



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



Université des Sciences et de la Technologie Houari Boumediene

Faculté d'informatique

Département IA&SD

Rapport RCR

Spécialité :

Systèmes Informatiques Intelligents

Thème :

RAPPORT DES TPs RCR

Présenté par :

MEKDOUD Rachid

191931092011

G4

HEZOUAT Abdeldjalil

191931069453

G4

Mai 2023

TABLES DES MATIERES

TP1 : Inférence logique basée sur un solveur SAT	6
Résumé :	6
Etape 1 : Répertoire UBCSAT :	6
Etape 2 : Exécution solveur SAT sur les deux exemples.	6
Etape 3 :	9
Etape 4 :	15
TP2 : LOGIQUE DES PREDICATS :	18
Résumé :	18
Exploitation de l'outils :	18
Déroulement sur un exemple concret :	18
TP3 : LOGIQUE MODALE.....	20
Résumé :	20
Exploitation de l'outils MLP :	20
Exploitation de l'outils TWEETY :	23
TP4 : Logique des defaults :	25
Résumé :	25
Outil utilisé :	25
Exploration de l'outil :	25
Utilisation de l'outil :	25
Exemple 1 : (est dans Extras Exemple)	25
Exemples 2(Exo 2 serie Td)	26
Exemple 3(exemple 2 de exo1) :	26
TP5 : Réseau Sémantique	28
Outils utilise :	28
Explication :	28
Classe Node :	28
Classe Edge:	29
TP6 : LOGIQUES DES DESCRIPTIONS :	33
Objectif :	33
Outil utilisé :	33

Exploration de l'outil :.....	33
Utilisation de l'outil :.....	34
Définition des Concepts Atomique :.....	34
Définition des Rôles atomiques :	35
Représentation des Connaissances T-BOX:	35
Représentation des Connaissances A-BOX :	38

TABLES DES FIGURES

Figure 1:Repertoire UBCSAT	6
Figure 2:Exécution solveur SAT exemple 1	6
Figure 3:Resultat d'exécution	7
Figure 4:Solution solveur	7
Figure 5:Affichage des paramètres	8
Figure 6:test2.cnf	8
Figure 7:exécution test2.cnf	9
Figure 8:résultats de test2.cnf	9
Figure 9: fichier cnf pour Céphalopode	11
Figure 10:exécution de cephal.cnf	11
Figure 11:Résultats cephal	12
Figure 12:Fichier cnf cephal1	13
Figure 13:Solution trouve de cephal1	14
Figure 14: Exécution de cephal1	14
Figure 15: Résultats d'exécution fichier Benchmark	15
Figure 16:Navigation.cnf	16
Figure 17:Programme C pour l'etape4	17
Figure 18:Compilation du programme C	17
Figure 19:Résultat d'exécution du programme C	17
Figure 20:Définition des constantes	18
Figure 21:Définition des prédicats	19
Figure 22:Définition de fonction	19
Figure 23:Creation des formules logiques	19
Figure 24:Résultats d'inférence	19
Figure 25:Modale Logic Playground	20
Figure 26:Tweety pour Logique Modale	23
Figure 27:Resultat des formules pour logique Modale	24
Figure 28:Création d'un default	25
Figure 29:Execution exemple 1 Logique des defaults	26
Figure 30:Execution Exemple 2 logiques des defaults	26

Figure 31:Exécution Exemple 3 Default.....	27
Figure 32:Resultats Exemple 3.....	27
Figure 33:Attribut Réseau sémantique	28
Figure 34:Classe Node	29
Figure 35:Classe Edge.....	29
Figure 36:Methode pour faire l'héritage	30
Figure 37: main programme de réseau sémantique	31
Figure 38: Fonctionnalité de l'outils protégé.....	33
Figure 39:création des concepts Atomique	34
Figure 40:Rôle atomique.....	35
Figure 41:Représentation des connaissances T-BOX	35
Figure 42:Création de classe Equivalent.....	36
Figure 43:Représentation des connaissances A-Box.....	38

TP1 : 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 Bench-marking et simuler l'inférence d'une BC avec un algorithme.

Etape 1 : Répertoire UBCSAT :

test1.cnf	06/04/2023 18:23	Fichier CNF	1 Ko
test1	07/04/2023 17:49	Document texte	1 Ko
test2.cnf	06/04/2023 18:35	Fichier CNF	1 Ko
ubcsat	09/05/2008 19:37	Fichier	330 Ko
ubcsat	09/05/2008 19:36	Application	356 Ko

Figure 1: Répertoire UBCSAT

Etape 2 : Exécution solveur SAT sur les deux exemples.

1.2.1 Exemple 1 : Exécution solveur SAT sur l'exemple 1 :

Pour L'essai numéro 1 Nous avons exécuter la commande suivante :

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

```
C:\Users\rachi\Bureau\M1\Semestre02\Représentation des connaissances et raisonnement 1\ubcsat-1-1-0>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 2012735864
# -solve 1
# -find,-numsol 1
# -findunique 0
# -srestart 0
# -prestart 0
# -drestart 0
#
# -alpha 1.3
# -rho 0.8
```

Figure 2: Exécution solveur SAT exemple 1

```

# -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      1      1
#
# 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 = 1
Steps_CoeffVariance = 0
Steps_Median = 1
CPUtime_Mean = 0
CPUtime_CoeffVariance = 0
CPUtime_Median = 0

```

Figure 3:Resultat d'exécution

Cette première partie affiche les informations sur la version de l'UBCSAT, un site d'aide et des paramètres du solveur utilisé et le rapport de résultat de traitement.

```

# Solution found for -target 0

1 2 -3 -4 5

```

Figure 4:Solution solveur

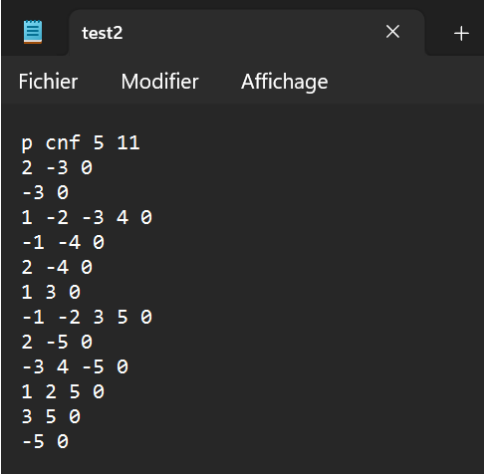
Ceci représente la sortie de l'UBCSAT (le résultat du traitement), dans notre cas il nous affiche qu'il a pu trouver une solution. On peut donc déduire que le fichier « test1.cnf » est satisfiable.

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

Figure 5:Affichage des paramètres

Quant à cet affichage, c'est un rapport qui contient le nombre de variables contenues dans le fichier (5 variables), le nombre de clauses . . . etc

1.2.2 Example 2:



```
p cnf 5 11
2 -3 0
-3 0
1 -2 -3 4 0
-1 -4 0
2 -4 0
1 3 0
-1 -2 3 5 0
2 -5 0
-3 4 -5 0
1 2 5 0
3 5 0
-5 0
```

Figure 6:test2.cnf

```
ubcsat -alg saps -i test2.cnf -solve
```



```

C:\Users\rachi\Bureau\M1\Semestre02\Représentation des connaissances et raisonnement 1\ubcsat-1-1-0>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
# -gttimeout 0
# -noimprove 0
# -target 0
# -wtarg 0
# -seed 1376810353
# -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:

```

Figure 7:exécution test2.cnf

```

# 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      6      6
#
# Solution found for -target 0
#
# 1 2 -3 -4 5
#
Variables = 5
Clauses = 11
TotalLiterals = 27
TotalCPUtimeElapsed = 0.002
FlipsPerSecond = 3000
RunsExecuted = 1
SuccessfulRuns = 1
PercentSuccess = 100.00
Steps_Mean = 6
Steps_CoeffVariance = 0
Steps_Median = 6

```

Figure 8:résultats de test2.cnf

Etape 3 :

Traduction de la base de connaissances relative aux connaissances zoologiques(céphalopodes) sous forme CNF, et tester la satisfiabilité de cette base.

Enoncé du cours :

- Les nautilus sont des céphalopodes ;
- Les céphalopodes sont des mollusques ;
- Les mollusques ont généralement une coquille ;
- Les nautilus en ont une.

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

Soient les 12 symboles non logiques suivant que nous interprétons tel que :

$\{Na, Nb, Nc\}$: {nautilus de a , nautilus de b, nautilus de c}

$\{Céa, Céb, Céc\}$: {céphalopode de a, céphalopode de b, céphalopode de c}

$\{Ma, Mb, Mc\}$: {mollusque de a, mollusque de b, mollusque de c}

$\{Coa, Cob, Coc\}$: {coquille de a, coquille de b , coquille de c}

En ignorant pour l'instant les connaissances utilisant le mot **généralement** nous avons :

$(Na \supset CEa) ; (Nb \supset CEB) ; (Nc \supset CEC);$

$(CEa \supset Ma); (CEb \supset Mb) ; (CEc \supset Mc);$

$(Na \supset COa); (Nb \supset COb); (Nc \supset COc);$

$(Ma \supset COa); (Mb \supset COb) ; (Mc \supset COc);$

$(CEa \supset \neg COa) ; (CEb \supset \neg COb) ; (CEc \supset \neg COc);$

$Na ; CEB ; Mc$

La mise sous forme CNF :

Si on applique la règle $(a \Rightarrow b) = (\neg a \vee b)$ sur les clauses précédentes :

— $(\neg Na \vee Céa) ; (\neg Nb \vee Céb) ; (\neg Nc \vee Céc) ;$

— $(\neg Céa \vee Ma) ; (\neg Céb \vee Mb) ; (\neg Céc \vee Mc) ;$

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

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

— $(\neg Céa \vee \neg Coa) ; (\neg Céb \vee \neg Cob) ; (\neg Céc \vee \neg Coc) ;$

