))");
62 w.addFormula("(r ->"+e.NOT+"s)");
63
64
65
66
67     /*****execution world *****/
68     try {
69         a.e.println("/*****execution for empty World *****/\n\n");
70         DefaultReasoner r = new DefaultReasoner(w, rules); //cr ation du raisonneur
71
72
73         HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
74         a.e.println("W1 : \n\t { " + w.toString()
75             + " }\n D: \n\t { " + rules.toString() + " }");
76         a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
extension");
77         for (String c : scenarios) {
78             a.e.println("\t E: Th(W U ( " + c + " ))");
79             // Added closure operator
80             a.e.incIndent();
81             WFF world_and_ext = new WFF("(( " + w.getWorld() + " ) & ( "
82                 + c + " ))");
83             a.e.println("= " + world_and_ext.getClosure());

```

```

84         a.e.decIndent();
85     }
86     a.e.println("");
87 } catch (Exception e){
88
89     }
90 }
91 }
92
93 public static void main(String[] args) {
94     exo7();
95 }
96 }

```

```

/*****execution for World *****/

Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
Trying (eeee) & (p -> (q &r)) & (r ->~s)
W1 :
      { (p -> (q &r)) & (r ->~s) }

D:
      {{{[]:(p ==> (p)) , [[r]:(~q) ==> (~q)] , [[s):(t) ==> (t)] , [(p):(v) ==> (v)] , [(q):(~v) ==> (~v)] , [(v):(u) ==> (u)] }
Par clôture déductive et minimalité, cette théorie admet une seule extension
      E: Th(W U (u & v & p))
      = (r | ~p) & eeee & (~s | ~r) & p & u & v & q & r & ~s & (q | ~p)
      E: Th(W U (~v & p))
      = (r | ~p) & eeee & (~s | ~r) & p & ~v & q & r & ~s & (q | ~p)

```

FIGURE 17 – Résultat de l'exercice 7

Toolbox 2 : Default logic Simulation Online Tool

Cette toolbox ne marche pas pour le moment et sera disponible bientôt selon le site.

Toolbox 3, 5, 6, 7 and 8 : Orbital Library, DefaultLogicModelCkeck, defaultLogic (la meme), default logic reasoner and Java Tweety

Ces Toolboxes sont similaires à DefaultLogic Library.

Toolbox 4 : Extension Calculator

Voici l'interface de l'outil

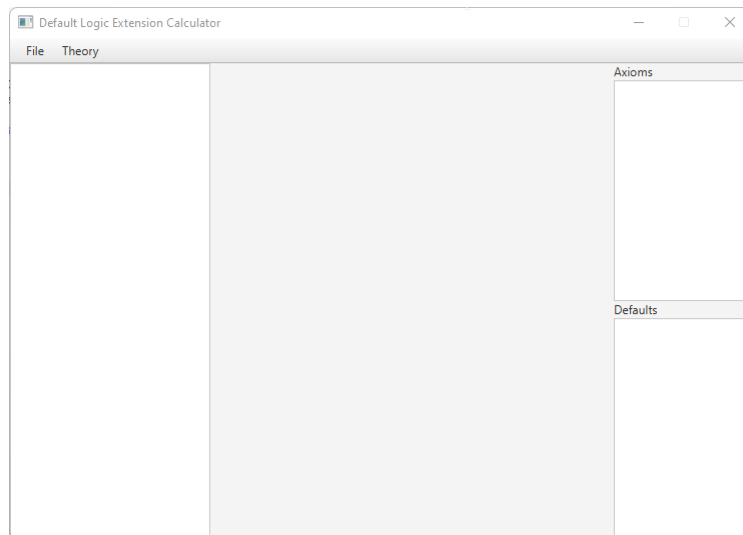


FIGURE 18 – Interface de l'outil Extension Calculator

Exercice 1

$D1 = \{ A : B/C \}$
 $D2 = \{ A / \neg C/D \}$

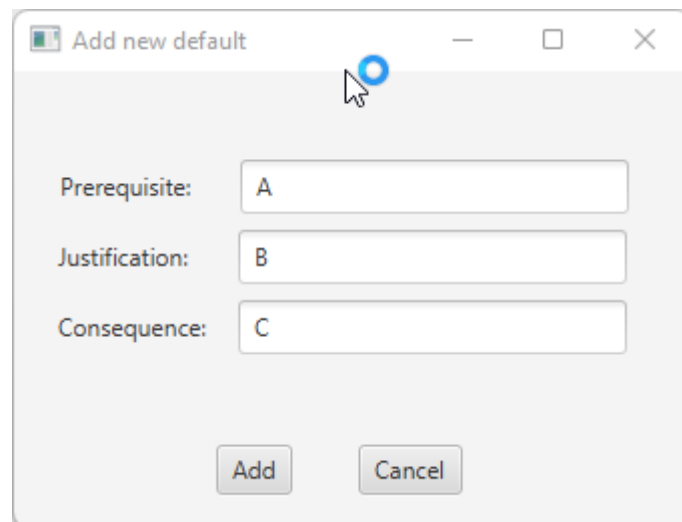


FIGURE 19 – Création du défaut d1

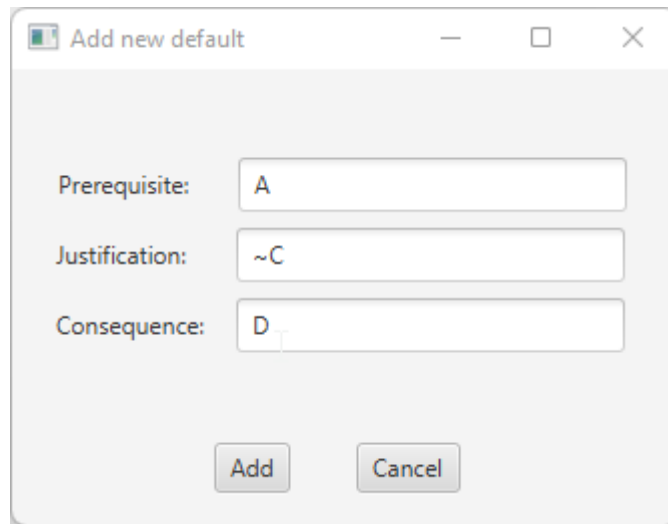


FIGURE 20 – Création du défaut d2

- $w_1 = \{ \neg A \}$

