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

BOURROUINA Rania
181831052716

CHIBANE Ilies
181831072041

Résumé

Dans ce TP, nous allons implémenter quelques exercices de la série de TD en utilisant la toolbox «defaultlogicc» 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");
```

```
        d1.setJustificatoin("B");
        d1.setConsequence("C");
        rules.addRule(d1);


        DefaultRule d2 = new DefaultRule();  //cr ation d'un d faut d2
        /****************** D finition dun d faut d2 *********************/
        d2.setPrerequisite("A");
        d2.setJustificatoin(e.NOT+"C");
        d2.setConsequence("D");
        rules.addRule(d2);



        /****************** D finition dun monde w1 *********************/
        WorldSet w1= new WorldSet();
        w1.addFormula(e.NOT+"A");



        /****************** D finition dun monde w2 *********************/
        WorldSet w2= new WorldSet();
        w2.addFormula("A");
        w2.addFormula(e.NOT+"B");


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


        /****************** D finition dun monde w4 *********************/
        WorldSet w4= new WorldSet();
        w4.addFormula("A");
        w4.addFormula("("+e.NOT+"B"+ e.AND+"C)");




        /****************execution W1 ************/
        try {
            a.e.println("/****************execution World 1 ************/\n\n\n");
            DefaultReasoner r = new DefaultReasoner(w1, rules); //cr ation du raisonneur


            HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
            a.e.println("W1 : \n\t { " + w1.toString()
                    + " }\n D: \n\t {" + rules.toString() +" }");
            a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
    extension");
            for (String c : scenarios) {
                a.e.println("\t E: Th(W U (" + c + "))");
                // Added closure operator
                a.e.incIndent();
                WFF world_and_ext = new WFF("(( " + w1.getWorld() + " ) & ("
                        + c + "))");
                a.e.println("= " + world_and_ext.getClosure());
                a.e.decIndent();
            }
            a.e.println("");
        }catch(Exception e){

        }
        /****************execution W2 ************/
        try {
          a.e.println("/****************execution World 2 ************/\n\n\n");
            DefaultReasoner r = new DefaultReasoner(w2, rules);
            HashSet<String> scenarios = r.getPossibleScenarios();
            a.e.println("W1 : \n\t { " + w2.toString()
            + " }\n D: \n\t {" + rules.toString() +" }");
```

```
            a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
    extension");
            for (String c : scenarios) {
                a.e.println("\t E: Th(W U (" + c + "))");
                // Added closure operator
                a.e.incIndent();
                WFF world_and_ext = new WFF("(( " + w2.getWorld() + " ) & ("
                        + c + "))");
                a.e.println("= " + world_and_ext.getClosure());
                a.e.decIndent();
            }
            a.e.println("");
        }catch(Exception e){
        }

        /****************execution W3 ************/
        try {
          a.e.println("/****************execution World 3 ************/\n\n\n");
            DefaultReasoner r = new DefaultReasoner(w3, rules);
            HashSet<String> scenarios = r.getPossibleScenarios();
            a.e.println("W1 : \n\t { " + w3.toString()
            + " }\n D: \n\t {" + rules.toString() +" }");
            a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
    extension");
            for (String c : scenarios) {
                a.e.println("\t E: Th(W U (" + c + "))");
                // Added closure operator
                a.e.incIndent();
                WFF world_and_ext = new WFF("(( " + w3.getWorld() + " ) & ("
                        + c + "))");
                a.e.println("= " + world_and_ext.getClosure());
                a.e.decIndent();
            }
            a.e.println("");
        }catch(Exception e){
        }

        /****************execution W4************/
        try {
            a.e.println("/****************execution World 4 ************\n\n\n");
            DefaultReasoner r = new DefaultReasoner(w4, rules);