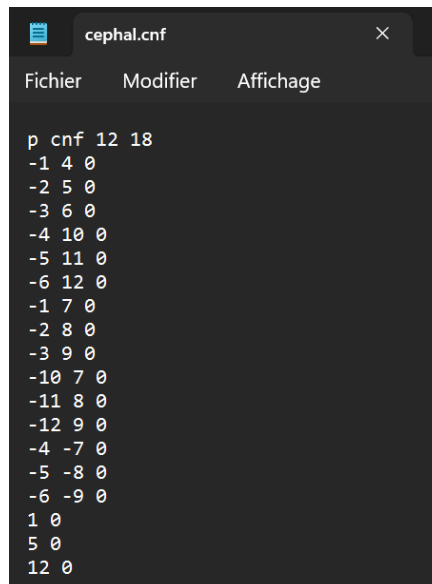
— $Na ; Céb ; Mc$

On va représenter ces clauses dans un fichier CNF sachant que :

$Na=1 , Céa=4 , Coa=7 , Ma=10$

$Nb=2 , Céb=5 , Cob=8 , Mb=11$

Nc=3 ,Céc=6 ,Coc=9 ,Mc=12



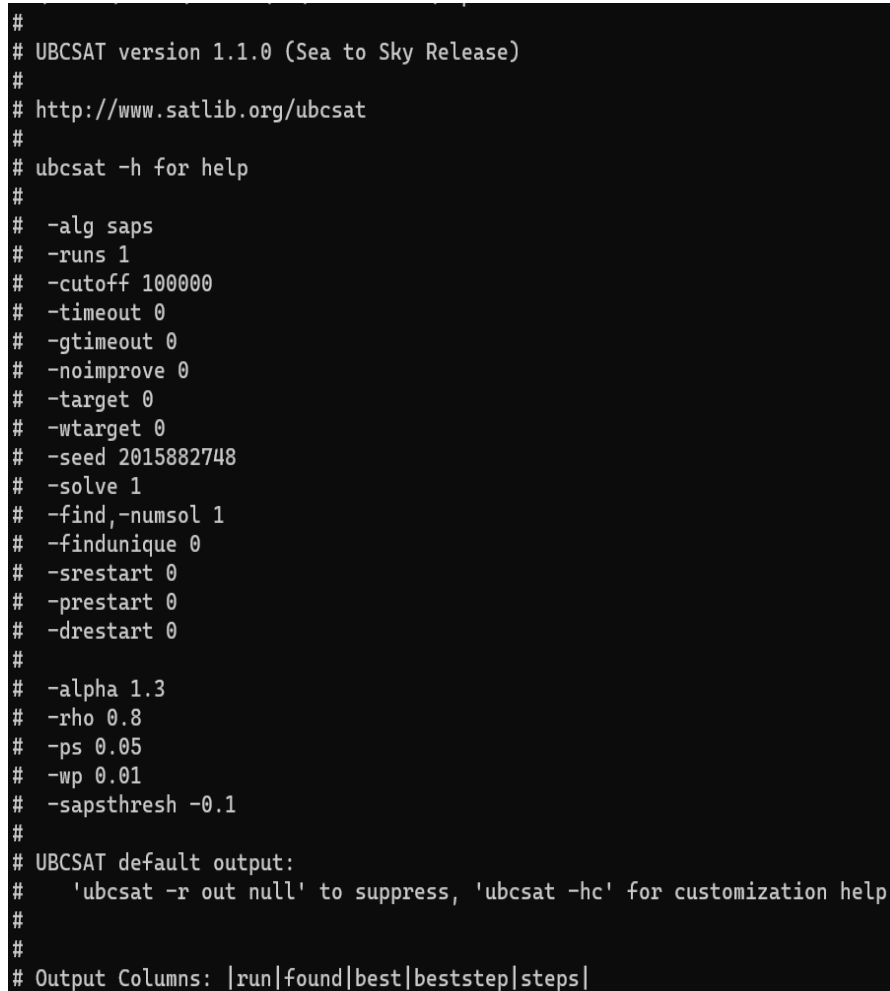
```
cephal.cnf
Fichier  Modifier  Affichage

p cnf 12 18
-1 4 0
-2 5 0
-3 6 0
-4 10 0
-5 11 0
-6 12 0
-1 7 0
-2 8 0
-3 9 0
-10 7 0
-11 8 0
-12 9 0
-4 -7 0
-5 -8 0
-6 -9 0
1 0
5 0
12 0
```

Figure 9: fichier cnf pour Céphalopode

```
ubcsat -alg saps -i cephal.cnf -solve
```

Résultat :



```
#
# 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 2015882748
# -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|
```

Figure 10:exécution de cephal.cnf

```

# 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      2      6      100000
# No Solution found for -target 0

Variables = 12
Clauses = 18
TotalLiterals = 33
TotalCPUTimeElapsed = 0.006
FlipsPerSecond = 16666550
RunsExecuted = 1
SuccessfulRuns = 0
PercentSuccess = 0.00
Steps_Mean = 100000
Steps_CoeffVariance = 0
Steps_Median = 100000
CPUTime_Mean = 0.00600004196167
CPUTime_CoeffVariance = 0
CPUTime_Median = 0.00600004196167

```

Figure 11: Résultats cephal

On remarque qu'aucune solution n'a été trouvée, donc le fichier « cephal.cnf » n'est pas satisfiable.

La seule solution pour éviter ceci, on doit modifier la traduction de généralement : on exclut explicitement les exceptions connues

“ Les céphalopodes qui ne sont pas des nautilus n'ont pas de coquille ”

“ Les mollusques qui ne sont pas des céphalopodes non nautilus en ont une ”

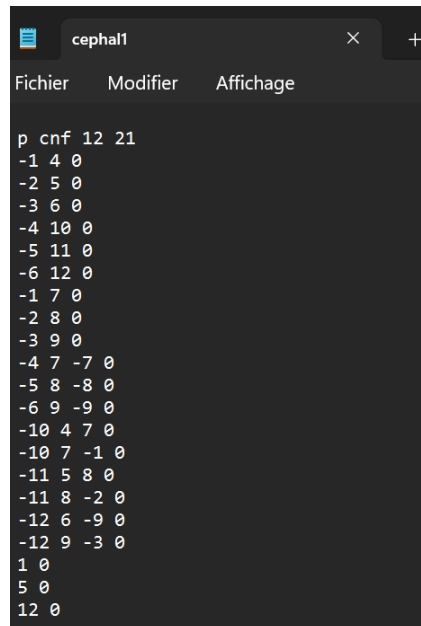
- (Na Cea) ; (Nb Céb) ; (Nc Céc) ;
- (Cea Ma) ; (Céb Mb) ; Céc Mc) ;
- (Na Coa) ; (Nb Cob) ; (Nc Coc) ;
- ((Cea \wedge \neg Na) \supset \neg Coa) ;
- ((Céb \wedge \neg Nb) \supset \neg Cob) ;
- ((Céc \wedge \neg Nc) \supset \neg Coc) ;
- ((Ma \wedge \neg (Cea \wedge \neg Na)) \supset Coa) ;
- ((Mb \wedge \neg (Céb \wedge \neg Nb)) \supset Cob) ;
- ((Mc \wedge \neg (Céc \wedge \neg Nc)) \supset Coc).
- Na ; Céb ; Mc.

Si on applique la règle $:(a \vee b) = (\neg a \wedge \neg b)$ sur les clauses précédentes :

- (\neg Na Cea) ; (\neg Nb Céb) ; (\neg Nc Céc) ;

- $(\neg \text{Céa Ma}) ; (\neg \text{Céb Mb}) ; (\neg \text{Céc Mc}) ;$
- $(\neg \text{Na Coa}) ; (\neg \text{Nb Cob}) ; (\neg \text{Nc v Coc}) ;$
- $\neg \text{Céa} \vee \text{Na} \vee \neg \text{Coa}$
- $\neg \text{Céb} \vee \text{Nb} \vee \neg \text{Cob}$
- $\neg \text{Céc} \vee \text{Nc} \vee \neg \text{Coc}$
- $\neg \text{Ma Céa Coa} ; \neg \text{Ma Coa} \neg \text{Na}$
- $\neg \text{Mb Céb Cob} ; \neg \text{Mb Cob} \neg \text{Nb}$
- $\neg \text{Mc Céc Coc} ; \neg \text{Mc Coc} \neg \text{Nc}$
- $\text{Na} ;$
- $\text{Céb} ;$
- Mc

Représentations des clauses dans le fichier cnf :



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

Figure 12: Fichier cnf cephal1

On execute avec:

```
ubcsat -alg saps -i cephal1.cnf -solve
```

Résultats :

```
C:\Users\rachi\Bureau\M1\Semestre02\Représentation des connaissances et raisonnement 1\ubcsat-1-1-0>ubcsat -alg saps -i cephal1.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
# -wtargt 0
# -seed 2016471737
# -solve 1
# -find,-numsol 1
# -findunique 0
# -restart 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|
#
```

Figure 14: Exécution de cephal1

```
# 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      13      13
#
# Solution found for -target 0
#
# 1 2 -3 4 5 -6 7 8 -9 10
# 11 12
#
Variables = 12
Clauses = 21
TotalLiterals = 48
TotalCPUtimeElapsed = 0.001
FlipsPerSecond = 13001
RunsExecuted = 1
SuccessfulRuns = 1
PercentSuccess = 100.00
Steps_Mean = 13
Steps_CoeffVariance = 0
Steps_Median = 13
CPUtime_Mean = 0.000999927520752
CPUtime_CoeffVariance = 0
CPUtime_Median = 0.000999927520752
```

Figure 13: Solution trouve de cephal1

Solution: $\text{Na} \wedge \neg \text{Nb} \wedge \neg \text{Nc} \wedge \text{Cea} \wedge \text{Ceb} \wedge \neg \text{Cec} \wedge \text{Coa} \wedge \text{Cob} \wedge \neg \text{Coc} \wedge \text{Ma} \wedge \text{Mb} \wedge \text{Mc}$

1.3.1 Exemple d'un fichier Benchmarks :

Exécution du fichier Benchmarks par la commande :

```
C:\Users\rachi\Bureau\M1\Semestre02\Représentation des connaissances et raisonnement 1\ubcsat-1-1-0>ubcsat -alg saps -i benchmarks.cnf -solve
```

Résultats obtenus :

```
# 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      11      11
#
# 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

Variables = 20
Clauses = 91
TotalLiterals = 273
TotalCPUtimeElapsed = 0.001
FlipsPerSecond = 11001
RunsExecuted = 1
SuccessfulRuns = 1
PercentSuccess = 100.00
Steps_Mean = 11
Steps_CoeffVariance = 0
Steps_Median = 11
CPUtime_Mean = 0.000999927520752
CPUtime_CoeffVariance = 0
CPUtime_Median = 0.000999927520752
```

Figure 15: Résultats d'exécution fichier Benchmark

Etape 4 :

Dans cette étape, nous allons écrire un algorithme qui simule l'inférence d'une base de Connaissances

Supposons que nous avons une base de connaissances représentant un petit système d'itinéraire pour une voiture :

Si le GPS est activé (Na), alors la voiture peut se diriger automatiquement (Ma).