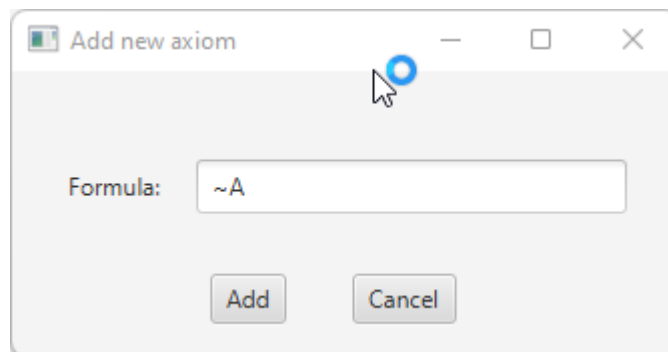


FIGURE 21 – Création du monde w1

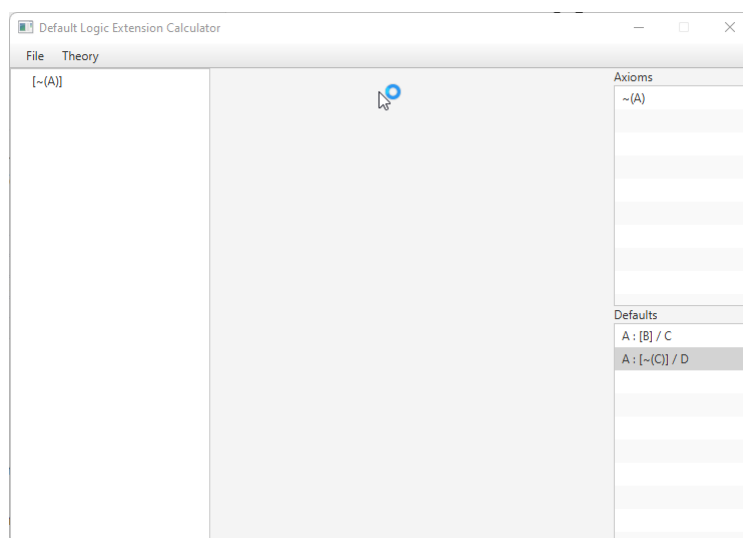


FIGURE 22 – Résultat de l'extension de w1

Si la colonne à gauche est égale à la colonne des axiomes, alors il n'y a pas d'extension

$$- w_2 = \{ A, \neg B \}$$

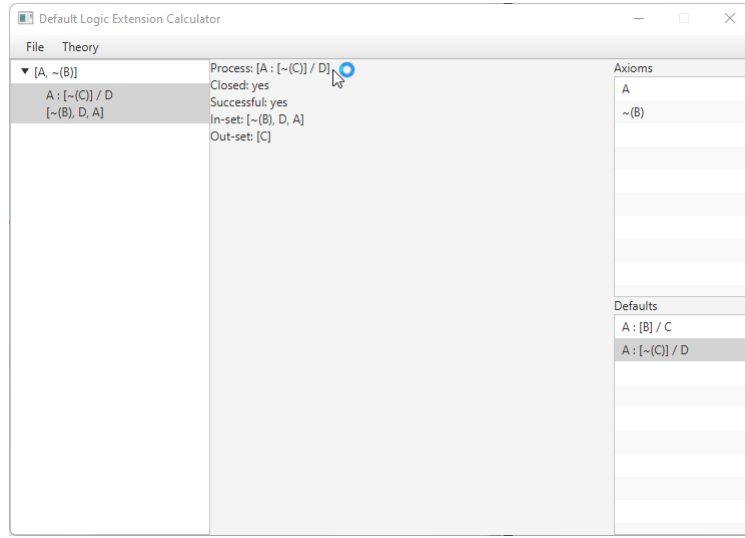


FIGURE 23 – Résultat de l'extension de w_2

$$- w_3 = \{ A, \neg C \vee \neg D \}$$

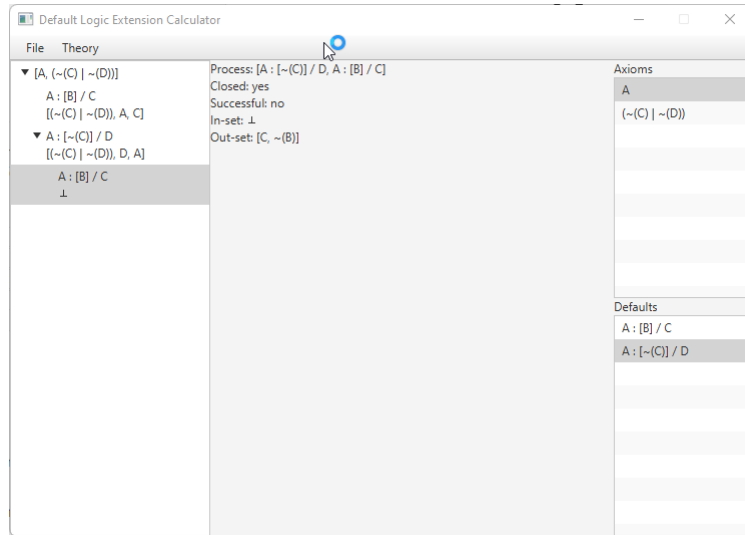


FIGURE 24 – Résultat de l'extension de w_3

$$- w_4 = \{ A, \neg B \wedge C \}$$

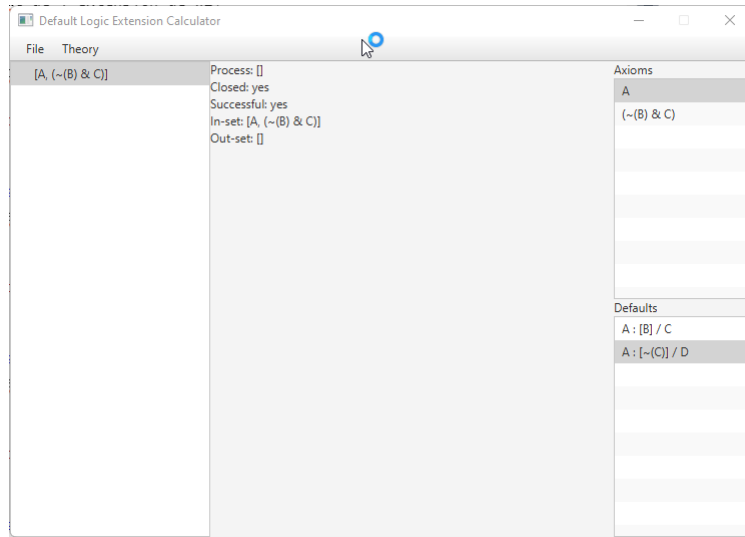


FIGURE 25 – Résultat de l’extension de w_4

Exercice 2

Nous avons donc :

$W=\{A\}$ et $D=\{A : \neg B/B\}$.

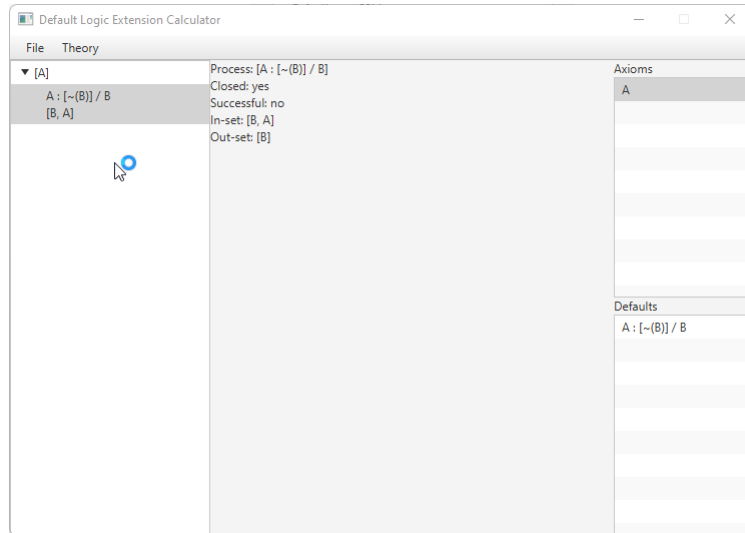


FIGURE 26 – Résultat de l’exercice 2

Exercice 3

Nous avons donc :

$W=\{A, B\}$ et $D=\{A \wedge B : \neg C/\neg C\}$.

$W'=\{A, B, C\}$ et $D=\{A \wedge B : \neg C/\neg C\}$.

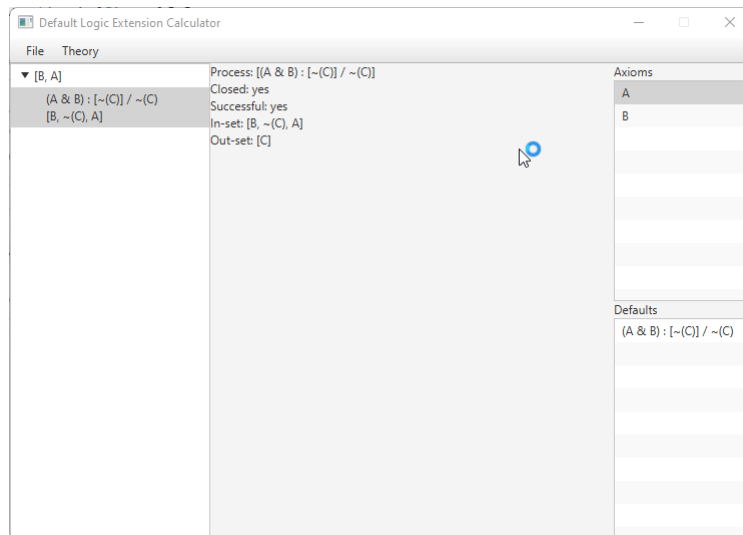


FIGURE 27 – Résultat de l'exercice 3 avec w

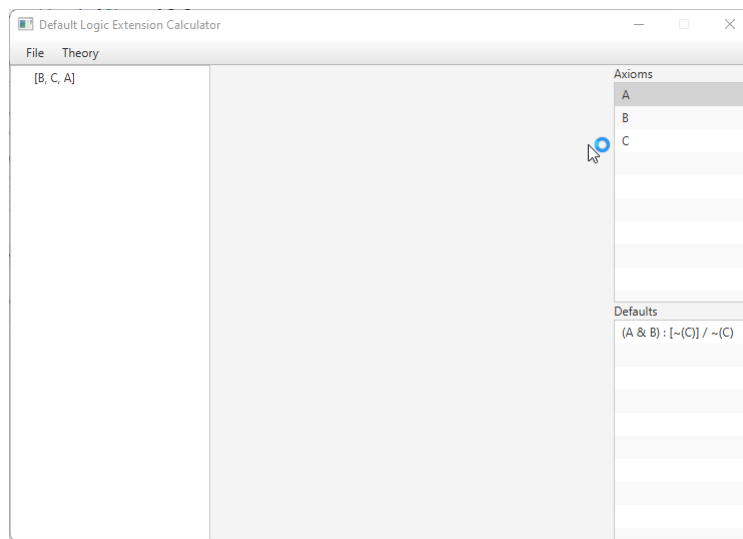


FIGURE 28 – Résultat de l'exercice 3 avec w'

Exercice 4 et 5

Cette toolbox ne permet pas de saisir le symbole quel que soit : « \forall » et le symbole d'implication « \supset »

Exercice 6 et 7

Cette toolbox ne permet pas de saisir le symbole EMPTY_FORMULA c'est à dire que tous les champs doivent contenir quelque chose. Par conséquent, les deux exercices ne peuvent pas être résolus avec cette toolbox.

Chapitre 5

TP 4 : Réseaux sémantiques

1 Introduction

Dans ce TP, nous allons nous intéresser à l'implémentation de réseaux sémantiques, et plus précisément à l'implémentation de différents algorithmes des réseaux sémantiques.

2 Réseaux sémantiques

Un réseau sémantique est un graphe marqué destiné à la représentation des connaissances. Pour notre TP, un réseau sémantique est défini dans un fichier JSON récupéré via le site qui a été joint dans le TP ou chaque nœud possède un label et un id qui sont par la suite utilisés pour représenter les différentes relations entre les nœuds. Nous allons réaliser les différents algorithmes avec le langage python étant particulièrement adapté pour travailler avec les fichiers JSON.

Voici l'exemple de réseau sémantique que nous allons utiliser pour le reste du TP.

3 Partie 1 : implémenter l'algorithme de propagation de marqueurs dans les réseaux sémantiques

Dans cette partie nous allons implémenter l'algorithme de propagation de marqueurs vu en cours :

