

Rapport TP3  
Applications en logique des défauts  
Représentation des Connaissances et Raisonnement 1

BOUROUINA Rania  
181831052716

CHIBANE Ilies  
181831072041

# 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("\t = " + 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("\t = " + 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("\t = " + 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.setJustificatoion(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("= " + 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\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)
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

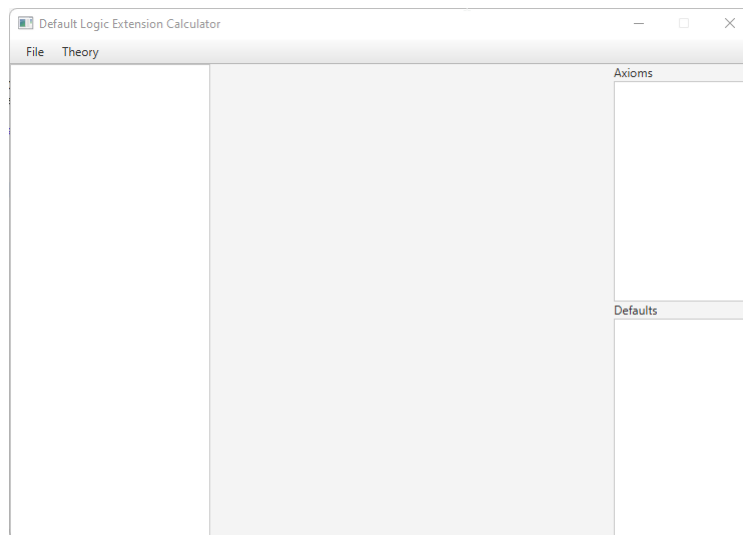


FIGURE 18 – Interface de l'outil Extension Calculator

## Exercice 1

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

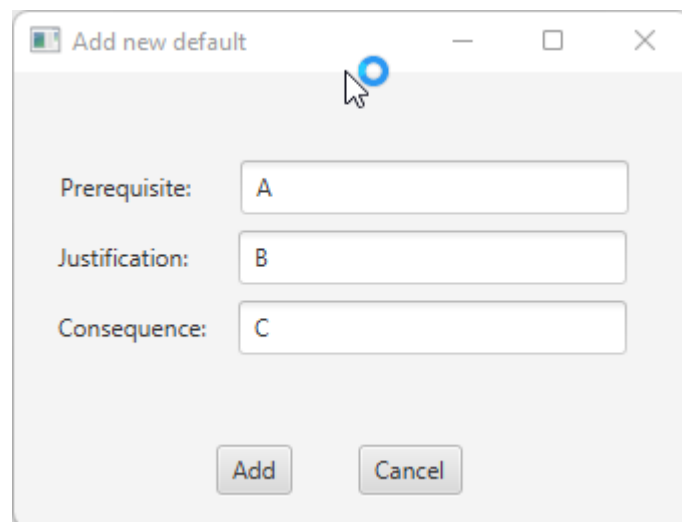