Si la caméra arrière est activée (Cea), alors la voiture peut reculer automatiquement (Cob).

Si le capteur de distance avant est activé (Ceb), alors la voiture peut s'arrêter automatiquement (Coa).

Si le capteur de distance arrière est activé (Cec), alors la voiture peut reculer automatiquement (Cob).

Si le bouton de frein est enfoncé (Nb), alors la voiture s'arrête automatiquement (Coa).

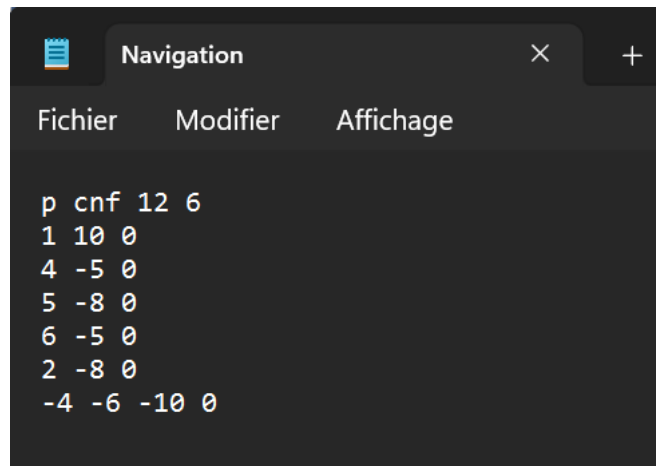
Si la voiture est en train de rouler (Nc), alors la voiture peut être dirigée manuellement (Mb).

Maintenant, supposons que notre but est de s'assurer que la voiture ne recule pas automatiquement (Cob)

Na=1 ,Céa=4 ,Coa=7 ,Ma=10

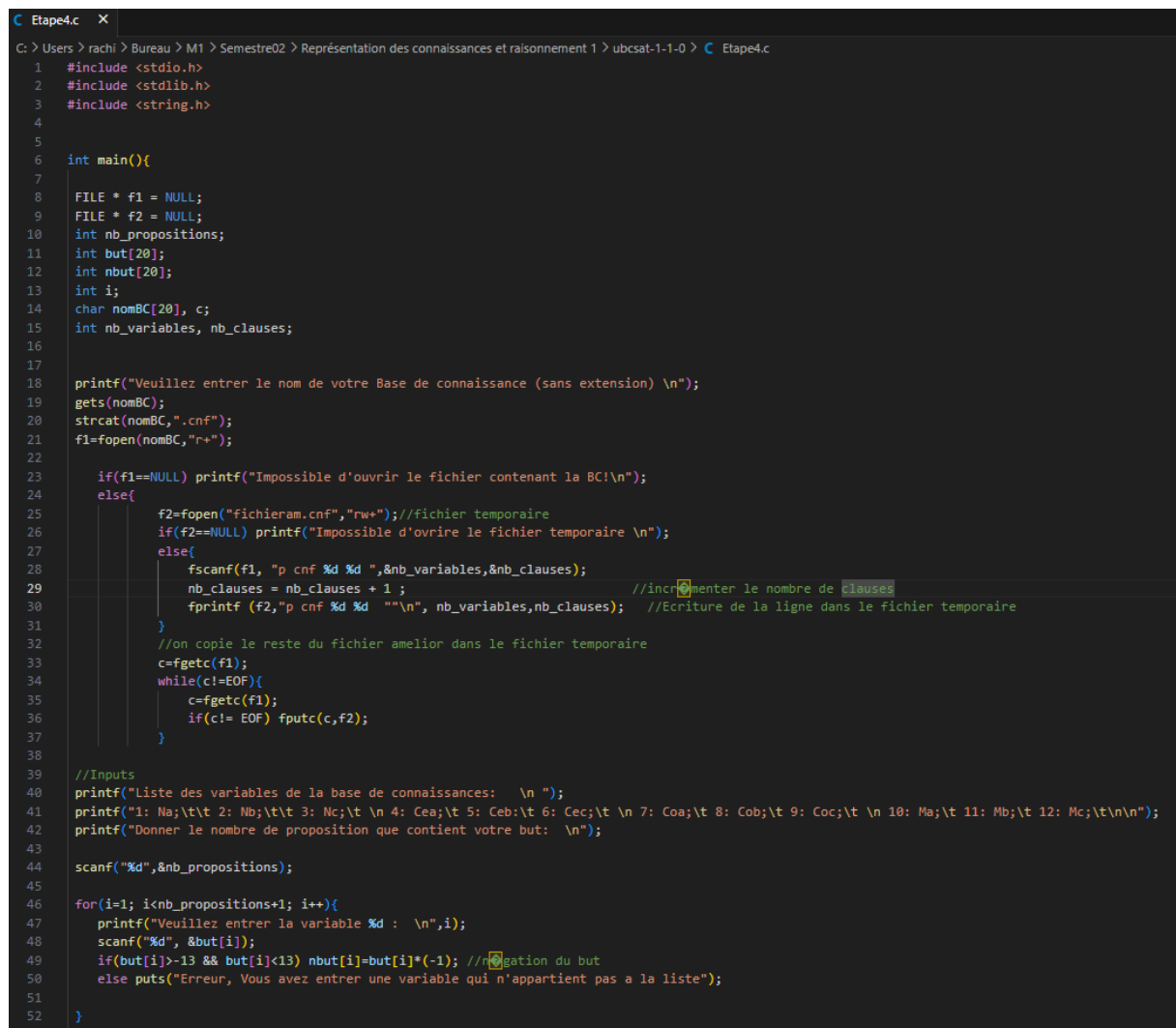
Nb=2 ,Céb=5 ,Cob=8 ,Mb=11 Nc=3 ,Céc=6 ,Coc=9 ,Mc=12

Exécution :



```
p cnf 12 6
1 10 0
4 -5 0
5 -8 0
6 -5 0
2 -8 0
-4 -6 -10 0
```

Figure 16:Navigation.cnf



```
C Etape4.c X
C: > Users > rachi > Bureau > M1 > Semestre02 > Représentation des connaissances et raisonnement 1 > ubcsat-1-1-0 > C Etape4.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5
6  int main(){
7
8      FILE * f1 = NULL;
9      FILE * f2 = NULL;
10     int nb_propositions;
11     int but[20];
12     int nbut[20];
13     int i;
14     char nomBC[20], c;
15     int nb_variables, nb_clauses;
16
17
18     printf("Veuillez entrer le nom de votre Base de connaissance (sans extension) \n");
19     gets(nomBC);
20     strcat(nomBC, ".cnf");
21     f1=fopen(nomBC, "r+");
22
23     if(f1==NULL) printf("Impossible d'ouvrir le fichier contenant la BC!\n");
24     else{
25         f2=fopen("fichieram.cnf", "rw+"); //fichier temporaire
26         if(f2==NULL) printf("Impossible d'ouvrir le fichier temporaire \n");
27         else{
28             fscanf(f1, "p cnf %d %d", &nb_variables, &nb_clauses);
29             nb_clauses = nb_clauses + 1; //incrémenter le nombre de clauses
30             fprintf(f2, "p cnf %d %d\n", nb_variables, nb_clauses); //Ecriture de la ligne dans le fichier temporaire
31         }
32         //on copie le reste du fichier amelior dans le fichier temporaire
33         c=fgetc(f1);
34         while(c!=EOF){
35             c=fgetc(f1);
36             if(c!= EOF) fputc(c, f2);
37         }
38
39         //Inputs
40         printf("Liste des variables de la base de connaissances: \n ");
41         printf("1: Na;\t\t 2: Nb;\t\t 3: Nc;\t\t \n 4: Céa;\t 5: Céb;\t 6: Céc;\t \n 7: Coa;\t 8: Cob;\t 9: Coc;\t \n 10: Ma;\t 11: Mb;\t 12: Mc;\t\n\n");
42         printf("Donner le nombre de proposition que contient votre but: \n");
43
44         scanf("%d", &nb_propositions);
45
46         for(i=1; i<nb_propositions+1; i++){
47             printf("Veuillez entrer la variable %d : \n", i);
48             scanf("%d", &but[i]);
49             if(but[i]>13 && but[i]<13) nbut[i]=but[i]*(-1); //négation du but
50             else puts("Erreur, Vous avez entré une variable qui n'appartient pas à la liste");
51         }
52     }
53 }
```



```

53 //Ajout du 0 representant la fin d une clause dans le fichier cnf
54 fprintf(f2, "\n");
55 for(i=1; i<nb_propositions+1; i++)fprintf(f2,"%d ",nbut[i]); //Ajout du non but au fichier tmp
56 fprintf(f2,"0");
57 fclose(f2); //fermeture du fichier
58
59 //Appel du solver pour le test
60 system("ubcsat -alg saps -i fichieram.cnf -solve > resultat.txt");
61 }
62 fclose(f2);
63
64 //Affichage des r[0]ultats:
65 int quit=0;
66 FILE *f =fopen("resultat.txt","r+");
67 if(f1==NULL) printf("impossible d'ouvrir le fichier resultat.txt \n");
68 else{
69     char texte[1000];
70     while(fgets(texte, 1000, f) && !quit){
71         if(strstr(texte, "# Solution found for -target 0")){
72             printf("\n BC U {Non but} est satisfiable\nSolution trouvee : \n");
73             fscanf(f, "\n");
74             while(!strstr(fgets(texte, 1000, f), "Variables"))
75                 printf("%s", texte); //la solution
76             quit=1;
77         }
78     }
79     if(quit==0){
80         int k;
81         for(k=1;k<nb_propositions+1;k++){
82             printf("-%d ",but[k]);
83         }
84         if(k>2){
85             printf("ne peuvent pas etre atteints");
86         }
87         else {printf("ne peut pas etre atteint\n");}
88     }
89 }
90
91 fclose(f2);
92 }