```
1 import json
2
3 def get_label(reseau_semantique, node, relation):
4     node_relation_edges = [edge["from"] for edge in reseau_semantique["edges"] if (edge["to"] == node["id"] and edge["label"] == relation)]
5     node_relation_edges_label = [node["label"] for node in reseau_semantique["nodes"] if node["id"] in node_relation_edges]
6     reponse = "il y a un lien entre les 2 noeuds : " + ", ".join(node_relation_edges_label)
7     return reponse
8
9 def propagation_de_marqueurs(reseau_semantique, node1, node2, relation):
10     nodes = reseau_semantique["nodes"]
11
12     solutions_found = []
13
14     for i in range(min(len(node1), len(node2))):
15         solution_found = False
16
17         try:
18             M1 = [node for node in nodes if node["label"] == node1[i][0]]
19             M2 = [node for node in nodes if node["label"] == node2[i][0]]
20
21             edges = reseau_semantique["edges"]
22             propagation_edges = [edge for edge in edges if (edge["to"] == M1["id"] and edge["label"] == "is a")]
23             while len(propagation_edges) != 0 and not solution_found:
24                 temp_node = propagation_edges.pop()
25                 temp_node_contient_edges = [edge for edge in edges if (edge["from"] == temp_node["from"] and edge["label"] == relation)]
26                 solution_found = any(d["to"] == M2["id"] for d in temp_node_contient_edges)
27                 if not solution_found:
28                     temp_node_is_a_edges = [edge for edge in edges if (edge["to"] == temp_node["from"] and edge["label"] == "is a")]
29                     propagation_edges.extend(temp_node_is_a_edges)
30
31             solutions_found.append(get_label(reseau_semantique, M2, relation) if solution_found else "il n'y a pas un lien entre les 2 noeuds")
32         except IndexError:
33             solutions_found.append("Aucune reponse n'est fournie par manque de connaissances.")
34
35     return(solutions_found)
```

FIGURE 1 – Algorithme de propagation de marqueur

Pour cet Algorithme nous allons utiliser le réseau sémantique fourni par le lien donnée dans la série de TP détaillé dans l'image suivante :



FIGURE 2 – Réseau sémantique utilisé pour la partie 1 du TP

Utilisant plusieurs nœuds marqués en entrée, on obtient les résultats suivants :