            HashSet<String> scenarios = r.getPossibleScenarios();
            a.e.println("W1 : \n\t { " + w4.toString()
            + " }\n D: \n\t {" + rules.toString() +" }");
            a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
    extension");
            for (String c : scenarios) {
                a.e.println("\t E: Th(W U (" + c + "))");
                // Added closure operator
                a.e.incIndent();
                WFF world_and_ext = new WFF("(( " + w4.getWorld() + " ) & ("
                        + c + "))");
                a.e.println("= " + world_and_ext.getClosure());
                a.e.decIndent();

            }
            a.e.println("");

        }catch(Exception e){

        }

}

    public static void main(String[] args) {


      exo1();
}
  }
```

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

2.3 Résultat de l'exercice 1

D1 ={ A : B/C}
D2 = { A/ ¬ C/D}

- w_1 ={ ¬ A}

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



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

FIGURE 4 – Résultats pour World 1

Pas d'extensions, les défauts ne sont pas générateurs d'extension.
- w_2 ={ A, ¬ 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, ¬ C ∨¬ 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, ¬ B ∧ 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 appliable.

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 Δ=<W,D> telle que W={A} et D={A : ¬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 : ¬ B/B}.

2.5 Code Source pour l'exercice 2

```java
package be.fnord.DefaultLogic;




import a.e;
import be.fnord.util.logic.DefaultReasoner;
import be.fnord.util.logic.WFF;
import be.fnord.util.logic.defaultLogic.DefaultRule;
import be.fnord.util.logic.defaultLogic.RuleSet;
import be.fnord.util.logic.defaultLogic.WorldSet;

import java.util.HashSet;

public class tp3 {


  public static void exo2() {
    RuleSet rules = new RuleSet(); //pour mettre les d fauts
    DefaultRule d = new DefaultRule(); //cr ation d'un d faut d1
    d.setPrerequisite("A");
    d.setJustificatoin(e.NOT+"B");
    d.setConsequence("B");
    rules.addRule(d);

    WorldSet w= new WorldSet();
    w.addFormula("A");


        /****************execution ************/
        try {
            a.e.println("/****************execution World ************/\n\n\n");
            DefaultReasoner r = new DefaultReasoner(w, rules); //cr ation du raisonneur


            HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
            a.e.println("W1 : \n\t { " + w.toString()
                    + " }\n D: \n\t {" + rules.toString() +" }");
            a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
    extension");
            for (String c : scenarios) {
                a.e.println("\t E: Th(W U (" + c + "))");
                // Added closure operator
                a.e.incIndent();
                WFF world_and_ext = new WFF("(( " + w.getWorld() + " ) & ("
                        + c + "))");
                a.e.println("= " + world_and_ext.getClosure());
                a.e.decIndent();
            }
            a.e.println("");
        }catch(Exception e){

        }
  }

}

    public static void main(String[] args) {


      exo2();
}
  }
```

```
/***************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 Δ=<W,D> et Δ'=<W',D> telles que ;
W={A,B},
W'={A,B,C} et
D={A∧B : ¬C /¬C}.

FIGURE 10 – Enoncé de l'exercice 3

Nous avons donc :
W={A, B} et D={A ∧ B : ¬ C/¬ C}.
W'={A, B, C} et D={A ∧ B : ¬ C/¬ C}.