```

Figure 17:Programme C pour l'etape4

Pour compiler le programme c :

```

C:\Users\rachi\Bureau\M1\Semestre02\Représentation des connaissances et raisonnement 1\ubcsat-1-1-0>gcc ETAPE4.c -o
ETAPE4

```

Figure 18:Compilation du programme C

```

C:\Users\rachi\Bureau\M1\Semestre02\Représentation des connaissances et raisonnement 1\ubcsat-1-1-0>ETAPE4
Veuillez entrer le nom de votre Base de connaissance (sans extension)
Navigation
Impossible d'ovrire le fichier temporaire
Liste des variables de la base de connaissances:
1: Na;      2: Nb;      3: Nc;
4: Cea;     5: Ceb;     6: Cec;
7: Coa;     8: Cob;     9: Coc;
10: Ma;     11: Mb;     12: Mc;

Donner le nombre de proposition que contient votre but:
1
Veuillez entrer la variable 1 :
8
-8 ne peut pas etre atteint

```

Figure 19:Résultat d'exécution du programme C

TP2 : LOGIQUE DES PREDICATS :

Résumé :

Ce travail pratique porte sur l'exploitation de la bibliothèque Tweety pour la modélisation des connaissances en logique des prédicats. L'objectif est d'illustrer des exemples de modélisation logique en utilisant cette bibliothèque.

Dans ce contexte, nous avons utilisé Tweety pour définir des ensembles de symboles logiques tels que des prédicats, des constantes et des fonctions. Nous avons ensuite créé des structures logiques pour représenter des relations entre ces symboles, en utilisant des règles logiques.

Exploitation de l'outils :

Nous allons voir l'utilisation de la librairie Tweety sur un exemple afin de modéliser les différentes approches qu'on a appris dans le cours de la logique du premier ordre.

Déroulement sur un exemple concret :

Le scénario utilisé concerne la modélisation de relations entre animaux et plantes, en utilisant des prédicats et des fonctions. Voici une description détaillée de l'exemple :

- Définition des constantes :

Deux constantes sont créées : "**titi**" et "**bob**", qui sont des animaux, et "**emma**", qui est une plante.

```
FolSignature sig = new FolSignature();
Sort animal = new Sort( name: "animal");
sig.add(animal);
Constant anna = new Constant( name: "titi", animal); //titi est un animal
sig.add(anna);
Constant bob = new Constant( name: "bob", animal); //bob est un animal
sig.add(bob);
Sort plante = new Sort( name: "plante");
sig.add(plante);
Constant emma = new Constant( name: "emma", plante); //emma est une plante
sig.add(emma);
```

Figure 20: Définition des constantes

- Définition des prédicats :

Un prédicat "**Animal**" est défini avec une **arité de 1**, indiquant qu'il s'agit d'un prédicat prenant un terme de type animal en argument.

Un prédicat "**Mange**" est défini avec une **arité de 2**, indiquant qu'il s'agit d'un prédicat prenant deux termes en argument.

```
List<Sort> l = new ArrayList<Sort>();
l.add(animal);
Predicate animals = new Predicate( name: "Animal", l); //creation du prédicat Animal d'arité 1
sig.add(animals);
l = new ArrayList<Sort>();
l.add(animal);
l.add(plante);
Predicate eats = new Predicate( name: "Mange", l); //creation du prédicat Mange qui est d'arité 2
sig.add(eats);
l = new ArrayList<Sort>();
l.add(animal);
```

Figure 21:Définition des prédicats

- Définition des fonctions :

Une fonction "pereDe" est définie avec une arité de 1, indiquant qu'elle prend un terme de type animal en argument.

```
// pereDe est une fonction qui prend un terme qui est de type animal
Functor fatherOf = new Functor( name: "pereDe",l,animal);
sig.add(fatherOf);
```

Figure 22:Définition de fonction

- Création d'un ensemble des formules logiques :

Les formules représentent les connaissances suivantes :

- Pour tout X, si X est un animal, alors il existe un Y que X mange.
- Pour tout X, si X est un animal, alors pereDe(X) est aussi un animal.
- Titi est un animal.
- Titi mange Emma.

```
b.add(parser.parseFormula( text: "forall X:(Animal(X) => (exists Y:(Mange(X,Y))))");
//si X est un animal alors fatherOf(X) est un animal
b.add(parser.parseFormula( text: "forall X:(Animal(X) => Animal(pereDe(X)))");
b.add(parser.parseFormula( text: "Animal(titi)")); //titi est un animal
b.add(parser.parseFormula( text: "Mange(titi,emma)")); //bob mange emma
```

Figure 23:Creation des formules logiques

Affichage des formules logiques dans la console :

```
"C:\Program Files\Java\jdk-19\bin\java.exe" ...
forall X: ((Animal(X)=>exists Y: (Mange(X,Y))))
Mange(titi,emma)
forall X: ((Animal(X)=>Animal(pereDe(X))))
Animal(titi)
Process finished with exit code 0
```

Figure 24:Résultats d'inférence

TP3 : LOGIQUE MODALE

Résumé :

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

- Modale Logic Playground
- La librairie de java Tweety

Exploitation de l'outil MLP :

Dans modal logic playground on choisit le nombre de variable propositionnelle = 3 tel que $a = p, b = q, c = r$

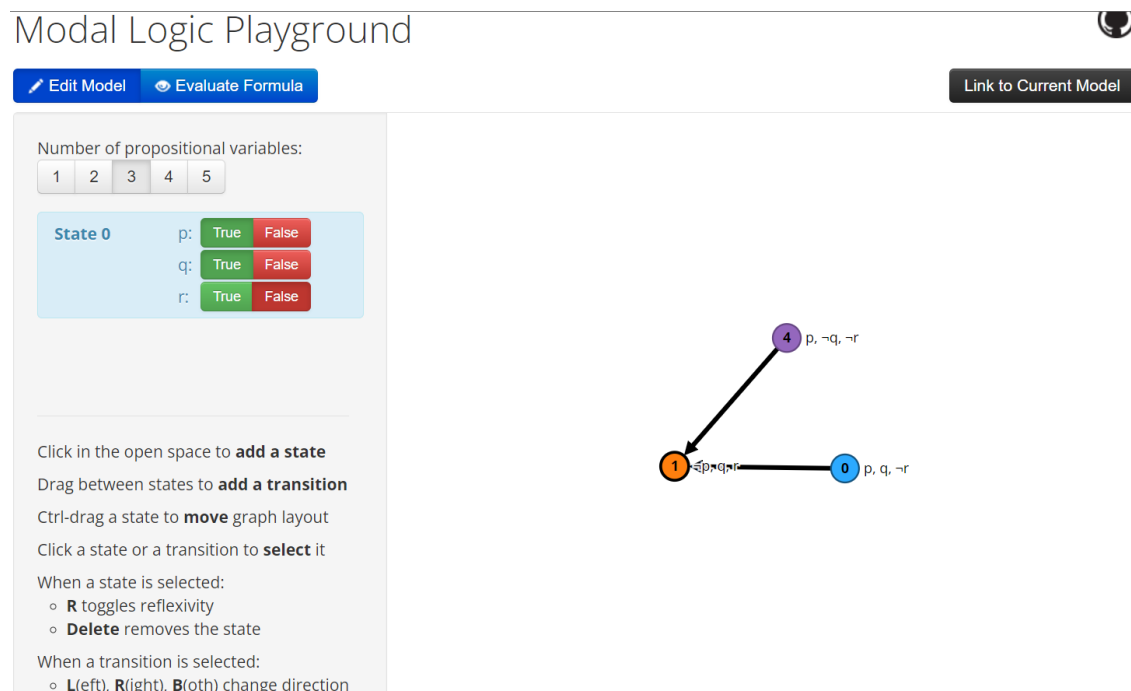
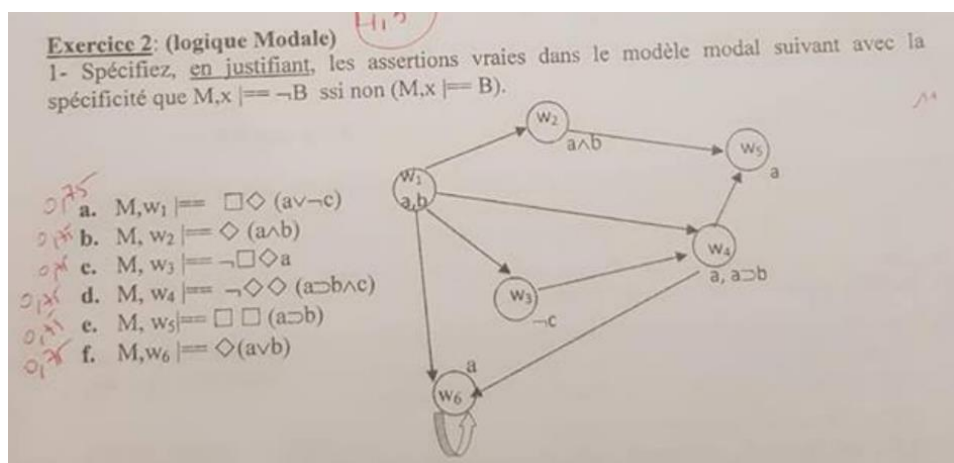
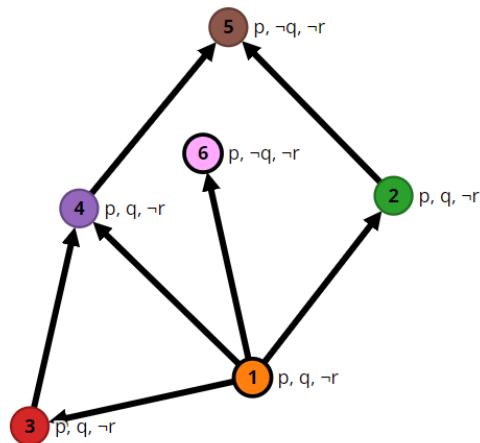