```
Partie 1: l'algorithme de propagation de marqueurs
Modes de Représentations des connaissances contient Axiome A7
il y a un lien entre les 2 noeuds : Systeme T, Systeme S5
Modes de Représentations des connaissances contient Axiome A4
il y a un lien entre les 2 noeuds : Logique D ordre 1
Modes de Représentations des connaissances contient Axe-IA
il n'y a pas un lien entre les 2 noeuds
Modes de Représentations des connaissances contient Axiome A9
Aucune reponse n'est fournie par manque de connaissances.
```

FIGURE 3 – Résultat obtenu par l'algorithme de propagation de marqueurs

On voit qu'effectivement l'algorithme à trouver le lien entre ces deux nœuds.

4 Partie 2 : implémenter l'algorithme d'héritage

Dans cette partie, nous allons implémenter l'algorithme d'héritage vu en cours :

```

1 import json
2
3 def get_label(reseau_semantique, node_id):
4     label = [node["label"] for node in reseau_semantique["nodes"] if node["id"] == node_id]
5     return " ,".join(label)
6
7 def heritage(reseau_semantique, name):
8     the_end = False
9
10    nodes = reseau_semantique["nodes"]
11    edges = reseau_semantique["edges"]
12
13    node = [node for node in nodes if node["label"] == name][0]
14    legacy_edges = [edge["to"] for edge in edges if (edge["from"] == node["id"] and edge["label"] == "is_a")]
15    legacy = []
16    properties = []
17    while not the_end:
18        n = legacy_edges.pop()
19        legacy.append(get_label(reseau_semantique, n))
20        legacy_edges.extend([edge["to"] for edge in edges if (edge["from"] == n and edge["label"] == "is_a")])
21        properties_nodes = [edge for edge in edges if (edge["from"] == n and edge["label"] != "is_a")]
22        for pn in properties_nodes:
23            properties.append(" : ".join([pn["label"], get_label(reseau_semantique, pn["to"])]))
24        if len(legacy_edges) == 0:
25            the_end = True
26
27    return legacy, properties

```

FIGURE 4 – Algorithme d'héritage

Pour cet Algorithme, nous allons utiliser un autre réseau sémantique exploitant mieux les propriétés de cet algorithme et qui a, lui aussi, été vu en cours et qui est détaillé dans l'image suivante :

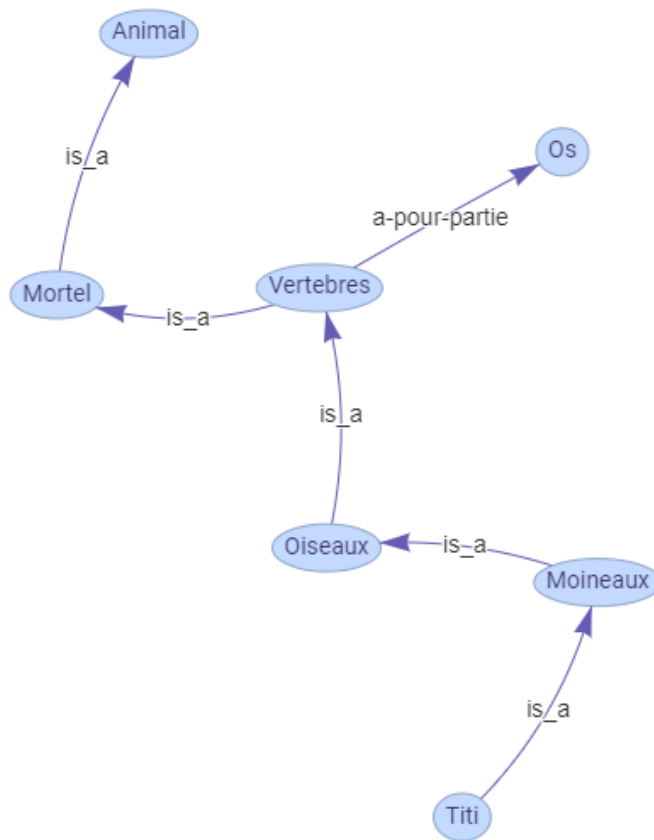


FIGURE 5 – Réseau sémantique utilisé pour la partie 2 du TP

En utilisant le nœud “Titi” en entrée, on obtient le résultat suivant :