2.7 Code Source pour l'exercice 3

```java
package be.fnord.DefaultLogic;




import a.e;
import be.fnord.util.logic.DefaultReasoner;
import be.fnord.util.logic.WFF;
import be.fnord.util.logic.defaultLogic.DefaultRule;
import be.fnord.util.logic.defaultLogic.RuleSet;
import be.fnord.util.logic.defaultLogic.WorldSet;

import java.util.HashSet;

public class tp3 {

  public static void exo3() {

    RuleSet rules = new RuleSet(); //pour mettre les d fauts
    DefaultRule d = new DefaultRule(); //cr ation d'un d faut d1
    d.setPrerequisite("A"+e.AND+"B");
    d.setJustificatoin(e.NOT+"C");
    d.setConsequence(e.NOT+"C");
    rules.addRule(d);

    WorldSet w1= new WorldSet();
    w1.addFormula("A");
```

```
    w1.addFormula("B");

    WorldSet w2= new WorldSet();
    w2.addFormula("A");
    w2.addFormula("B");
    w2.addFormula("C");

        /****************execution world 1************/
        try {
            a.e.println("/****************execution World 1 ************/\n\n\n");
            DefaultReasoner r = new DefaultReasoner(w1, rules); //cr ation du raisonneur


            HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
            a.e.println("W1 : \n\t { " + w1.toString()
                    + " }\n D: \n\t {" + rules.toString() +" }");
            a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
    extension");
            for (String c : scenarios) {
                a.e.println("\t E: Th(W U (" + c + "))");
                // Added closure operator
                a.e.incIndent();
                WFF world_and_ext = new WFF("(( " + w1.getWorld() + " ) & ("
                        + c + "))");
                a.e.println("= " + world_and_ext.getClosure());
                a.e.decIndent();
            }
            a.e.println("");
        }catch(Exception e){

        }

        /****************execution world 2 ************/
        try {
            a.e.println("/****************execution World 2************/\n\n\n");
            DefaultReasoner r = new DefaultReasoner(w2, rules); //cr ation du raisonneur


            HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
            a.e.println("W1 : \n\t { " + w2.toString()
                    + " }\n D: \n\t {" + rules.toString() +" }");
            a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
    extension");
            for (String c : scenarios) {
                a.e.println("\t E: Th(W U (" + c + "))");
                // Added closure operator
                a.e.incIndent();
                WFF world_and_ext = new WFF("(( " + w2.getWorld() + " ) & ("
                        + c + "))");
                a.e.println("= " + world_and_ext.getClosure());
                a.e.decIndent();
            }
            a.e.println("");
        }catch(Exception e){

        }

  }

}
    public static void main(String[] args) {
    exo3();
}
  }
```

```
/***************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={(∀ x) (CHRETIENS-LIBANAIS(X) ⊃ LIBANAIS(X)) ; (∀ x) (MELKITE(X) ⊃ ARABE(X)) ; (∀ x) (MELKITE(X) ⊃ ¬ PARLE(X,FRANCAIS)) ; (∀ x) ((MELKITE(X) ⊃ CHRETIENS-LIBANAIS(X) ∧¬ MARONITE(X)) ; }.
D={ CHERTIENC-LIBANAIS(X) :MARONITE(X)/MARONITE(X) ; CHRETIENS-LIBANAIS(X) : ¬ ARABE(X)/¬ ARABE(X) ; LIBANAIS(X) : PARLE(X,FRANÇAIS)/PARLE(X,FRANÇAIS)}.

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 Δ=<W,D,≺> 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 Δ=<W,D, ≺> suivante :
W = ∅
D={a:b/b; :¬a/¬a; :a/a } et
≺: {$d_1 \prec d_3 \prec d_2$}.

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={∅} et D={a :b/b ; :¬ a/¬ a ; :a/a}.

2.10 Code Source pour l'exercice 6

```
package be.fnord.DefaultLogic;




import a.e;
import be.fnord.util.logic.DefaultReasoner;
import be.fnord.util.logic.WFF;
import be.fnord.util.logic.defaultLogic.DefaultRule;
import be.fnord.util.logic.defaultLogic.RuleSet;
import be.fnord.util.logic.defaultLogic.WorldSet;

import java.util.HashSet;
import java.util.LinkedList;

public class tp3 {

  public static void exo6() {


    RuleSet rules = new RuleSet(); //pour mettre les d fauts


    DefaultRule d1 = new DefaultRule(); //cr ation d'un d faut d1
    d1.setPrerequisite("a");
    d1.setJustificatoin("b");
    d1.setConsequence("b");
    rules.addRule(d1);


    DefaultRule d2 = new DefaultRule(); //cr ation d'un d faut d1
    d2.setPrerequisite(e.EMPTY_FORMULA);
    d2.setJustificatoin(e.NOT+"a");
    d2.setConsequence(e.NOT+"a");
    rules.addRule(d2);

```