FIGURE 19 – Création du défaut d1



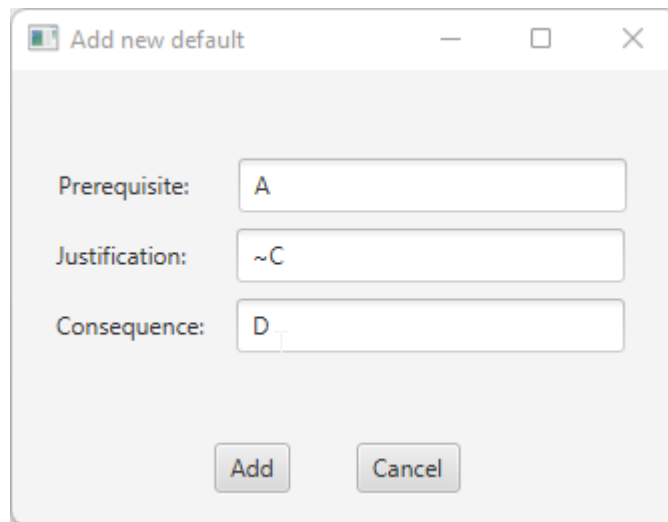


FIGURE 20 – Création du défaut d2

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

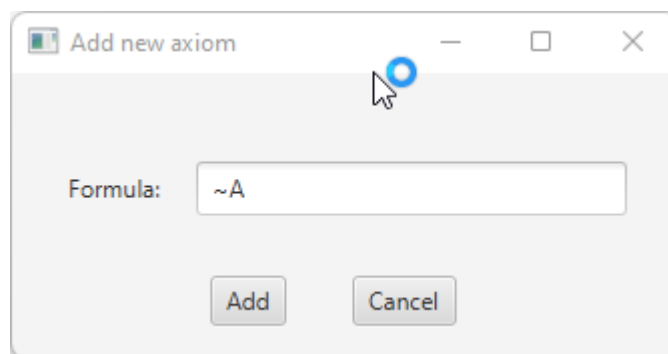


FIGURE 21 – Création du monde w1

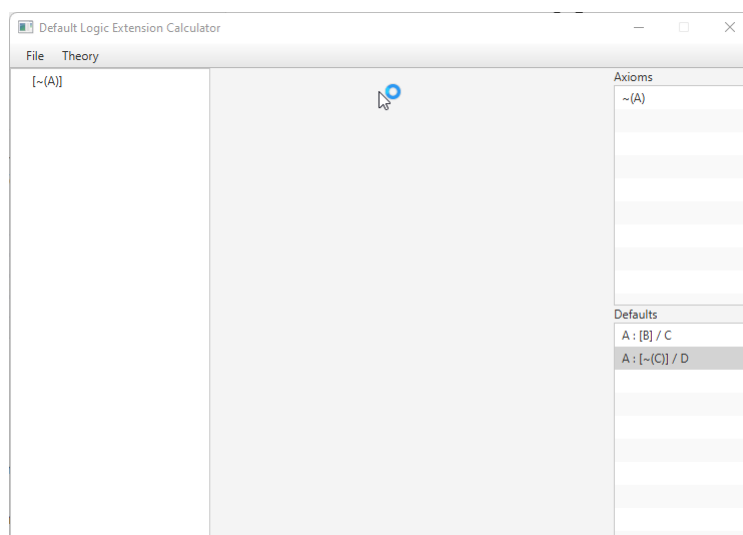


FIGURE 22 – Résultat de l'extension de w1

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

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

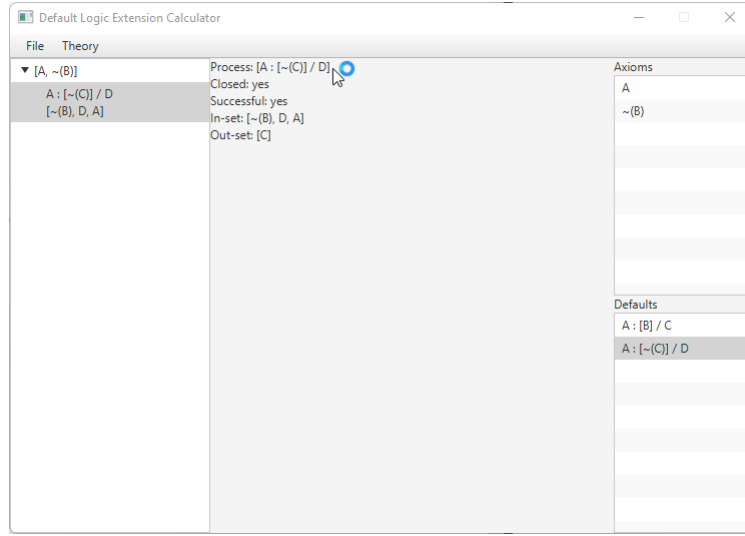


FIGURE 23 – Résultat de l'extension de  $w_2$

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

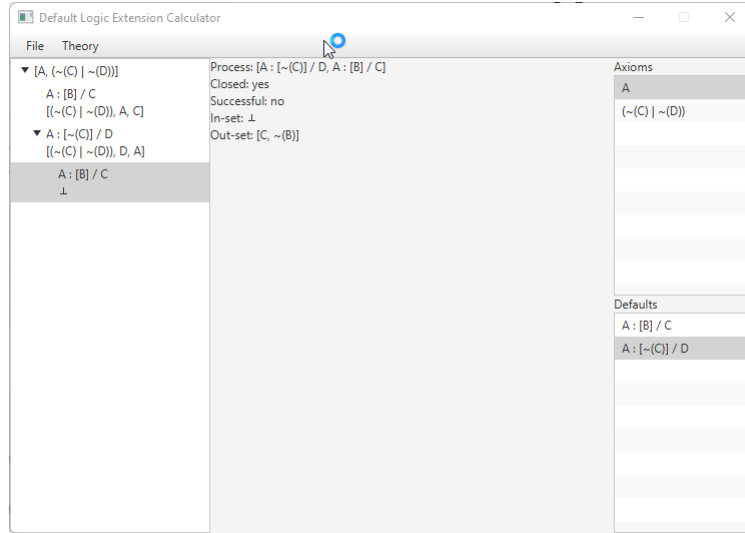


FIGURE 24 – Résultat de l'extension de  $w_3$

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

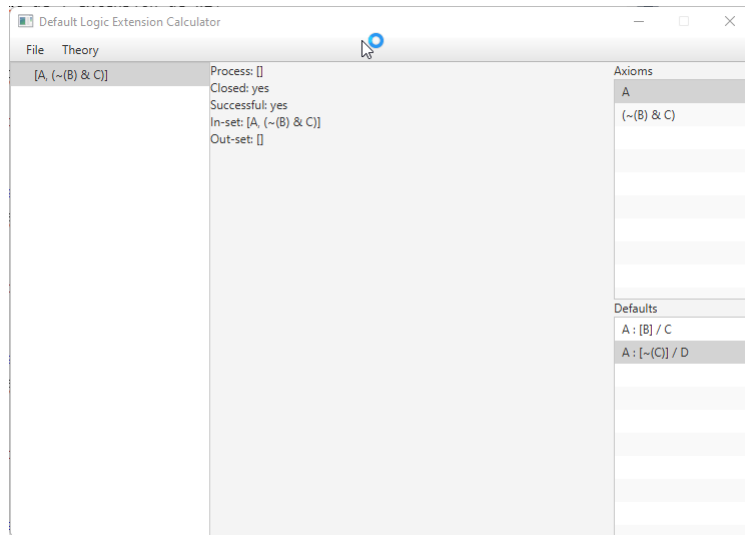


FIGURE 25 – Résultat de l'extension de  $w_4$

## Exercice 2

Nous avons donc :