```
Partie 2: l'algorithm d'heritage
Resultat de l'inference utiliser:
Titi
Moineaux
Oiseaux
Vertebres
Mortel
Animal
Deduction des priorites:
a-pour-partie: 0s
```

FIGURE 6 – Résultat obtenu par l’algorithme d’héritage

5 Partie 3 : implémentez un algorithme qui permet d’inhiber la propagation dans le cas des liens d’exception

Cet algorithme est très similaire à celui de la partie 1 la seule différence est dans le fait qu’on ne prend pas en compte les arcs de types exception, l’algorithme est plus détaillé ci-dessous :

```
1 import json
2
3 def get_label(reseau_semantique, node, relation):
4     node_relation_edges = [edge["from"] for edge in reseau_semantique["edges"] if (edge["to"] == node["id"] and edge["label"] == relation)]
5     node_relation_edges_label = [node["label"] for node in reseau_semantique["nodes"] if node["id"] in node_relation_edges]
6     reponse = "il y a un lien entre les 2 noeuds : " + ", ".join([node_relation_edges_label[1]])
7     return reponse
8
9 def propagation_de_marqueurs(reseau_semantique, node1, node2, relation):
10     nodes = reseau_semantique["nodes"]
11
12     solutions_found = []
13
14     for i in range(min(len(node1), len(node2))):
15         solution_found = False
16
17         try:
18             M1 = [node for node in nodes if node["label"] == node1[i]][0]
19             M2 = [node for node in nodes if node["label"] == node2[i]][0]
20
21             edges = reseau_semantique["edges"]
22             propagation_edges = [edge for edge in edges if (edge["to"] == M1["id"] and edge["label"] == "is a" and edge["edge_type"] != "exception")]
23             while len(propagation_edges) != 0 and not solution_found:
24                 temp_node = propagation_edges.pop()
25                 temp_node_contient_edges = [edge for edge in edges if (edge["from"] == temp_node["from"] and edge["label"] == relation and edge["edge_type"] != "exception")]
26                 solution_found = any(d["to"] == M2["id"] for d in temp_node_contient_edges)
27                 if not solution_found:
28                     temp_node_is_a_edges = [edge for edge in edges if (edge["to"] == temp_node["from"] and edge["label"] == "is a" and edge["edge_type"] != "exception")]
29                     propagation_edges.extend(temp_node_is_a_edges)
30
31             solutions_found.append(get_label(reseau_semantique, M2, relation) if solution_found else "il n'y a pas un lien entre les 2 noeuds")
32         except IndexError:
33             solutions_found.append("Aucune reponse n'est fournie par manque de connaissances.")
34
35     return(solutions_found)
```

FIGURE 7 – Algorithme d’inhibition de la propagation dans le cas des liens d’exception

Pour cet Algorithme nous allons utiliser le même réseau sémantique avec l’ajout d’un lien d’exception pour voir la différence avec le premier algorithme. Le réseau est détaillé dans l’image suivante :

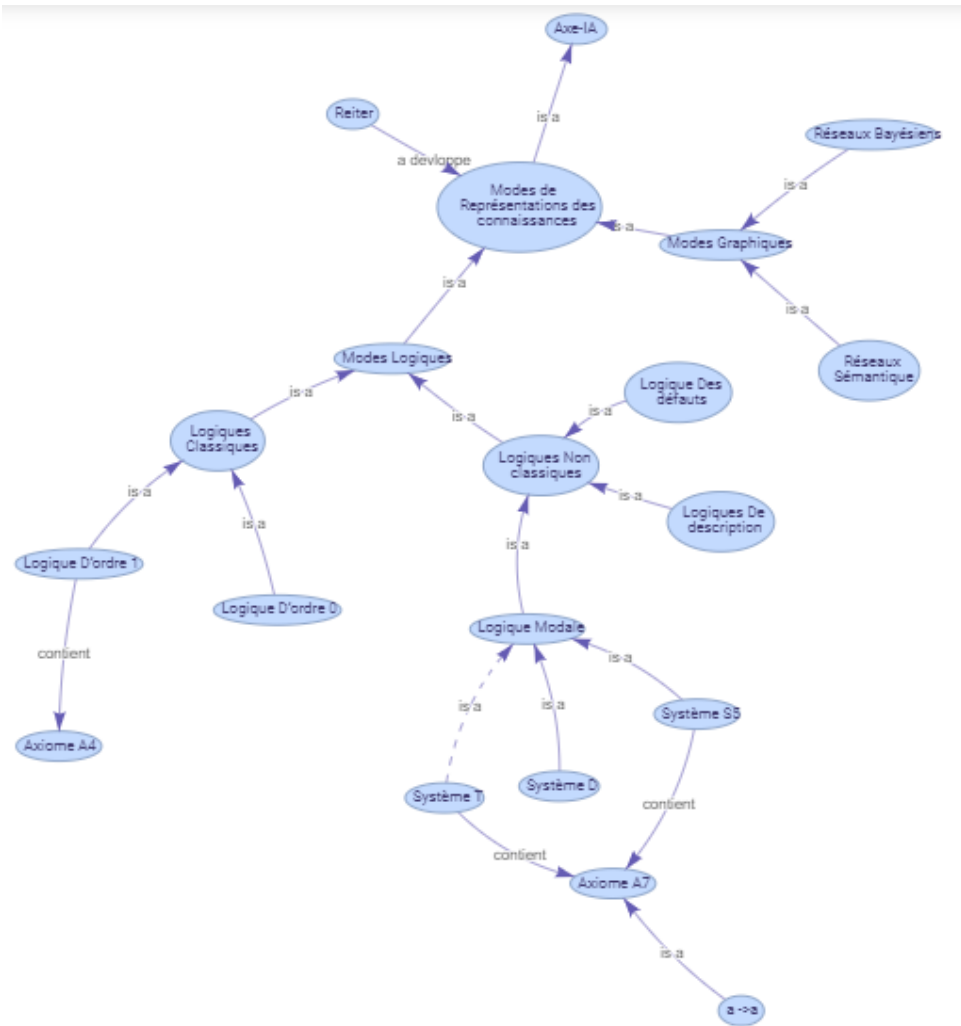


FIGURE 8 – Réseau sémantique utilisé pour la partie 3 du TP

Utilisant la question “Mode de représentation des connaissances contient Axiome A7” on obtient la réponse suivante :

Partie 3: l'algorithme de propagation de marqueurs avec exception
 Modes de Représentations des connaissances contient Axiome A7
 il y a un lien entre les 2 noeuds : Systeme S5

FIGURE 9 – Résultat obtenu par l'algorithme d'inhibition de la propagation dans le cas des liens d'exception

Chapitre 6

TP 5 : Logique des descriptions

Résumé

Dans ce TP, nous allons manipuler des données ontologique en logique des descriptions en optant pour le raisonneur **Pellet** avec deux options d'outils :

- 1 - La librairie Python **OwlReady**.
- 2 - Le logiciel **WebProtégé**.

Nous commençons par la présentation de notre TBOX et ABOX

Tbox : Entités atomiques et composées

Cette box est composée d'entités atomiques ayant des concepts et des roles. Les premiers représentantt des classes et les deuxièmes leurs méthodes. Par ailleurs, les entités composées sont des sous ensembles des concepts.

2.1 Tbox : Entités atomiques

2.1.a Concepts

Personne
Aliment
University

2.1.b Roles

Mange
Mange_par
Enseigne
Enseigne_par
PartieDe

2.2 Entités composées

Faculty : Sous ensemble de University
Departement : Sous ensemble de Faculty
Etudiant :Sous ensemble de Personne
Enseignant : Sous ensemble de Personne
Malbouffe : Sous ensemble de Aliment

2.3 Abox : Instances

Personne(Khellaf)
Pesonne(Mohamed)

Tool 1 : Python Owl Ready Library

3.0.a Code Source