```
    DefaultRule d3 = new DefaultRule(); //cr ation d'un d faut d1
    d3.setPrerequisite(e.EMPTY_FORMULA);
    d3.setJustificatoin("a");
    d3.setConsequence("a");
    rules.addRule(d3);

    WorldSet w= new WorldSet();
    w.addFormula(e.EMPTY_FORMULA);




        /***************execution world ************/
        try {
            a.e.println("/***************execution for empty World ************/\n\n\n");
            DefaultReasoner r = new DefaultReasoner(w, rules); //cr ation du raisonneur


            HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
            a.e.println("W1 : \n\t { " + w.toString()
                    + " }\n D: \n\t {" + rules.toString() +" }");
            a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
    extension");
            for (String c : scenarios) {
                a.e.println("\t E: Th(W U (" + c + "))");
                // Added closure operator
                a.e.incIndent();
                WFF world_and_ext = new WFF("(( " + w.getWorld() + " ) & ("
                        + c + "))");
                a.e.println("= " + world_and_ext.getClosure());
                a.e.decIndent();
            }
            a.e.println("");
        }catch(Exception e){

        }

  }


}
    public static void main(String[] args) {

      exo6();
}
  }
```

```
/***************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 Λ=<W,D,$\prec$> 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

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

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
```

```
import java.util.HashSet;
import java.util.LinkedList;

public class tp3 {

  public static void exo7() {
RuleSet rules = new RuleSet(); //pour mettre les d fauts


    DefaultRule d1 = new DefaultRule(); //cr ation d'un d faut d1
    d1.setPrerequisite(e.EMPTY_FORMULA);
    d1.setJustificatoin("p");
    d1.setConsequence("p");
    rules.addRule(d1);


    DefaultRule d2 = new DefaultRule(); //cr ation d'un d faut d1
    d2.setPrerequisite("r");
    d2.setJustificatoin(e.NOT+"q");
    d2.setConsequence(e.NOT+"q");
    rules.addRule(d2);

    DefaultRule d3 = new DefaultRule(); //cr ation d'un d faut d1
    d3.setPrerequisite("s");
    d3.setJustificatoin("t");
    d3.setConsequence("t");
    rules.addRule(d3);


    DefaultRule d4 = new DefaultRule(); //cr ation d'un d faut d1
    d4.setPrerequisite("p");
    d4.setJustificatoin("v");
    d4.setConsequence("v");
    rules.addRule(d4);

    DefaultRule d5 = new DefaultRule(); //cr ation d'un d faut d1
    d5.setPrerequisite("q");
    d5.setJustificatoin(e.NOT+"v");
    d5.setConsequence(e.NOT+"v");
    rules.addRule(d5);

    DefaultRule d6 = new DefaultRule(); //cr ation d'un d faut d1
    d6.setPrerequisite("v");
    d6.setJustificatoin("u");
    d6.setConsequence("u");
    rules.addRule(d6);

    WorldSet w= new WorldSet();
    w.addFormula("(p -> (q "+e.AND+"r))");
    w.addFormula("(r ->"+e.NOT+"s)");




        /****************execution world ************/
        try {
            a.e.println("/****************execution for empty World ************/\n\n\n");
            DefaultReasoner r = new DefaultReasoner(w, rules); //cr ation du raisonneur


            HashSet<String> scenarios = r.getPossibleScenarios(); //faire l'extension
            a.e.println("W1 : \n\t { " + w.toString()
                    + " }\n D: \n\t {" + rules.toString() +" }");
            a.e.println("Par cl ture d ductive et minimalit , cette th orie admet une seule
    extension");
            for (String c : scenarios) {
                a.e.println("\t E: Th(W U (" + c + "))");
                // Added closure operator
                a.e.incIndent();
                WFF world_and_ext = new WFF("(( " + w.getWorld() + " ) & ("
                        + c + "))");
                a.e.println("= " + world_and_ext.getClosure());
```

```
                a.e.decIndent();
            }
            a.e.println("");
        }catch(Exception e){

        }
  }
}

    public static void main(String[] args) {
      exo7();
}
  }
```

```
/***************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)
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 bientot selon le site.