Figure 25: Modale Logic Playground

On va traiter exercice d'un examen (2018) en utilisant Modale Logic Playground





Evaluation:

$\Box \Diamond (a \mid \sim c)$

<p>Enter a formula:</p> <input type="text" value="[]<>(p ~r)"/> <p>Evaluate</p> <p>True: w_1, w_3, w_5, w_6</p> <p>False: w_2, w_4</p> <p>When entering a formula:</p> <ul style="list-style-type: none"> use $\sim A$ for $\neg A$ use $\Box A$ for $\Box A$ 	<p>Current formula: $\Box \Diamond (p \vee \neg r)$</p>
---	---

$\Diamond (a \ \& \ b)$

<p>Enter a formula:</p> <input type="text" value="<>(p&q)"/> <p>Evaluate</p> <p>True: w_1, w_3</p> <p>False: w_2, w_4, w_5, w_6</p> <p>When entering a formula:</p> <ul style="list-style-type: none"> use $\sim A$ for $\neg A$ use $\Box A$ for $\Box A$ 	<p>Current formula: $\Diamond (p \wedge q)$</p>
--	---

$\sim [] \Diamond a$

Enter a formula:

Evaluate

True:
 w_2, w_4

False:
 w_1, w_3, w_5, w_6

Current formula:
 $\neg \Box \Diamond p$

When entering a formula:

- use $\neg A$ for $\neg A$
- use $[]A$ for $\Box A$
- use $\langle \rangle A$ 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 $(A \leftrightarrow B)$

$\neg \langle \rangle \langle \rangle (a \rightarrow b \ \& \ c)$

Enter a formula:

Evaluate

True:
 $w_1, w_2, w_3, w_4, w_5, w_6$

False:
 \emptyset

Current formula:
 $\neg \Diamond \Diamond (p \rightarrow (q \wedge r))$

When entering a formula:

- use $\neg A$ for $\neg A$
- use $[]A$ for $\Box A$
- use $\langle \rangle A$ 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 $(A \leftrightarrow B)$

$\Box \Box (a \rightarrow b)$

Enter a formula:

Evaluate

True:
 w_2, w_4, w_5

False:
 w_1, w_3, w_6

Current formula:
 $\Box \Box (p \rightarrow q)$

When entering a formula:

- use $\neg A$ for $\neg A$
- use $[]A$ for $\Box A$
- use $\langle \rangle A$ 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 $(A \leftrightarrow B)$

$\Diamond(a \mid b)$

Enter a formula:

Evaluate

True:
 w_1, w_2, w_3, w_4, w_6

False:
 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 $(A \mid B)$ for $(A \vee B)$
- use $(A \rightarrow B)$ for $(A \rightarrow B)$
- use $(A \leftrightarrow B)$ for $(A \leftrightarrow B)$

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

Exploitation de l'outil TWEETY :

Voici l'exemple qui démontre comment utiliser Tweety pour travailler avec des formules modales et des formules de logique du premier ordre. Il montre comment créer une base de connaissances modale, ajouter des formules, utiliser un parser pour analyser les formules, et utiliser un raisonneur pour répondre aux requêtes sur la base de connaissances modale.

```
public static void main(String[] args) throws ParseException, IOException {
    MlBeliefSet bs = new MlBeliefSet(); // Mlbeliefset est utilise pour stocker les formules modales
    MlParser parser = new MlParser(); // Mlparser est utilise pour parser les formules modales
    FolSignature sig = new FolSignature(); // FolSignature est utilise pour stocker les symboles de la logique du premier ordre
    // On ajoute les symboles de la logique du premier ordre dans la signature de la logique modale
    sig.add(new Predicate( name: "p", arity: 0));
    sig.add(new Predicate( name: "q", arity: 0));
    // On ajoute la signature de la logique du premier ordre dans le parser de la logique modale
    parser.setSignature(sig);
    // On ajoute les formules modales dans la base de connaissances modale
    bs.add((RelationalFormula)parser.parseFormula( text: "◊(p && q)"));
    bs.add((RelationalFormula)parser.parseFormula( text: "[ ](!p || q)"));
    bs.add((RelationalFormula)parser.parseFormula( text: "[ ](q && ◊(!q))"));
    // On affiche la base de connaissances modale
    System.out.println("Modal knowledge base: " + bs);
    //reasoner est utilise pour repondre aux requetes sur la base de connaissances modale par la methode query
    SimpleMlReasoner reasoner = new SimpleMlReasoner();
    System.out.println("[ ](!p) " + reasoner.query(bs, (FolFormula)parser.parseFormula( text: "[ ](!p)")) + "\n");
    System.out.println("◊(p && q) " + reasoner.query(bs, (FolFormula)parser.parseFormula( text: "◊(p && q)")) + "\n");
    System.out.println("! (q) " + reasoner.query(bs, (FolFormula)parser.parseFormula( text: "! (q)")) + "\n");
}
```

Figure 26:Tweety pour Logique Modale

Les formules représentent les connaissances suivantes :

Il existe un monde où "p" et "q" sont tous deux vrais.

Dans tous les mondes, soit "p" est faux, soit "q" est vrai.

Dans tous les mondes, "q" est vrai et il existe un état où "q" est faux.

Utilisation du raisonneur (SimpleMIRasoner) :

Un objet de type **SimpleMIRasoner** est créé pour répondre aux requêtes sur la base de connaissances modale.

Plusieurs requêtes sont formulées et le résultat est affiché à l'écran.

Resultats d'exécution :

```
"C:\Program Files\Java\jdk-19\bin\java.exe" ...  
Modal knowledge base: { [](!p||q), <>(p&&q), [](q&&<>(!q)) }  
[](p)      true  
  
<>(p && q)  true  
  
!(q)  true
```

Figure 27:Resultat des formules pour logique Modale

TP4 : Logique des defaults :

Résumé :

L'objectif de ce TP est d'exploiter un outil de raisonnement basé sur la logique des defaults

Outil utilisé :

L'outil utilisé pour ce TP est un raisonneur de logique par défaut basé sur la logique par défaut de Reiter, développé par Evan Morrison. Disponible par le lien :

<http://edm92.github.io/defaultlogic/>

Cet outil utilise la librairie Orbital qui est une bibliothèque de classes Java qui offre des représentations et des algorithmes pour la logique mathématiques et informatique (calcul formel, algorithmes numériques, démonstrations de théorèmes, recherches et planification)

Exploration de l'outil :

Code de création d'un default :

```
DefaultRule rule1 = new DefaultRule();
rule1.setPrerequisite("A");
rule1.setJustification("B");
rule1.setConsequence("C");
```

Figure 28:Création d'un default

Dans ce TP nous allons nous intéresser à l'application du raisonneur pour la logique des defaults. La classe **DefaultReasoner** implémente le processus de raisonnement de la logique des defaults en définissant une théorie $\Delta = \langle w, d \rangle$ comme un ensemble w de formules du langage (WorldSet) et un ensemble d de defaults (RuleSet). Un défaut est défini par son prérequis, sa justification et sa conséquence. La méthode **getPossibleScenarios()** retourne pour une théorie donnée un HashSet d'extension valide après avoir vérifié l'utilisabilité et l'applicabilité de chaque default.

La classe DefaultLogicExample offre 13 exemples prédéfinies de théories, la classe ExtraExample offre 2 exemples prédéfinies de théories prenant en considération des connaissances concrètes.

Utilisation de l'outil :

Dans ce qui suit nous allons voir l'application du raisonneur sur plusieurs exemples certains sont prédéfinis par Evan Morrison dans son projet.

Exemple 1 : (est dans Extras Example)

- Tweety est un oiseau
- En général, les oiseaux volent
- Tweety est un pingouin
- Les pingouins ne volent pas

$\Delta = \langle w, d \rangle$

$w = \{ \text{tweety_is_a_bird}, \text{tweety_is_a_penguin}, \text{tweety_is_a_penguin} \rightarrow \neg \text{tweety_can_fly} \}$

$d = \{ \text{tweety_is_a_bird} : \text{tweety_can_fly} / \text{tweety_can_fly}, \\ \text{tweety_is_a_bird} : \text{tweety_is_a_penguin} / \neg \text{tweety_can_fly} \}$

Résultats :

```
C:\Users\rachi\IdeaProjects\edm92-defaultlogic-d6e0a11\lib\log4j-1.2.15.jar;C:\Users\rachi\IdeaProjects\edm92-defaultlogic-d6e0a11\lib\orbital-core.jar
be.fnord.DefaultLogic.ExtraExample
We one day learn that tweety is penguin.
Trying  tweety_is_a_bird & tweety_is_a_penguin & (tweety_is_a_penguin -> ~tweety_can_fly)
Trying  tweety_is_a_bird & tweety_is_a_penguin & (tweety_is_a_penguin -> ~tweety_can_fly)
Given the world:
    tweety_is_a_bird & tweety_is_a_penguin & (tweety_is_a_penguin -> ~tweety_can_fly)
And the rules
    [(tweety_is_a_bird):(tweety_can_fly) ==> (tweety_can_fly)] , [(tweety_is_a_bird):(tweety_is_a_penguin) ==> (~tweety_can_fly)]
Possible Extensions
    Ext: Th(W U (~tweety_can_fly))
    = ~tweety_can_fly & tweety_is_a_bird & tweety_is_a_penguin & (~tweety_can_fly | ~tweety_is_a_penguin)

Process finished with exit code 0
```

Figure 29:Execution exemple 1 Logique des defaults

Exemples 2(Exo 2 serie Td)

$\Delta = \langle W, D \rangle$

$W = \{A\}$

$D = \{A : \neg B / B\}$

```
Given the world:
    A
And the rules
    [(A):(~B) ==> (B)]
Possible Extensions
Execution time was 360 ms.

Process finished with exit code 0
```