```
1  ###
2  from owlready2 import *
3
4  onto = get_ontology("http://testxyz.org/onto.owl") #create ontology using iri
5
6  with onto: #defining our ontology
7
8      ##Defining concepts##
9      class Personne(Thing):
10         pass
```

```

11
12 class Aliment(Thing):
13     pass
14
15 class University(Thing):
16     pass
17
18 AllDisjoint([Personne, Aliment, University]) #pour dire qu'un individu ne peut pas etre une
19 personne et un aliment
20
21 ##Defining roles##
22 class mange(Personne >> Thing):
23     pass
24
25 class enseigne(Personne >> Thing): #persone est thing
26     pass
27
28 class enseigne_par(ObjectProperty):
29     inverse_property = enseigne
30
31 class mange_par(ObjectProperty):
32     inverse_property = mange
33
34 class PartieDe(Thing >> Thing):
35     pass
36
37 ##Defining composed entities##
38 class Faculty(Thing):
39     equivalent_to = [Thing & PartieDe.some(University)]
40
41 class Departement(Thing):
42     equivalent_to = [Thing & PartieDe.some(Faculty)]
43
44 class Enseignant(Personne):
45     equivalent_to = [Personne & enseigne.only(Personne)]
46
47 class Etudiant(Thing):
48     equivalent_to = [Personne & enseigne_par.only(Enseignant)]
49
50 ##defining instances ABOX##
51
52 class Mohamed(Thing):
53     equivalent_to = [Personne & mange.only(Aliment)]
54
55 class Khellaf(Personne):
56     equivalent_to = [Enseignant & mange.some(Aliment) & enseigne.only(Etudiant)]
57
58 class MalBouffe(Thing):
59     equivalent_to = [Aliment & (mange_par.some(Personne))]
60
61 AllDisjoint([Etudiant, Enseignant])
62 AllDisjoint([Khellaf, Mohamed])
63 AllDisjoint([MalBouffe, Departement, Faculty, University])
64
65 sync_reasoner_pellet(infer_property_values=True)
66 onto.save(file = "tp_rc1.owl", format = "rdxml")
67
68
69 ###
70 with onto:
71
72     USTHB = onto.University()
73
74     Ranya = onto.Etudiant()
75
76     Chocolat = onto.Aliment()
77
78     Moulai = onto.Personne()
79
80
81     SI = Thing() #department

```

```

82 INFO = Thing() #faculty
83
84 INFO.PartieDe.append(USTHB)
85 SI.PartieDe.append(INFO)
86
87 Ranya.mange = [Chocolat]
88 Moulai.enseigne = [Ranya]
89
90 sync_reasoner_pellet(infer_property_values=True)
91 onto.save(file = "tp_rc2.owl", format = "rdxml")
92
93 # %%

```

Voici les résultats déduits par le raisonneur :

Tool 2 : WebProtégé

Voici l'interface de l'outil Web Protégé :

```

* Owlready2 * Pellet took 1.63551926612854 seconds
* Owlready * Reparenting onto.MalBouffe: {owl.Thing} => {onto.Aliment}
* Owlready * Reparenting onto.Khellaf: {onto.Personne} => {onto.Enseignant}
* Owlready * Reparenting onto.Etudiant: {owl.Thing} => {onto.Personne}
* Owlready * (NB: only changes on entities loaded in Python are shown, other changes are done but not listed)

```

FIGURE 1 – Résultats déduits sur les instances

Ici, le raisonneur a déduit que :

- La Mallbouffe est un Aliment.
- Khellaf est Enseignant(e).
- Etudiant est une Personne.

NB : Seulement les instances changées sont mentionnées par le raisonneur.