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

Ces Toolboxs sont similaires à DefaultLogic Library.

Toolbox 4 : Extension Calculator

Voici l'interface de l'outil

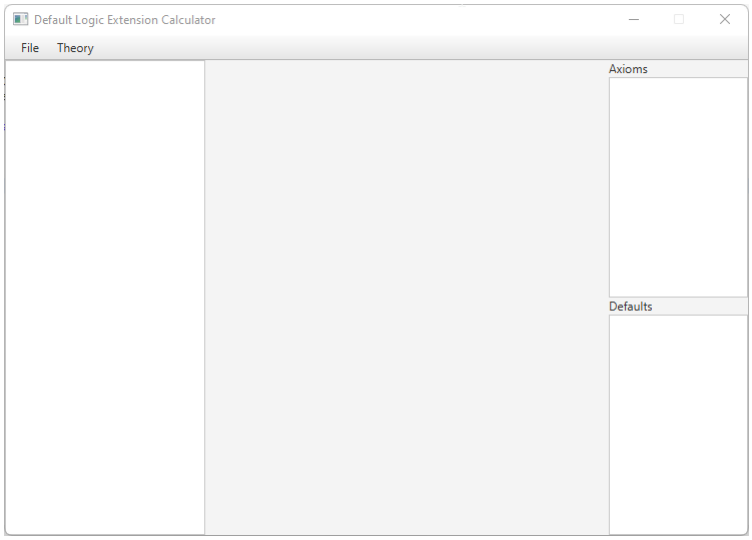

FIGURE 18 – Interface de l'outil Extension Calculator

Exercice 1

D1 ={ A : B/C}
D2 = { A/ ¬ C/D}

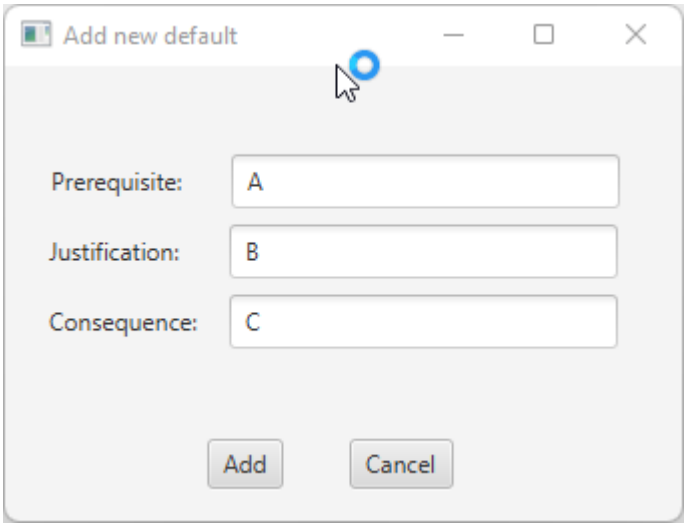