Figure 30:Execution Example 2 logiques des defaults

On remarque bien que on n'a pas d'extension

Exemple 3(exemple 2 de exo1) :

$\Delta = \langle w, d \rangle$

$w = \{A, \neg B\}$

$d = \{A : B/C, A : \neg C/D\}$

Code :

```
public static void example2exo1() {  
    WorldSet myWorld = new WorldSet();  
    myWorld.addFormula(_wff: "A");  
    myWorld.addFormula(_wff: a.e.NOT + "B");  
  
    DefaultRule rule1 = new DefaultRule();  
    rule1.setPrerequisite("A");  
    rule1.setJustificatoin("B");  
    rule1.setConsequence("C");  
  
    DefaultRule rule2 = new DefaultRule();  
    rule2.setPrerequisite("A");  
    rule2.setJustificatoin(a.e.NOT + "C");  
    rule2.setConsequence("D");  
}
```

Figure 31:Exécution Exemple 3 Default

Résultats :

```
be.fnord.DefaultLogic.TDExemples  
Trying  A & ~B  
Trying  A & ~B  
Given the world:  
    A & ~B  
And the rules  
    [(A):(B) ==> (C)] , [(A):(~C) ==> (D)]  
Possible Extensions  
    Ext: Th(W U (D))  
    = D & ~B & A  
  
Process finished with exit code 0
```

Figure 32:Resultats Example 3

TP5 : Réseau Sémantique

Outils utilise :

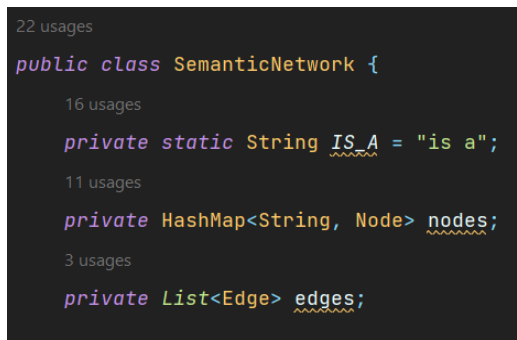
On a utilisé le langage de programmation JAVA afin de programmer le réseau

Explication :

La classe "SemanticNetwork" encapsule la fonctionnalité liée au réseau sémantique, incluant les nœuds, les relations et l'algorithme de marquage. Elle possède deux attributs :

- "nodes", qui est une collection associant des clés (sous forme de chaînes de caractères) à des valeurs (nœuds), représentant les différents nœuds de notre réseau sémantique.
- "edges", qui est une liste d'objets de type "Edge", représentant les différentes relations entre les nœuds.

De plus, la classe dispose d'un attribut de classe appelé "IS_A" qui est utilisé pour définir la relation "est un".

A screenshot of a code editor showing the Java code for the SemanticNetwork class. The code is as follows:

```
22 usages
public class SemanticNetwork {
    16 usages
    private static String IS_A = "is a";
    11 usages
    private HashMap<String, Node> nodes;
    3 usages
    private List<Edge> edges;
```

Figure 33:Attribut Réseau sémantique

Classe Node :

Attributs : Elle contient 4 attributs :

- name : qui est l'étiquette du nœud
- edges : toutes les relations du nœud avec les autres.
- exceptionLinks: tous les liens d'exceptions.
- visitedFrom : un attribut qui sert pour l'algorithme de marquage.

Méthodes : Les getters et les setters, et les méthodes suivantes :

- addEdge() : permet d'ajouter une relation au nœud.
- getExceptionLinks() : Cette méthode renvoie la liste des liens d'exception du nœud.
- addExceptionLink(Edge edge): Cette méthode ajoute un lien d'exception (passé en paramètre) à la liste des liens d'exception du nœud.

```

public static class Node {
    private String name;
    private List<Edge> edges;
    private int visitedFrom;
    private List<Edge> exceptionLinks;

    public Node(String name) { ...

    public String getName() { ...

    public void addEdge(Edge edge) { ...

    public List<Edge> getEdges() { ...

    public int getVisitedFrom() { ...

    public void setVisitedFrom(int visitedFrom) { ...

    public List<Edge> getExceptionLinks() { ...

    public void addExceptionLink(Edge edge) { ...
}

```

Figure 34:Classe Node

Classe Edge:

Attributs :

Elle contient 3 attributs :

- from: le nœud de départ de la relation.
- to:le nœud final de la relation.
- name: qui est l'étiquette de la relation.

```

public static class Edge {
    private Node from;
    private Node to;
    private String name;

    public Edge(Node from, Node to, String name) { ...

    public Node getFrom() { ...

    public Node getTo() { ...

    public String getName() { ...
}

```

Figure 35:Classe Edge

addNode et addEdge permettent de rajouter un nœud et une relation dans “nodes” et “edges” respectivement

public List<Node> propagate(Node start,int mark1,Node end ,int mark2) :

Cette méthode va prendre en paramètre le nœud de départ et le nœud d'arrivée généralement on applique l'algorithme de marquage pour répondre à une question du genre “Quelles sont les guerres qui ont généré des pertes financières ?” Ici le nœud de départ c’est “guerre” et le nœud d’arrivée est “pertes financières” et le but de l’algorithme c’est de retrouver toutes les guerres qui génère des pertes financières.

Le paramètre “mark1” désigne la marque M1 et “mark2” désigne la marque M2 de l’algorithme de marquage. La méthode propagate va faire appel à une autre méthode propagate deux fois qui est surchargé en commençant à propager la marque M1 depuis le noeud de départ (appel 1) puis de propager la marque M2 (appel 2) puis retourner les noeuds qui ont la marque M1 et qui sont en relation avec les noeuds de la marque M2.

private boolean isException(Node node, Edge edge): Cette méthode vérifie si un lien donné (edge) est une exception pour un nœud donné (node). Elle parcourt les liens d'exception du nœud et vérifie si l'un d'entre eux correspond exactement au lien donné.

public List<Node> inheritance(Node node) : Cette méthode implémente un algorithme pour obtenir les propriétés héritées d'un nœud. Elle parcourt les arêtes sortantes d'un nœud donné, elle vérifie si le lien est une exception en appelant la méthode ***isException***. Si le lien est identifié comme une exception, la méthode ignore cette arête et passe à l'arête suivante sans ajouter le nœud de destination correspondant à la liste des propriétés héritées. Ensuite, elle continue à explorer les arêtes sortantes des nœuds nouvellement ajoutés jusqu'à ce qu'il n'y ait plus de liens "is a" à suivre. La méthode renvoie finalement la liste des propriétés héritées.

```
public List<Node> inheritance(Node node) {
    List<Node> properties = new ArrayList<>();
    List<Node> parents = new ArrayList<>();
    parents.add(node);
    while (!parents.isEmpty()) {
        Node parent = parents.remove(0);
        for (Edge edge : parent.getEdges()) {
            if (edge.getName().equals(SemanticNetwork.IS_A)) {
                if (isException(node, edge)) {
                    continue;
                }
                Node child = edge.getTo();
                if (!properties.contains(child)) {
                    properties.add(child);
                    parents.add(child);
                }
            }
        }
    }
    return properties;
}
```

Figure 36:Methode pour faire l'héritage

Dans le *main*, on crée un réseau sémantique en utilisant la classe **SemanticNetwork** et effectue différentes opérations sur ce réseau. Tout d'abord, le réseau est initialisé avec des nœuds représentant des concepts liés à la guerre, tels que "guerre conventionnelle", "guerre nucléaire", "guerre froide", etc. Ensuite, des relations sont établies entre ces nœuds, en utilisant des liens "est un" et des liens spécifiques comme "a généré". Par exemple, il est spécifié que "guerre conventionnelle" est une sorte de "guerre", et "guerre froide" est également une sorte de "guerre". De plus, certaines relations sont établies entre des nœuds spécifiques, comme le lien "adhère" entre "fln" et "parti politique".

```
public static void main(String[] args) {
    semanticNetwork.addEdge(from:"guerre nucleaire", to:"guerre", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"guerre technologique", to:"guerre", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"guerre technologique collaborative mondiale", to:"guerre technologique",
        SemanticNetwork.IS_A);
    Edge perteEdge = semanticNetwork.addEdge(from:"guerre technologique", to:"pertes financieres",
        edgeName:"genere");
    semanticNetwork.addNode(nodeName:"guerre technologique sans pertes financieres");
    semanticNetwork.addEdge(from:"usa russie", to:"guerre froide", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"guerre froide", to:"guerre", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"guerre algerie", to:"guerre conventionnelle", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"guerre algerie", to:"chouhada", edgeName:"a genere");
    semanticNetwork.addEdge(from:"1 guerre mondiale", to:"guerre conventionnelle", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"1 guerre mondiale", to:"poilus", edgeName:"a genere");
    semanticNetwork.addEdge(from:"2 guerre mondiale", to:"guerre conventionnelle", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"chouhada", to:"victime militaire", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"chouhada", to:"fln", edgeName:"adhere");
    semanticNetwork.addEdge(from:"poilus", to:"victime militaire", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"victime militaire", to:"perte humaine", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"victime militaire", to:"parti politique", edgeName:"adhere pas");
    semanticNetwork.addEdge(from:"fln", to:"parti politique", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"larbi benmhidi", to:"chouhada", SemanticNetwork.IS_A);
    semanticNetwork.addEdge(from:"hassiba benbouali", to:"chouhada", SemanticNetwork.IS_A);

    // Ajout de liens d'exception
    semanticNetwork.addExceptionLink(fromNode:"guerre technologique collaborative mondiale", perteEdge);
}
```