```

* Owlready * Reparenting onto.aliment1: {onto.Aliment} => {onto.MalBouffe}
* Owlready * Reparenting onto.thing1: {owl.Thing} => {onto.Département}
* Owlready * Reparenting onto.thing2: {owl.Thing} => {onto.Faculté}
* Owlready * Reparenting onto.personne1: {onto.Personne} => {onto.Enseignant}
* Owlready * (NB: only changes on entities loaded in Python are shown, other changes are done but not listed)

```

FIGURE 2 – Résultats du raisonneur

Ici, le raisonneur a déduit que :

- Aliment1 (Chocolat) est une Malbouffe.
- Thing1 (Objet) est un département.
- Thing2 (Objet) est une Faculté.
- Personne1 est un(e) enseignant(e).

Tool 2 : WebProtégé

Voici l'interface de l'outil Web Protégé :

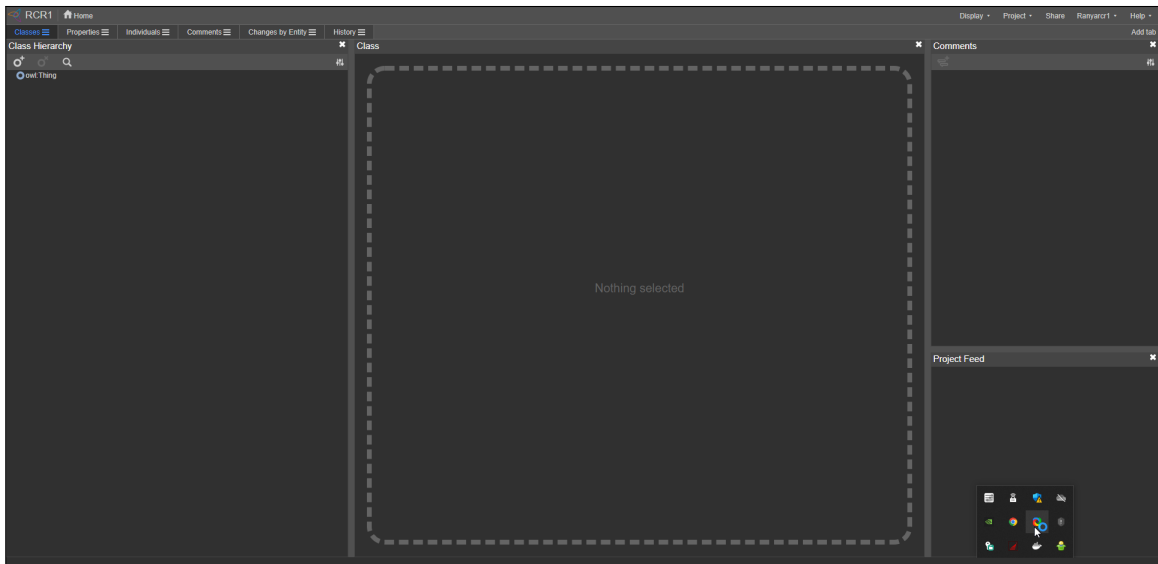


FIGURE 3 – Interface de WebProtégé

Voici donc notre description :

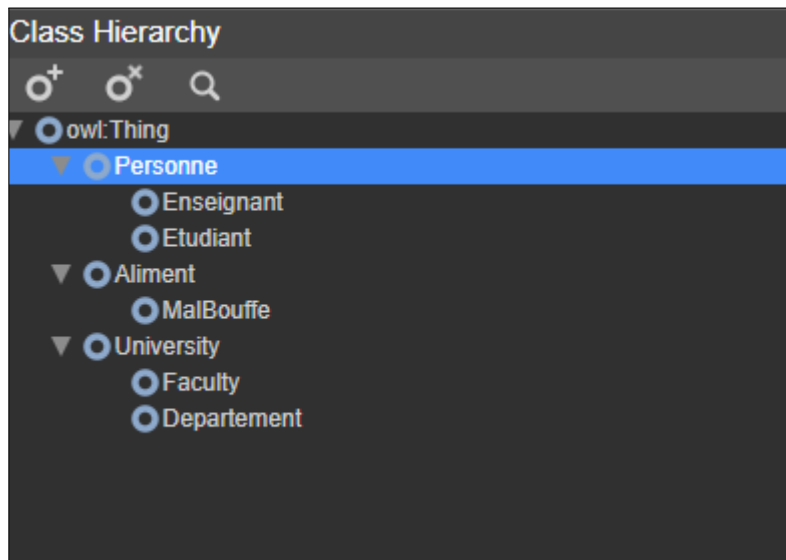


FIGURE 4 – Concepts et entités composées

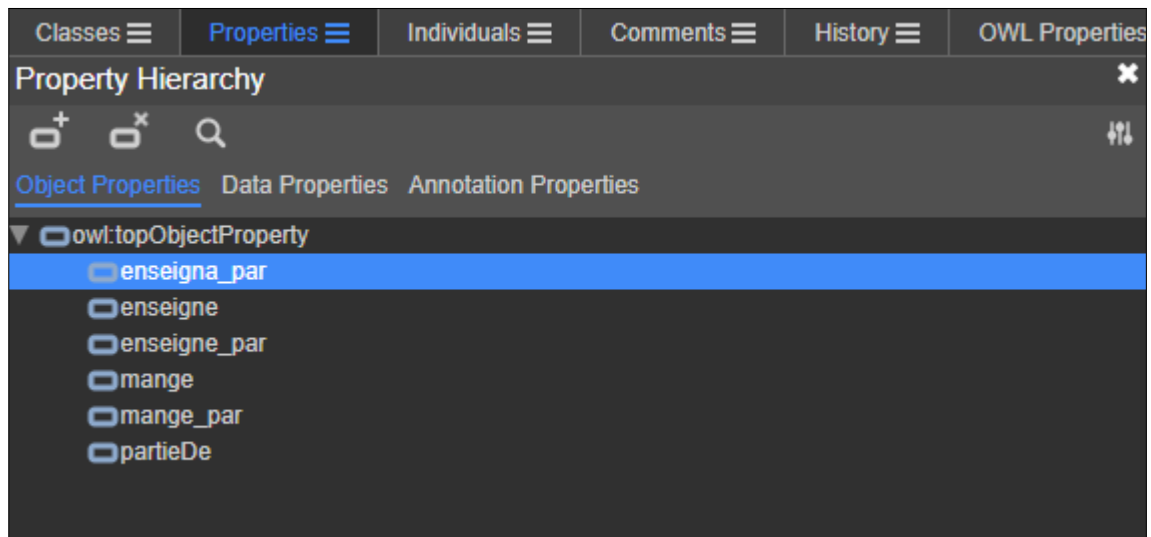


FIGURE 5 – Roles

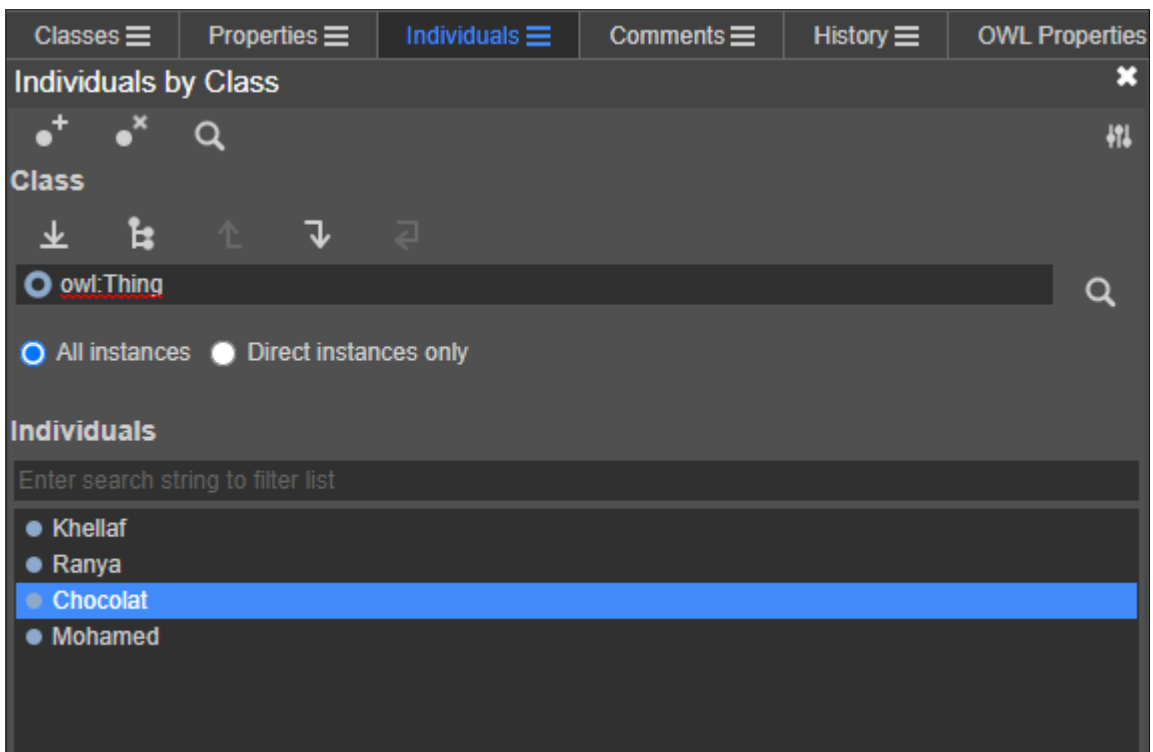





FIGURE 6 – Instances


Individual: Chocolat






IRI

http://webprotege.stanford.edu/RDTeSJcXe3AcDiSNjOn3RIIm

Annotations

 rdfs:label

 Chocolat


Enter property


Enter value

Types

Enter a class name

Relationships

 mange_par

 Mohamed

Enter property

Enter value

Same As

Enter an individual name

FIGURE 7 – Exemple details d'un individu

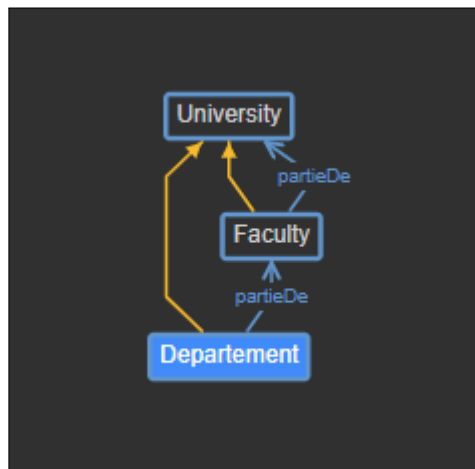


FIGURE 8 – Schéma de departement

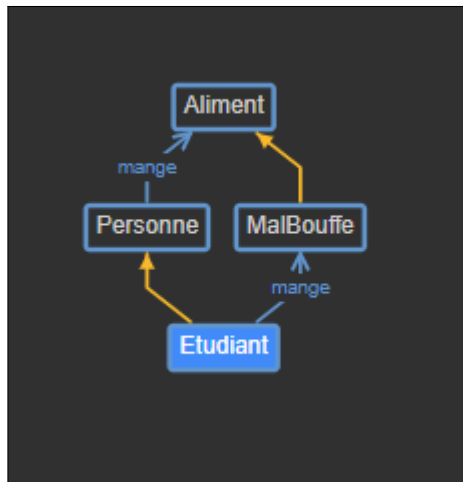


FIGURE 9 – Schéma de etudiant