FIGURE 19 – Création du défaut d1

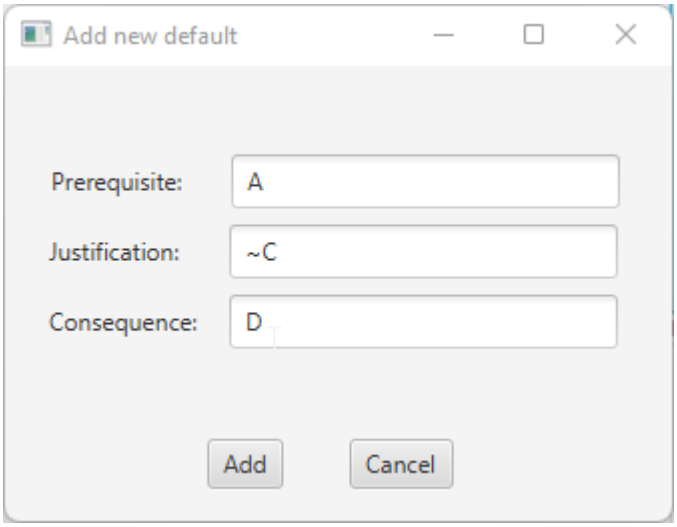

FIGURE 20 – Création du défaut d2

- w_1 ={ ¬ A}

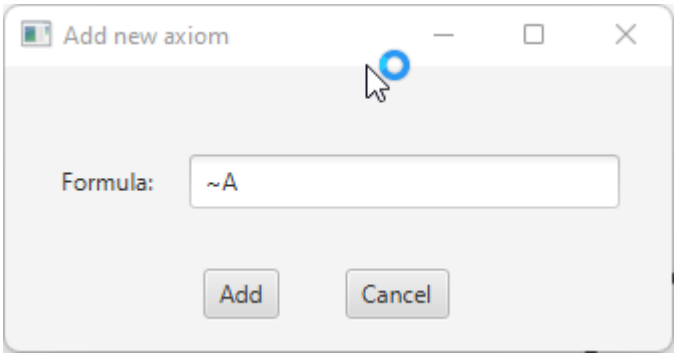

FIGURE 21 – Création du monde w1

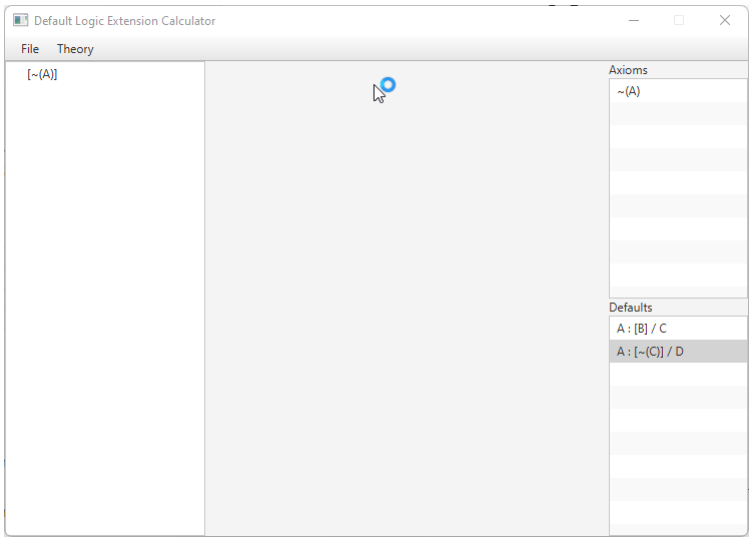

FIGURE 22 – Résultat de l'extension de w1

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

- w_2 ={ A, ¬ B}