Figure 37: main programme de réseau sémantique

Ensuite, un lien d'exception est ajouté pour spécifier qu'il y a une exception à la relation "génère" entre "guerre technologique" et "pertes financières". Cela signifie que dans la plupart des cas, la guerre technologique génère des pertes financières, mais il existe des situations exceptionnelles où cela n'est pas le cas.

```

Node n1 = semanticNetwork.getNode(nodeName:"guerre");
Node n2 = semanticNetwork.getNode(nodeName:"victime militaire");

// propager les marqueurs 1 et 2 (guerre et victime militaire)
List<Node> nodes = semanticNetwork.propagate(n1, mark1:1, n2, mark2:2);

// pour chaque noeud dans la liste nodes1
System.out.println("Quelle guerre a des victime militaire?\n");
for (Node node : nodes) {
    System.out.println(node.getName());
}

System.out.println("\n*****\n");
// inheritance
System.out.println("Les props de la guerre d'algerie:\n");
Node guerreAlgerie = semanticNetwork.getNode(nodeName:"guerre algerie");
List<Node> properties = semanticNetwork.inheritance(guerreAlgerie);
for (Node node : properties) {
    System.out.println(node.getName());
}

System.out.println("\n*****\n");
// exception link
System.out.println("Guerre technologique avec des pertes financieres?\n");
Node guerreTechnologique = semanticNetwork.getNode(nodeName:"guerre technologique");

```

Ensuite, on effectue différentes opérations sur le réseau sémantique. Tout d'abord, on utilise la méthode **propagate** pour propager des marqueurs à travers le réseau. Dans cet exemple, un marqueur 1 est propagé à partir du nœud "guerre" et un marqueur 2 à partir du nœud "victime militaire". La méthode retourne une liste de nœuds qui répondent aux critères de propagation, c'est-à-dire les nœuds qui sont connectés d'une manière spécifique à ces marqueurs. Ces nœuds sont affichés en réponse à la question "Quelle guerre a des victimes militaires ?".

```

Quelle guerre a des victime militaire?

1 guerre mondiale
guerre algerie

```

Ensuite, la méthode **inheritance** est utilisée pour récupérer les propriétés héritées d'un nœud spécifique. Dans cet exemple, les propriétés héritées de "guerre algérie" sont affichées. Cela signifie que les nœuds qui sont directement ou indirectement connectés à "guerre algérie" par une relation "est un" sont récupérés.

```

Les props de la guerre d'algerie:

guerre conventionnelle
guerre
conflit

```

Enfin, la méthode **propagate** est utilisée à nouveau pour trouver les nœuds qui répondent à une autre question, à savoir "Quelles guerres technologiques génèrent des pertes financières ?". Le nœud "guerre technologique" est propagé avec un marqueur 3 et le nœud

```

Guerre technologique avec des pertes financieres?

guerre technologique

```

"pertes financières" avec un marqueur 4. Les nœuds qui répondent à cette propagation sont affichés.

TP6 : LOGIQUES DES DESCRIPTIONS :

Objectif :

L'objectif de ce TP consiste à utiliser un raisonneur pour simuler le raisonnement en exploitant les TBOX et les Abox des exercices précédents, L'exercice pris comme exemple est l'exercice 4

Outil utilisé :

L'outil utilisé pour ce TP est "Protégé", c'est un éditeur graphique d'ontologies open source utilisé pour effectuer des raisonnements sur ces ontologies à l'aide de raisonneur tels que "Pellet" et "HermiT". Il est disponible en téléchargement sur ce lien :

<https://protege.stanford.edu>

Exploration de l'outil :

L'outil est un Éditeur graphique il contient beaucoup de fonctionnalité mais les plus intéressantes dans ce TP sont dans la figure suivante :

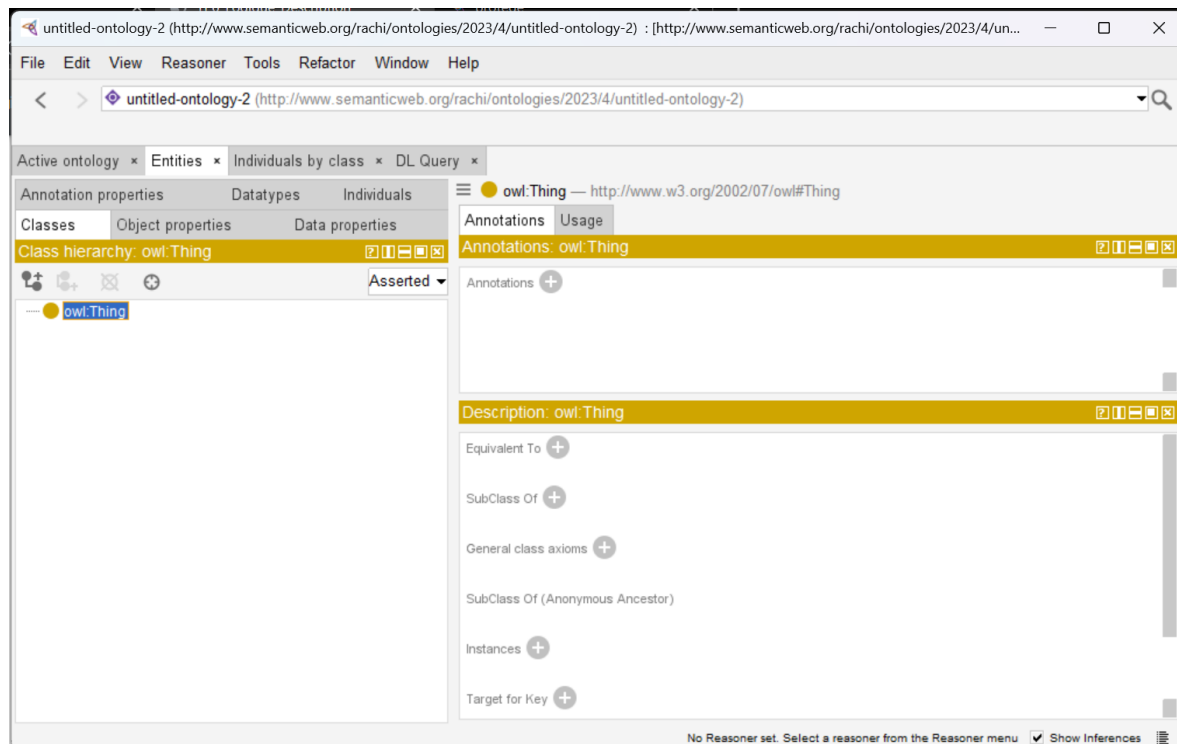


Figure 38: Fonctionnalité de l'outils protégé

1 - **Classes** : Dans cette section on peut représenter les concepts atomiques et les différentes connaissances.

2 - **Objects Properties** : Ici on définit les différents rôles.

3 - **Individuals by Class** : Dans cette onglet les différentes instances des classes sont définies donc c'est ici qu'on définit la A-BOX.

Utilisation de l'outil : Pour cette partie on va utiliser le raisonneur pour définir la TBOX et ABOX de l'exercice 4 du TD :

Exercice 4 :

Représentez les connaissances en utilisant la logique de description dans laquelle un concept complexe est défini par :

$$C \rightarrow A \mid \top \mid \perp \mid \neg C \mid C \sqcap D \mid \forall R.C \mid \exists R.C \mid C \sqcup D \mid \text{au moins } n R \mid \text{au plus } n R$$

Les concepts atomiques sont dénotés par A et B et un rôle atomique est dénoté par R. Dans la A-box, C(a) représente l'assertion d'un concept alors que R(b,c) représente l'assertion de rôle.

Les concepts atomiques et les rôles doivent être prédéfinis au préalable

Soient les concepts atomiques EVENEMENT, DOMMAGE-NATURE, ATTEINTE-ECOSYS, CHIMIE-EAUX

Et le rôle atomique : provoque

- a- Les catastrophes écologiques sont des événements qui provoquent toujours des dommages à la nature et des atteintes à l'écosystème.
- b- Les phénomènes géophysiques sont des événements qui provoquent des catastrophes écologiques.
- c- Les séismes, les cyclones et les éruptions volcaniques sont des phénomènes géophysiques.
- d- Les risques naturels, les feux de forêts sont des catastrophes écologiques.
- e- Les accidents nucléaires sont des catastrophes technologiques.
- f- Les catastrophes écologiques et les catastrophes technologiques sont distinctes.
- g- Les catastrophes technologiques n'induisent pas des dommages à la nature.
- h. Les inondations sont des risques naturels qui provoquent un déséquilibre sur la chimie des eaux.
- h- Tchernobyl est un accident nucléaire.
- i- Les feux en Amazonie sont des catastrophes écologiques.

Définition des Concepts Atomique :

Avant de commencer à définir les connaissances on définit tout d'abord par les concepts atomiques, on va les ajouter dans l'onglet classes :

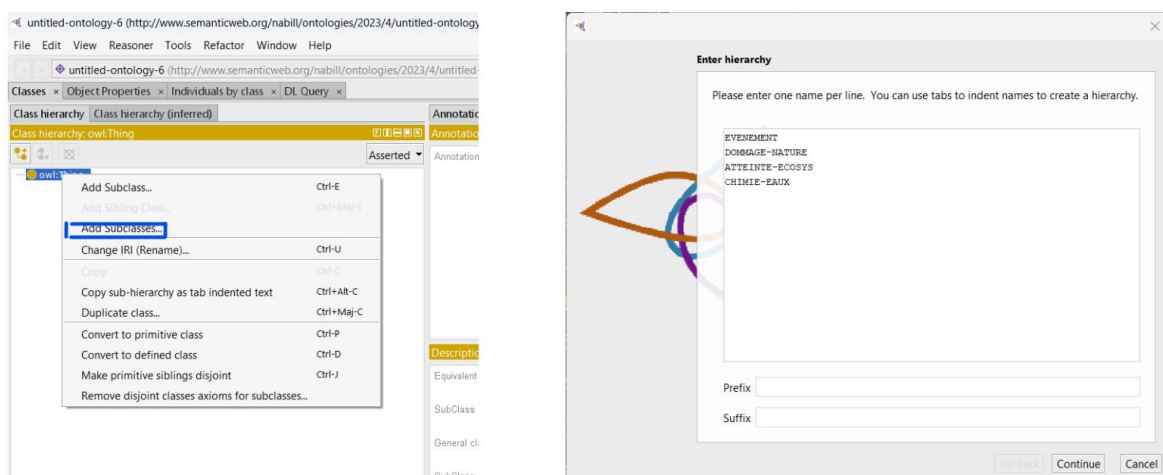


Figure 39:création des concepts Atomique

Après l'ajout on remarque que les Concepts ont été ajouté sous la classe Thing qui est le "Tout" :



Définition des Rôles atomiques :

Comme pour les Concepts on ajoute les rôles atomiques avec les mêmes étapes sauf qu'on le fait dans la section "Object Propertés" :

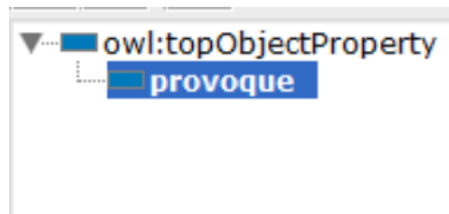


Figure 40:Rôle atomique

Représentation des Connaissances T-BOX:

Pour définir un nouveau concept on ajoute une sous-classe de la classe Thing puis on sélectionne la nouvelle classe créée et on la définit soit par l'inclusion (Sub Class of) ou l'équivalence (Equivalent to).

