Rapport de TP03

Réalisé par :

SOUIDI Mohamed Amine

181831044438

DJEKOUNE Imad Eddine

181831089663

M2 IV G2 2021-2022

L'énoncé de TP

Il s'agit d'exploiter une des boites à outils relatives au raisonnement basé sur la logique des

Défauts, nous allons implémenter quelques exercices de la série de TD en utilisant la toolbox «defaultlogicc» conçue en java par Evan Morrison.

Solution :

On 'a Traiter les 3 premiers exercices :

Code source :

Main : donner le choix d'utilisateur a choisit l'exercice

```java
package be.fnord.DefaultLogic;

import java.util.Scanner;

public class Main {

     public static void main( String[] args )
        {
         Scanner sc= new Scanner(System.in);
         System.out.print("Entrer le numéro de l'exercice : ");
         int choix= sc.nextInt();

         switch(choix)
         {

           case 1:
               Exo1.solution();
               break;

           case 2:
               Exo2.solution();
               break;

           case 3:
               Exo3.solution();
               break;
           default:
               System.out.println("Choix incorrect");
               break;
       }

        }

}
```

Exo1 :

Soit l'ensemble de défauts D={d1,d2} avec d1= A: B/C et d2= A:¬C/D .

Quelles sont les extensions qui peuvent se déduire si on considère les ensembles de formules suivantes:

1. W={ ¬A }
2. W={A, ¬B }
3. W={A, ¬C∨¬D }
4. W={A, ¬B∧C }

```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 Exo1 {

 public static void solution (){

     RuleSet rules = new RuleSet (); // pour mettre les dfault

    DefaultRule d1 = new DefaultRule (); // cration d'un dfault d1
    /* ****************** Difinition dun dfault d1 ******************** */
     d1.setPrerequisite ("A");

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


     DefaultRule d2 = new DefaultRule (); // cration d'un dfault d2
     /* ****************** dfault d2 ******************** */
     d2.setPrerequisite ("A");
    d2.setJustificatoin (e.NOT +"C");
    d2.setConsequence ("D");
    rules.addRule (d2);



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



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

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


    /* ******************  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 ); // cration 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 clture  dductive et minimalit ");
     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 clture  dductive et minimalit");
    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 clture  dductive et minimalit ");
    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 clture  dductive et minimalit ");
    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){

    }

}


}
```

Output :

```
Entrer le numéro de l'exercice : 1
/**************** execution World 1 ************/



W1 :
        { ~A }
 D:
        {[(A):(B) ==> (C)] , [(A):(~C) ==> (D)] }
Par clture  dductive et minimalit

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



Trying eeee & A & ~B
Trying eeee & A & ~B
W1 :
        { A & ~B }
 D:
        {[(A):(B) ==> (C)] , [(A):(~C) ==> (D)] }
Par clture  dductive et minimalit
        E: Th(W U (D))
       = D & ~B & eeee & A

 /**************** 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 clture  dductive et minimalit
        E: Th(W U (C))
       = C & ~D & eeee & (~D | ~C) & A
```

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



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

Pour w1 pas d'extensions, les défauts ne sont pas générateurs d'extension.

Exo2 :

```
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 Exo2
{

    public static void solution () {
                    RuleSet rules = new RuleSet(); // pour mettre les dfauts
                    DefaultRule d =  new  DefaultRule ();  // c r a t i o n  d' un  d  f a u t
d1 d.set Prerequisite (" 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 ); // cration 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 clture dductive et minimalit Extensions :  ");
                    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){
                    }

    }


}
```

Output :

```
<terminated> Main (9) [Java Application] D:\programmes\jdk-12.0.2\bin\javaw.exe  (Jan 13, 2023,
Entrer le numéro de l'exercice : 2
 /**************** execution World ************/



 W1 :
        { A }
 D:
        {[([]):(~ B) ==> (B)] }
 Par clture dductive et minimalit Extensions :
```

Exo 3 :

Quelles sont les extensions des théories □=<W,D> et □'=<W',D> telles que ;

W={A,B},

W'={A,B,C} et

D={A□B : ¬C /¬C}.

```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  Exo3 {
public static void solution () {

        RuleSet rules = new RuleSet(); // pour mettre les defauts
        DefaultRule  d =  new  DefaultRule();  // creation  d'un  defaut  d1
        d.setPrerequisite("A&B");
        d.setJustificatoin("~C");
        d.setConsequence("~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 ); // cration du raisonneur


            HashSet<String> scenarios  = r.getPossibleScenarios();  // faire  l' extension
            a.e.println (" W1 :\n\t { " + w1.toString() + " }\nD:\n\t {" + rules.toString() +"
}");
            a.e.println (" Par clture  dductive et minimalit , cette thorie admet une
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 ); // c r a t i o n 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 clture  dductive et minimalit , cette thorie admet une
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){

                    }

                }


        }
```

Out put :

```
Entrer le numéro de l'exercice : 3
 /**************** execution World 1 ************/



Trying eeee & A &  B
 W1 :
        { A &  B }
D:
        {[(A&B):(~C) ==> (~C)] }
 Par clture  dductive et minimalit , cette thorie admet une  extension
        E: Th( W U (~C))
        = B & ~C & eeee & A

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



 W1 :
        { A & B & C }
 D:
        {[(A&B):(~C) ==> (~C)] }
 Par clture  dductive et minimalit , cette thorie admet une extension
```