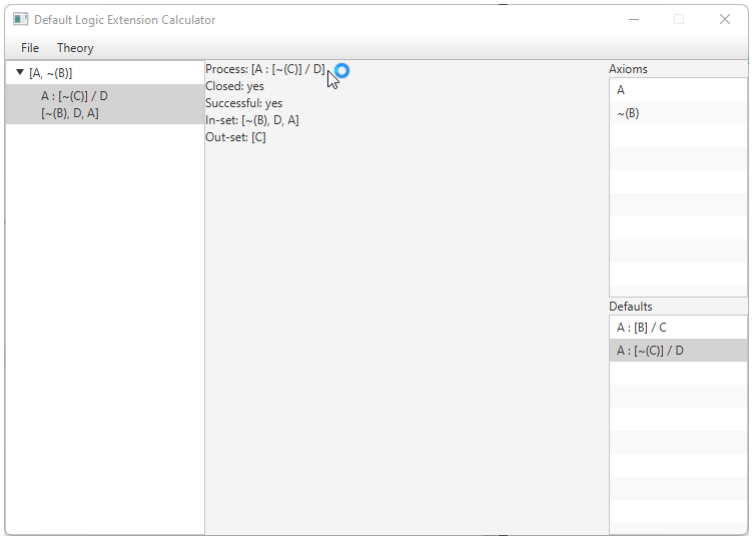

FIGURE 23 – Résultat de l'extension de w2

- w_3 ={ A, ¬ C ∨¬ D}

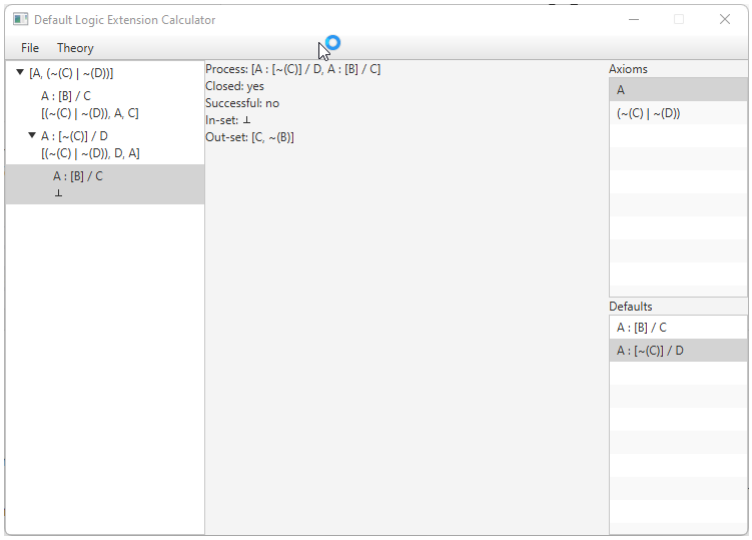

FIGURE 24 – Résultat de l'extension de w3

- w_4 ={ A, ¬ B ∧ C}

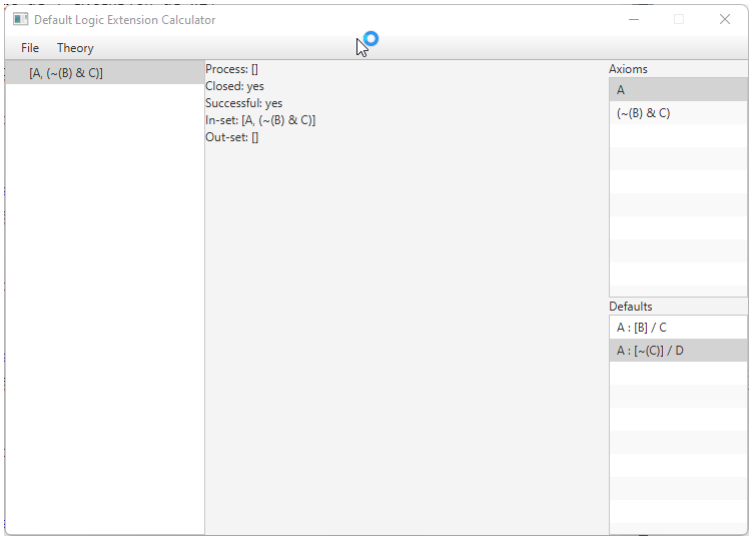

FIGURE 25 – Résultat de l'extension de w4

Exercice 2

Nous avons donc :
W={A} et D={A : ¬ B/B}.