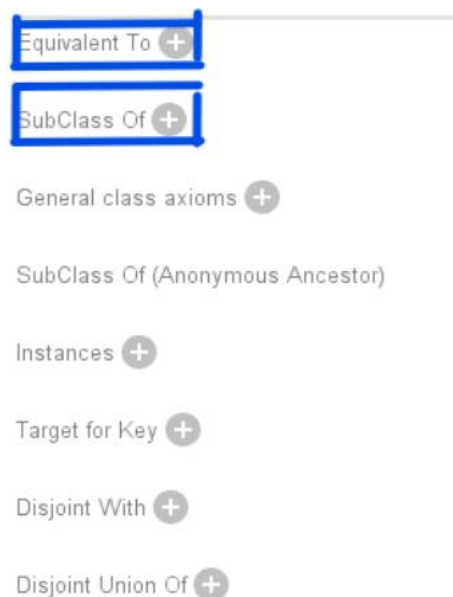


Figure 41:Représentation des connaissances T-BOX

On va commencer à représenter le premier concept de l'exercice :

- a- Les catastrophes écologiques sont des événements qui provoquent toujours des dommages à la nature et des atteintes à l'écosystème.

$CATASTROPHES-ECOLOGIQUE \equiv EVENEMENT \sqcap \forall \text{provoque.}(DOMMAGE-NATURE \sqcap ATTEINTE-ECOSYS)$

Ici on crée une nouvelle classe appelé "CATASTROPHES-ECOLOGIQUE" puis on clique sur le + à côté de "Equivalent To" puisque c'est une équivalence :

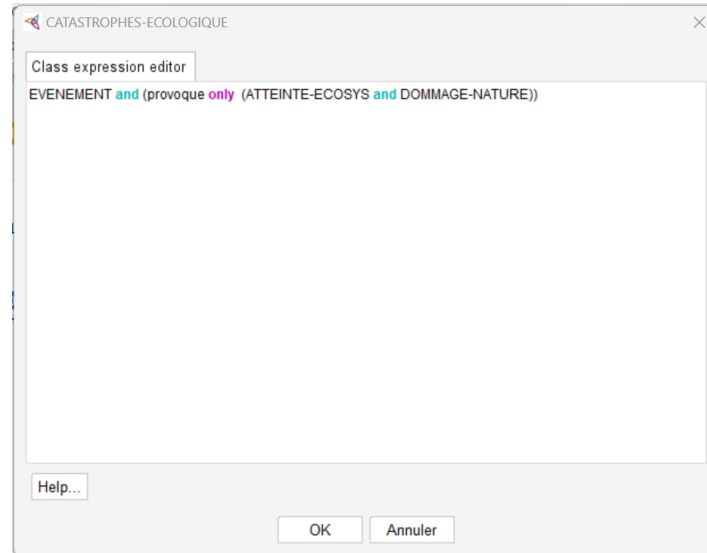


Figure 42:Création de classe Equivalent

Le "and" représente \sqcap et le "only" représente le \forall et on obtient la description suivante pour la classe CATASTROPHES-ECOLOGIQUE :



On fait la même chose pour chaque connaissance sachant que :

"or" représente \sqcup .

"some" équivaut au \exists .

"min" représente "au moins n R".

"not" équivaut a \neg

et enfin le "max" représente "au plus n R".

- b- Les phénomènes géophysiques sont des évènements qui provoquent des catastrophes écologiques.

$PHENOMENES-GEOPHYSIQUES \equiv EVENEMENT \sqcap \exists \text{provoque.CATASTROPHES-ECOLOGIQUE}$

Description: PHENOMENES-GEOPHYSIQUES

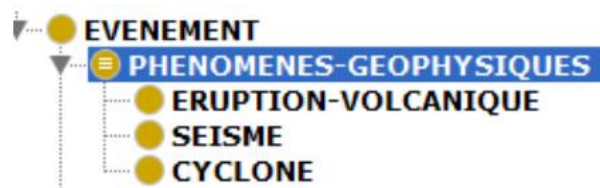
Equivalent To +

● **EVENEMENT**
and (provoque some CATASTROPHES-ECOLOGIQUE)

- c- Les séismes, les cyclones et les éruptions volcaniques sont des phénomènes géophysiques.

$SEISME \subseteq PHENOMENES-GEOPHYSIQUES$ $CYCLONE \subseteq PHENOMENES-GEOPHYSIQUES$ $ERUPCION-VOLCANIQUE \subseteq PHENOMENES-GEOPHYSIQUES$

L'inclusion peut aussi être représenté par une arborescence :



- d- Les risques naturels, les feux de forêts sont des catastrophes écologiques.

$RISQUE-NATUREL \subseteq CATASTROPHES-ECOLOGIQUE$ $FEUX-FORETS \subseteq CATASTROPHES-ECOLOGIQUE$



- e- Les accidents nucléaires sont des catastrophes technologiques.

$ACCIDENT-NUCLEAIRE \subseteq CATASTROPHE-TECHNOLOGIQUES$ Ici on doit d'abord définir le concept "CATASTROPHE-TECHNOLOGIQUES" puis le concept "ACCIDENT-NUCLEAIRE".

Description: ACCIDENT-NUCLEAIRE

Equivalent To +

SubClass Of +

● **CATASTROPHE-TECHNOLOGIQUES**

- f- Les catastrophes écologiques et les catastrophes technologiques sont distinctes.
 $CATASTROPHES-ECOLOGIQUE \sqcap CATASTROPHE-TECHNOLOGIQUES \equiv \perp$

Dans cette outil le \perp doit être représenté par une classe créée appelé "Nothing" puis on la définit.

Description: owl:Nothing

Equivalent To +

● **CATASTROPHES-ECOLOGIQUE** and **CATASTROPHE-TECHNOLOGIQUES**

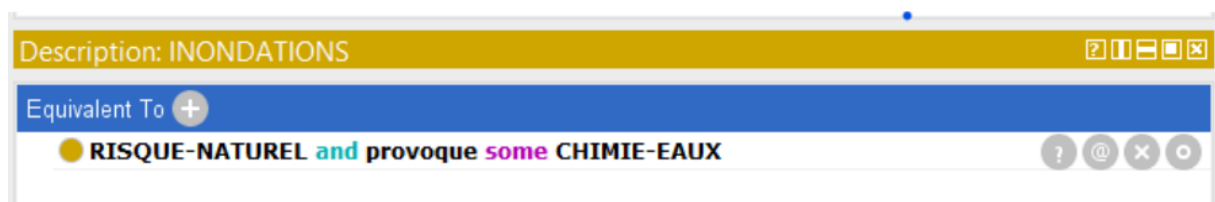
SubClass Of +

- g- Les catastrophes technologiques n'induisent pas des dommages à la nature.
 $CATASTROPHE\text{-}TECHNOLOGIQUES \sqsubseteq \exists \text{provoque.}(\neg \text{DOMMAGE-NATURE})$

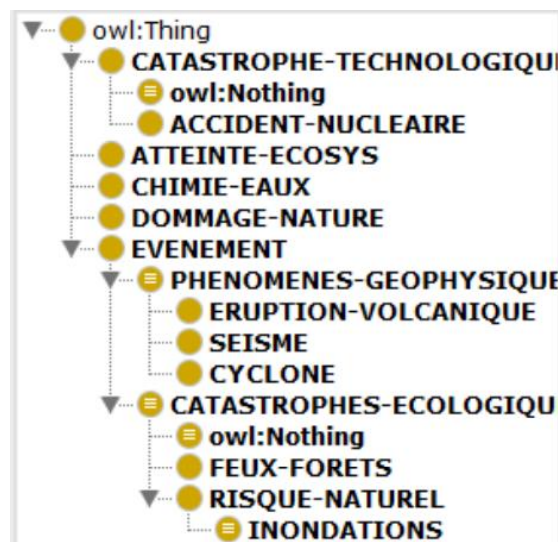


- h. Les inondations sont des risques naturels qui provoquent un déséquilibre sur la chimie des eaux.

$INONDATIONS \equiv RISQUE\text{-}NATUREL \sqcap \exists \text{provoque.CHIMIE-EAUX}$



Après la représentation des connaissances on obtient cet arbre hiérarchique :



Représentation des Connaissances A-BOX : On représente la ABOX dans la section Individuals by class en sélectionnant la classe puis crée une instance

- h- Tchernobyl est un accident nucléaire.

ACCIDENT-NUCLEAIRE(TCHERNOBYL)

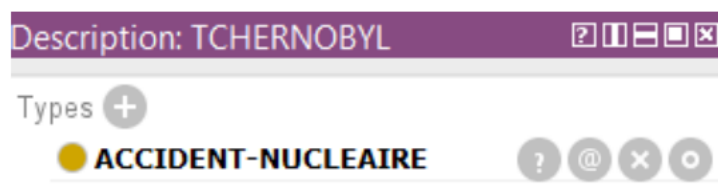


Figure 43: Représentation des connaissances A-Box

- i- Les feux en Amazonie sont des catastrophes écologiques.

CATASTROPHES-ECOLOGIQUE (feux-Amazonie)