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

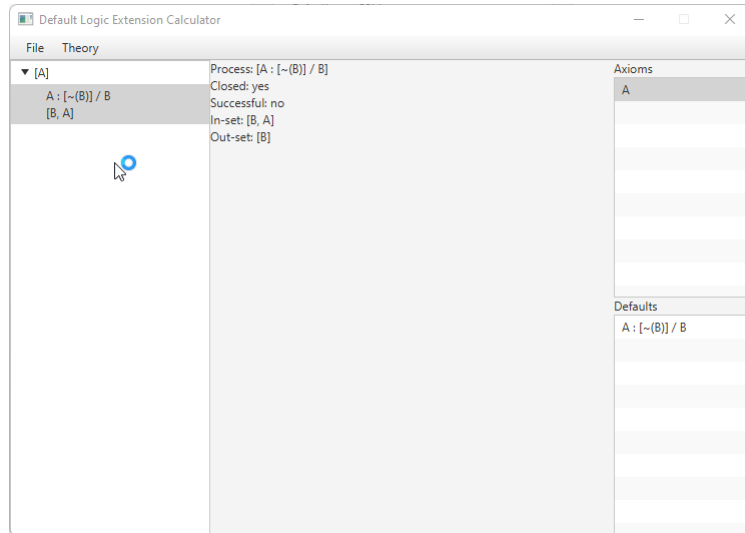


FIGURE 26 – Résultat de l'exercice 2

## Exercice 3

Nous avons donc :

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

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

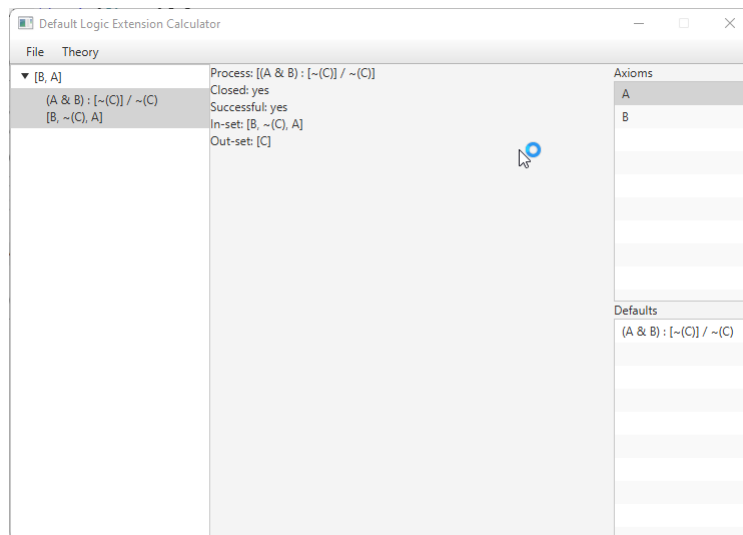


FIGURE 27 – Résultat de l'exercice 3 avec w

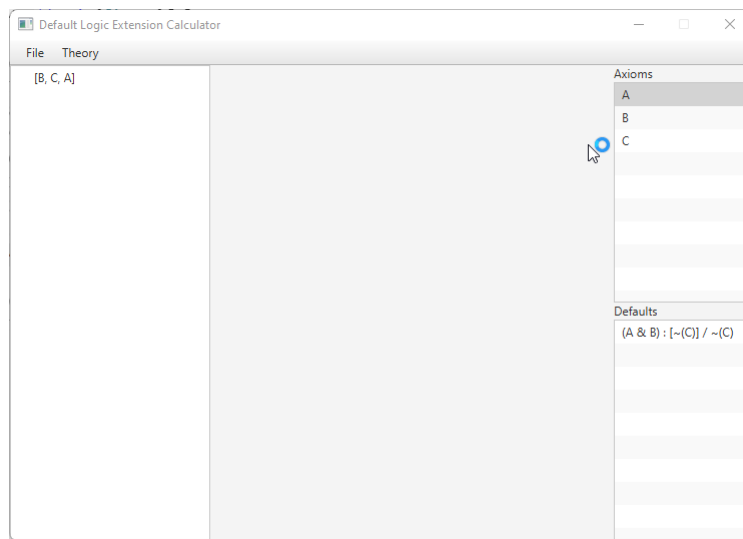


FIGURE 28 – Résultat de l'exercice 3 avec w'

## 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.