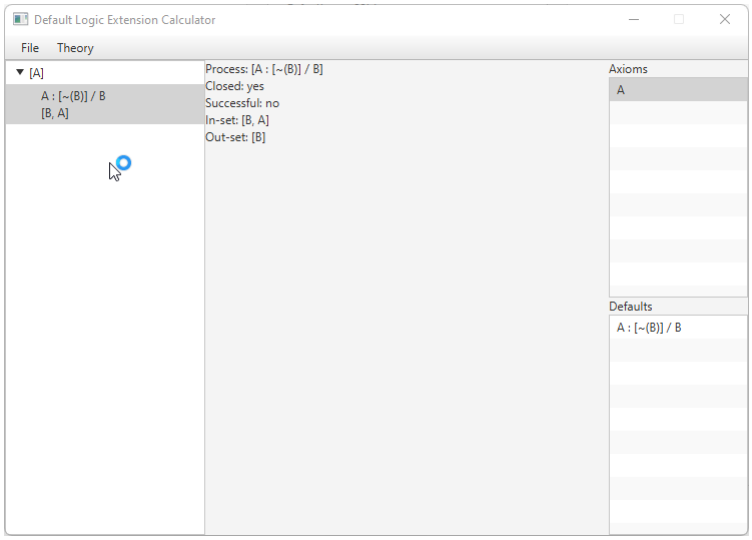

FIGURE 26 – Résultat de l'exercice 2

Exercice 3

Nous avons donc :
W={A, B} et D={A ∧ B : ¬ C/¬ C}.
W'={A, B, C} et D={A ∧ B : ¬ C/¬ C}.

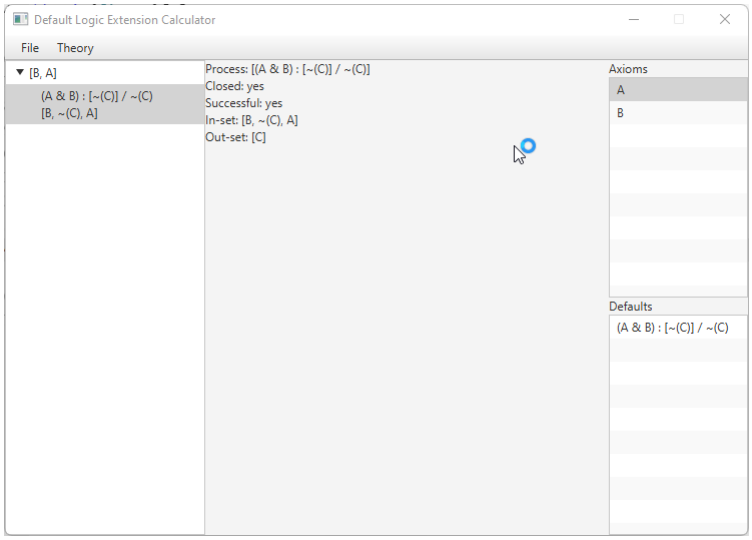

FIGURE 27 – Résultat de l'exercice 3 avec w

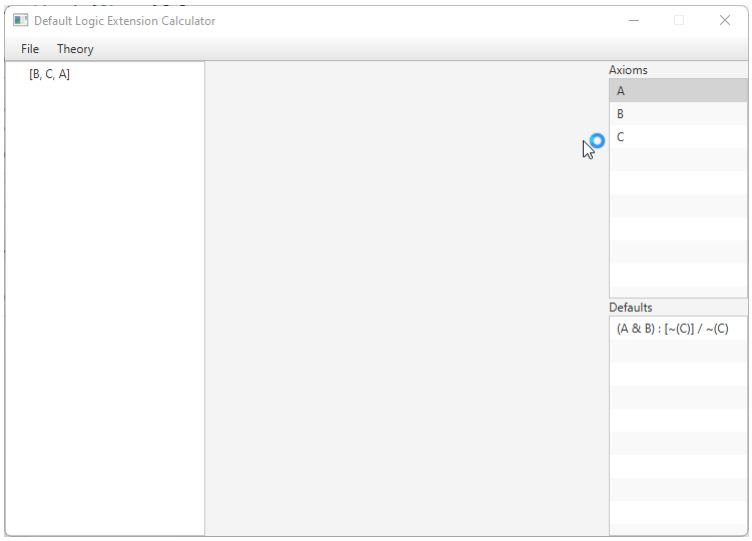

FIGURE 28 – Résultat de l'exercice 3 avec w'

Exercice 4 et 5

Cette toolbox ne permet pas de saisir le symbole quel que soit : «∀» et le symbole d'implication «⊃»

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 etre résolu avec cette toolbox.