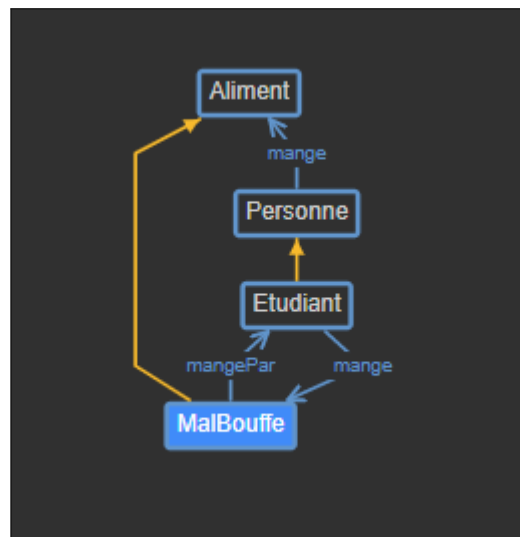


FIGURE 10 – Schéma de MalBouffe

Individual: Ranya

IRI
http://webprotege.stanford.edu/R9byeWKSjb8Zmwk9dyuiAC

Annotations

☒ rdfs:label Ranya lang ✕

Enter property Enter value lang

Types

☒ Etudiant ✕

Enter a class name

Relationships

☒ mange ☒ MalBouffe ✕

Enter property Enter value lang

Same As

☒ Mohamed ✕

Enter an individual name

FIGURE 11 – Dédution pour Ranya

Individual: Khellaf

IRI
http://webprotege.stanford.edu/R9NsdKsPYcVoFoVmETx7vRu

Annotations

☒ rdfs:label Khellaf lang ✕

Enter property Enter value lang

Types

☒ Personne ✕

Enter a class name

Relationships

☒ enseigne ☒ Ranya ✕

☒ mange ☒ Aliment ✕

Enter property Enter value lang

Same As

Enter an individual name

FIGURE 12 – Dédution pour Khellaf

5.1 Conclusion

Nous avons remarqué que les deux outils sont différent à utiliser. La librairie Python donne un peu plus de flexibilité à l'utilisateur le laissant créer n'importe quelle hiérarchie de classes et relations. Cependant, ceci peut induire à une logique non consistante. Dans ce cas là, l'outil protégé est le choix optimale pour avoir une logique bien définie.

Chapitre 7

Conclusion Générale

Ces TPs nous ont fait découvrir les différentes logiques sur lesquelles un système informatique peut opérer. Nous avons appris à modéliser des problèmes de la vie réelle, générer des inférences, utiliser des raisonneurs pour étudier la véracité de formule, représenter graphiquement des informations et finalement déduire des faits.